Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**«Μοντέλα Συστημικής Δυναμικής: Εφαρμογές του AnyLogic»**
Ονοματεπώνυμο Φοιτητή	**Ηλίας Μακρυγιάννης**
Πατρώνυμο	**Αθανάσιος**
Αριθμός Μητρώου	**ΜΠΠΛ/06009**
Επιβλέπων	**Νικήτας Ασημακόπουλος, Καθηγητής**

Ημερομηνία Παράδοσης **Φεβρουάριος 2012**

Μεταπτυχιακή Διατριβή

Τριμελής Εξεταστική Επιτροπή

Νικήτας Ασημακόπουλος Καθηγητής	Αριστέα Σινανιώτη Καθηγήτρια	Χαράλαμπος Κωνσταντόπουλος Λέκτορας

Ευχαριστίες

Τελειώνοντας αυτό το προσωπικό ταξίδι αναζήτησης νέων γνώσεων στο ΠΜΣ «ΠΛΗΡΟΦΟΡΙΚΗ» του Τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς θα ήθελα να ευχαριστήσω, για την υπομονή που επέδειξαν, όλους όσους κούρασα με το καθημερινό μου πρόγραμμα τα τελευταία χρόνια.

Εκφράζω τις θερμές μου ευχαριστίες σε όλους τους συντελεστές του ΠΜΣ «Πληροφορική» για την πολύ καλή συνεργασία μας και για τα όσα μου προσέφεραν κατά τη διάρκεια της φοίτησης μου στο Μεταπτυχιακό Πρόγραμμα Σπουδών. Ευχαριστώ το Διδάκτορα κ. Ιωάννη Θεοχαρόπουλο, για την καθοδήγηση που μου προσέφερε κατά την ανάληψη και κατά τα πρώτα στάδια εκπόνησης της συγκεκριμένης Μεταπτυχιακής Διατριβής.

Ιδιαίτερα ευχαριστήρια στον Καθηγητή κ. Νικήτα Ασημακόπουλο για τη συνεργασία, την καθοδήγηση, τη βοήθεια και την ανοχή (που έδειξε στην καταστρατήγηση από μέρους μου του χρονοδιαγράμματος περάτωσης της εργασίας).

Περίληψη

Η κεφαλαιώδης σημασία της μοντελοποίησης και η αναγκαιότητα της προσομοίωσης για την επίλυση επιστημονικών, όχι μόνο θεωρητικών αλλά και καθημερινών πρακτικών, προβλημάτων είναι αναμφισβήτητη. Η αποτελεσματικότητα της προσομοίωσης εξαρτάται απόλυτα από τα εργαλεία μοντελοποίησης που έχουμε, από τις δυνατότητες που αυτά έχουν και ασφαλώς από το πόσο καλά και αποτελεσματικά τα χρησιμοποιούμε για τη δημιουργία των μοντέλων.

Η παρούσα Μεταπτυχιακή Διατριβή πραγματεύεται την παρουσίαση παραδειγμάτων μοντέλων Συστημικής Δυναμικής, εφαρμογών μοντελοποίησης και προσομοίωσης, που αναπτύχθηκαν με το λογισμικό AnyLogic της εταιρείας XJ Technologies. Αναδεικνύονται ο τρόπος ανάλυσης, η διαδικασία σχεδιασμού και η δημιουργία των μοντέλων. Παρουσιάζονται, τέλος, τα αποτελέσματα της προσομοίωσης μέσα από το περιβάλλον ανάπτυξης εφαρμογών του λογισμικού AnyLogic.

Πιστεύουμε ακράδαντα ότι με το πέρας της ανάγνωσης αυτής της Μεταπτυχιακής Διατριβής, ο αναγνώστης θα έχει σταθμίσει το κατά πόσο το συγκεκριμένο λογισμικό μπορεί να θεωρηθεί ως προσθετικός παράγοντας γνώσης και ισχύος στο επιστημονικό πεδίο της Συστημικής Ανάλυσης.

Λέξεις κλειδιά: Προσομοίωση, Μοντελοποίηση, Μοντέλα, AnyLogic, XJTek

Abstract

The essential importance of modeling and the necessity of simulation to solve scientific, not only theoretical but also everyday practical, problems are undeniable. The effectiveness of simulation depends entirely on the modeling tools we have, on the sufficiency they have and of course on how well and effectively used for modeling.

The present master thesis deals with the presentation of examples of Systemic Dynamics models, modeling and simulation applications, developed with AnyLogic software by XJ Technologies Company. It reveals the analysis way, the planning process and modeling. Presented, finally, the simulation results through the application development environment of AnyLogic.

We firmly believe that by the end of reading this master thesis, the reader will weigh whether this software can be considered as additive agent of knowledge and power in the scientific field of Systemic Analysis.

Keywords: Simulation, Modeling, Model, AnyLogic, XJTek

Πρόλογος

Ένα παλαιό λαϊκό γνωμικό ισχυρίζεται πως τα καλά εργαλεία είναι αυτά που κάνουν έναν τεχνίτη καλό. Ίσως αρκετοί αναγνώστες να έχουν αντιρρήσεις, όμως προσωπικά συμφωνώ με το παραπάνω. Διαστέλλοντας την ερμηνεία του παραπάνω γνωμικού μπορούμε να θεωρήσουμε πως όσο καλός και να είναι κάποιος στην εργασία του, μια μικρή ή μεγαλύτερη βοήθεια θα τον αναδείξει ακόμη περισσότερο και θα γίνει ακόμη καλύτερος στο χώρο δραστηριότητας του. Κατά γενική ομολογία, ισχύει στις μέρες μας ότι αυτός που κατέχει ή που έχει πρόσβαση στα κατάλληλα ποιοτικά εργαλεία ΚΑΙ παράλληλα έχει τη γνώση να τα χρησιμοποιήσει εποικοδομητικά μπορεί, στο χώρο που δραστηριοποιείται, να θεωρηθεί καλός τεχνίτης.

Τα καλά εργαλεία στις μέρες μας έχουν πάψει να είναι μόνο υλικά. Με την ανάπτυξη των επιστημών, των τεχνολογικών επιτευγμάτων αλλά κυρίως με την άνθηση της Πληροφορικής τα εργαλεία λογισμικού είναι αυτά που άλλαξαν το τοπίο σε σχεδόν όλους τους επιστημονικούς κλάδους και προσέδωσαν νέες δυνατότητες στους ερευνητές και στους επαγγελματίες όλων των χώρων που βρίσκονται στην αιχμή του δόρατος του σύγχρονου τεχνοοικονομικού περιβάλλοντος.

Το λογισμικό που θα παρουσιαστεί στη συνέχεια αυτής της Μεταπτυχιακής Διατριβής είναι από τα πλέον σύγχρονα που έχει να επιδείξει ο επιστημονικός κλάδος της Συστημικής Ανάλυσης και της Μοντελοποίησης. Η αξία του λογισμικού θα φανεί και πιστεύουμε θα αναγνωριστεί μέσα από την παράθεση μερικών παραδειγμάτων προσομοίωσης, παραδείγματα που προέρχονται από εφαρμογές μοντέλων σε κλάδους της επιστήμης.

Στο σημείο αυτό ευχαριστώ για μια ακόμη φορά τον Καθηγητή κ. Νικήτα Ασημακόπουλο για τη βοήθεια που μου έτεινε, για την εμπιστοσύνη που επέδειξε στο πρόσωπο μου και για την καθοδήγηση, (η οποία δεν ήταν σχετιζόμενη αποκλειστικά με την εργασία), που μου προσέφερε.

Τέλος, προσπάθησα να αποφύγω τα λάθη και τις ανακρίβειες και να φανώ αντικειμενικός σε αυτά που παραθέτω. Τα τυχόν λάθη που ίσως εντοπισθούν, οι πιθανές παραβλέψεις, ατέλειες ή παραλείψεις παρατηρηθούν με βαρύνουν αποκλειστικά.

Φεβρουάριος 2012 - Ηλίας Αθ. Μακρυγιάννης

Περιεχόμενα

Εισαγωγή

Η αξία της μοντελοποίησης και της προσομοίωσης έγκειται στο γεγονός ότι μπορούμε, είτε εκ των προτέρων είτε εκ των υστέρων, να αναλύσουμε πολύπλοκα συστήματα και να σχεδιάσουμε συστήματα που αναπαριστούν φυσικά φαινόμενα, που περιγράφουν επιχειρηματικές, οικονομικές, κοινωνικές διαδικασίες κ.λ.π. που συμβαίνουν καθημερινά γύρω μας.

Το χαρακτηριστικό γνώρισμα των παραπάνω, (των φαινομένων και των διαδικασιών που συναντάμε καθημερινά), είναι η πολύ-παραμετρική φύση τους και η ύπαρξη στοχαστικών (αστάθμητων ή άλλων απρόβλεπτων) παραγόντων που επηρεάζουν την πορεία των φαινομένων και των διαδικασιών. Με άλλα λόγια τα προς εξέταση συστήματα είναι δύσκολο να μοντελοποιηθούν με κλειστά μαθηματικά μοντέλα, (δημιουργώντας ένα σύνολο από κλειστές μαθηματικές εξισώσεις), τις περισσότερες φορές και προσπαθούμε μέσω της προσομοίωσης να κατασκευάσουμε αναπαραστάσεις που θα «μοιάζουν» με τα πραγματικά φαινόμενα και τις διαδικασίες. Εντοπίζοντας τους παράγοντες που είναι υπεύθυνοι για τις μεταβολές και τις αποκλίσεις εντός των διαδικασιών και μειώνοντας τα περιθώρια σφάλματος στο ελάχιστο φτάνουμε να μιμούμαστε όσο καλύτερα και αποδοτικά γίνεται τα πραγματικά συστήματα.

Διακρίνονται τρείς (3) μεγάλες κατηγορίες μοντελοποίησης ανάλογα με ποια μεθοδολογία χρησιμοποιείται. Έχουμε τη μοντελοποίηση Διακριτού Γεγονότος, τη μοντελοποίηση της Δυναμικής Συστημάτων και τη μοντελοποίηση μέσω Πρακτόρων Λογισμικού. Συνδυάζοντας δύο ή περισσότερες από τις προαναφερθείσες κατηγορίες οδηγούμαστε στην Υβριδική μοντελοποίηση.

Η διατριβή χωρίζεται σε τρία (3) μέρη. Το πρώτο μέρος αποτελείται από τα κεφάλαια 1 και 2. Στο πρώτο κεφάλαιο επεξηγείται συνοπτικά η έννοια της μοντελοποίησης και της προσομοίωσης ώστε ο αναγνώστης να εισαχθεί στην έννοια της υβριδικής μοντελοποίησης και στην προσομοίωση πολύπλοκων καθημερινών φαινομένων. Στο δεύτερο κεφάλαιο περιγράφεται το λογισμικό AnyLogic της εταιρείας XJ Technologies. Δίνεται μια πολύ συνοπτική περιγραφή του περιβάλλοντος του λογισμικού και γίνεται στοχευόμενη παρουσίαση σημείων που θεωρούμε ότι είναι σημαντικά για την ανάλυση των παραδειγμάτων που θα ακολουθήσουν στα επόμενα κεφάλαια.

Το δεύτερο μέρος της διατριβής αποτελείται από τα κεφάλαια 3 μέχρι και 10. Σε κάθε ένα από τα κεφάλαια αυτά παρουσιάζεται ένα παράδειγμα προσομοίωσης. Τα παραδείγματα των κεφαλαίων 3, 4, 7 και 8 προέχονται από το χώρο της υγείας. Τα παραδείγματα των κεφαλαίων 5, 6 και 9 προέρχονται από το χώρο της κοινωνιολογίας και των οικοσυστημάτων ενώ το παράδειγμα του κεφαλαίου 10 προέρχεται από το χώρο της τεχνολογίας της πληροφορικής και των τηλεπικοινωνιών και αφορά σε ζητήματα υποδομών. Περιγράφεται το θεωρητικό πρόβλημα, παρουσιάζεται ο τρόπος ανάλυσης και η υλοποίηση με το λογισμικό, παρατίθεται ο κώδικας και η διαδικασία δημιουργίας του μοντέλου και τέλος δίνεται η προσομοίωση του μοντέλου του παραδείγματος.

Στο τρίτο κεφάλαιο παρουσιάζεται το προβλήματος της διασποράς μιας μολυσματικής νόσου σε ορισμένη γεωγραφική περιοχή και σε συγκεκριμένο πληθυσμό. Μελετάται η εξάπλωση μιας επιδημίας με τη χρήση του μοντέλου SIR των Kermack και McKendrick. Για τη μοντελοποίηση του προβλήματος χρησιμοποιείται η μέθοδος της Δυναμικής Συστημάτων, (System Dynamics). Στο τέταρτο κεφάλαιο μελετάται το ίδιο πρόβλημα όπως στο κεφάλαιο 3 αλλά χρησιμοποιείται η μέθοδος των Πρακτόρων Λογισμικού, (Agent Based), για τη μοντελοποίηση του προβλήματος. Στο πέμπτο κεφάλαιο μας απασχολεί το πρόβλημα της επιβίωσης και της διακύμανσης του πληθυσμού δύο ειδών, όπου το ένα είδος είναι το θήραμα και το άλλο είδος ο θηρευτής, σε μια απομονωμένη γεωγραφική περιοχή. Η μοντελοποίηση γίνεται μέσω της Συστημικής Δυναμικής. Στο έκτο κεφάλαιο μελετάται το ίδιο πρόβλημα όπως στο κεφάλαιο 5 αλλά χρησιμοποιείται η μέθοδος των Πρακτόρων Λογισμικού για τη μοντελοποίηση του προβλήματος. Στο έβδομο κεφάλαιο εξετάζουμε με τη μέθοδο των Πρακτόρων Λογισμικού την εξάπλωση του ιού HIV σε μια γεωγραφική περιοχή και σε συγκεκριμένες κατηγορίες πληθυσμού. Στο όγδοο κεφάλαιο παρουσιάζεται η προσομοίωση ενός τμήματος επειγόντων περιστατικών ενός νοσοκομείου. Η μέθοδος μοντελοποίησης είναι αυτή

των Διακριτών Γεγονότων, (Discrete Event). Στο ένατο κεφάλαιο το προς εξέταση ζήτημα είναι η μοντελοποίηση μιας κοινωνικής εξέγερσης μέσω της υβριδικής μεθόδου, (Hybrid ή Combined) και τέλος στο δέκατο κεφάλαιο μας ενδιαφέρει η μοντελοποίηση του ζητήματος της εκλογής αρχηγού σε ένα δίκτυο κατανεμημένων υπολογιστών. Η μοντελοποίηση γίνεται μέσω της μεθόδου των Πρακτόρων Λογισμικού.

Στο τρίτο μέρος της διατριβής καταγράφονται τα συμπεράσματα, ακολουθούν τα παραρτήματα όπου παρουσιάζονται τα διαγράμματα των κλάσεων σε UML και ο κώδικας των παραδειγμάτων και τέλος η βιβλιογραφία.

Η συγγραφή της Μεταπτυχιακής Διατριβής βασίστηκε στην έκδοση του λογισμικού AnyLogic 6.4.0 με ακαδημαϊκή άδεια χρήσης.

ΚΕΦΑΛΑΙΟ 1

Εισαγωγικές έννοιες

1.1 Μοντελοποίηση - Προσομοίωση

Η σημασία της κατανόησης ενός φυσικού φαινομένου, ενός μαθηματικού προβλήματος ή γενικότερα η επίλυση κάποιου ζητήματος της καθημερινής ζωής, μέσα από την επιστημονική θεώρηση των πραγμάτων είναι μεγάλη. Η προσπάθεια των ειδικών να ολοκληρώσουν με επιτυχία το έργο τους δηλαδή να ανακαλύψουν την λύση ή τις λύσεις του επιστημονικού ζητήματος γίνεται ερευνώντας τους φυσικούς νόμους και τις αλληλεπιδράσεις των παραμέτρων και των μεταβλητών που αφορούν το εν λόγω ζήτημα.

Γενικότερα, ένα σύνολο αλληλεπιδρώντων στοιχείων τα οποία συνεργάζονται μεταξύ τους ή λειτουργούν συλλογικά για την επίτευξη κάποιου σκοπού ονομάζεται σύστημα, (system). Ως μοντελοποίηση, (modeling), ενός συστήματος ορίζεται η διαδικασία της περιγραφής και της αναπαράστασης όλων των εννοιών που το αποτελούν και όλων των σχέσεων που επηρεάζουν τη συμπεριφορά του. Το αποτέλεσμα της μοντελοποίησης είναι η δημιουργία του μοντέλου, (model), του συστήματος. Η ανάλυση του συστήματος με βάση τον παραπάνω ορισμό μας δίνει, στη γενική περίπτωση, ένα αφαιρετικό μοντέλο αναπαράστασης της πραγματικότητας διότι πιθανά ένα μέρος της πληροφορίας του πραγματικού συστήματος δεν μπορεί να εξαχθεί. Σε μια τέτοια περίπτωση ενδιαφερόμαστε για την ελαχιστοποίηση του λάθους μας κατά τη δημιουργία του τεχνητού συστήματος.

Συνοψίζοντας τα παραπάνω, μια γενική μεθοδολογία για τη δημιουργία ενός μοντέλου θα αποτελούνταν από τα επόμενα στάδια:

1. Αναγνώριση του προβλήματος προς επίλυση
2. Αποτύπωση της λειτουργίας του συστήματος
3. Δημιουργία του μοντέλου του συστήματος
4. Έλεγχος και παρακολούθηση του μοντέλου
5. Αξιολόγηση του αποτελέσματος και πιθανή επαναδιατύπωση των προηγούμενων σταδίων.

Σκοπός των ειδικών είναι η μελέτη ενός συστήματος με μαθηματικές μεθόδους, (απαιτείται η πλήρης γνώση του συστήματος και η δυνατότητα αναπαράστασης του με μαθηματικά μοντέλα), η ανάλυση του συστήματος στα πρωταρχικά του στοιχεία και ύστερα, η προσπάθεια σύνθεσης του συστήματος, η αναπαράσταση του με μαθηματικά μοντέλα, η αντιγραφή της συμπεριφοράς του με τη χρήση γνωστών στοιχειωδών δομικών μονάδων. Η προσπάθεια όμως αυτή δεν είναι (τις περισσότερες φορές) εύκολη διότι η πολυπλοκότητα των συστημάτων είναι εξαιρετικά μεγάλη. Με την ανάπτυξη των υπολογιστών, μετακινήθηκε το ενδιαφέρον από τις μαθηματικές μεθόδους, (που προσφέρουν ακρίβεια υπολογισμών), σε λιγότερο ακριβείς μεθοδολογίες μελέτης, ανάλυσης και σύνθεσης συστημάτων, τις λεγόμενες πειραματικές μεθόδους και μεθοδολογίες, οι οποίες εξαρτώνται από τον ορισμό του υπό εξέταση μοντέλου του συστήματος καθώς και των συσχετιζόμενων παραμέτρων του. Το μειονέκτημα όλων αυτών των πειραματικών μεθοδολογιών είναι η μειωμένη ακρίβεια τους.

Οι πειραματικές μέθοδοι χωρίζονται σε δύο (2) μεγάλες κατηγορίες που διαφοροποιούνται ανάλογα με το σκοπό για τον οποίο χρησιμοποιούνται. Αν ο τελικός στόχος μας σε κάθε περίπτωση είναι η υλοποίηση ενός συγκεκριμένου συστήματος τότε μιλάμε για τη διαδικασία της εξομοίωσης, ενώ αν ο απώτερος σκοπός μας είναι η ενδελεχής μελέτη ενός συστήματος τότε αναφερόμαστε στην διαδικασία της προσομοίωσης του. Οι γενικοί ορισμοί των δύο (2) κατηγοριών είναι:

(α) Εξομοίωση, (emulation), θεωρείται μια μέθοδος αναπαραγωγής ενός συστήματος μέσω ενός άλλου συστήματος παρόμοιου με το πρώτο, ενώ

(β) Προσομοίωση, (simulation), είναι μια μέθοδος μελέτης ενός συστήματος και ανάλυσης των χαρακτηριστικών του με τη βοήθεια ενός άλλου συστήματος, το οποίο στις περισσότερες περιπτώσεις είναι ένας ηλεκτρονικός υπολογιστής ή μια συστοιχία υπολογιστών.

Η προσομοίωση είναι λοιπόν μια διαδικασία μίμησης του πραγματικού συστήματος από ένα «κατασκευασμένο» με στόχο την εξαγωγή των χαρακτηριστικών του πραγματικού συστήματος. Χρησιμοποιείται ευρέως σε κάθε επιστημονική περιοχή διότι μέσω αυτής μπορούμε να παρακολουθήσουμε τις συνέπειες των εκτελουμένων πειραμάτων, να προχωρήσουμε στη συλλογή και στον έλεγχο στοιχείων δεδομένων όταν ο απευθείας πειραματισμός στο πραγματικό σύστημα είναι ανέφικτος για πρακτικούς ή και για ηθικούς λόγους.

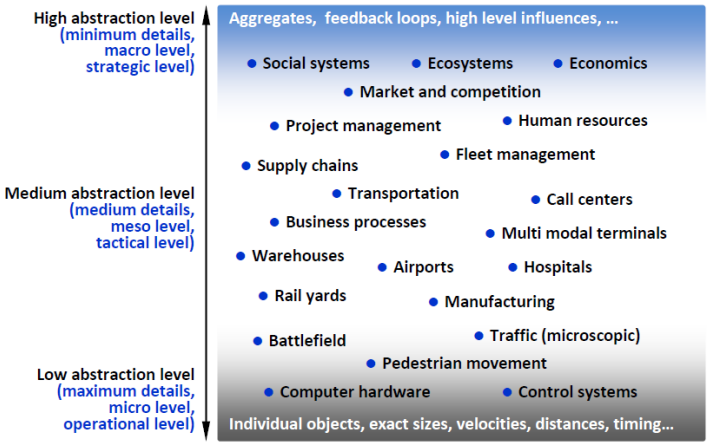
Απώτερος σκοπός της προσομοίωσης είναι η δημιουργία μοντέλων που θα μιμούνται «τέλεια» τα πραγματικά συστήματα λαμβάνοντας υπόψη οποιαδήποτε λεπτομέρεια. Για τη δημιουργία του μοντέλου κατασκευάζονται και εκτελούνται διαφορετικά σενάρια, (για κάθε ένα από αυτά τα «ειδικά συστήματα» εισάγονται διεγέρσεις, μεταβάλλονται οι αρχικές συνθήκες και οι τιμές κάποιων παραμέτρων με σκοπό την ανακάλυψη των αλληλεπιδράσεων και των σημαντικών παραγόντων που συμμετέχουν στις εσωτερικές και εξωτερικές διαδικασίες), τα οποία και αξιολογούνται. Τα αποτελέσματα που λαμβάνουμε από αυτά τα σενάρια μας αποκαλύπτουν τις εσωτερικές ισορροπίες που κρύβουν τα προς εξέταση συστήματα και μας επιτρέπουν να αντιληφθούμε τις ιδιαιτερότητες και τα χαρακτηριστικά της κάθε περίπτωσης, να διορθώσουμε τα λάθη μας και εν τέλει να δημιουργήσουμε το επιθυμητό μοντέλο.

Για να καταφέρουμε να κάνουμε την διαδικασία της προσομοίωσης όσο το δυνατό ευέλικτη και άμεσα υλοποιήσιμη, με άλλα λόγια να γίνει η διαδικασία παραγωγική χρειάζεται να ακολουθήσουμε ορισμένους κανόνες που έχουν σχέση με την επαναχρησιμοποίηση των μοντέλων, την προσαρμοστικότητα των μοντέλων και την ικανότητα συντήρησης, τη λεγόμενη συντηρισιμότητα των μοντέλων. Οι τρείς αυτές αρχές αν και έχουν γενική ισχύ στη σύγχρονη επιστήμη, πρωταγωνιστούν στην επιστήμη της πληροφορικής. Δεν είναι τυχαίο το ότι ένας πολύ ενεργός κλάδος της προσομοίωσης είναι η αντικειμενοστραφής προσομοίωση.

Αντικειμενοστραφής ορίζεται το είδος της προσομοίωσης που για την περιγραφή του συστήματος χρησιμοποιούνται αντικείμενα και παράλληλα ορίζονται οι σχέσεις και οι αλληλεπιδράσεις μεταξύ των αντικειμένων. Για την ολοκλήρωση της περιγραφής ορίζονται οι κλάσεις των αντικειμένων, τα γενικά πρότυπα που ορίζουν τις γενικές ιδιότητες των αντικειμένων. Ως αντικείμενο ορίζεται ένας αφηρημένος τύπος δομής δεδομένων που έχει τη δυνατότητα να διατηρεί τις δικές της ιδιωτικές παραμέτρους, μεταβλητές, διαδικασίες και μεθόδους κρυφές από τα άλλα αντικείμενα.

Το επίπεδο της ανάλυσης, το πόσο ενδελεχώς εξετάζουμε το σύστημα, του κάθε μοντέλου εξαρτάται από το λεγόμενο επίπεδο αφαίρεσης των οντοτήτων του μοντέλου. Ο προγραμματιστής «αποκαλύπτει» τις βασικές οντότητες του συστήματος που πρόκειται να μοντελοποιηθεί και μέσω της ανάλυσης του συστήματος δημιουργεί τις κλάσεις που θα περιγράψουν το μοντέλο του συστήματος. Τέλος ορίζει το πως τα αντικείμενα των κλάσεων θα αλληλεπιδρούν με τα υπόλοιπα αντικείμενα των κλάσεων. Η κληρονομικότητα και ο πολυμορφισμός, χαρακτηριστικές έννοιες της αντικειμενοστραφούς προσομοίωσης, καθορίζουν αφενός τη μεταφορά ιδιοτήτων από κλάση σε κλάση και αφετέρου τη κλωνοποίηση των περιεχομένων μιας κλάσης σε περισσότερα από ένα νέα αντικείμενα, ορισμένα από το χρήστη, που όμως έχουν διαφορετικές ιδιότητες και συμπεριφορές, αντίστοιχα.

Η προσομοίωση μέσω μοντέλων συχνά αναφέρεται και με τον όρο μοντελοποίηση. Καθώς η τεχνολογία εξελίσσεται και αυξάνεται η ισχύς των υπολογιστών η μοντελοποίηση αναπτύσσεται επίσης και εφαρμόζεται σε όλο και μεγαλύτερο φάσμα προβλημάτων και τελευταία εμφανίζεται δυναμικά στη προσπάθεια χειρισμού συστημάτων παροχής υπηρεσιών οποιασδήποτε πολυπλοκότητας.



Εικόνα 1-1: Παρουσίαση ενδεικτικών κατηγοριών εφαρμογών προς μοντελοποίηση σύμφωνα με το επίπεδο αφαίρεσης τους.

Στην Εικόνα 1-1 παρουσιάζονται μερικές εφαρμογές της προσομοίωσης που έχουν ταξινομηθεί με βάση το επίπεδο αφαίρεσης των αντίστοιχων μοντέλων. Τα μοντέλα στο κάτω μέρος του σχήματος είναι μοντέλα όπου τα πραγματικά αντικείμενα έχουν τη μέγιστη λεπτομέρεια και βρίσκονται στο φυσικό επίπεδο. Σε αυτό το επίπεδο ενδιαφερόμαστε για τη φυσική αλληλεπίδραση μεταξύ των αντικειμένων και για τις πραγματικές τιμές των φυσικών μεγεθών, π.χ. ακριβής διάσταση στο χώρο, ταχύτητα, απόσταση και χρόνος. Παραδείγματα αυτού του τύπου προβλημάτων που απαιτούν μοντελοποίηση σε χαμηλό επίπεδο αφαίρεσης θεωρούνται η κυκλοφορία των αυτοκινήτων σε μια διασταύρωση που ελέγχεται από ένα φανάρι, η μοντελοποίηση ενός συστήματος ελέγχου, (π.χ. σύστημα πέδησης), αυτοκινήτου και η αλληλεπίδραση των στρατιωτών σε ένα πεδίο μάχης.

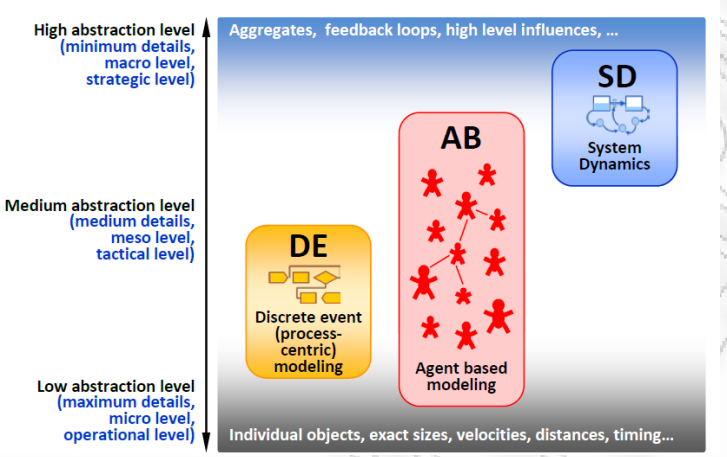
Τα μοντέλα που βρίσκονται στην κορυφή του διαγράμματος έχουν πολύ υψηλό επίπεδο αφαίρεσης. Τα μεμονωμένα αντικείμενα συνήθως αντικαθίστανται από σύνολα αντικειμένων. Αντί να μοντελοποιείται κάθε οντότητα ξεχωριστά, αυτό γίνεται για ένα δεδομένο σύνολο ή για συγκεκριμένα υποσύνολα. Αναλόγως, η αλληλεπίδραση μεταξύ των αντικειμένων κατά τη μοντελοποίηση αυξάνεται σε πολυπλοκότητα, σε διαφορετικό, υψηλότερο, επίπεδο αφαίρεσης. Για παράδειγμα, στις προσομοιώσεις συστημάτων που έχουν να κάνουν με πωλητές και αγοραστές, τα αντικείμενα που αντιπροσωπεύουν τις πωλήσεις αντί να μοντελοποιούνται για κάθε μεμονωμένο καταναλωτή ξεχωριστά, προσομοιώνονται σε κατηγορίες καταναλωτών.

Επίσης, η ενδιάμεση κατάσταση μεταξύ των δύο (2) παραπάνω περιπτώσεων είναι όταν μοντελοποιούμε συστήματα των οποίων το επίπεδο αφαίρεσης βρίσκεται στο ενδιάμεσο μεταξύ χαμηλής και υψηλής αφαίρεσης. Ένα τέτοιο παράδειγμα είναι η υλοποίηση ενός μοντέλου για το τμήμα επειγόντων περιστατικών ενός νοσοκομείου. Εδώ ο φυσικός χώρος είναι σημαντικός διότι μας ενδιαφέρει π.χ. πόσος χρόνος απαιτείται για να διανυθεί η απόσταση από τα δωμάτια της εντατικής στο χώρο του ακτινολογικού εργαστηρίου. Η φυσική αλληλεπίδραση μεταξύ των ανθρώπων μπορεί να μην είναι σημαντικός παράγοντας στη κατάστρωση του μοντέλου γιατί υποθέτουμε ότι δεν θα έχουμε συμφόρηση στο κτίριο. Σε ένα μοντέλο μιας επιχειρηματικής διαδικασίας π.χ. μεταφορές αγαθών μπορεί να μην ενδιαφερόμαστε για το πως γίνεται η αποστολή των αγαθών αλλά μας ενδιαφέρει η αποστολή της παραγγελίας να διαρκεί συγκεκριμένες ημέρες. Αντίστοιχα, στη μοντελοποίηση ενός τηλεφωνικού κέντρου δεν ενδιαφερόμαστε για τη χρονική διάρκεια των λειτουργιών του κέντρου αλλά για τις ίδιες τις λειτουργίες.

Συμπερασματικά, η επιλογή του σωστού επιπέδου αφαίρεσης είναι κρίσιμη για την επιτυχία της συνολικής προσομοίωσης και της επιτυχημένης διαμόρφωσης του ιδανικού μοντέλου. Μετά την επιλογή αυτή, η επιλογή της μεθόδου μοντελοποίησης, ο προγραμματισμός των εργασιών και η υλοποίηση του μοντέλου γίνεται αρκετά απλή. Βέβαια, κατά τη διαδικασία της ανάπτυξης του μοντέλου είναι φυσιολογικό, πολλές φορές ακόμα και επιθυμητό, να

επανεξεταστεί το επίπεδο της αφαίρεσης του μοντέλου. Συνήθως ξεκινάμε με υψηλό επίπεδο αφαίρεσης και προσθέτουμε τα στοιχεία που χρειάζονται.

Στην Εικόνα 1-2 παρουσιάζονται οι τρεις βασικές μέθοδοι μοντελοποίησης συστημάτων που χρησιμοποιούνται και είναι γνωστές από τη βιβλιογραφία. Κάθε μέθοδος, συνήθως, χρησιμοποιείται σε συγκεκριμένο εύρος «τιμών» των επιπέδων αφαίρεσης.



Εικόνα 1-2: Κατηγορίες μοντελοποίησης και επίπεδα αφαίρεσης των μοντέλων.

1.2 Μέθοδος Μοντελοποίησης Διακριτών Γεγονότων (Discrete Event Modeling - DE)

Η μοντελοποίηση Διακριτών Γεγονότων ή η προσομοίωση Διακριτών Γεγονότων, όπως επίσης είναι γνωστή, χρησιμοποιείται σε προβλήματα που έχουν μεσαίο ή χαμηλό επίπεδο αφαίρεσης.

Στη μέθοδο αυτή, το προς εξέταση σύστημα παρουσιάζεται σαν μια διαδικασία, μια σειρά από εργασίες που εκτελούνται έχοντας συγκεκριμένο χρόνο αρχής και χρόνο τέλους για κάθε μια από αυτές. Μεταξύ των δύο (2) αυτών χρόνων, δεν επιτρέπονται αλλαγές, στο μοντέλο του συστήματος, κατά την προσομοίωση. Όμως, μας ενδιαφέρει η στιγμή που συμβαίνει μια συγκεκριμένη αλλαγή στο σύστημα και η οποία είναι σημαντική για εμάς διότι τροποποιείται το μοντέλο που αντιπροσωπεύει το εξεταζόμενο σύστημα. Αν και γενικότερα ο κόσμος μας αλλάζει συνεχώς και συνεπώς θεωρείται χρονικά συνεχής, οι αλλαγές δεν θεωρείται ότι γίνονται σε μια στιγμή αλλά κάθε μια από αυτές χρειάζεται μη μηδενικό χρόνο για να συμβεί και μπορεί να αναλυθεί σε περαιτέρω φάσεις σύμφωνα με το επίπεδο αφαίρεσης του μοντέλου.

1.3 Μέθοδος Δυναμικής Μοντελοποίησης Συστημάτων (System Dynamics - SD)

Στη δεκαετία του 1950 αναπτύχθηκε η θεωρία της Δυναμικής των Συστημάτων (System Dynamics) από τον Jay W. Forrester, η οποία αρχικά εφαρμόστηκε στις επιχειρήσεις και στη συνέχεια στη βιομηχανική παραγωγή.

Είναι χρονική μέθοδος, άρα αφορά δυναμικά συστήματα και τα κύρια χαρακτηριστικά της είναι η εισαγωγή των εννοιών του σημείου συσσώρευσης (stock) που αναπαριστά μια διαδικασία του πραγματικού κόσμου στην οποία έρχονται και φεύγουν πληροφορίες και της ροής (flow) η οποία αντιπροσωπεύει τη μεταφερόμενη πληροφορία. Αντίθετα από την προηγούμενη μέθοδο, η Δυναμική μοντελοποίηση ή προσομοίωση Συστημάτων είναι μια μέθοδος με την οποία για τη δημιουργία του μοντέλου χρησιμοποιούνται διαφορικές εξισώσεις και μέθοδοι επίλυσης χωρικών εξισώσεων κατάστασης γραμμικών συστημάτων συνεχούς και διακριτού χρόνου. Η μέθοδος αυτή επιλέγεται να μοντελοποιήσει συστήματα χρονικά συνεχή όπου το ζητούμενο είναι η κατανόηση του ίδιου του μοντέλου ως ενιαίο σύνολο, ως δομή αλλά και κάθε παραμέτρου που επηρεάζει το μοντέλο κατά περίπτωση. Η συμπεριφορά των προς

εξέταση συστημάτων γίνεται μέσα από τη δημιουργία πολύπλοκων δυναμικών μοντέλων, (αναλυτικών μοντέλων συνεχούς χρόνου), που αποτελούνται κυρίως από θετικούς και αρνητικούς βρόχους ανάδρασης και από δομές καθυστέρησης που αλληλεπιδρούν με τη βοήθεια των κατά περίπτωση μοναδικών παραμέτρων του συστήματος. Η μέθοδος αυτή λειτουργεί σε εφαρμογές υψηλού επιπέδου αφαίρεσης και χρησιμοποιείται κυρίως στη μοντελοποίηση συστημάτων στρατηγικής, σε συστήματα κοινωνικής δυναμικής και σε συστήματα με διαπιστωμένη ενδογενή αβεβαιότητα.

1.4 Μέθοδος Μοντελοποίησης με τη χρήση Πρακτόρων Λογισμικού (Agent Based Modeling - AB)

Η ιδέα του πράκτορα λογισμικού αναπτύχθηκε ως ιδέα στα τέλη της δεκαετίας του 1940 και συνδέεται με τη μηχανή του Von Neumann αλλά δεν είχε γίνει ευρέως γνωστή μέχρι τη δεκαετία του 1990 λόγω του ότι δεν είχαμε τις απαραίτητες τεχνολογικές λύσεις στην τεχνολογία λογισμικού. Η μηχανή του von Neumann είναι μια θεωρητική μηχανή που μπορεί να αναπαράγει τον εαυτό της εαν ακολουθήσει ακριβείς και λεπτομερείς οδηγίες. Η πρώτη μηχανή αυτού του τύπου ολοκληρώθηκε από τον ίδιο τον von Neumann και αργότερα ονομάστηκε κυτταρικό αυτόματο, (cellular automaton).

Ο ορισμός του Πράκτορα δεν έχει ακριβή και μοναδικό ορισμό. Στην Πληροφορική και στην Επιστήμη των Υπολογιστών ο Πράκτορας έχει την ικανότητα να προβαίνει σε ανεξάρτητες αποφάσεις. Σε πρακτικό επίπεδο, ένας Πράκτορας έχει συγκεκριμένα χαρακτηριστικά και κανόνες να κυβερνούν τη συμπεριφορά και την ικανότητα του να αποφασίζει. Μπορεί να μαθαίνει και να προσαρμόζει τη συμπεριφορά του με βάσει τις εμπειρίες του μέσα στο χρόνο. Υπάρχει συγκεκριμένο περιβάλλον στο οποίο βρίσκεται και αλληλεπιδρά με τους άλλους Πράκτορες.

Η μοντελοποίηση με τη χρήση Πρακτόρων Λογισμικού (εμφανίζεται επίσης ως σύστημα πολλαπλών πρακτόρων ή άλλες φορές ως πολυπρακτορικά συστήματα προσομοίωσης) είναι μια κατηγορία από υπολογιστικά μοντέλα για την προσομοίωση των ενεργειών και των αλληλεπιδράσεων των αυτόνομων παραγόντων, (που μπορεί να αναφέρονται τόσο σε ατομικές όσο και σε συλλογικές οντότητες όπως οργανώσεις ή ομάδες), με σκοπό την αξιολόγηση των επιπτώσεών τους στο συνολικό προς εξέταση σύστημα. Οι πράκτορες κατά τη μοντελοποίηση μπορεί να είναι συγκεκριμένα φυσικά αντικείμενα ή πολύ αφηρημένα, όπως ανταγωνιστικές εταιρείες ή κυβερνήσεις. Για το λόγο αυτό, το επίπεδο αφαίρεσης των μοντέλων ποικίλλει. Τα μοντέλα που υλοποιούνται, προσομοιώνουν τις ταυτόχρονες κινήσεις, τις προσπάθειες των πρακτόρων να καταφέρουν το σκοπό τους και τις αλληλεπιδράσεις μεταξύ των πρακτόρων σε μια προσπάθεια να προβλεφθούν οι καταστάσεις και τα αποτελέσματα των σύνθετων φαινομένων που μοντελοποιούνται μέσα σε ένα ορισμένο περιβάλλον δράσης των πρακτόρων. Για την κατασκευή των μοντέλων συνδυάζονται στοιχεία της θεωρίας των παιγνίων, των πολύπλοκων συστημάτων, της υπολογιστικής κοινωνιολογίας, της κοινωνικής αλληλεπίδρασης και του προγραμματισμού.

1.5 Υβριδική μοντελοποίηση (Hybrid ή Combined Modeling)

Η υβριδική μοντελοποίηση δεν είναι τίποτε άλλο παρά ο συνδυασμός των παραπάνω μεθόδων μοντελοποίησης και αντιπροσωπεύει το άλμα που έγινε τις τελευταίες δεκαετίες στην επιστημονική εξέλιξη της προσομοίωσης. Με την πρόοδο των υπολογιστών, σε υλικό και λογισμικό, έγινε δυνατή η προσομοίωση πολύπλοκων συστημάτων που έχουν συνιστώσες και συνεχούς αλλά και διακριτού χρόνου. Με άλλα λόγια, η συμπεριφορά αυτών των συστημάτων μπορεί να αναπαρασταθεί με αναλυτικό τρόπο, με αλγεβρικές και διαφορικές εξισώσεις, όσο αυτά βρίσκονται σε σταθερή κατάσταση αλλά η αλλαγή της κατάσταση τους εξαρτάται από την εμφάνιση ενός γεγονότος, μιας διακριτής κατάστασης που προκύπτει λόγω της ύπαρξης μιας μηχανής καταστάσεων στο παρασκήνιο.

ΚΕΦΑΛΑΙΟ 2

Το Λογισμικό AnyLogic της Εταιρείας XJTek

2.1 Η Ιστορία του Λογισμικού AnyLogic

Στις αρχές της δεκαετίας του '90 υπήρχε μεγάλο επιστημονικό ενδιαφέρον για την μαθηματική προσέγγιση του προβλήματος της μοντελοποίησης και της προσομοίωσης των παράλληλων διαδικασιών, (Parallel Processes), μέσω λογισμικού. Αυτή η προσέγγιση θα μπορούσε να χρησιμοποιηθεί για τον έλεγχο της ακρίβειας και της ορθότητας των παράλληλων και των κατανεμημένων προγραμμάτων λογισμικού εκείνης της εποχής. Μια ερευνητική ομάδα, (Distributed Computer Network - DCN), στο Τεχνικό Πανεπιστήμιο της Αγίας Πετρούπολης, (Saint Petersburg Technical University), που είχε ως ερευνητικό αντικείμενο τα κατανεμημένα δίκτυα υπολογιστών ανέπτυξε ένα λογισμικό. Το λογισμικό είχε την ονομασία COVERS (Concurrent Verification and Simulation) και είχε δημιουργηθεί για την ανάλυση της ορθότητας και της ακρίβειας άλλων προγραμμάτων λογισμικού. Το εργαλείο αυτό πέρα από την παράλληλη επαλήθευση και την προσομοίωση, επέτρεπε κατά τη μοντελοποίηση τον γραφικό συμβολισμό της δομής και της συμπεριφοράς του συστήματος.

Στα μέσα της δεκαετίας του ΄90, το εργαστήριο DCN λόγω της επιτυχίας του με το λογισμικό COVERS αποφάσισε να προχωρήσει στην δημιουργία μιας εταιρείας ανάπτυξης λογισμικού προσομοιώσεων, της XJ Technologies. Η εταιρεία XJ Technologies πρωτοπαρουσίασε το λογισμικό που δημιούργησε, ως ανεξάρτητο εργαλείο αντικειμενοστραφούς προσομοίωσης (Object Oriented Simulation), στο Winter Simulation Conference το έτος 2000. Κύρια χαρακτηριστικά του λογισμικού, που έκαναν εντύπωση, ήταν η αντικειμενοστραφής προσέγγιση, η χρήση του προτύπου της UML, η χρήση της γλώσσας προγραμματισμού Java και το γραφικό περιβάλλον του.

Το λογισμικό ονομάστηκε AnyLogic, (θα ονομάζεται Λογισμικό στο εξής, εκτός αν οριστεί στο κείμενο κάτι διαφορετικό), διότι είναι το μοναδικό, (μέχρι σήμερα, εξ' όσων γνωρίζουμε), Λογισμικό που μπορεί όχι μόνο να δημιουργεί και να διαχειρίζεται μοντέλα που έχουν υλοποιηθεί βάσει των τριών (3) μεθοδολογιών προσομοίωσης που αναφέρθηκαν στο Κεφάλαιο 1 αλλά κάνοντας χρήση της Υβριδικής Μοντελοποίησης να διαχειρίζεται και να κατασκευάζει υβριδικά μοντέλα. Χαρακτηριστικά γνωρίσματα του Λογισμικού είναι: (α) το πλήρως γραφικό περιβάλλον ανάπτυξης και παρουσίασης, (β) ότι χρησιμοποιηεί γνώριμα περιβάλλοντα ανάπτυξης λογισμικού, οι πλατφόρμες της Java και του Eclipse (σε λειτουργικά περιβάλλοντα GNU-Linux, Mac, Windows) και (γ) επιτρέπει τη χρήση της τεχνικής «Σύρω και Αφήνω», (Drug and Drop), για τη δημιουργία των μοντέλων σε όλες τις μεθοδολογίες προσομοίωσης.

Η πρώτη έκδοση του Λογισμικού ήταν η 4.0, επειδή η αρίθμηση συνεχίστηκε από την αρίθμηση του COVERS 3.0. Ένα μεγάλο βήμα έγινε το 2003, όταν βγήκε η έκδοση του Λογισμικού 5.0 με την οποία η εταιρεία στράφηκε στην επιχειρησιακή προσομοίωση και παρουσίασε παραδείγματα μοντέλων για τις ακόλουθες περιοχές επαγγελματικού και ερευνητικού ενδιαφέροντος, όπως για την αγορά και τον επιχειρηματικό ανταγωνισμό, την υγειονομική περίθαλψη, τον κατασκευαστικός τομέα, τις αλυσίδες εφοδιασμού, τη διοικητική μέριμνα (Logistics), το λιανικό εμπόριο, τις επιχειρησιακές διαδικασίες, την άμυνα, τα δίκτυα υπολογιστών και τηλεπικοινωνιών, τη διαχείριση της κυκλοφορίας κ.λ.π.

Σήμερα, η εταιρεία XJ Technologies Company Ltd, (http://www.xjtek.com/), είναι μία από τις κορυφαίες εταιρείες στον τομέα των εργαλείων και των επιχειρηματικών εφαρμογών της προσομοίωσης στον κόσμο. Εκτός από το συγκεκριμένο Λογισμικό, η εταιρεία παρέχει επίσης συμβουλευτικές υπηρεσίες για κάθετι που έχει σχέση με την μοντελοποίηση συστημάτων σε επαγγελματικό επίπεδο. Το Λογισμικό της εταιρείας βρίσκεται ήδη, (Δεκέμβριος 2011), στην

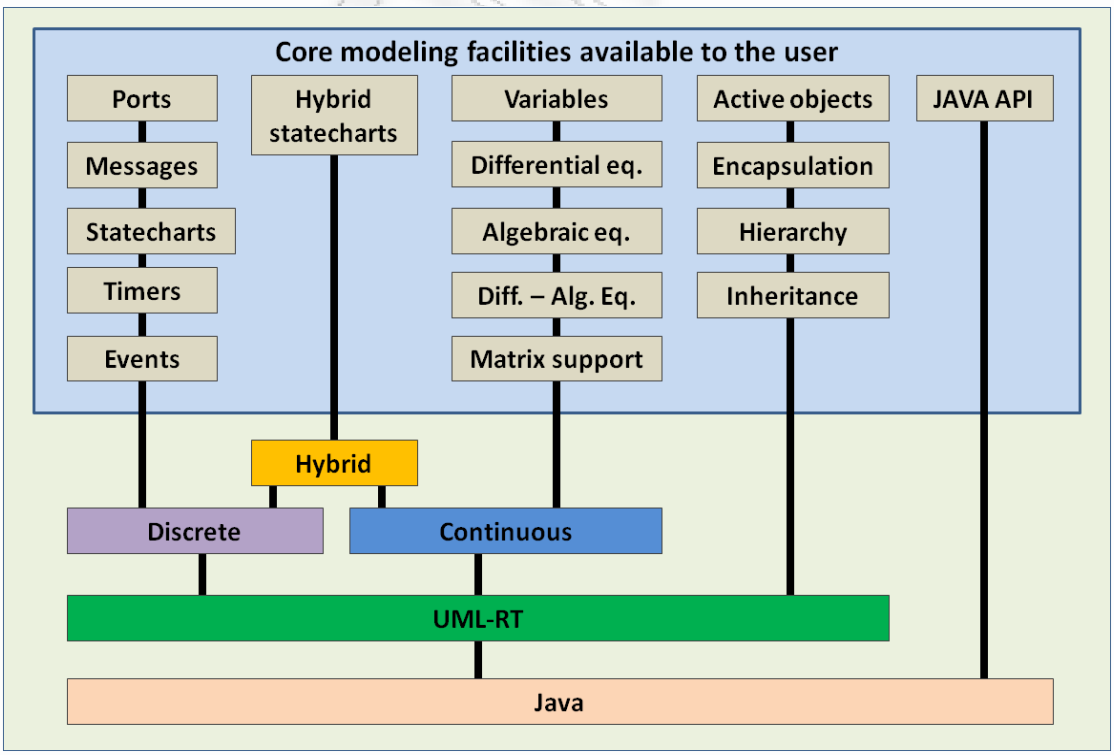
έκδοση 6.7 και στο ιστότοπο http://www.xjtek.com/anylogic/overview/editions/ παρουσιάζονται οι διαφορετικές του εκδόσεις.

2.2 AnyLogic και Java

Ο αντικειμενοστραφής προγραμματισμός (Object Oriented Programming, OOP) είναι στις μέρες μας εξαιρετικά δημοφιλής και έχει ωριμάσει ως τεχνολογία ανάπτυξης λογισμικού. Βασικοί λόγοι που οδήγησαν στη ραγδαία ανάπτυξη του είναι: (α) η μεθοδολογία του μοιάζει με τον τρόπο σκέψης του ανθρώπου, (β) μπορεί να χρησιμοποιηθεί στην επίλυση προβλημάτων διαφορετικής δυσκολίας και κλίμακας και κυρίως (γ) είναι κατάλληλος για την οργάνωση μεγάλων έργων που για την ανάπτυξη και την ολοκλήρωση τους θα χωριστούν και θα αναπτυχθούν σε τμήματα.

Η δυνατότητα της αφαίρεσης της πληροφορίας και της σκόπιμης απόκρυψης των επιμέρους χαρακτηριστικών προσφέρει στους προγραμματιστές τη δυνατότητα του τεμαχισμού και έπειτα της συνένωσης των τμημάτων κώδικα και αργότερα τη δυνατότητα συντήρησης του λογισμικού. Τα αντικειμενοστραφή προγράμματα (και γενικότερα τα προϊόντα λογισμικού) είναι οργανωμένα γύρω από τα δεδομένα (από αυτό που πρόκειται να επηρεαστεί) και όχι γύρω από τον κώδικα (αυτό που επηρεάζει, αυτό που δίνει εντολές στα δεδομένα). Ο προγραμματιστής είναι αυτός καθορίζει τις διαδικασίες που θα επιδράσουν στα δεδομένα.

Πίσω από τις αρχές του αντικειμενοστραφούς προγραμματισμού βρίσκονται τρία σημαντικότατα χαρακτηριστικά: η ενθυλάκωση (encapsulation), ο πολυμορφισμός (polymorphism) και η κληρονομικότητα (inheritance). Η ενθυλάκωση είναι ο μηχανισμός που συνδέει τον κώδικα με τα δεδομένα σε ενιαίες «κατασκευές» ώστε και τα δύο να είναι προστατευμένα από εξωτερικές επιδράσεις. Αποτέλεσμα αυτής της σύνδεσης είναι η δημιουργία των αντικειμένων, των αυτόνομων αφηρημένων οντοτήτων, που μπορεί να έχουν δημόσια (επιτρέπεται η προσπέλαση τους από άλλα τμήματα του κώδικα) και ιδιωτικά (επιτρέπεται η προσπέλαση τους μόνο από το αντικείμενο) χαρακτηριστικά. Το αντικείμενο είναι στιγμιότυπο, (instance), μιας κλάσης, (class). Η κλάση είναι αυτή που προσδιορίζει, (με αφηρημένο τρόπο), τη μορφή του αντικειμένου. Η κλάση καθορίζει τις κοινές ιδιότητες και συμπεριφορές του.



Εικόνα 2-1: Η δομή των πλαισίων στα οποία βασίζεται το Λογισμικό.

Η εικόνα 2-1 παρουσιάζει τη διασύνδεση των πλαισίων πάνω στα οποία είναι δομημένο το Λογισμικό, ξεκινώντας από το χαμηλό επίπεδο και φτάνοντας στο υψηλότερο. Στη βάση βρίσκεται το περιβάλλον ανάπτυξης της γλώσσας προγραμματισμού Java το οποίο συνδέεται με τη ενοποιημένη γραφική γλώσσα μοντελοποίησης για περιβάλλοντα πραγματικού χρόνου UML-RT. Τα επόμενα επίπεδα αφορούν στις μεθόδους μοντελοποίησης και σε αυτό το σημείο τοποθετείται το πλαίσιο της υβριδικής προσομοίωσης, το κύριο πλεονέκτημα του Λογισμικού. Στο κεντρικό πλαίσιο παρουσιάζονται συνοπτικά οι κεντρικές διευκολύνσεις που προσφέρονται στο χρήστη για να δημιουργήσει μοντέλα.

Η εισαγωγή του Λογισμικού στην ανάπτυξη μοντέλων μας οδηγεί σε μετρήσιμα συμπεράσματα για την αποτελεσματικότητα του, κυρίως όμως ξεχωρίζει ότι έχουμε τη δυνατότητα ανάπτυξης μοντέλων με τη χρήση περισσότερων της μιας μεθοδολογίας με το ίδιο εργαλείο που αυτόματα οδηγεί στη μείωση του χρόνου και του κόστους ανάπτυξης. Το γραφικό περιβάλλον ανάπτυξης είναι πλήρως δομημένο και ακολουθεί τα πρότυπα της αντικειμενοστραφούς μεθοδολογίας ενώ υπάρχουν έτοιμες βιβλιοθήκες αντικειμένων, στοιχείων προσομοίωσης τα οποία είναι εύκολο να χρησιμοποιηθούν, να τροποποιηθούν και να επαναχρησιμοποιηθούν. Δίνεται η δυνατότητα δημιουργίας διαδραστικών μοντέλων και συνθηκών προσομοίωσης με το χρήστη, σε εξελιγμένα περιβάλλοντα προσομοίωσης.

Επίσης, δίνεται η δυνατότητα εξαγωγής των υλοποιημένων μοντέλων με τη μορφή Java Applets, οπότε έχουμε εφαρμογές Java ανεξάρτητες του περιβάλλοντος ανάπτυξης χωρίς να χρειάζονται άδειες χρήσης λογισμικού.

2.3 Εισαγωγή στο Λογισμικό

Στη συνέχεια παρουσιάζονται μερικά στοιχεία από το περιβάλλον του Λογισμικού, χωρίς βέβαια να καλύψουμε πλήρως και σε βάθος το θέμα. Ο αναγνώστης θα βρει τις περαιτέρω πληροφορίες που αναζητά στο κεφάλαιο της βιβλιογραφίας. Ένας πακτωλός γνώσεων για το Λογισμικό βρίσκεται στην ιστοσελίδα της εταιρείας που δημιούργησε, αναπτύσσει και υποστηρίζει το Λογισμικό.

Μετά την εγκατάσταση του Λογισμικού, ο χρήστης στην πρώτη επαφή με το περιβάλλον του Λογισμικού αντικρίζει την κεντρική σελίδα καλωσορίσματος όπου παρουσιάζονται τα υλοποιημένα μοντέλα που συνοδεύουν το Λογισμικό κατά κατηγορία επιστημονικού πεδίου και κατά τύπο μεθόδου μοντελοποίησης. Κάτω ακριβώς από το μήνυμα καλωσορίσματος βρίσκονται ενεργά εικονίδια που μέσω Υπερσυνδέσμων επιτελούν συγκεκριμένες λειτουργίες, όπως ενημέρωση για την έκδοση, τις αλλαγές και τις νέες λειτουργίες, πρόγραμμα μετάπτωσης δεδομένων στη νέα έκδοση για τους χρήστες παλαιότερης έκδοσης, μεταφορά στην ιστοσελίδα της εταιρείας, μεταφορά στο περιβάλλον εργασίας και ανάπτυξης του Λογισμικού.

Τα υλοποιημένα μοντέλα που συνοδεύουν το Λογισμικό προσφέρουν πληρότητα στο προσφερόμενο προϊόν όμως εκτός αυτού έχουν και εκπαιδευτική αξία. Εξερευνώντας τα παραδείγματα, ένας υποψήφιος χρήστης του Λογισμικού μπορεί να γνωρίσει άμεσα τη φιλοσοφία του Λογισμικού, να είναι σε θέση να τροποποιήσει ο ίδιος τα υφιστάμενα μοντέλα με την τεχνική του «δοκιμάζω και μαθαίνω», να δημιουργήσει ο ίδιος νέα μοντέλα, να αντλήσει γνώσεις για τον τρόπο υλοποίησης των μοντέλων σε κάθε διαφορετική διαδικασία προσομοίωσης και να αποκτήσει εμπειρία συνολικά. Η εκπαιδευτική αξία των παραδειγμάτων είναι σίγουρα σημαντική και η προσεκτική ανάλυση των υλοποιημένων μοντέλων θα δώσει στον επιμελή χρήστη την ώθηση προς την κατανόηση της έννοιας και της φιλοσοφίας που βρίσκεται πίσω από την προσομοίωση.

Εικόνα 2-2: Τα πολλά έτοιμα παραδείγματα μοντέλων που συνοδεύουν το Λογισμικό.

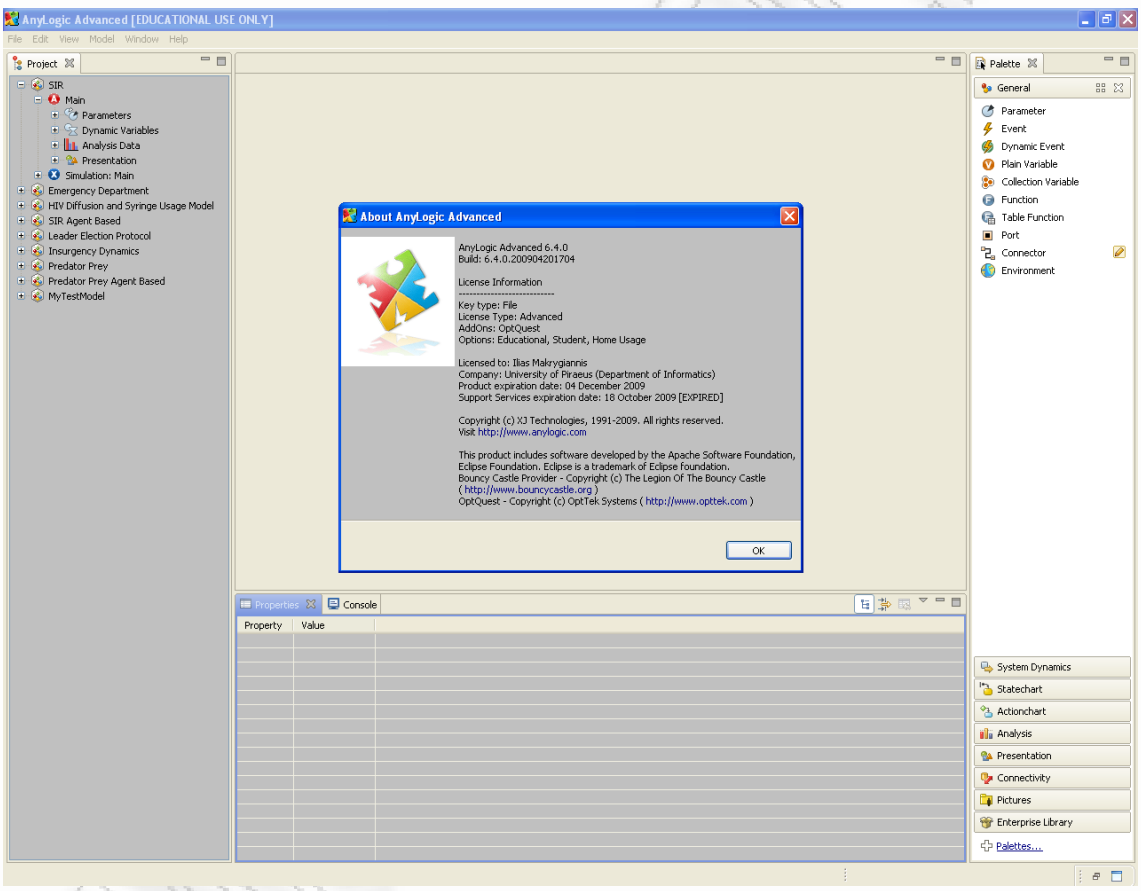
Πατώντας με το ποντίκι ο χρήστης στο «Χ», όπως φαίνεται στη διπλανή εικόνα, μεταφερόμαστε στην κεντρική σελίδα του περιβάλλοντος εργασίας και ανάπτυξης του Λογισμικού, εικόνα 2-3.

Η κεντρική σελίδα του περιβάλλοντος εργασίας και ανάπτυξης του Λογισμικού μπορεί να χωριστεί σε τέσσερα ευδιάκριτα τμήματα, (τα δύο (2) ακραία κάθετα τμήματα και ενδιάμεσα αυτών των δύο, τα δύο (2) οριζόντια τμήματα), που όμως δεν είναι λειτουργικά ανεξάρτητα μεταξύ τους καθώς το ένα συμπληρώνει το άλλο στην ανάπτυξη των μοντέλων. Τα τμήματα αυτά στη «γλώσσα» του Λογισμικού ονομάζονται Όψεις (Views), κάθε μια έχει το ρόλο της και η χρήση τους ξεκινά από την απλή πλοήγηση στη δομή των μοντέλων και φτάνει στην καταγραφή λαθών κατά τη διαδικασία εκτέλεσης των ολοκληρωμένων πια μοντέλων. Έχουν κρυμμένα μενού από όπου μπορούμε να έχουμε πρόσβαση σε πληροφορίες, σε στοιχεία και λειτουργίες χρησιμοποιώντας το δεξί κουμπί του ποντικιού. Εξ' ορισμού αυτή είναι η απεικόνιση, (η οποία αποτελείται από τις όψεις Palette, Project, Properties κλπ που χρησιμοποιούνται για την κατασκευή των μοντέλων μέσω της δημιουργίας των κλάσεων και των ενεργών αντικειμένων), που εμφανίζεται στην επιφάνεια εργασίας και ονομάζεται Model από το Λογισμικό. Υπάρχουν δύο (2) επιλογές, η Model και η Debug, όπου μπορούμε όμως να επιλέξουμε μόνο τη μία κάθε φορά. Η κάθε μια από αυτές χρησιμοποιείται για την εκτέλεση συγκεκριμένου τύπου εργασιών και επιτρέπει την παροχή στοχευμένων λειτουργιών. Ορίζονται σύνολα ενεργειών που είναι

ορατά στο χρήστη και μπορούν να τροποποιηθούν κατά περίπτωση, να αποθηκευτούν και να επαναχρησιμοποιηθούν σε μια άλλη χρονική στιγμή. Η Debug χρησιμοποιείται για την αποσφαλμάτωση του κώδικα (των μοντέλων) και περιλαμβάνει τις όψεις Debug, Breakpoints, Variables και Expressions.

Πριν ξεκινήσουμε την περιγραφή του Λογισμικού, να σημειώσουμε σε αυτό το σημείο πως χρησιμοποιήσαμε την έκδοση 6.4.0 Advance του Λογισμικού, με εκπαιδευτική άδεια χρήσης καταχωρημένη στο Τμήμα Πληροφορικής του Πανεπιστημίου Πειραιώς.

Το Λογισμικό χρησιμοποιεί κώδικα που έχει αναπτυχθεί από την Eclipse, (μια κοινότητα ανοικτού λογισμικού που έχει επικεντρωθεί στη δημιουργία μιας ανοικτής και επεκτάσιμης πλατφόρμας ανάπτυξης λογισμικού και της διαχείρισης αυτού του λογισμικού σε όλο τον κύκλο της ζωής του λογισμικού), για την ανάπτυξη των μοντέλων προσομοίωσης. Επίσης, χρησιμοποιείται ο βελτιστοποιητής, της γλώσσας προγραμματισμού Java της εταιρείας OptTek INC., OptQuest, ο οποίος έχει σχεδιαστεί ώστε να είναι συμβατός και να συνεργάζεται με τα μοντέλα προσομοιώσεων που αναπτύσσονται από το Λογισμικό.



Εικόνα 2-3:Η πρώτη εικόνα του περιβάλλοντος εργασίας του Λογισμικού.

Όπως αναφέραμε παραπάνω και εμφανίζεται και στη εικόνα 2-3, στα αριστερά βρίσκεται η Όψη των Έργων, (Project View), η οποία μας εμφανίζει σε δενδρική δομή όλα τα μοντέλα που έχουμε τοποθετήσει στην επιφάνεια, στο χώρο εργασίας και τα οποία μπορούμε να προσπελάσουμε σε πρώτο χρόνο. Μόνο ένα από αυτά τα μοντέλα μπορεί να είναι ενεργό, (στην έκδοση που χρησιμοποιήσαμε), ανοικτό στην επιφάνεια και να το επεξεργαζόμαστε. Η δενδρική δομή των μοντέλων προφανώς προσφέρει τη δυνατότητα της εύκολης πρόσβασης-πλοήγησης στα στοιχεία που τα αποτελούν αφού η ιεραρχική οργάνωση των μοντέλων είναι ευδιάκριτη. Στην οργάνωση αυτή το ίδιο το μοντέλο βρίσκεται στο υψηλότερο επίπεδο της ιεραρχίας ενώ τα ενεργά αντικείμενα, (active objects) και οι κλάσεις της java, (java classes), είναι αυτές που

ακολουθούν. Τα στοιχεία που ορίζουν τη δομή των ενεργών αντικειμένων και των κλάσεων οργανώνονται, σε ανεξάρτητους κλάδους, σε κατώτερα ιεραρχικά επίπεδα.

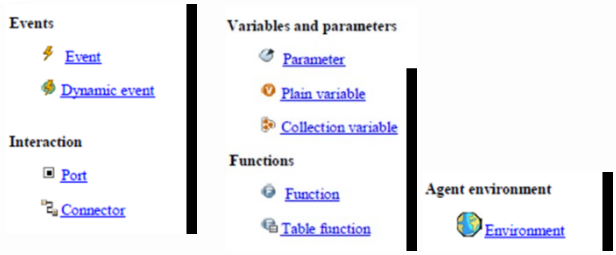
Δεξιά στην επιφάνεια εργασίας βρίσκεται η όψη της παλέτας, (Palette View), ένα παράθυρο δομημένο με τέτοιο τρόπο που βοηθά τον χρήστη στην επιλογή των κατάλληλων δομικών στοιχείων κατά τη διαδικασία της δημιουργίας των μοντέλων. Αυτή η παλέτα έχει το χαρακτηριστικό να απλουστεύει τη διαδικασία της σχεδίασης των προσομοιώσεων διότι αποτελείται από σύνολα έτοιμων δομικών στοιχείων σε μορφή εικονιδίων, (stencils), που χρησιμοποιούνται επανηλημένα για το σχεδιασμό των μοντέλων. Το Λογισμικό προσφέρει αυτά τα έτοιμα εργαλεία, συγκεντρωμένα και κατηγοριοποιημένα σε βιβλιοθήκες, ανά είδος μοντελοποίησης, αποκλειστικά διαμορφωμένα για αυτή τη χρήση.

Στο κέντρο, χαμηλά βρίσκεται η όψη των ιδιοτήτων, (Properties View), η οποία εμφανίζει τις ιδιότητες ενός επιλεγμένου από το χρήστη δομικού στοιχείου που χρησιμοποιείται στην κατασκευή ενός μοντέλου. Η επιλογή γίνεται με τη χρήση του ποντικιού, (πίεση του δεξιού πλήκτρου του ποντικιού πάνω στο στοιχείο που μας ενδιαφέρει). Οι ιδιότητες του δομικού στοιχείου, που ανήκει στο προς εξέταση μοντέλο, μπορούν να τροποποιηθούν άμεσα με την αλλαγή τιμών π.χ. στα πεδία κειμένου, στα κουμπιά επιλογής, στα κουμπιά τσεκαρίσματος κλπ.

Τέλος, ο χώρος πάνω ακριβώς από το παράθυρο της όψης των ιδιοτήτων, (στην εικόνα 2-3 δεν εμφανίζεται ευκρινώς), είναι το ενεργό παράθυρο της σχεδίασης των μοντέλων προσομοιώσεων.

Το Λογισμικό έχει τη δυνατότητα να ανιχνεύει αυτόματα κάποια προβλήματα ή λάθη που ανακύπτουν κατά τη διάρκεια της ανάπτυξης και της δημιουργίας του μοντέλου. Υποστηρίζεται ο έλεγχος συντακτικών λαθών στους τύπους και στις παραμέτρους του κώδικα σε γλώσσα προγραμματισμού Java αλλά και στην κατασκευή των διαγραμμάτων, (θα αναφερθούμε παρακάτω σε αυτά). Τα λάθη εμφανίζονται στην όψη των προβλημάτων, (Problems View). και παράλληλα παρουσιάζεται η θέση του λάθους στον συνολικό κώδικα του μοντέλου και συνοδεύεται από την περιγραφή του λάθους. Η όψη της αναζήτησης, (Search View), είναι αυτή που χρησιμεύει στην αναζήτηση λημάτων σχετικών με τις λειτουργίες του Λογισμικού ενώ η όψη της κονσόλας, (Console View), είναι αυτή που σε μορφή στοίβας παρουσιάζει πληροφορίες για τις τιμές των μεταβλητών εισόδου, των μεταβλητών εξόδου και των λαθών με τη δυνατότητα εντοπισμού των θέσεων στον κώδικα μέσω Υπερσυνδέσμων. Η όψη της βοήθειας, (Help View), είναι αυτή στην οποία γίνεται η διαχείριση της ενσωματωμένης βοήθειας του Λογισμικού που παραθέτει στο χρήστη κάθε είδους πληροφορίες σχετικά με τη δομή και τη λειτουργία του Λογισμικού. Η όψη του εντοπισμού σφαλμάτων, (Debug View), βοηθά στην εμφάνιση των νημάτων (threads) είτε κατά τη μετάφραση του προγράμματος σε γλώσσα προγραμματισμού Java με την οποία υλοποιείται το μοντέλο, είτε κατά την εκτέλεση του προγράμματος. Κάθε νήμα του προγράμματος εμφανίζεται – παρουσιάζεται ως ένας κόμβος μιας δενδρικής δομής. Στη δενδρική δομή κάθε διαδικασία που εκτελείται, εμφανίζεται. Η όψη παράθεσης των σημείων διακοπής, (Breakpoints View), μας παρουσιάζει τα σημεία διακοπής που έχουν ορισθεί κατά την ανάπτυξη ενός μοντέλου στον κώδικα ανάπτυξης του. Η όψη των μαθηματικών παραστάσεων, (Expressions View) είναι αυτή στην οποία εμφανίζονται οι τιμές των παραμέτρων, των σταθερών κατά την εκτέλεση της εφαρμογής. Η όψη παράθεσης των μεταβλητών, (Variables View), εκτελεί την ίδια εργασία για την περίπτωση των μεταβλητών.

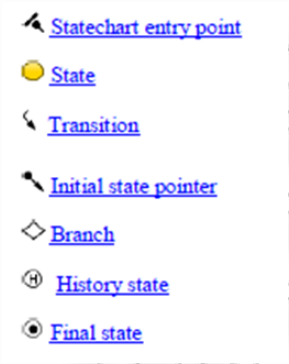
Όπως αναφέρθηκε προηγουμένως, στην όψη της παλέτας είναι συγκεντρωμένα, κατά κατηγορία, τα σύνολα των έτοιμων δομικών στοιχείων που εμφανίζονται σε μορφή εικονιδίων που χρησιμοποιούνται για το σχεδιασμό των μοντέλων.



Εικόνα 2-4: Τα αντικείμενα της κατηγορίας General.

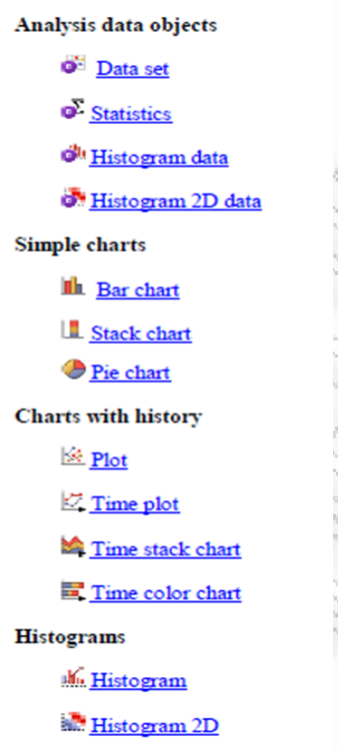
Στη γενική κατηγορία, General, περιέχονται όλα τα απαραίτητα στοιχεία για την κατασκευή ενός μοντέλου. Η δομή, τα χαρακτηριστικά και η δυναμική του μοντέλου περιγράφονται από τα ενεργά αντικείμενα της κατηγορίας αυτής.

Εικόνα 2-5(α): Τα αντικείμενα της κατηγορίας System Dynamics.

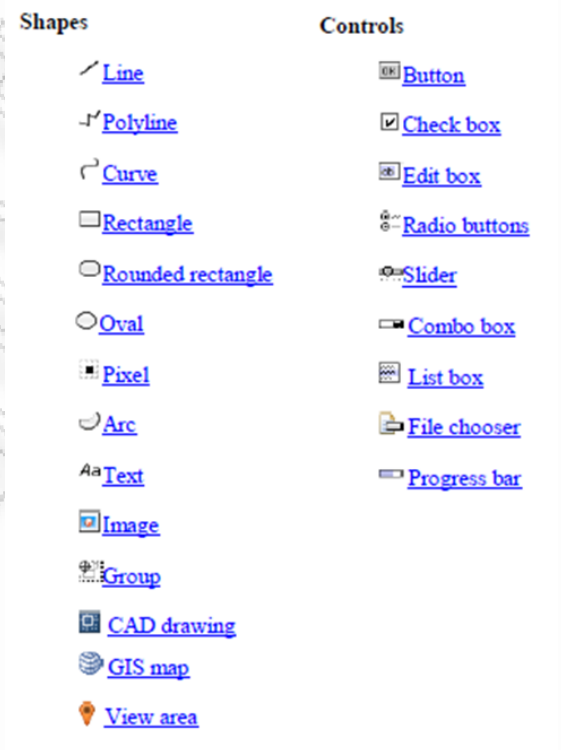


Εικόνα 2-5(β): Τα αντικείμενα της κατηγορίας Statechart.

Εικόνα 2-5(γ): Τα αντικείμενα της κατηγορίας Actionchart.



Εικόνα 2-5(δ): Τα αντικείμενα της κατηγορίας Analysis.



Εικόνα 2-5(ε): Τα αντικείμενα της κατηγορίας Presentation.

Στην κατηγορία System Dynamics περιέχονται τα στοιχεία που χρησιμοποιούνται για τη δημιουργία των μοντέλων της κατηγορίας της Δυναμικής Συστημάτων. Στην κατηγορία Statechart βρίσκονται τα στοιχεία για τη δημιουργία των διαγραμμάτων καταστάσεων ενώ η κατηγορία Actionchart περιέχει δομημένα διαγράμματα block που επιτρέπουν τον ορισμό

αλγορίθμων με τη χρήση γραφικών. Η κατηγορία της Ανάλυσης, Analysis, περιέχει τα στοιχεία για τη συλλογή, την προβολή και την ανάλυση των δεδομένων εξόδου. Η κατηγορία της Παρουσίασης, Presentation, περιέχει τα σχήματα για το σχηματισμό των παρουσιάσεων και των ελέγχων, εικόνες 2-5(α) έως 2-5(ε).

2.3.1 Η κατηγορία των δομικών στοιχείων της Ανάλυσης

Το Λογισμικό επιτρέπει στον χρήστη να απεικονίσει, με τη χρησιμοποίηση διαφόρων διαγραμμάτων και ιστογραμμάτων, τα δεδομένα εξόδου του εκάστοτε μοντέλου κατά την διάρκεια της εκτέλεσης της προσομοίωσης του τόσο δυναμικά όσο και με τη μορφή κειμένου.

Ο γενικός όρος «διάγραμμα» (chart) χρησιμοποιείται για την περιγραφή των απλών διαγραμμάτων, (Simple charts), των διαγραμμάτων που περιέχουν και παρελθόντα στοιχεία (Charts with history) καθώς και των ιστογραμμάτων, (Histograms). Η διαφορά τους βρίσκεται στο ότι τα απλά διαγράμματα εμφανίζουν τις τρέχουσες τιμές των δεδομένων εξόδου ενώ τα ιστογράμματα δέχονται δείγματα των δεδομένων εξόδου, τα επεξεργάζονται στατιστικά και εμφανίζουν τις τιμές των αποτελεσμάτων και του δείγματος. Τα απλά διαγράμματα κατηγοριοποιούνται και χωρίζονται σε ραβδογράμματα (Bar Chart), σε διαγράμματα σωρού (Stack Chart) και σε κυκλικά διαγράμματα (Pie Chart). Σαν διαγράμματα που περιέχουν και παρελθόντα στοιχεία ορίζονται όλα τα διαγράμματα που έχουν ως παράμετρο το χρόνο (Plot, Time Plot, Time Stack Chart, Time Color Chart) ενώ τα ιστογράμματα χωρίζονται σε αυτά των μιας και δύο διαστάσεων (Histogram, Histogram 2D).

Τα ραβδογράμματα επιδεικνύουν πλήθος στοιχείων δεδομένων με μορφή παραλληλόγραμμων των οποίων το μέγεθος είναι ανάλογο της τιμής του δεδομένου που αντιπροσωπεύουν. Στην περίπτωση αρνητικών τιμών οι ράβδοι αυξάνονται προς την αντίθετη κατεύθυνση. Το μεγέθος, το ύψος και το πλάτος, εξαρτάται από τις προτιμήσεις του χρήστη. Η κατεύθυνση των ράβδων μπορεί να είναι σε ένα από τα τέσσερα σημεία του ορίζοντα.

Τα διαγράμματα σωρού απεικονίζουν τα στοιχεία δεδομένων εξόδου της κάθε κατηγορίας ενός συνόλου δεδομένων με τη μορφή σωρού ράβδων. Το μέγεθος των ράβδων είναι ανάλογο των τιμών των αντίστοιχων στοιχείων. Οι αρνητικές τιμές δεν επιτρέπονται σε αυτό το είδος διαγράμματος.

Το κυκλικό διάγραμμα παρουσιάζει τη συμβολή της κάθε κατηγορίας ενός συνόλου δεδομένων ως τμήμα ενός κύκλου. Το μέγεθος του τόξου του κυκλικού τμήματος είναι ανάλογο της τιμής της αντίστοιχης κατηγορίας δεδομένων. Τα στοιχεία δεδομένων αυτού του διαγράμματος δεν μπορούν να έχουν αρνητική τιμή.

Τα διαγράμματα επί του καρτεσιανού επιπέδου απεικονίζουν σύνολα δεδομένων εξόδου των οποίων οι τιμές είναι σε μορφή ζεύγους, <χ,ψ>. Μία ορισμένη συνάρτηση υπολογίζει την τιμή του ψ συναρτήσει δοσμένης τιμής του χ. Οι τιμές χ και ψ παρουσιάζονται στους αντίστοιχους άξονες χ και ψ του καρτεσιανού επιπέδου.

Η γραφική παράσταση συναρτήσει του χρόνου παρουσιάζει ουσιαστικά την εξελικτική πορεία των τιμών με το πέρασμα του χρόνου. Ο οριζόντιος άξονας αντιπροσωπεύει πάντα το χρόνο και έχει φορά προς τα δεξιά. Ανάλογα με τον μαθηματικό τύπο που συσχετίζει ένα δεδομένο με το χρόνο, προκύπτει αντίστοιχη γραφική παράσταση.

Το διάγραμμα σωρού συναρτήσει του χρόνου παρουσιάζει την εξέλιξη της συμβολής διάφορων δεδομένων σε ένα σύνολο κατά τη διάρκεια του χρόνου. Κάθε χρονική στιγμή οι τιμές τοποθετούνται η μία πάνω στην άλλη δημιουργώντας μία στοίβα στον πάτο της οποίας βρίσκεται η πρώτη τιμή που εισήχθη. Οι αρνητικές δεν τιμές δεν επιτρέπονται σε αυτό το είδος διαγραμμάτων και ο οριζόντιος άξονας με φορά προς τα δεξιά απεικονίζει τη χρονική εξέλιξη.

Τα χρονικά διαγράμματα χρώματος, (Time Color Chart), παρουσιάζουν την μεταβολή διάφορων συνόλων στοιχείων κατά τη διάρκεια του χρόνου με τη μορφή οριζόντιων χρωματιστών ράβδων όπου οι χρωματικές αλλαγές είναι απόρροια της λογικής τιμής μιας συνθήκης που χαρακτηρίζει τα δεδομένα. Όταν η συνθήκη είναι αληθής, (TRUE), χρησιμοποιείται ένα συγκεκριμένο χρώμα και αντίστοιχα ένα διαφορετικό χρώμα επιλέγεται για την περιγραφή του ψευδούς, (FALSE) της συνθήκης. Ο οριζόντιος άξονας με φορά προς τα

δεξιά αντιπροσωπεύει πάντα το χρόνο. Αυτό το είδος διαγραμμάτων επιλέγεται στην περίπτωση που ο χρήστης επιθυμεί να παρακολουθήσει τις διακριτές αλλαγές ενός αντικειμένου που τελούνται στο πέρασμα του χρόνου, π.χ. απασχολημένος/ελεύθερος χρήστης, ανέβασμα/κατέβασμα αρχείου κλπ.

Τα ιστογράμματα απεικονίζουν την στατιστική επεξεργασία των αντικειμένων που ορίζονται στα δεδομένα ιστογράμματος (Histogram Data). Ο άξονας χ διαμορφώνεται ώστε να υποστηρίζει όλα τα είδη των δεδομένων του ιστογράμματος. Τα ιστογράμματα δύο διαστάσεων παρουσιάζουν δισδιάστατα διαγράμματα.

2.3.2 Η κατηγορία των δομικών στοιχείων της Παρουσίασης

Το Λογισμικό επιτρέπει τον σχεδιασμό πολύπλοκων δισδιάστατων αναπαραστάσεων χρησιμοποιώντας την αντικειμενοστραφή δομή κατά την διάρκεια της μοντελοποίησης. Στο επίπεδο της παρουσίασης του κάθε μοντέλου, οργανώνονται όλα τα σχήματα και οι ελεγκτικές διαδικασίες. Το κάθε σχήμα προσδιορίζεται από ένα πλήθος ιδιοτήτων που καθορίζουν την οπτική παρουσίαση του, π.χ. τη θέση, το μέγεθος, το χρώμα κλπ. Τα σχήματα μπορούν να αποκτήσουν δυναμικό χαρακτήρα χρησιμοποιώντας τις δυναμικές τους ιδιότητες οι οποίες επιτρέπουν στον χρήστη να καθορίσει τόσο τις τιμές αυτών των ιδιοτήτων όσο και τη εξάρτηση της εμφάνισης τους από τα δεδομένα κάποιου ενεργού αντικειμένου. Το αποτέλεσμα είναι η άμεση παρακολούθηση των εσωτερικών αλλαγών του μοντέλου χάρη της οπτικής τους αναπαράστασης κατά την διάρκεια της προσομοίωσης. Το Λογισμικό επιτρέπει στο χρήστη να μπορεί να παρέμβει και να τροποποιήσει τις διαδικασίες που συμβαίνουν μέσω πλήθους κουμπιών που του δίνουν τη δυνατότητα ελέγχου και χειρισμού.

Για τη σχεδίαση ευθύγραμμων τμημάτων και γραμμών χρησιμοποιείται το εργαλείο γραμμή, (Line). Το εργαλείο Polyline χρησιμοποιείται για τη σχεδίαση τεθλασμένων γραμμών ενώ για τη σχεδίαση καμπύλων γραμμών χρησιμοποιείται το εργαλείο Curve. Για τη σχεδίαση παραλληλόγραμμων σχημάτων ο χρήστης μπορεί να επιλέξει ανάμεσα στα εργαλεία Rectangle και Rectangle1. Με τη χρήση του εργαλείου Oval μπορούν να σχεδιαστούν κυκλικά και ελλειψοειδή σχήματα. Για την εισαγωγή σημείων το κατάλληλο εργαλείο είναι το Pixel και για τους κυκλικούς τομείς το εργαλείο Arc.

Στην περίπτωση που χρήστης επιθυμεί να εισάγει κείμενο και σχόλια στο μοντέλο, τα οποία δεν επηρεάζουν τη συμπεριφορά του, επιλέγει το εργαλείο Text. Αν επιθυμεί την εισαγωγή εικόνων επιλέγει το εργαλείο Image. Τέλος, με την επιλογή του εργαλείου Group δίνεται η δυνατότητα ομαδοποίησης των σχημάτων παρουσίασης.

Τα εργαλεία ελέγχου που το Λογισμικό προσφέρει προκειμένου το μοντέλο να αποκτήσει αλληλεπιδραστικό χαρακτήρα είναι το εργαλείο Button, ένα κουμπί που μπορεί ο χρήστης να πατήσει κατά την διάρκεια της προσομοίωσης του μοντέλου. Μπορεί να έχει ένα κωδικό όνομα, (Label). Καθορίζεται πότε είναι πατημένο και πότε όχι από μία δίτιμη έκφραση που ορίζεται από το χρήστη και επίσης ορίζεται ποιά θα είναι η ενέργεια που αυτό πυροδοτεί. Το εργαλείο Check χρησιμοποιείται για την επιλογή ή την ακύρωση μιας ενέργειας. Μπορεί να φέρει ένα κωδικό όνομα (Label), να το διέπει μία δίτιμη έκφραση που να καθορίζει αν θα είναι επιλεγμένο ή όχι κατά τη διάρκεια της προσομοίωσης και ποια ενέργεια θα ξεκινήσει αν αυτό επιλεγεί. Το εργαλείο Edit δημιουργεί ένα πεδίο στο οποίο ο χρήστης μπορεί να εισάγει δεδομένα. Το εργαλείο αυτό μπορεί να είναι ενεργοποιημένο ή απενεργοποιημένο κατά τη διάρκεια της προσομοίωσης και εκτελείται κώδικας αν αυτό επιλεγεί από το χρήστη. Το εργαλείο Radio Buttons χρησιμοποιείται για την επιλογή και την ενεργοποίηση μιας συγκεκριμένης επιλογής από μία λίστα αντικειμένων. Έχει συγκεκριμένο κωδικό όνομα και για κάθε ένα κουμπί της λίστας δίνεται μια τιμή μέσω μεταβλητών συγκεκριμένου τύπου. Οι επιλογές μπορούν να εμφανίζονται οριζόντια ή κάθετα κατά την επιλογή της ιδιότητας Orientation. Το εργαλείο Slider, οι ρυθμιστές ολίσθησης χρησιμοποιούνται όταν ζητάμε την αλλαγή των τιμών των ενεργών αντικειμένων με ορισμένο τρόπο. Ορίζεται η ελάχιστη τιμή, (Minimum value), και η μέγιστη τιμή, (Maximum value), και επίσης η αρχική τιμή, (Default value). Ταυτόχρονα, ένα κομμάτι κώδικα προστίθεται και εκτελείται σε κάθε αλλαγή της τιμής του ρυθμιστή. Το εργαλείο μπορεί να εμφανίζεται οριζόντια ή κάθετα.

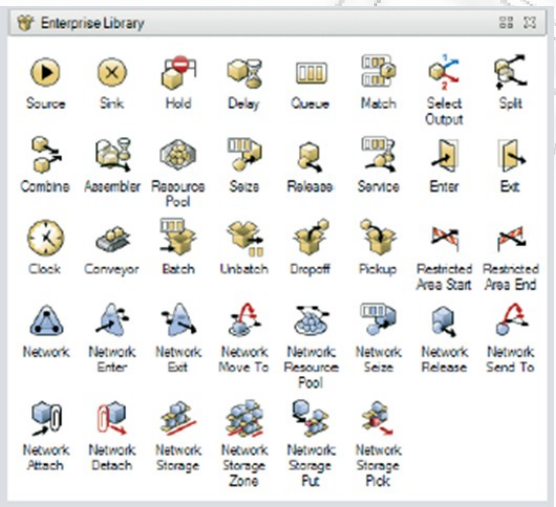
Η κατηγορία της Συνδεσιμότητας, Connectivity, περιέχει τα στοιχεία για τη διασύνδεση των βάσεων δεδομένων ενώ η κατηγορία των Εικόνων, Pictures, περιέχει ένα σύνολο από εικόνες που χρησιμοποιούνται πιο συχνά από τους χρήστες του Λογισμικού.

Το ενεργό παράθυρο της σχεδίασης των μοντέλων προσομοιώσεων, η περιοχή ανάπτυξης του Λογισμικού, ουσιαστικά υποκρύπτει ένα γραφικό συντάκτη (Graphical Editor). Για την επιλογή των στοιχείων από τις κατηγορίες, πατώ το δεξί πλήκτρο επάνω στο ενεργό αντικείμενο που με ενδιαφέρει στην όψη Project και επιλέγω το άνοιγμα μέσω του γραφικού συντάκτη. Επιλέγονται σχήματα, λειτουργίες π.χ. αντιγραφής και μεγέθυνσης.

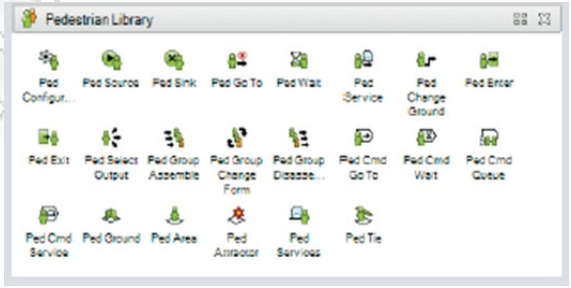
Ο γραφικός συντάκτης χρησιμοποιεί συντεταγμένες για την τοποθέτηση των αντικειμένων στο χώρο εργασίας. Το σημείο με συντεταγμένες (0,0) βρίσκεται, εξ' ορισμού, στο πάνω αριστερά σημείο της οθόνης. Ο συντάκτης είναι ο χώρος αυτός στον οποίο καθορίζονται οι δομές των κλάσεων ενεργού αντικειμένου, μέσα από μια σειρά καθορισμένων βημάτων που είναι :

1. Ο καθορισμός του ενδιάμεσου κάθε κλάσης
2. Ο καθορισμός του εικονιδίου του ενεργού αντικειμένου και της αναπαράστασης του με τη χρήση αντικειμένων από την κατηγορία Presentation, σχημάτων και ελέγχων. Σε αυτό το βήμα γίνεται η σύνδεση των ιδιοτήτων των σχημάτων με τα δεδομένα των ενεργών αντικειμένων και των ενσωματωμένων αντικειμένων.
3. Ο καθορισμός της συμπεριφοράς των στοιχείων π.χ. των γεγονότων (Events) ή των διαγραμμάτων καταστάσεων (Statecharts)
4. Ο καθορισμός των ενσωματωμένων αντικειμένων και του τρόπου διασύνδεσης τους.

Πέρα από τη δυνατότητα αποτύπωσης των ιδιοτήτων των ενεργών αντικειμένων υπάρχει η δυνατότητα να δούμε τον κώδικα των κλάσεων και των ενεργών αντικειμένων που το Λογισμικό δημιουργεί. Η εμφάνιση του κώδικα μπορεί να γίνει μέσω της χρήσης του συντάκτη της γλώσσας προγραμματισμού Java, (Java Editor). Μετά τη μετάφραση του μοντέλου από το μεταγλωτιστή (Build model), επιλέγοντας με δεξί κλικ του ποντικιού την κλάση που μας ενδιαφέρει, διαλέγουμε στην όψη Project View την επιλογή Open with Java Editor.



Εικόνα 2-6: Η βιβλιοθήκη Enterprise.



Εικόνα 2-7: Η βιβλιοθήκη Pedestrian.

Οι βιβλιοθήκες Enterprise (Εικόνα 2-6) και Pedestrian (Εικόνα 2-7), είναι δύο (2) ενσωματωμένες βιβλιοθήκες που συνοδεύουν το Λογισμικό. Η βιβλιοθήκη Enterprise περιέχει αντικείμενα με τα οποία μπορούμε να προσομοιώσουμε πολύπλοκα συστήματα διακριτών γεγονότων, όπως διαδικασίες παραγωγής με λεπτομερή γραφικά σχεδιαγράμματα, απλά και σύνθετα συστήματα υπηρεσιών π.χ. αεροδρόμια, τράπεζες, νοσοκομεία κλπ, εφοδιαστικές αλυσίδες και άλλα. Η βιβλιοθήκη αυτή μας επιτρέπει τη δημιουργία, την επικύρωση και την

παρουσίαση ευέλικτων μοντέλων προσομοίωσης που έχουν τη δυνατότητα συλλογής και ανάλυσης στατιστικών δεδομένων, τη δυνατότητα οπτικής αναπαράστασης των προσομοιούμενων διαδικασιών.

Αντίθετα, η βιβλιοθήκη Pedestrian εξετάζει λεπτομερώς σε φυσικό επίπεδο την δυναμική των συστημάτων στα οποία οι πεζοί είναι τα κύρια αντικείμενα παρατήρησης. Παράμετροι όπως τα μεγέθη των επιμέρους αντικειμένων, οι κανόνες συμπεριφοράς, η ικανότητά των πεζών να επιταχύνουν και να επιβραδύνουν, οι τοίχοι, τα εμπόδια, οι σκάλες, οι προτεραιότητες στην κίνηση κλπ αποτελούν στοιχεία που επιτρέπουν την μοντελοποίηση και τη βαθύτερη κατανόηση του προς εξέταση συστήματος. Μπορούν να μετρηθούν με ακρίβεια και να βελτιστοποιηθούν οι ιδιότητες του συστήματος, να ανακαλυφθούν τα σημεία συμφόρησης και να προβλέφθούν επικίνδυνες καταστάσεις που διαφορετικά θα παρέμεναν αφανέρωτα. Κάθε προσέγγιση μπορεί να διαμορφωθεί προσδιορίζοντας τα ειδικά χαρακτηριστικά στα τελικά μοντέλα.

Επίσης, μια νέα βιβλιοθήκη, η Yard Rail, επιτρέπει τη δημιουργία μοντέλων, την προσομοίωση και την οπτική απεικόνιση σιδηροδρομικών μοντέλων οποιασδήποτε πολυπλοκότητας και κλίμακας. Μπορεί να χρησιμοποιηθεί συνδυαστικά με μοντέλα διακριτών γεγονότων ή και με μοντέλα πρακτόρων, δημιουργώντας υβριδικά μοντέλα προσομοίωσης για την εξέταση μοντέλων που σχετίζονται με τη μεταφορά, τη φόρτωση και την εκφόρτωση, την κατανομή των πόρων, την συντήρηση των μηχανημάτων και του υλικού γενικότερα, επιχειρηματικές διαδικασίες, κλπ. Η βιβλιοθήκη παράγει πολύ λεπτομερείς προσομοιώσεις και είναι δυνατό να προσδιοριστεί η βέλτιστη λύση όταν χρησιμοποιείται ο βελτιστοποιητής, (optimizer).

2.3.3 Μέθοδοι μοντελοποίησης και ενεργά αντικείμενα.

Όπως έχουμε ήδη αναφέρει προηγούμενα, στη γλώσσα μοντελοποίησης του Λογισμικού κεντρικό ρόλο έχει η έννοια του μοντέλου, η περιγραφή του προβλήματος που υλοποιείται μέσα από ένα σύνολο ενεργών αντικειμένων με τη βοήθεια του γραφικού συντάκτη που ορίζει όχι μόνο τη δομή των ενεργών αντικειμένων και των κλάσεων που τα αποτελούν, αλλά ορίζει επίσης τον τρόπο διασύνδεσης των αντικειμένων, καθορίζει τη συμπεριφορά τους, επιτρέπει κάνοντας χρήση των κατάλληλων σχημάτων και των ελέγχων από την όψη της Παρουσίασης την εικονοποίηση των αντικειμένων και ορίζει την διεπαφή τους με το χρήστη.

Στη μέθοδο μοντελοποίησης Διακριτών Γεγονότων, κεντρικό ρόλο κατέχει το Γεγονός, (Event), το οποίο μπορεί να παραμένει ενεργό και να εκτελείται περισσότερες από μια φορές, χρειάζεται μηδενικό χρόνο για να εκτελεστεί, είναι ατομικό (δεν αλληλεπιδρά με κανένα άλλο εκτελούμενο Γεγονός) και μπορεί να αλλάξει το μοντέλο προκαλώντας αλληλουχία άλλων Γεγονότων οπότε και μπορεί να έχουμε την διαδοχική εκτέλεση Γεγονότων. Υπάρχουν περιπτώσεις που η σειρά εκτέλεσης έχει σημασία και θα πρέπει ο χρήστης να προβλέψει τέτοιου είδους λεπτομέρειες κατά τη δημιουργία του μοντέλου. Στο Λογισμικό Γεγονότα μπορούν να υλοποιηθούν μέσα από δύο τύπους αντικειμένων: τα Γεγονότα (Events) και τα Διαγράμματα Μετάβασης (Statecharts). Σε επίπεδο βιβλιοθηκών του Λογισμικού, σε ανώτερο επίπεδο, η χρήση της Βιβλιοθήκης Enterprise βοηθά στη δημιουργία διαδικασιοκεντρικών μοντέλων.

2.3.3.1 Στατικό Γεγονός, (Static Event)

Είναι ο απλούστερος τρόπος προγραμματισμού κάποιας ενέργειας, λειτουργίας σε ένα μοντέλο. Συνήθως χρησιμοποιείται για τη μοντελοποίηση καθυστερήσεων και την επιβολή χρονικών ορίων στις διαδικασίες. Για τον προγραμματισμό κάποιας ενέργειας με στατικό Γεγονός, είναι απαραίτητο να καθοριστεί το πότε θα συμβεί και το τι θα εκτελέσει. Το τι ενέργεια θα εκτελεστεί προγραμματίζεται από τον χρήστη μέσω της γλώσσας προγραμματισμού Java. Το πότε θα πραγματοποιηθεί, εξαρτάται από τον τύπο πυροδότησης που ο χρήστης ορίζει μεταξύ των εξής τριών τύπων:

1. Συγκεκριμένη στιγμή (Timeout triggered event): το γεγονός πραγματοποιείται σε μία γνωστή εκ των προτέρων χρονική στιγμή.

2. Υπό συνθήκη (Condition triggered event): όταν η συνθήκη που έχει εισάγει ο χρήστης ισχύει, το γεγονός πραγματοποιείται.
3. Ρυθμού (Rate triggered event): εάν ο χρήστης καθορίσει ένα γεγονός να συμβαίνει βάσει ρυθμού, τότε αυτό θα πραγματοποιείται περιοδικά, ανάλογα με τον καθορισμένο ρυθμό.

2.3.3.2 Δυναμικό Γεγονός (Dynamic Event)

Τα δυναμικά γεγονότα χρησιμοποιούνται για τον προγραμματισμό αριθμού ταυτόχρονων και ανεξάρτητων γεγονότων. Για παράδειγμα, η μοντελοποίηση ενός καναλιού επικοινωνίας μπορεί να γίνει χρησιμοποιώντας δυναμικά γεγονότα όπου το κάθε γεγονός αντιστοιχεί σε κάθε ένα μήνυμα που αποστέλλεται μέσω του διαύλου επικοινωνίας. Άλλο παράδειγμα μπορεί να θεωρηθεί ο εξυπηρετητής δικτύου με απεριόριστη χωρητικότητα. Έχουν την ικανότητα να αρχικοποιήσουν κάθε στιγμιότυπο των Δυναμικών Γεγονότων με ορισμό των αρχικών τιμών των δεδομένων χρησιμοποιώντας τις παραμέτρους των γεγονότων.

2.3.3.3 Σύγκριση των Γεγονότων και των Δυναμικών Γεγονότων

Και οι δύο (2) τύποι είναι υπεύθυνοι για τον προγραμματισμό από το χρήστη ορισμένων ενεργειών σε ένα μοντέλο. Όμως τα Δυναμικά Γεγονότα συμβαίνουν μόνο μια φορά, αντίθετα από τα Στατικά Γεγονότα που μπορούν να επαναλαμβάνονται. Επίσης, αντίθετα από τα Στατικά Γεγονότα, τα Δυναμικά Γεγονότα μπορούν να περνάνε παραμέτρους, η αρχικοποίηση κάθε αντιγράφου των Δυναμικών Γεγονότων γίνεται με συγκεκριμένο κώδικα. Καλό είναι να χρησιμοποιούνται όταν α) όταν η ενέργεια που θα γίνει εξαρτάται από ορισμένες συγκεκριμένες πληροφορίες και β) όταν περιμένουμε πολλά γεγονότα, που θα εκτελέσουν παρόμοιες ενέργειες, αυτά να προγραμματίζονται, στο μοντέλο, την ίδια χρονική στιγμή.

2.3.4 Διαγράμματα Μετάβασης (Statecharts)

Ορισμένες φορές χρειάζεται να μοντελοποιήσουμε πολύπλοκες συμπεριφορές, οι οποίες δεν είναι δυνατό να αναπαρασταθούν με τα στατικά ή/και τα δυναμικά γεγονότα. Τα διαγράμματα μετάβασης είναι κατάλληλα σε αυτές τις περιπτώσεις, αφού η χρονική σειρά πραγματοποίησης συγκεκριμένων γεγονότων είναι σημαντική. Τα διαγράμματα αυτά περιέχουν τόσο την κατάσταση που βρίσκεται το αντικείμενο όσο και την κατάσταση στην οποία θα μεταβεί. Οι μεταβάσεις συμβαίνουν σε συνθήκες που καθορίζονται από το χρήστη ενώ η εκτέλεση τους μπορεί να επιφέρει νέα αλλαγή κατάστασης η οποία να ενεργοποιεί ένα διαφορετικό σύνολο μεταβάσεων. Οι καταστάσεις μπορεί να είναι ιεραρχικές, δηλαδή να περιέχουν και άλλες καταστάσεις εσωτερικά. Το ίδιο το διάγραμμα καταστάσεων και η οργάνωση του, η δομή του, περιέχεται στο ενεργό αντικείμενο με το οποίο σχετίζεται.

Η Κατάσταση, (State), αντιπροσωπεύει μια θέση ελέγχου με ένα συγκεκριμένο σύνολο συνθηκών ή/και συμβάντων. Μια κατάσταση μπορεί να είναι είτε απλή είτε, εάν περιέχει άλλες καταστάσεις, σύνθετη. Ο έλεγχος βρίσκεται πάντα σε μία από τις απλές καταστάσεις, αλλά το τρέχον σύνολο των αντιδράσεων είναι μια ένωση της τρέχουσας απλής κατάσταση και όλων τα σύνθετα καταστάσεων που την περιέχουν. Η μετάβαση, (Transition), ορίζει την αλλαγή από μια κατάσταση σε μια άλλη και συμβαίνει όταν ένα γεγονός πυροδότησης λάβει χώρα και η συνθήκη αλλαγής είναι αληθής. Οι τύποι πυροδότησης είναι οι ίδιοι που ισχύουν και στα Γεγονότα. Ένας ειδικός τύπος μετάβασης ονομάζεται εσωτερικός, (Internal Transition), και είναι χρήσιμος για την υλοποίηση απλών εργασιών στο παρασκήνιο χωρίς να επιρρεάζουν και να διακόπτουν την κύρια δραστηριότητα της σύνθετης κατάστασης. Επίσης, ιδιαίτερη σημασία έχει η μετάβαση που πυροδοτείται από ένα μήνυμα, (Message Triggered Transition). Η μετάβαση εκτελείται όταν το διάγραμμα καταστάσεων λάβει ένα μήνυμα, (ένα αντίγραφο μιας κλάσης ή μια τιμή ενός θεμελιακού τύπου), που συμμορφώνεται με τον ορισμένο περιγραφέα του μηνύματος. Υπάρχουν τρεις τρόποι για να παρχθούν γεγονότα μηνυμάτων: (α) Μέσω της σύνδεσης μιας θύρας, (port), και ενός διαγράμματος καταστάσεων. Ότι μήνυμα έρχεται στην θύρα, οδηγείται

στο διάγραμμα, (β) με την κλήση της μεθόδου receiveMessage() ενός διαγράμματος καταστάσεων και (γ) με την κλήση της μεθόδου fireEvent() ενός διαγράμματος καταστάσεων. Στην περίπτωση αυτή το μήνυμα θα τοποθετηθεί στην ουρά του διαγράμματος καταστάσεων. Η ουρά είναι απαραίτητη σε αυτές τις περιπτώσεις που τα μηνύματα φτάνουν σε εκείνες τις στιγμές του χρόνου κατά την οποία το διάγραμμα καταστάσεων δεν μπορεί να αντιδράσει στα γεγονότα (π.χ. όταν εκτελείται μια μετάβαση).

Στη μέθοδο μοντελοποίησης της Δυναμικής Συστημάτων, δημιουργούνται πολύπλοκα δυναμικά μοντέλα με τη χρήση μεταβλητών δύο (2) τύπων, (α) τις απλές μεταβλητές, (Plain Variables), αυτές που ορίζονται τόσο σε μαθηματικούς τύπους όσο και ως χαρακτηριστικά μιας κλάσης Java και (β) τις Βοηθητικές Μεταβλητές Ροής, (Flow Auxiliaries Variables), με τις οποίες καθορίζονται οι δυναμικές μεταβλητές του μοντέλου οι οποίες μπορεί να αναπαριστούν είτε ροές (Flows), (επίσης είναι γνωστές ως ρυθμοί, (Rates) και μεταβάλλουν την τιμή των αποθεμάτων (Stocks)), είτε βοηθητικές μεταβλητές που χρησιμοποιούνται επικουρικά στον ορισμό των μοντέλων. Όταν εισάγεται μια μεταβλητή στον γραφικό συντάκτη θεωρείται βοηθητική μεταβλητή ενώ όταν λαμβάνει μέρος στην περιγραφή της εξίσωσης κάποιου αποθέματος τότε αντιμετωπίζεται ως ροή. Τα αποθέματα, (Stocks), που είναι επίσης γνωστά ως συσσωρεύσεις ή ως μεταβλητές κατάστασης, έχουν χρονικά μεταβαλλόμενες συνεχείς τιμές και καθορίζουν σε ένα σύστημα τις τιμές των ροών. Οι μεταβλητές μπορούν να αλλάζουν κατά την διάρκεια της εκτέλεσης του μοντέλου.

Εκτός των παραπάνω μεταβλητών μια άλλη κατηγορία είναι οι συλλογές μεταβλητών, (Collection Variables), οι οποίες χρησιμοποιούνται για τον ορισμό στοιχείων αντικειμένων που ομαδοποιούν πολλά ομοειδή στοιχεία σε μια ενιαία μονάδα. Οι συλλογές, (Collections), χρησιμοποιούνται για την αποθήκευση, την ανάκτηση και τον χειρισμό δεδομένων που σχηματίζουν φυσικές ομάδες, π.χ. μια ουρά αναμονής ή ένας τηλεφωνικό κατάλογο. Οι παραπάνω κατηγορίες μεταβλητών (απλές και συλλογές) μπορούν να είναι στατικές, η τιμή τους να διατηρείται.

Το Λογισμικό επιτρέπει τον καθορισμό συναρτήσεων, (Functions), στη γλώσσα προγραμματισμού Java από τον χρήστη. Οι συναρτήσεις είναι χρήσιμες όταν θέλουμε να επαναχρησιμοποιήσουμε τον ίδιο κώδικα αρκετές φορές κατά την υλοποίηση του μοντέλου. Άλλος τύπος συνάρτησης που το Λογισμικό επιτρέπει είναι οι συναρτήσεις πίνακα, (Table Functions), που χρησιμοποιούνται για την περιγραφή πολύπλοκων μη γραμμικών σχέσεων. Η συνάρτηση σε αυτή την περίπτωση χρησιμοποιεί ζευγάρι δεδομένων, όρισμα και τιμή, με τη χρήση κάποιου επιλεγμένου τύπου μαθηματικής παρεμβολής.

Μπορούμε να συνδέσουμε αντικείμενα συνδέοντας τις μεταβλητές τους. Οι συνδεόμενες μεταβλητές θα έχουν σε κάθε χρονική στιγμή την ίδια τιμή και οι αλλαγές τους θα επηρεάζουν τις υπόλοιπες μεταβλητές, (εξαρτημένες μεταβλητές). Επιτρέπεται από το Λογισμικό η αλληλεπίδραση των αντικειμένων σε οποιαδήποτε κλάση. Η επικοινωνία των ενεργών αντικειμένων είναι αφικτή μέσω της ανταλλαγής μηνυμάτων από τις θύρες επικοινωνίας (Ports). Ο μηχανισμός της επικοινωνίας είναι απλός: τα μηνύματα αποστέλλονται και λαμβάνονται μέσω των θυρών επικοινωνίας οι οποίες επιτρέπουν την είσοδο και την έξοδο των πληροφοριών από ένα ενεργό αντικείμενο σε ένα άλλο. Όταν αποστέλλεται ένα μήνυμα μέσω μιας θύρας επικοινωνίας, προωθείται σε όλες τις θύρες συνδέσεων έξω από το ενεργό αντικείμενο στο οποίο η θύρα ανήκει. Το στοιχείο Connector χρησιμοποιείται για να συνδέει θύρες ή μεταβλητές. Συνδέοντας δύο θύρες επιτρέπεται στο μήνυμα να περάσει μέσα από αυτές τις θύρες. Συνδέοντας δύο μεταβλητές καθιερώνεται η σύνδεση μεταβλητών και όταν συμβαίνουν αλλαγές στη μια μεταβλητή μεταφέρονται αμέσως στην άλλη όταν αυτή έχει δηλωθεί ως εξαρτημένη.

Στη μέθοδο μοντελοποίησης των Πρακτόρων Λογισμικού τα ενεργά αντικείμενα είναι τα βασικά δομικά στοιχεία των μοντέλων του Λογισμικού και μπορούν να χρησιμοποιηθούν για τη μοντελοποίηση πολύ διαφορετικών αντικειμένων του πραγματικού κόσμου. Όλα τα χαρακτηριστικά του αντικειμενοστραφούς προγραμματισμού λαμβάνονται υπόψη για τη κατασκευή των μοντέλων. Ένα χαρακτηριστικό του Λογισμικού που αναλύεται στη συνέχεια είναι η κατηγορία Actionchart.

2.3.5 Η κατηγορία των δομικών στοιχείων Actionchart

Η προσομοίωση σύνθετων μοντέλων απαιτεί αλγόριθμους ικανούς για επεξεργασία δεδομένων και υπολογισμούς. Το Λογισμικό προσφέρει τη δυνατότητα χρήσης των διαγραμμάτων δράσης (action charts) τα οποία είναι δομημένα διαγράμματα και επιτρέπουν τον γραφικό καθορισμό των αλγορίθμων. Τα διαγράμματα αυτά επιτρέπουν στον χρήστη να κατασκευάζει αλγόριθμους ακόμα και αν δεν είναι εξοικειωμένος με τους τελεστές της Java. Το πλεονέκτημα τους είναι ότι ο αλγόριθμος απεικονίζεται γραφικά και ο χρήστης κατανοεί άμεσα τι λειτουργία επιτελεί στο μοντέλο.

Για τη σχεδίαση ενός διαγράμματος ο χρήστης επιλέγει και στη συνέχεια τοποθετεί στον συντάκτη γραφικών τα δομικά στοιχεία του ActionChart. Δημιουργείται έτσι από το χρήστη ένα διάγραμμα που περιλαμβάνει μεταξύ της αφετηρίας και του σημείου επιστροφής του διαγράμματος τα υπόλοιπα δομικά στοιχεία που είναι αναγκαία για την περιγραφή του αλγόριθμου. Τα διαγράμματα δράσης λειτουργούν και οργανώνονται όπως οι συναρτήσεις σε ένα πρόγραμμα. Ο χρήστης καθορίζει τα ορίσματα για τα δεδομένα εισόδου αν αυτό είναι απαραίτητο. Τα ορίσματα αυτά είναι ορατά από όλες τις υπόλοιπες δομικές μονάδες του διαγράμματος και με αυτό τον τρόπο μπορούν να γίνουν οι υπολογισμοί που είναι αναγκαίοι για την εκτέλεση μιας σειράς υπολογιστικών εντολών. Επίσης, είναι δυνατή η χρήση τοπικών μεταβλητών για την αποθήκευση των αποτελεσμάτων βοηθητικών υπολογισμών. Οι τοπικές μεταβλητές είναι ορατές μόνο από το σημείο στο οποίο δηλώνονται και μετά.

Παρέχονται έτοιμα δομικά στοιχεία για τη δημιουργία βρόχων επαναλήψεων με ή χωρίς συνθήκη επανάληψης, οι βρόχοι While, Do While και For. Ο βρόχος For εμφανίζεται με δύο (2) μορφές, τη γενική (Generic) και αυτή που χρησιμοποιείται σαν επαναλήπτης στις ειδικές δομές της Java, στις Συλλογές (Collection Iterator).

Εκτός των παραπάνω σημαντικά δομικά στοιχεία είναι το Code που επιτρέπει την εισαγωγή κώδικα, (απλή εντολή ή σύνολο εντολών), με στόχο την εκτέλεση κάποιας ενέργειας εντός του διαγράμματος, το Decision που επιτρέπει την αλλαγή της ροής του διαγράμματος μετά από απόφαση ανάλογα με τα δεδομένα που ισχύουν τη δεδομένη χρονική στιγμή. Από προεπιλογή, ο κλάδος True είναι προς τα δεξιά και ο κλάδος False προς τα αριστερά. Το δομικό στοιχείο Return έχει δύο σημαντικούς ρόλους. Επιστρέφει την τιμή του διαγράμματος δράσης στο μοντέλο και κατ' επέκταση στον χρήστη και επιστρέφει συγκεκριμένη τιμή αν το διάγραμμα είναι τύπου Void. Κάθε είδους διακλάδωση που περιέχει το διάγραμμα πρέπει να καταλήγει στο στοιχείο Return, είναι το σημείο επιστροφής του διαγράμματος. Για αυτό το λόγο, στην αρχή της δημιουργίας του κάθε διαγράμματος το δομικό στοιχείο Return παράγεται αυτόματα από το Λογισμικό. Το δομικό στοιχείο Break ελέγχει τη ροή ενός βρόχου και μπορεί να αποφασίζει πότε ο βρόγχος θα τερματίζεται.

Η εντολή Run από την το μενού Model, ξεκινά την προσομοίωση του δημιουργηθέντος μοντέλου.

ΚΕΦΑΛΑΙΟ 3

Μοντελοποίηση της διασποράς μιας μολυσματικής νόσου με τη μέθοδο της Δυναμικής Συστημάτων

Η πρώτη εφαρμογή που θα μας απασχολήσει προέρχεται από την ιατρική και συγκεκριμένα από το χώρο της επιδημιολογίας. Το μοντέλο που θα παρουσιαστεί, εξετάζει την εξάπλωση μιας μολυσματικής επιδημίας σε έναν πληθυσμό λαμβάνοντας υπόψη μαθηματικές παραμέτρους με φυσική σημασία. Η εφαρμογή αποτελεί χαρακτηριστικό παράδειγμα διότι έχει εφαρμογή στην καθημερινή μας ζωή. Ας μην ξεχνάμε πως οι επιδημίες γρίπης, που συνεχώς μας ταλαιπωρούν, εξαπλώνονται πέρα από τα τοπικά όρια μιας περιοχής και συχνά παρουσιάζουν παγκόσμια διασπορά στη διάρκεια του χρόνου. Επίσης, όχι μόνο στο παρελθόν αλλά και στο παρόν, επιδημίες και μολυσματικές ασθένειες ήταν και είναι κύριες αιτίες θανάτου και αποδεκατισμού πληθυσμών, παγκοσμίως.

3.1 Θεωρητικό μοντέλο

Πολλά μοντέλα έχουν προταθεί στη βιβλιογραφία για την προσομοίωση των επιδημιών. Ένα από τα πλέον γνωστά είναι το μοντέλο SIR των Kermack και McKendrick που πρωτοπαρουσιάστηκε το 1927 και αποτελεί σημείο αναφοράς στην μοντελοποίηση των επιδημιών· είναι δε τόσο μεγάλη η επίδραση του στη μοντελοποίηση και στην επιστήμη γενικότερα που χρησιμοποιείται ακόμη και σήμερα. Βασικές παραδοχές του μοντέλου είναι: α) η επιδημία εξαπλώνεται μέσω της επαφής με μολυσμένα από την ίωση, τη μόλυνση, την αρρώστια άτομα β) τα άτομα αναρρώνουν από την ίωση, τη μόλυνση, την αρρώστια. Το μοντέλο δεν δέχεται την απώλεια ζωών κατά την προσομοίωση της επιδημίας και γ) τα άτομα αποκτούν ανοσία μετά την έκθεση τους στην ίωση, στη μόλυνση, στην αρρώστια.

Στο βασικό μοντέλο, (στο πέρασμα των ετών έχουν διατυπωθεί και προταθεί βελτιώσεις), θεωρούμε γνωστό το γεωγραφικό χώρο ως προς τα φυσικά όρια του και δεδομένο τον πληθυσμό των ατόμων. Ο πληθυσμός χωρίζεται σε τρεις κατηγορίες σύμφωνα με τις καταστάσεις της ασθένειας S, I, R:

- Η κατάσταση S (susceptible), κατάσταση ευπάθειας ή ευαισθησίας: Τα άτομα που δεν νόσησαν στο παρελθόν και δεν έχουν νοσήσει ακόμη. Μόλις νοσήσουν, αυτό συμβαίνει μέσω της επαφής με άτομα που νοσούν ήδη, μεταπίπτουν στην κατηγορία I.
- Η κατάσταση I (infectious), κατάσταση μολυσματική, λοιμώδης, μεταδοτική: Τα άτομα αυτής της κατηγορίας έχουν μολυνθεί από τη νόσο, είναι φορείς και μπορούν να μολύνουν άτομα της κατηγορίας S με συνέπεια τη διασπορά της ασθένειας. Τα άτομα της κατηγορίας I αναρρώνουν σε ορισμένο χρονικό διάστημα παύοντας να είναι φορείς της νόσου.

Ορίζεται ως μολυσματική περίοδος, η χρονική περίοδος κατά την οποία τα άτομα νοσούν και θεωρούνται μολυσματικά. Μόλις περάσει η περίοδος αυτή, τα άτομα μεταπίπτουν στην κατάσταση R.

- Η κατάσταση R (recovered) θεωρείται ως η θεραπευμένη κατάσταση: Τα άτομα αυτής της κατηγορίας δεν θα νοσήσουν ξανά στο μέλλον από τη συγκεκριμένη ασθένεια μιας που έχουν ανοσία πια στην συγκεκριμένη ασθένεια.

Το μοντέλο SIR εξετάζει τις τρεις αυτές καταστάσεις και τη μεταβολή των μεγεθών τους στη διάρκεια μιας χρονικής περιόδου για δεδομένο αριθμό πληθυσμού ατόμων. Τα μεγέθη είναι ανηγμένα στο συνολικό πληθυσμό και ισχύει ότι S+I+R=1.

Η μετάβαση μεταξύ των καταστάσεων γραφικά ακολουθεί τη διαδρομή:

Το μοντέλο είναι ντετερμινιστικό, για ίδιες αρχικές συνθήκες τα αποτελέσματα θα είναι ίδια, υλοποιείται με τη χρήση κανονικών διαφορικών εξισώσεων πρώτου βαθμού που μαθηματικά εκφράζεται ως:

$$\frac{dS}{dt} = -\rho SI$$

$$\frac{dI}{dt} = \rho SI - \alpha I$$

$$\frac{dR}{dt} = \alpha I$$

όπου ως ρ ορίζεται ο ρυθμός μετάδοσης της ασθένειας και ως α ο ρυθμός ίασης των ατόμων του πληθυσμού. Ο βασικός αριθμός αναπαραγωγής, (basic reproduction number), ορίζεται ως ο λόγος των ρυθμών ρ/α και αντιπροσωπεύει το μέσο αριθμό μολύνσεων που προκαλούνται από ένα μόνο φορέα που ανήκει στο συνολικό πληθυσμό των ευπαθών ατόμων. Ο αριθμός αυτός είναι σημαντικός διότι αν είναι μεγαλύτερος του 1, τότε έχουμε την εμφάνιση επιδημίας. Το ποσοστό των νέων μολύνσεων μπορεί να οριστεί ως ρSI, όπου είναι μια παράμετρος για τη μολυσματικότητα ενώ παράμετρος της αποθεραπείας είναι ο παράγοντας αI.

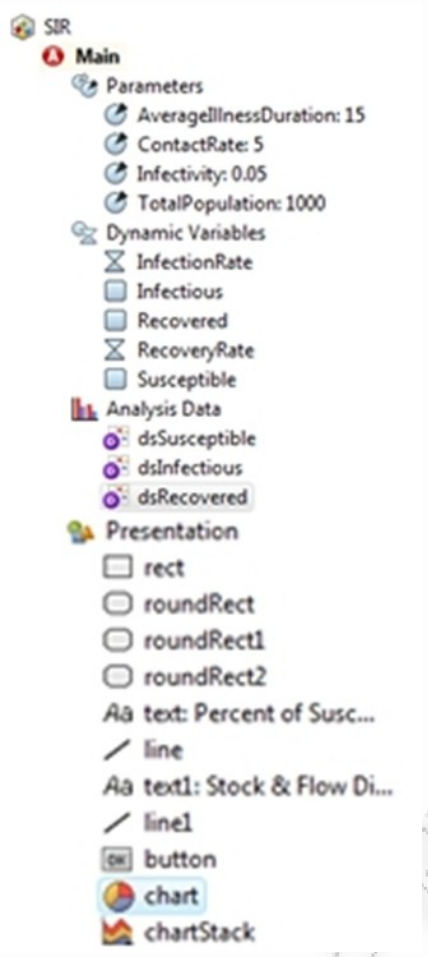
3.2 Μοντελοποίηση στο Λογισμικό - Η κλάση Main

Το μοντέλο υλοποιήθηκε χρησιμοποιώντας τη μέθοδο μοντελοποίησης της Δυναμικής Συστημάτων. Όπως παρουσιάζεται στην εικόνα 3-1, δημιουργήθηκε το έργο (project) SIR και μέσα σε αυτό ορίσαμε την κλάση Main, η οποία είναι κλάση ενεργού αντικειμένου (Active Object Class) που δεν περιέχει δημόσια στοιχεία (public elements). Η κλάση «χωρίζεται» σε δύο τμήματα, α) το υπολογιστικό τμήμα όπου ορίζονται οι παράμετροι (General Palette), οι μεταβλητές (System Dynamics Palette) και το τμήμα που περιέχει στοιχεία από την παλέτα συλλογής, ανάλυσης και εμφάνισης των στοιχείων δεδομένων (Analysis Palette) ώστε να κατασκευαστεί το μαθηματικό μοντέλο και β) το γραφικό τμήμα παρουσίασης (Presentation Palette), το τμήμα στο οποίο εμφανίζονται με γραφικό τρόπο τα συνδεδεμένα αντικείμενα δεδομένων κατά την προσομοίωση.

Παράμετροι της κλάσης Main

Όνομα παραμέτρου	Εξήγηση παραμέτρου	Τύπος παραμέτρου	Αρχική τιμή παραμέτρου
TotalPopulation	Ο συνολικός πληθυσμός προς εξέταση	Double	1000
Infectivity	Μολυσματικότητα	Double	0.05
ContactRate	Ο ρυθμός επαφής του πληθυσμού	Double	5
AverageIllnessDuration	Η μέση χρονική διάρκεια της ασθένειας	Double	15

Πίνακας 3-1: Οι παράμετροι της κλάσης Main.



Εικόνα 3-1: Η ιεραρχία του μοντέλου SIR.

Η ιεραρχική δενδρική δομή της κλάσης είναι εύκολο να γίνει αντιληπτή χωρίς ιδιαίτερη προσπάθεια. Επιγραμματικά, αποτελείται από τέσσερις (4) παραμέτρους (parameters, με εμφάνιση των τιμών τους), τρια (3) σημεία συσσώρευσης (stocks) και δύο (2) ροές (flows), τρια (3) σύνολα στοιχείων δεδομένων (data sets) και τα απαραίτητα σχήματα για την δημιουργία του γραφικού περιβάλλοντος παρουσίασης. Συγκεκριμένα χρησιμοποιήθηκαν ένα (1) παραλληλόγραμμο, τρια (3) παραλληλόγραμμα με στρογγυλεμένες γωνίες, δύο (2) κουτιά κειμένου, δύο (2) σχήματα γραμμές, ένα (1) σχήμα ελέγχου κουμπιού, ένα (1) κυκλικό γράφημα και ένα (1) ιστόγραμμα παρακολούθησης των στιγμιαίων τιμών των μεταβλητών σε πραγματικό χρόνο κατά την προσομοίωση του μοντέλου.

Θυμίζουμε σε αυτό το σημείο ότι τα σημεία συσσώρευσης χρησιμοποιούνται για την αναπαράσταση των πραγματικών διαδικασιών (real-world ή real-time processes) που η τιμή τους αλλάζει στο χρόνο με δεδομένη ροή. Η ροή είναι ο ρυθμός μεταφοράς μεταξύ των σημείων συσσώρευσης. Επίσης, κατά τη δημιουργία του μοντέλου κάνουμε την παραδοχή ότι έχουμε επαφές μεταξύ μολυσμένων και ευπαθών ατόμων σε ποσοστό ανάλογο με τους αντίστοιχους αριθμούς τους στον πληθυσμό. Η επιδημία σταματά μόλις η τιμή του Susceptible τείνει να μηδενιστεί.

Παρουσιάζουμε αναλυτικά, στον Πίνακα 3-1, τις παραμέτρους του μοντέλου με τη σειρά που εμφανίζονται στην ιεραρχική δομή. Οι δυναμικές μεταβλητές της κλάσης Main αποτελούνται από τις ροές InfectionRate και RecoveryRate, με μαθηματικές εκφράσεις τις:

$$\boldsymbol{InfectionRate} = \frac{\boldsymbol{Infectious} * \boldsymbol{ContactRate} * \boldsymbol{Susceptible}}{\boldsymbol{TotalPopulation} * \boldsymbol{Infectivity}}$$

και

$$\boldsymbol{RecoveryRate} = \frac{\boldsymbol{Infectious}}{\boldsymbol{AverageIllnessDuration}}$$

αντίστοιχα.

Τα σημεία συσσώρευσης Infectious, (με αρχική τιμή 1), Recovered και Susceptible, (με αρχική τιμή TotalPopulation –1), ορίζονται ως:

$$\frac{\boldsymbol{d(Infectious)}}{\boldsymbol{dt}} = \boldsymbol{InfectionRate} - \boldsymbol{RecoveryRate}$$

$$\frac{\boldsymbol{d(Recovered)}}{\boldsymbol{dt}} = \boldsymbol{RecoveryRate}$$

$$\frac{d(Susceptible)}{dt} = -InfectionRate$$

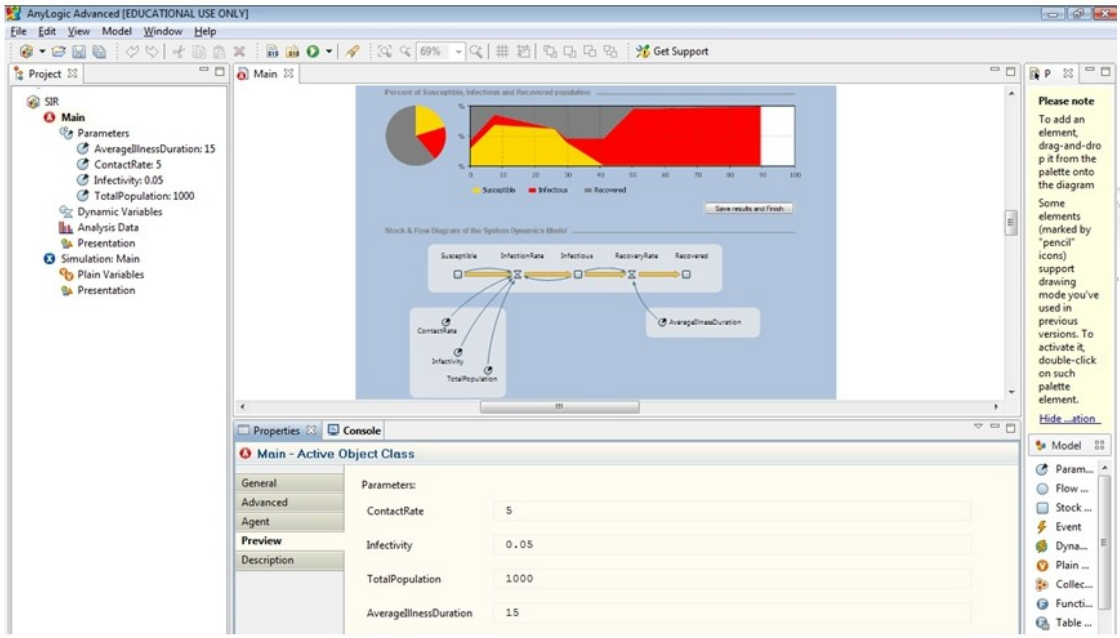
Οι ροές και τα σημεία συσσώρευσης είναι τύπου double. Στα σύνολα των στοιχείων δεδομένων dsSusceptible, dsInfectious και dsRecovered αποθηκεύονται οι στιγμιαίες τιμές των σημείων συσσώρευσης Infectious, Recovered και Susceptible κατά τη διάρκεια της προσομοίωσης. Τα σύνολα αυτά χρησιμεύουν στην γραφική απεικόνιση των αποτελεσμάτων.

Τα στοιχεία παρουσίασης (Presentation) της κλάσης Main σχηματίζουν γραφικά το μοντέλο προσομοίωσης και δημιουργούν το γραφικό περιβάλλον που βρίσκεται στο προσκήνιο της μοντελοποίησης. Στην εικόνα 3-2 παρουσιάζεται το ολοκληρωμένο προσκήνιο όπως εμφανίζεται στο περιβάλλον του λογισμικού ενώ εκτελείται η προσομοίωση και παρουσιάζονται «σε πραγματικό χρόνο» τα αποτελέσματα της προσομοίωσης.

Εικόνα 3-2: Το προσκήνιο της προσομοίωσης SIR.

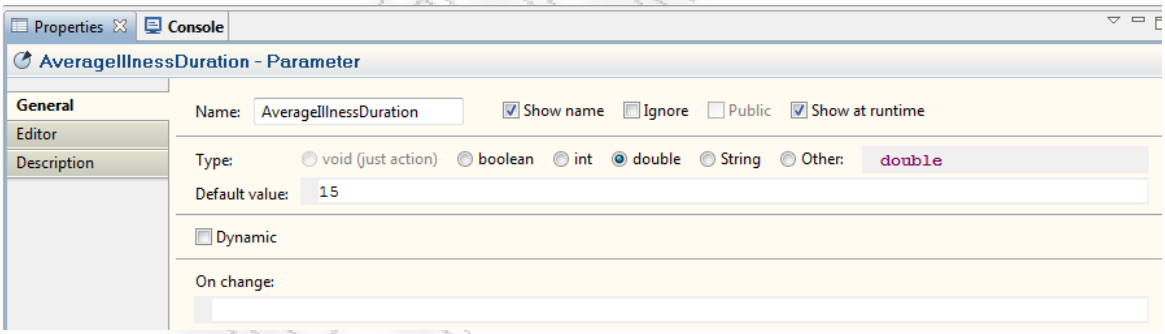
Στην εικόνα 3-2 παρουσιάζονται στο επάνω τμήμα της τα τρία σύνολα στοιχείων δεδομένων και ακριβώς από κάτω τα δύο διαγράμματα καταγραφής των αποτελεσμάτων. Το κάτω μέρος της εικόνας δείχνει τα στοιχεία της ανάλυσης με τη μεθοδολογία της Δυναμικής Συστημάτων. Παρατηρούμε την κίνηση S -> I -> R όπως δείχει η κατεύθυνση των παχέων βελών, τους δύο (2) βρόχους αρνητικής ανάδρασης (τα καμπύλα βέλη με κατεύθυνση από αριστερά προς τα δεξιά επάνω από τα παχιά βέλη) και τον ένα (1) βρόχο θετικής ανάδρασης (το καμπύλο βέλος με φορά από δεξιά προς τα αριστερά). Οι τρεις (3) παράμετροι συνδέονται και επιρρεάζουν τη ροή InfectionRate (η ομάδα των τριών αριστερά) ενώ μια παράμετρος σχετίζεται με τη ροή RecoveryRate.

Η ασθένεια εξαπλώνεται με την επαφή των φορέων της ασθένειας στα ευαίσθητα άτομα του πληθυσμού (ενεργεί ο βρόχος θετικής ανάδρασης) και ταυτόχρονα το πλήθος των ατόμων που πιθανά θα νοσήσουν μειώνεται (βρόχος αρνητικής ανάδρασης). Η θεραπεία από την ασθένεια μειώνει τον πληθυσμό των νοσούντων (βρόχος αρνητικής ανάδρασης). Στη συνέχεια παραθέτουμε στιγμιότυπα από την υλοποίηση του μοντέλου.

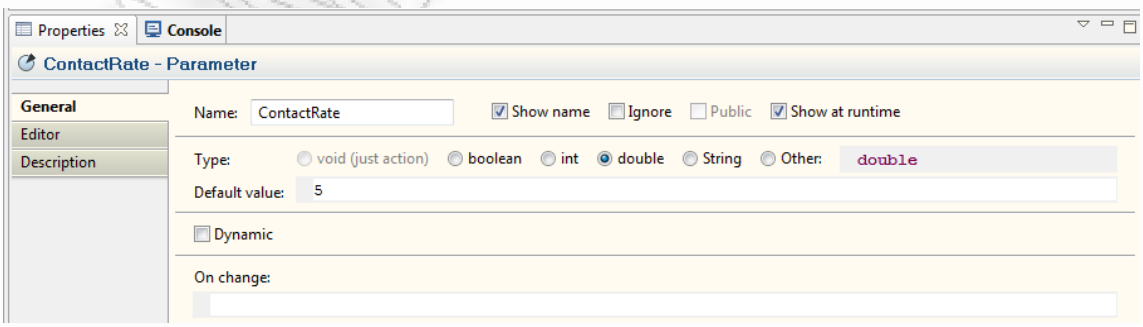


Εικόνα 3-3: Παρουσίαση του παραδείγματος όπως φαίνεται στον συντάκτη του λογισμικού και προβολή των παραμέτρων στην περιοχή των ιδιοτήτων.

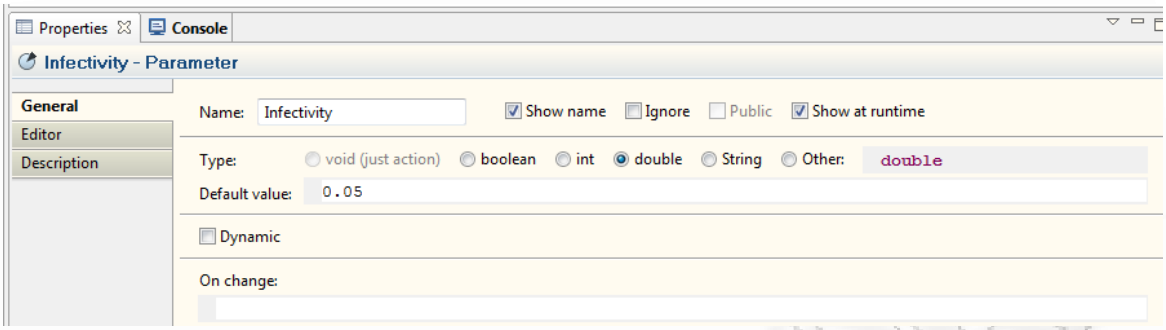
Στην εικόνα 3-3 εμφανίζεται ένα στιγμιότυπο του μοντέλου στο γραφικό περιβάλλον ανάπτυξης του Λογισμικού, ενώ στις εικόνες 3-4(α) έως και 3-4(γ), 3-5(α) έως 3-5(γ) και 3-6(α) έως και 3-6(β) παρουσιάζονται οι γενικές ιδιότητες των στοιχείων, των παραμέτρων, των σημείων συσσώρευσης και των ροών αντίστοιχα, που αποτελούν μέρος της κλάσης Main.



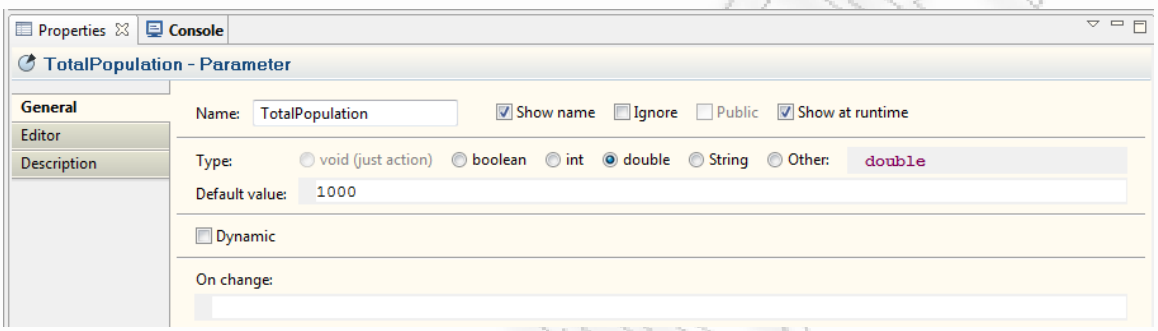
Εικόνα 3-4(α): Γενικές ιδιότητες της παραμέτρου AverageIllnessDuration.



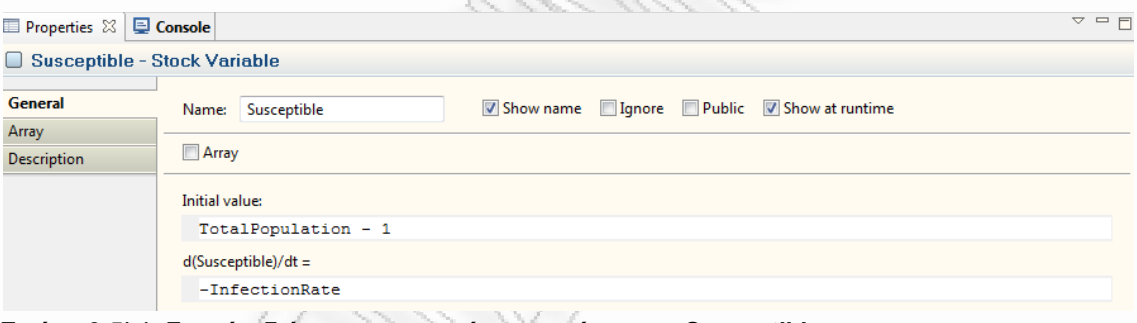
Εικόνα 3-4(β): Γενικές ιδιότητες της παραμέτρου ContactRate.



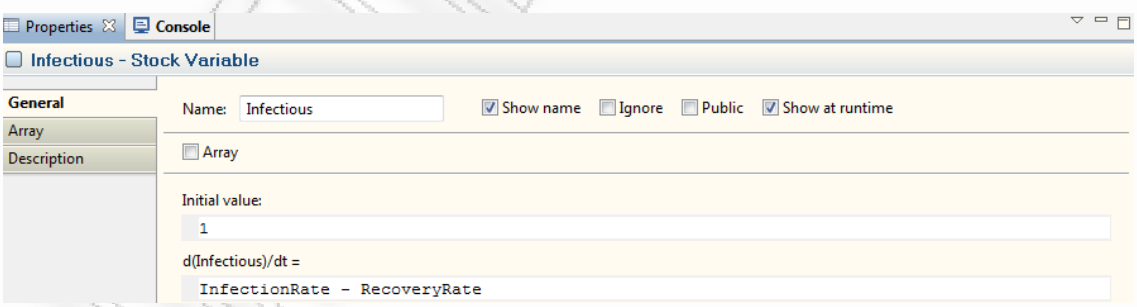
Εικόνα 3-4(γ): Γενικές ιδιότητες της παραμέτρου Infectivity.



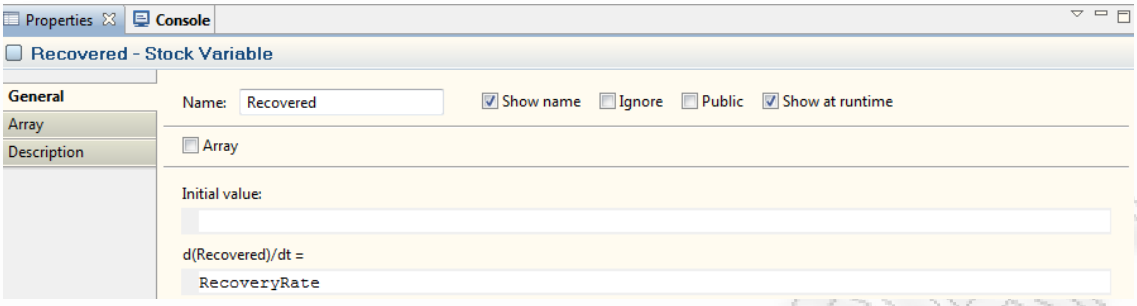
Εικόνα 3-4(δ): Γενικές ιδιότητες της παραμέτρου TotalPopulation.



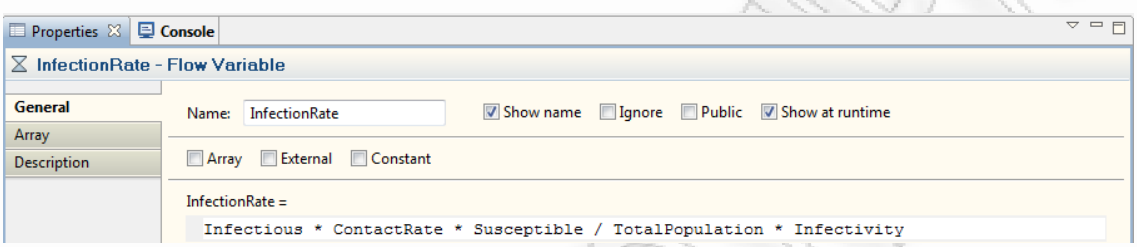
Εικόνα 3-5(α): Γενικές ιδιότητες του σημείου συσσώρευσης Susceptible.



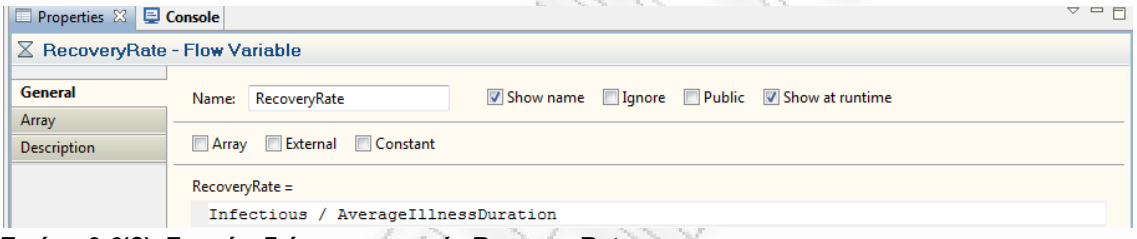
Εικόνα 3-5(β): Γενικές ιδιότητες του σημείου συσσώρευσης Infectious.



Εικόνα 3-5(γ): Γενικές ιδιότητες του σημείου συσσώρευσης Recovered.

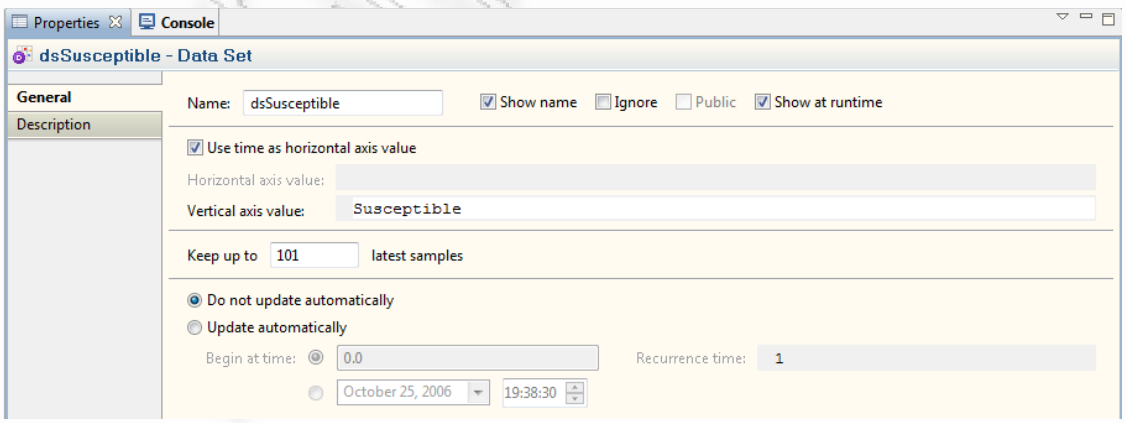


Εικόνα 3-6(α): Γενικές ιδιότητες της ροής InfectionRate.

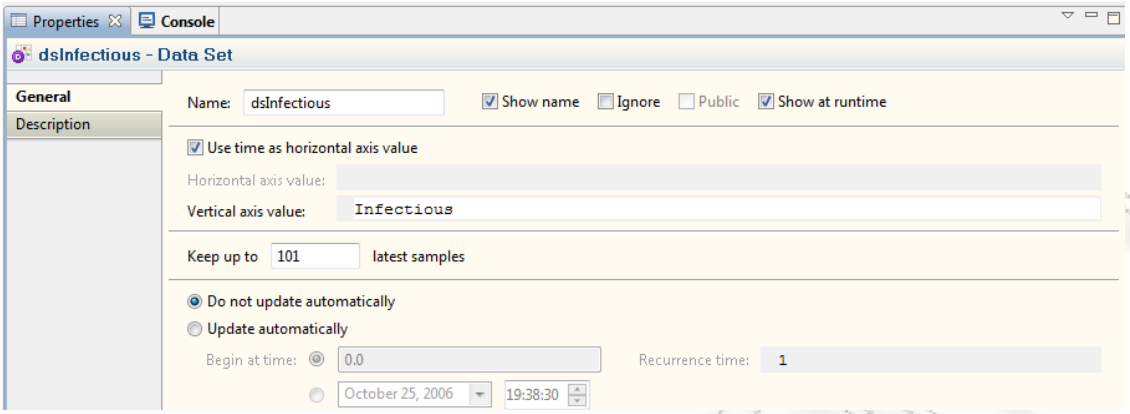


Εικόνα 3-6(β): Γενικές ιδιότητες της ροής RecoveryRate.

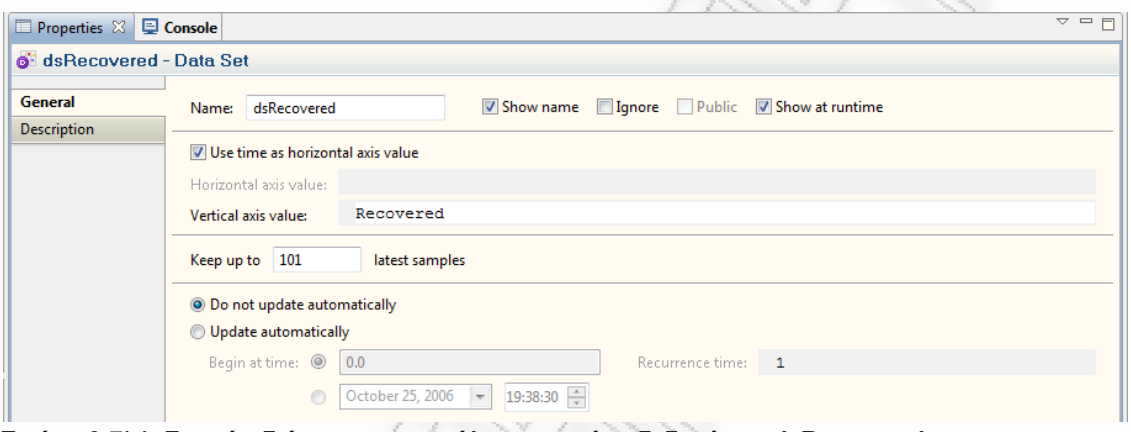
Στις εικόνες 3-7(α) έως και 3-7(γ), 3-8 και 3-9(α) έως και 3-9(γ) παρουσιάζονται οι γενικές ιδιότητες των στοιχείων παρουσίασης όπως αυτά ορίζονται στο μοντέλο. Δίνονται επίσης οι προχωρημένες ιδιότητες και οι ιδιότητες εμφάνισης του χρονικού διαγράμματος του σωρού chartStack. Τα παραπάνω στοιχεία παρουσίασης χρησιμοποιούνται κατά τη διαδικασία της προσομοίωσης του μοντέλου.



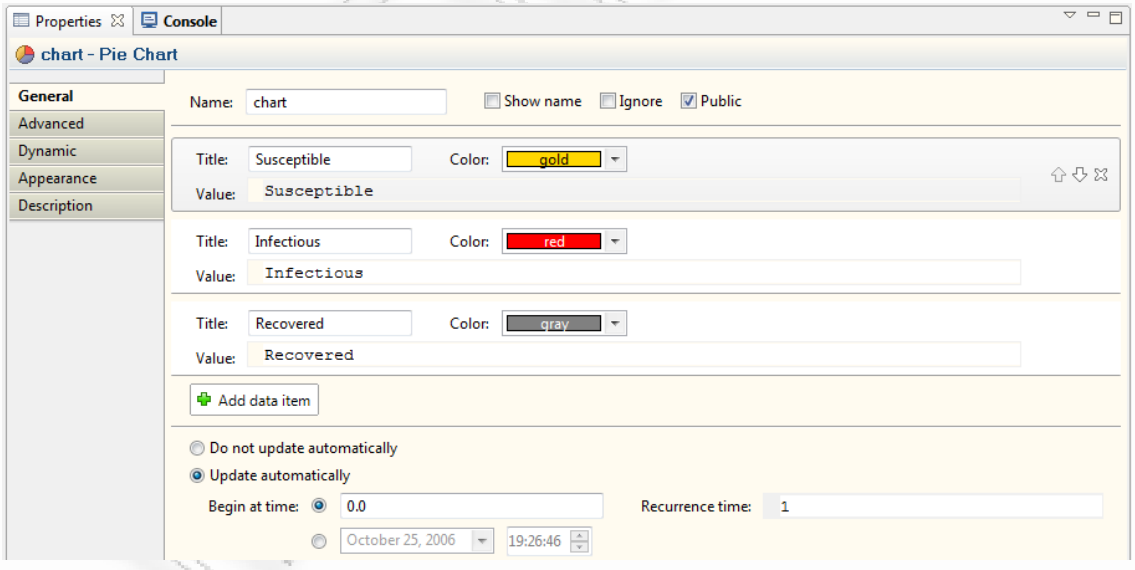
Εικόνα 3-7(α): Γενικές ιδιότητες του συνόλου στοιχείων δεδομένων dsSusceptible.



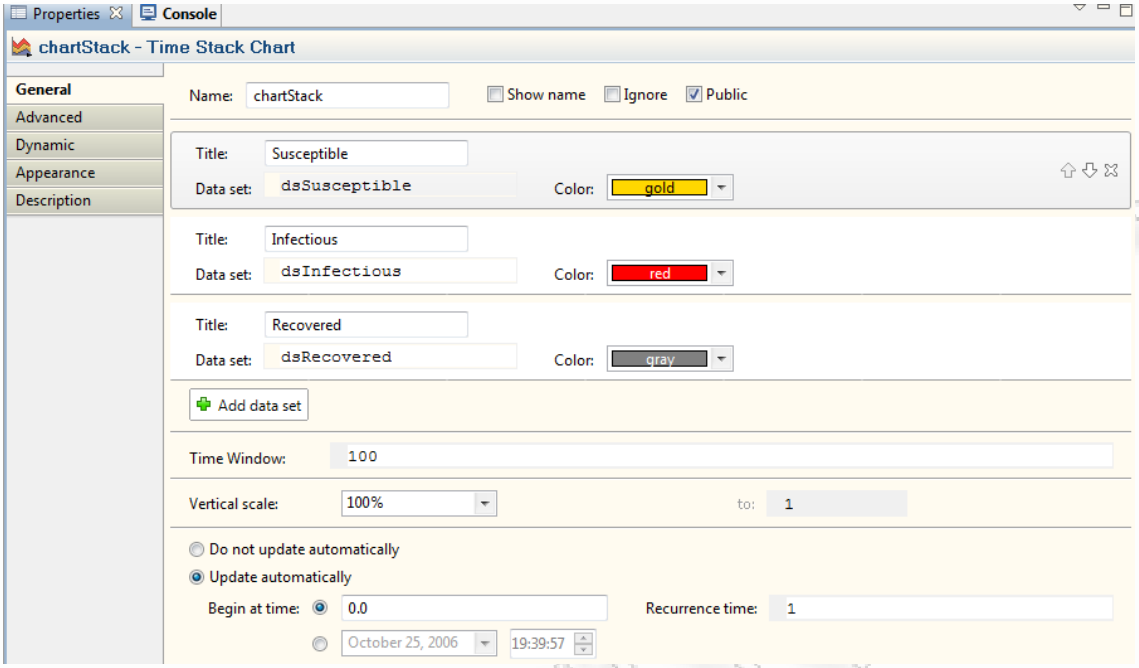
Εικόνα 3-7(β): Γενικές ιδιότητες του συνόλου στοιχείων δεδομένων dsInfectious.



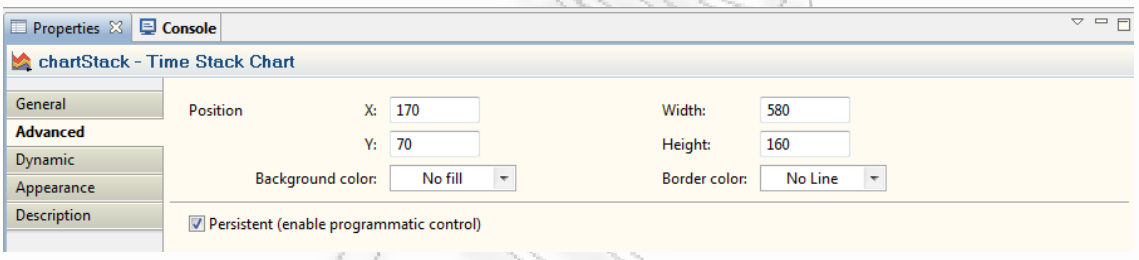
Εικόνα 3-7(γ): Γενικές ιδιότητες του συνόλου στοιχείων δεδομένων dsRecovered.



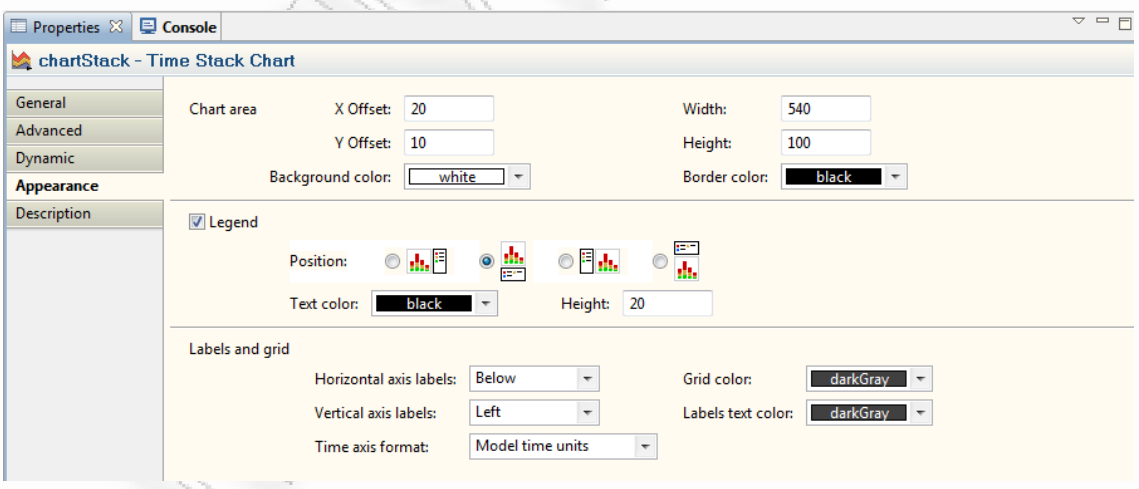
Εικόνα 3-8: Γενικές ιδιότητες του κυκλικού διαγράμματος chart.



Εικόνα 3-9(α): Γενικές ιδιότητες του χρονικού διαγράμματος σωρού chartStack.

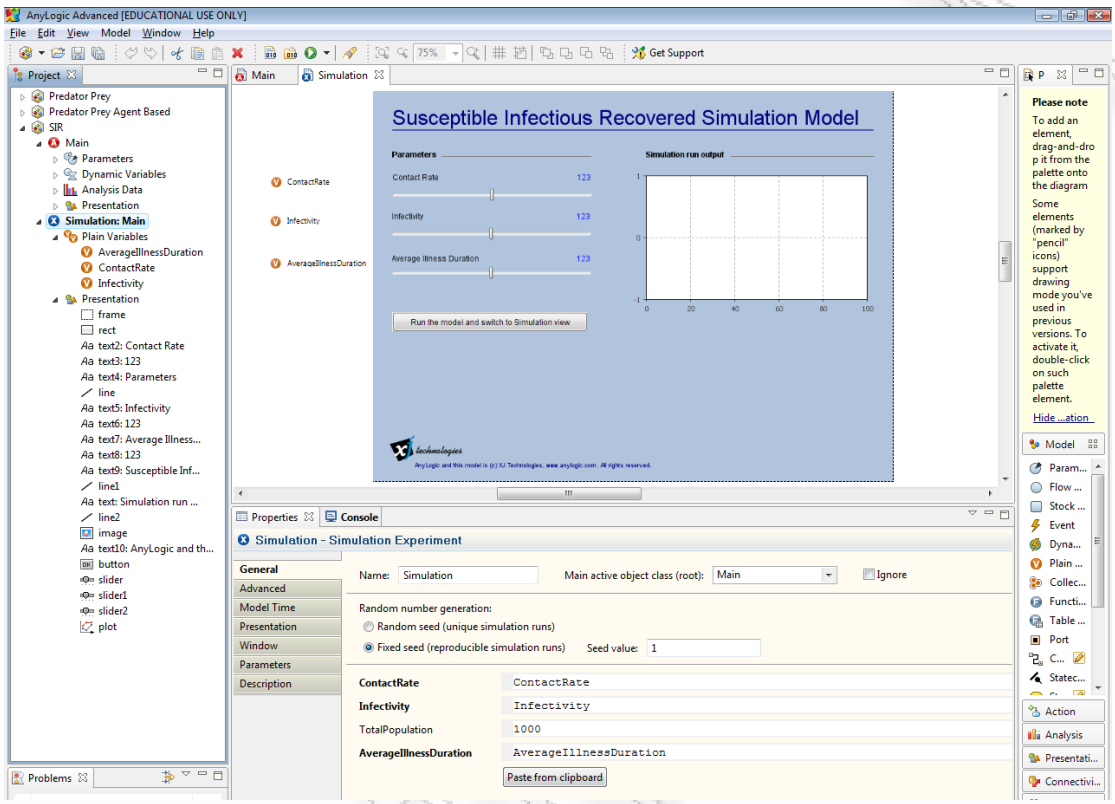


Εικόνα 3-9(β): Προχωρημένες ιδιότητες του χρονικού διαγράμματος σωρού chartStack.



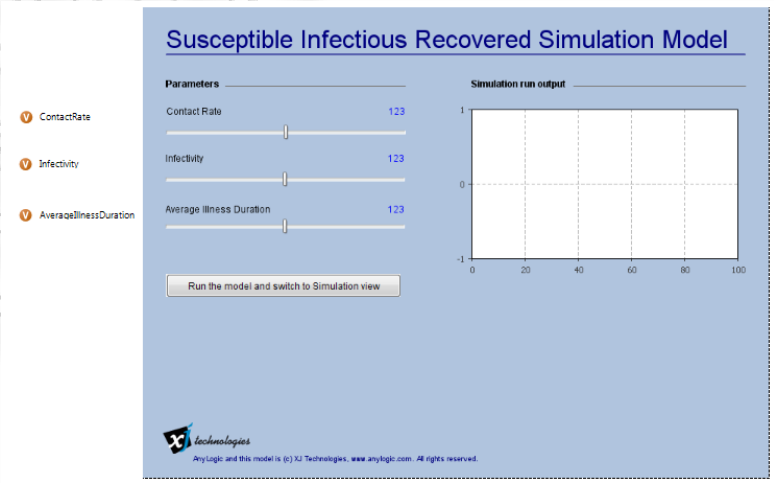
Εικόνα 3-9(γ): Ιδιότητες εμφάνισης του χρονικού διαγράμματος σωρού chartStack.

3.3 Η κλάση προσομοίωσης του έργου, Simulation:Main



Εικόνα 3-10(α): Η παρουσίαση της προσομοίωσης. Η πρώτη εικόνα κατά την προσομοίωση του μοντέλου όπως αυτή εμφανίζεται στο περιβάλλον του Λογισμικού.

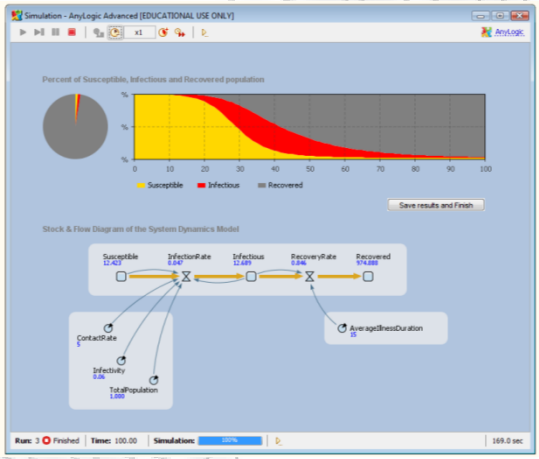
Το γραφικό περιβάλλον της προσομοίωσης, εικόνα 3-10 (α) και (β), αποτελείται από ένα (1) τμήμα κειμένου, ένα (1) ευδιάκριτο λογότυπο, τρεις (3) ράβδους κύλισης όπου γίνεται η αρχικοποίηση των τιμών των παραμέτρων ContactRate, Infectivity και AverageIllnessDuration από τον χρήστη, ένα (1) κομβίο για την έναρξη της προσομοίωσης και ένα (1) διάγραμμα χρόνου (Time Plot), στο οποίο απεικονίζεται το αποτέλεσμα της προσομοίωσης.



Εικόνα 3-10(β): Η πρώτη εικόνα σε μεγέθυνση με ικανοποιητική λεπτομέρεια.

Με την έναρξη της προσομοίωσης αλλάζει το γραφικό περιβάλλον και εμφανίζεται στο προσκήνιο ένα νέο παράθυρο, εικόνα 3-11 (α) και (β), που αποτελείται από δύο (2) τμήματα. Το πρώτο περιέχει ένα (1) κομβίο και δύο (2) ιστογράμματα που στη διάρκεια της προσομοίωσης εμφανίζουν τα επί τοις εκατό (%) ενδιάμεσα αποτελέσματα των τριών καταστάσεων του μοντέλου SIR ενώ το δεύτερο τμήμα παρουσιάζει το υλοποιημένο μοντέλο με τις ενδιάμεσες τιμές των παραμέτρων και των μεταβλητών να εμφανίζονται σε «πραγματικό» χρόνο.

Εικόνα 3-11(α): Στιγμιότυπο κατά την προσομοίωση του μοντέλου.

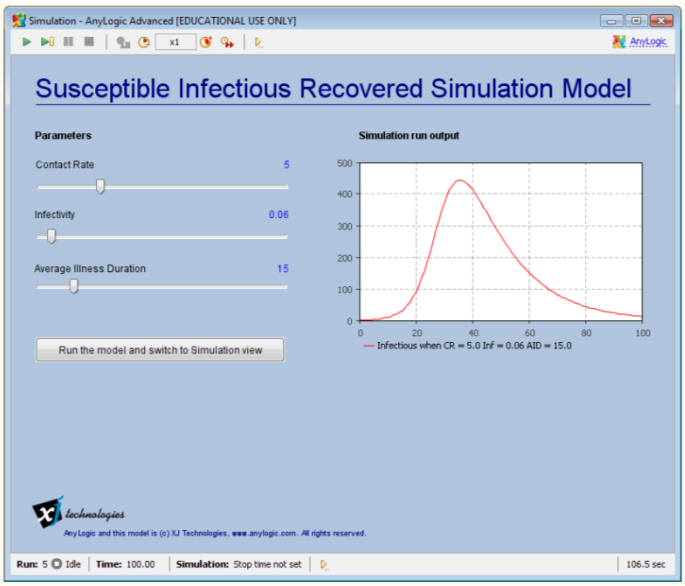


Εικόνα 3-11(β): Στιγμιότυπο μετά την ολοκλήρωση της προσομοίωσης του μοντέλου.

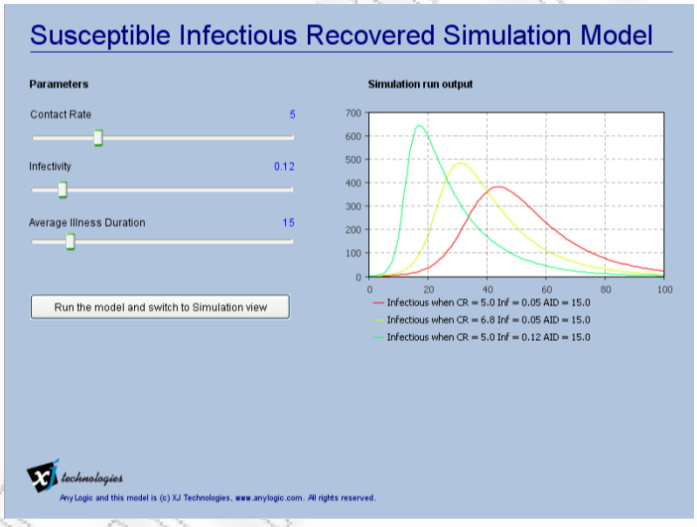
Στο τέλος της προσομοίωσης τα αποτελέσματα αποθηκεύονται με το πάτημα του κομβίου "Save results and Finish" και το γραφικό περιβάλλον αλλάζει στο προηγούμενο παράθυρο όπου εμφανίζεται στο διάγραμμα χρόνου η καμπύλη μεταβολής του σημείου συσσώρευσης Infectious στο χρόνο της προσομοίωσης.

Αλλάζοντας τις τιμές των παραμέτρων μπορούμε να εκτελέσουμε νέα/νέες προσομοιώσεις και να συγκρίνουμε τα εξαγόμενα, εικόνα 3-12 (α) και (β).

Τα διαγράμματα κλάσεων και ο πλήρης κώδικας του παραδείγματος βρίσκονται στο Παράρτημα 1.



Εικόνα 3-12(α): Το παράθυρο της προσομοίωσης μετά από μια (1) εκτέλεση του μοντέλου.



Εικόνα 3-12(β): Το παράθυρο της προσομοίωσης μετά από τρεις (3) εκτελέσεις του μοντέλου με διαφορετικές παραμέτρους.

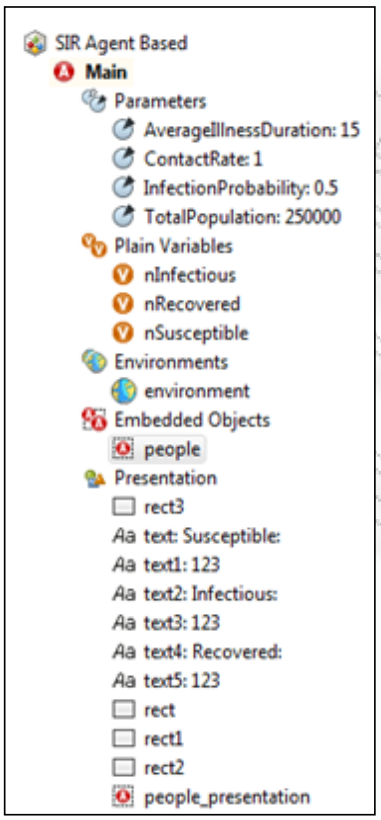
ΚΕΦΑΛΑΙΟ 4

Μοντελοποίηση της διασποράς μιας μολυσματικής νόσου με τη μέθοδο των Πρακτόρων Λογισμικού

Στο κεφάλαιο αυτό θα συνεχίσουμε την ανάλυση του μοντέλου SIR και θα επεκταθούμε στην υλοποίηση του μέσω της χρήσης πρακτόρων λογισμικού. Οι πράκτορες λογισμικού αναλαμβάνουν τον κυρίαρχο ρόλο σε αυτό το παράδειγμα προσομοίωσης, μπορούν δε, να βρίσκονται σε μια από τις τρεις, (όπως ορίστηκαν στο προηγούμενο κεφάλαιο), καταστάσεις (α) Susceptible, (β) Infectious και (γ) Recovered.

4.1 Η κλάση Main

Ο πράκτορας λογισμικού κινείται σε ένα οριοθετημένο περιβάλλον (environment) που έχει χαρακτηριστικά κλειστού συστήματος. Μέσα σε αυτό το χώρο, κάθε πράκτορας (άτομο) αλληλεπιδρά με τους γειτονικούς του, (άλλοι πράκτορες), οι οποίοι με τη σειρά τους βρίσκονται σε μια από τις επιτρεπτές καταστάσεις. Η ασθένεια μεταδίδεται με συγκεκριμένη πιθανότητα. Παρουσιάζεται γραφικά η διάχυση της ασθένειας στον οριοθετημένο χώρο. Εδώ να παρατηρήσουμε ότι οι πράκτορες αλληλεπιδρούν με τους άμεσα γειτονικούς τους και γραφικά, η διάχυση, η εξάπλωση της ασθένειας εμφανίζεται με τη μορφή ενός μετακινούμενου κύματος στα όρια του προς εξέταση χώρου.



Εικόνα 4-1: Η ιεραρχία του μοντέλου μετά την υλοποίηση του με το Λογισμικό.

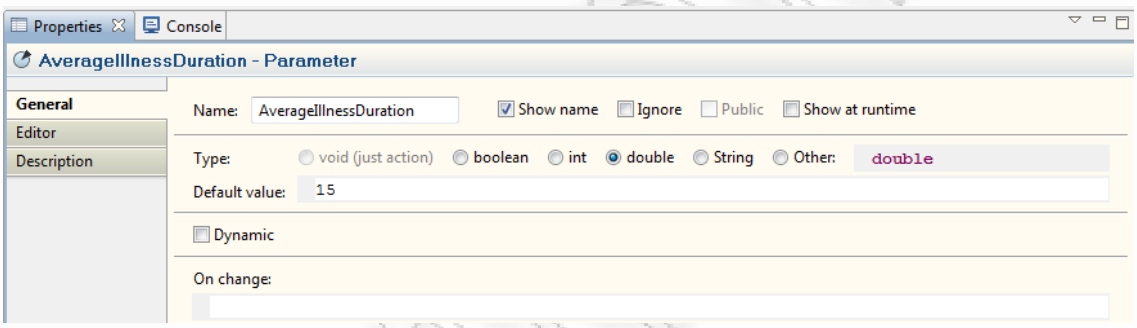
Η ιεραρχική δομή του παραδείγματος, του έργου, SIR Agent Based, όπως παρουσιάζεται στην εικόνα 4-1, έχει υλοποιηθεί κάτω από το έργο SIR Agent Based. Αποτελείται από την κλάση Main, την κλάση Person και την κλάση Simulation: Main.

Η δενδρική δομή της κλάσης Main αποτελείται από τέσσερις (4) παραμέτρους, τρεις (3) μεταβλητές, ένα (1) πρακτορικό περιβάλλον, ένα (1) ενσωματωμένο αντικείμενο και το τμήμα των γραφικών με τα στοιχεία που το αποτελούν. Συγκεκριμένα, οι παράμετροι της κλάσης, (Πίνακας 4-1), είναι οι AverageIllnessDuration, (ορίζεται η μέση χρονική διάρκεια σε ημέρες της ασθένειας), ContactRate, (ο παράγοντας δείκτης με τον οποίο ελέγχουμε αν η ασθένεια είναι επιδημία ή όχι), InfectionProbability, (η πιθανότητα που έχει ένα άτομο να νοσήσει από την ασθένεια) και TotalPopulation, (ο συνολικός πληθυσμός για τον οποίο θα γίνει η προσομοίωση). Όλοι οι παραπάνω παράμετροι, (εκτός από τον TotalPopulation που έχει ορισθεί τύπου int), είναι τύπου double. Οι μεταβλητές nInfectious, nRecovered και nSusceptible είναι τύπου double και εκφράζουν τον αριθμό των πρακτόρων (ατόμων) που έχουν μολυνθεί, που έχουν ιαθεί και που είναι υποψήφια να νοσήσουν αντίστοιχα. Στα επόμενα, εικόνες 4-2 έως και 4-4, παρουσιάζονται στιγμιότυπα από το Λογισμικό κατά την πορεία της υλοποίησης του μοντέλου.

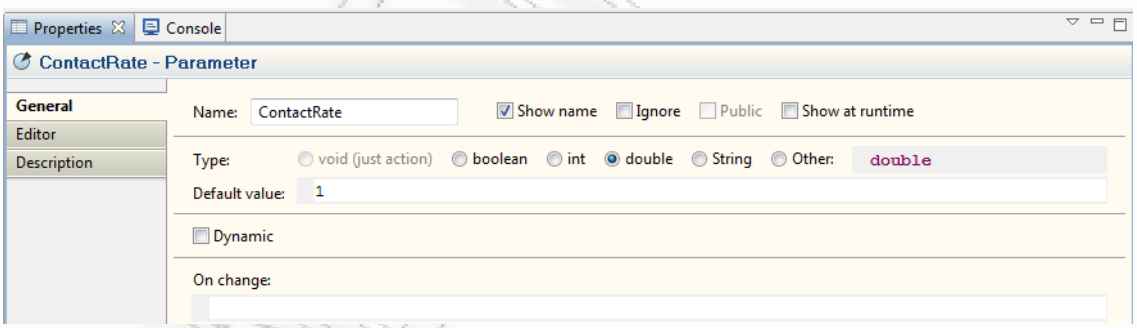
Α. Παράμετροι

Παράμετροι της κλάσης Main			
Όνομα παραμέτρου	Εξήγηση παραμέτρου	Τύπος παραμέτρου	Αρχική τιμή παραμέτρου
TotalPopulation	Ο συνολικός πληθυσμός προς εξέταση	Double	250000
InfectionProbability	Πιθανότητα Μόλυνσης	Double	0.5
ContactRate	Ο ρυθμός επαφής του πληθυσμού	Double	1
AverageIllnessDuration	Η μέση χρονική διάρκεια της ασθένειας	Double	15

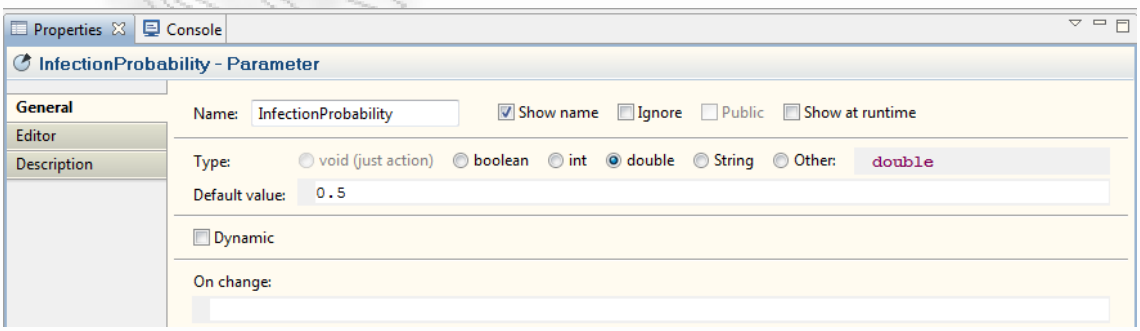
Πίνακας 4-1: Παράμετροι της κλάσης Main.



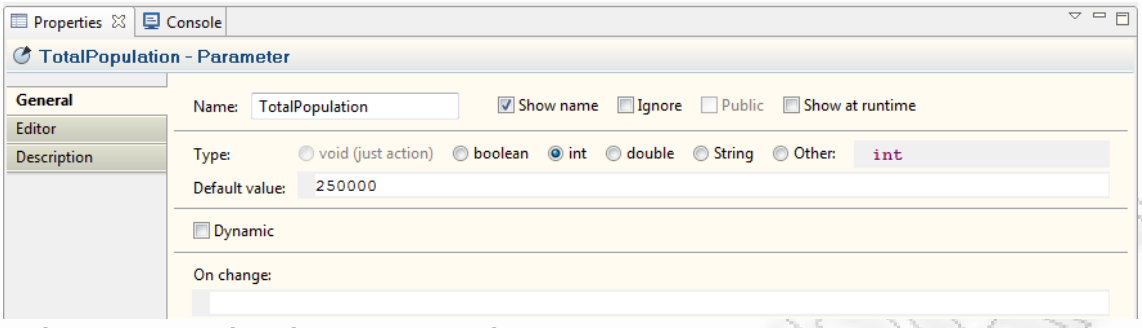
Εικόνα 4-2(α): Γενικές ιδιότητες της παραμέτρου AverageIllnessDuration.



Εικόνα 4-2(β): Γενικές ιδιότητες της παραμέτρου ContactRate.

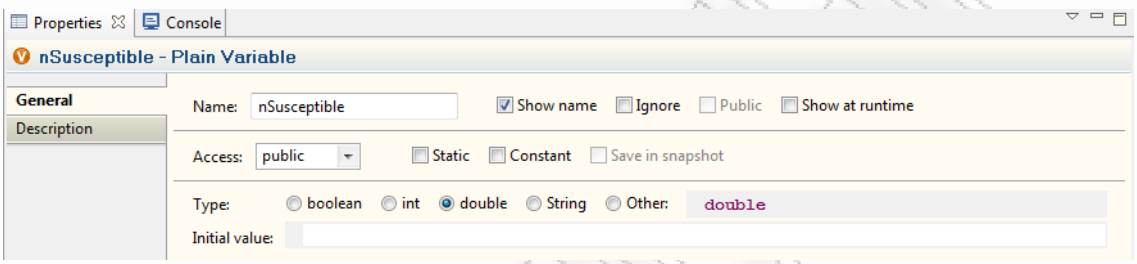


Εικόνα 4-2(γ): Γενικές ιδιότητες της παραμέτρου InfectionProbability.

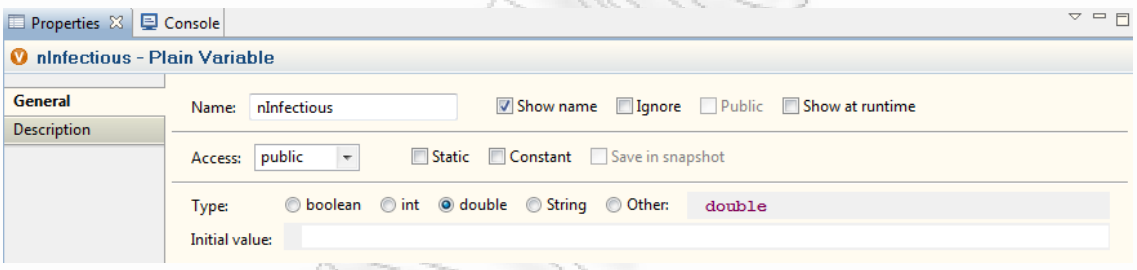


Εικόνα 4-2(δ): Γενικές ιδιότητες της παραμέτρου TotalPopulation.

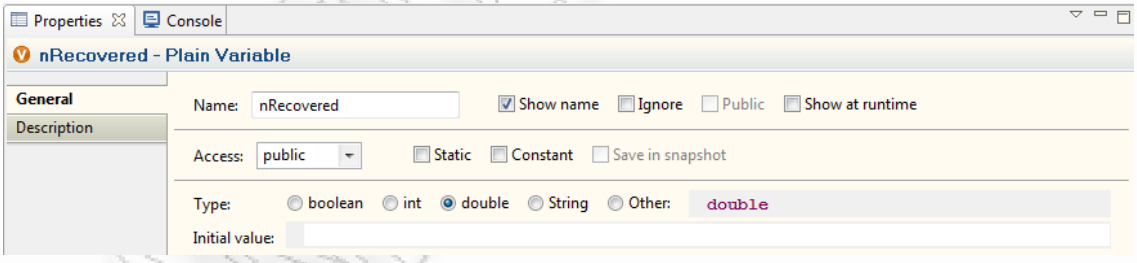
Β. Μεταβλητές



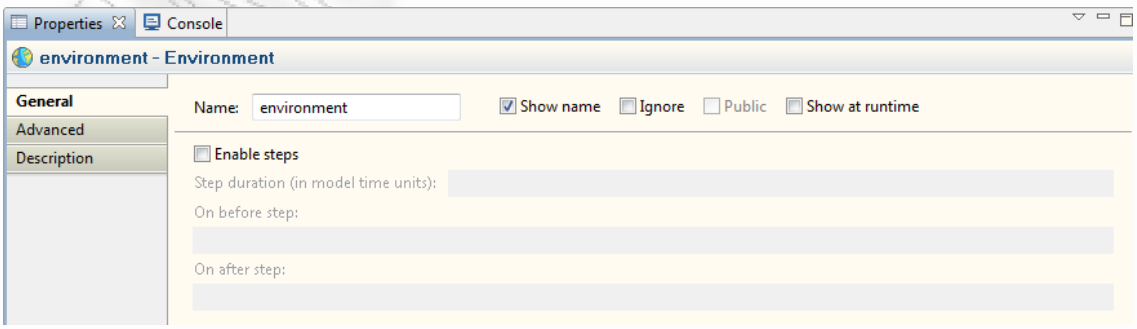
Εικόνα 4-3(α): Γενικές ιδιότητες της απλής μεταβλητής nSusceptible.



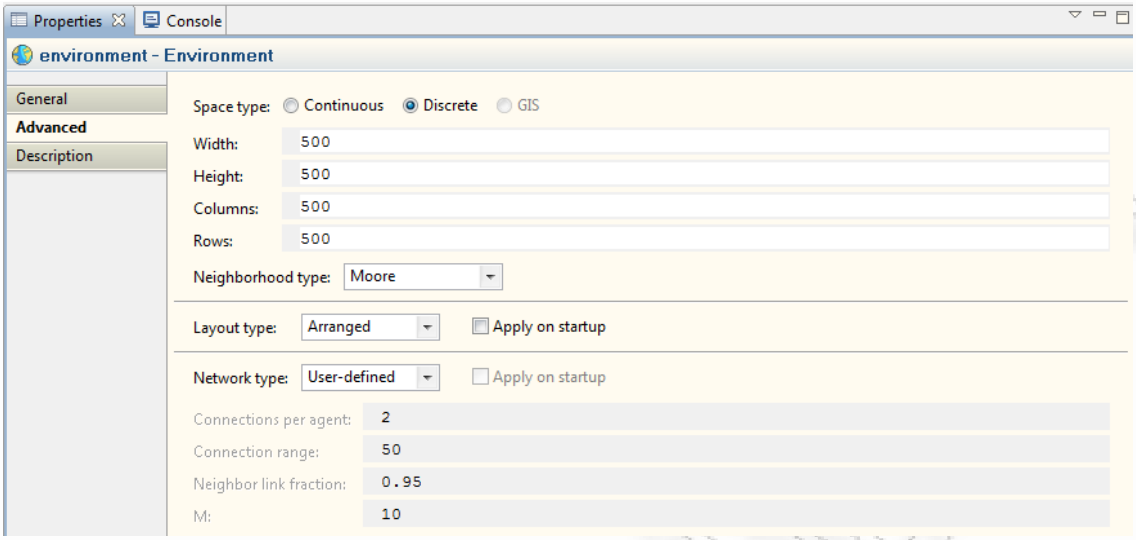
Εικόνα 4-3(β): Γενικές ιδιότητες της απλής μεταβλητής nInfectious.



Εικόνα 4-3(γ): Γενικές ιδιότητες της απλής μεταβλητής nRecovered.

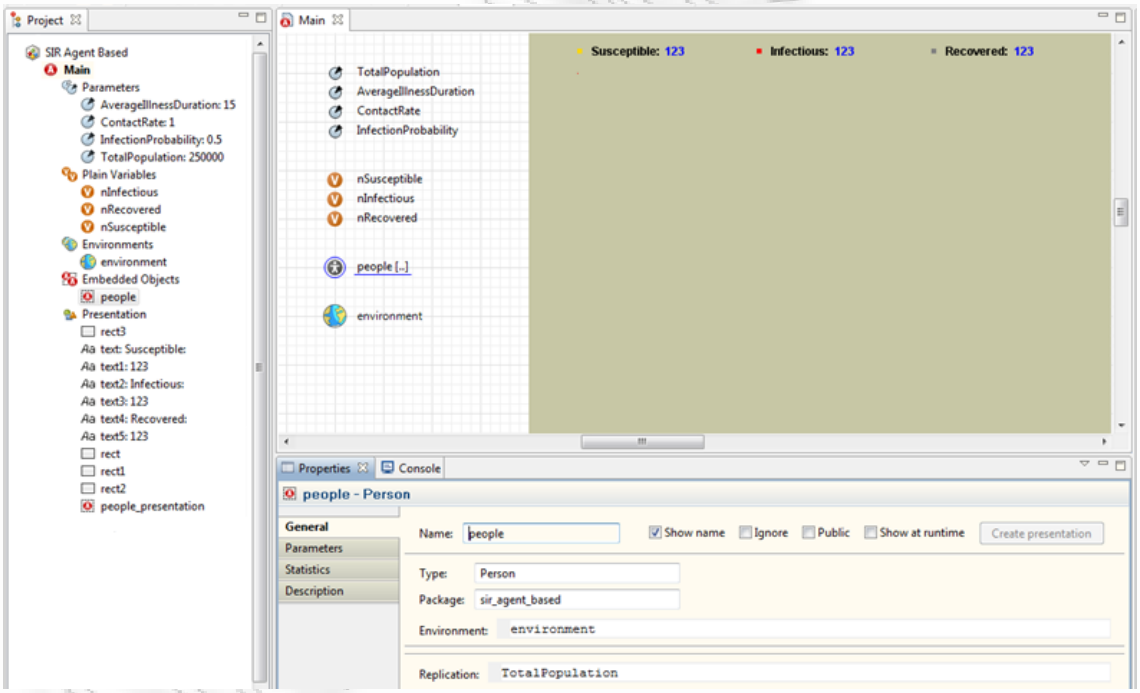


Εικόνα 4-4(α): Γενικές ιδιότητες του περιβάλλοντος environment.



Εικόνα 4-4(β): Προχωρημένες ιδιότητες του περιβάλλοντος environment.

Στο Λογισμικό, το πρακτορικό περιβάλλον (Environment) είναι μια ειδική κατασκευή με την οποία: (α) ορίζονται κοινές ιδιότητες σε ένα σύνολο πρακτόρων και (β) επιτρέπεται η πρόσβαση σε όλους τους εγγεγραμμένους πράκτορες με ομοιόμορφο τρόπο χωρίς να ενδιαφέρει αν ανήκουν (αν είναι στιγμιότυπο) στο ίδιο ενεργό αντικείμενο.

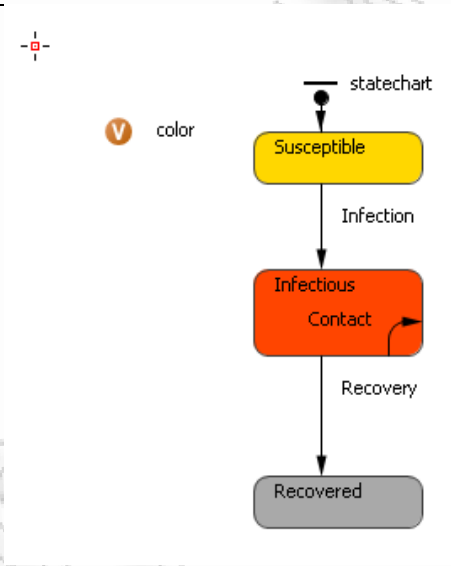


Εικόνα 4-5: Το ενσωματωμένο αντικείμενο people τύπου Person.

Με το ενσωματωμένο αντικείμενο people ορίζεται κάθε στιγμιότυπο της κλάσης Person. Το εικονίδιο που υποδεικνύει στο περιβάλλον του Λογισμικού το ενσωματωμένο αντικείμενο, είναι το  . Επίσης, το συγκεκριμένο εικονίδιο δηλώνει ότι το αντικείμενο δεν περιέχει στοιχεία public. Συσχετίζεται με τη μεταβλητή TotalPopulation και επενεργεί στο στοιχείο περιβάλλον, environment. Κατά την προσομοίωση κάθε στιγμιότυπο έχει το σχήμα ενός μικρού τετραγώνου, όπως φαίνεται στην εικόνα 4-5.

4.2 Η κλάση Person

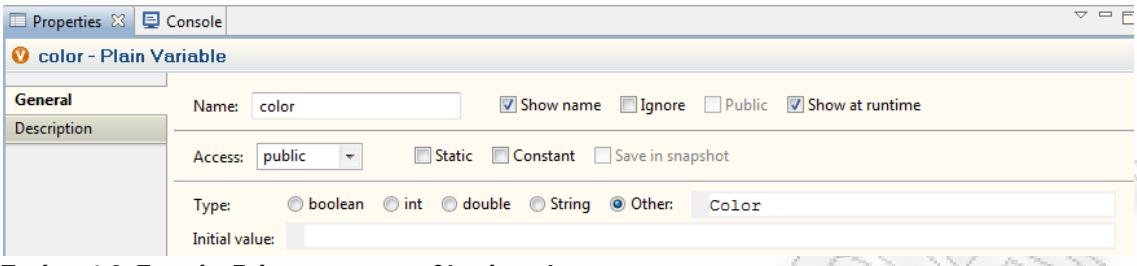
Εικόνα 4-6: Η ιεραρχική δομή της κλάσης (του πράκτορα) Person.



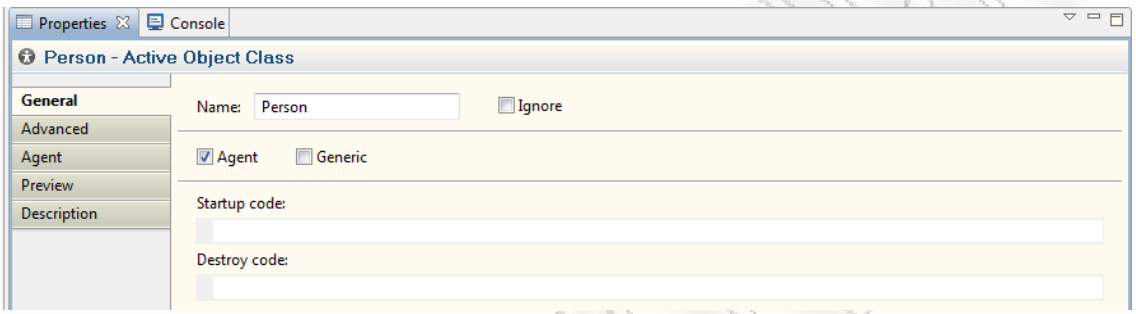
Εικόνα 4-7: Το διάγραμμα καταστάσεων της κλάσης Person.

Η ενεργή κλάση Person που δημιουργεί και ορίζει τον πράκτορα λογισμικού, Εικόνα 4-6, είναι υποκλάση της κλάσης Agent. Η κλάση Agent είναι η δομική μονάδα του Λογισμικού για την μοντελοποίηση με τη χρήση πρακτόρων λογισμικού. Η ενεργή κλάση Person αποτελείται από την μεταβλητή color τύπου Color, το διάγραμμα καταστάσεων statechart τύπου Statechart και τα αντικείμενα παρουσίασης τύπου Presentation. Παρακάτω, παρουσιάζονται ένα προς ένα.

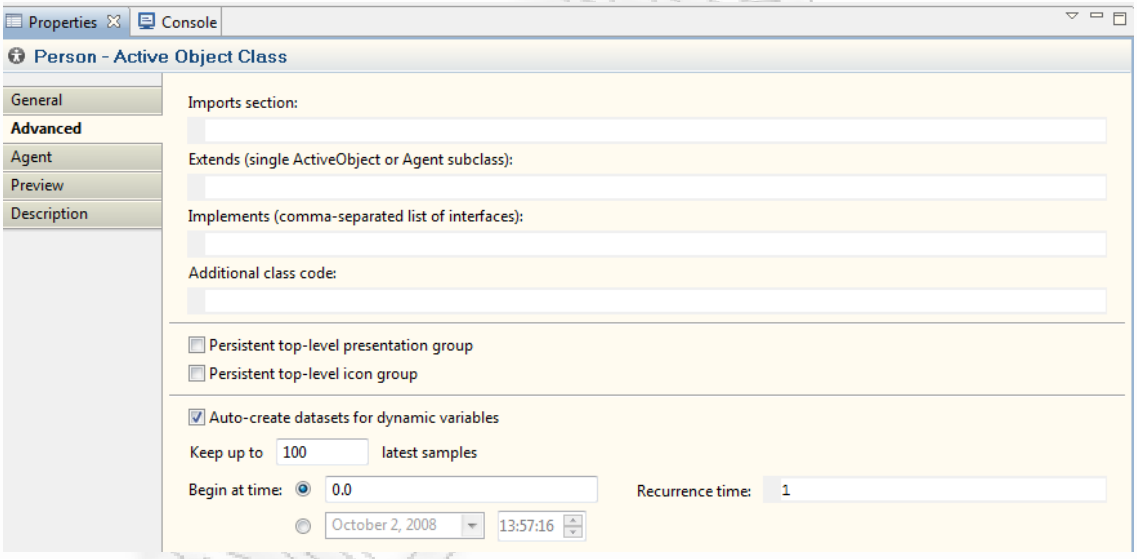
Το διάγραμμα καταστάσεων που υλοποιεί την κλάση Person, Εικόνα 4-7, έχει σαν αφετηρία την κατάσταση Susceptible, η οποία λόγω της μεταβλητής color έχει χρώμα χρυσό. Η μετάβαση Infection αλλάζει την κατάσταση του πράκτορα στην κατάσταση Infectious όταν δοθεί το έναυσμα, (μέσω του εσωτερικού μηχανισμού μεταφοράς μηνυμάτων του λογισμικού από έναν πράκτορα σε έναν άλλο ή και από το ίδιο τον πράκτορα στον εαυτό του), κατά τη διάρκεια της προσομοίωσης. Ο πράκτορας όταν βρίσκεται στην κατάσταση Infectious έχει κόκκινο χρώμα. Η μετάβαση Contact εξαρτάται από τις τιμές των παραμέτρων του μοντέλου, (είναι ανάλογη της μεταβλητής ContactRate και αντιστρόφως ανάλογη της μεταβλητής InfectionProbability) και των τιμών των παραμέτρων του περιβάλλοντος που σχετίζονται με τον τρόπο επαφής των στιγμιοτύπων των πρακτόρων. Η μετάβαση Contact αλλάζει την κατάσταση των πρακτόρων από Susceptible σε Infectious. Μόλις συντρέξουν οι συνθήκες, μέσω της μετάβασης Recovery, (είναι αντιστρόφως ανάλογη της μεταβλητής AverageIllnessDuration), ο πράκτορας μεταβαίνει στην κατάσταση Recovered και αποκτά μέσω της μεταβλητής color χρώμα γκρι. Ο πράκτορας αρχικά βρίσκεται στην κατάσταση Susceptible. Στην εικόνα 4-9, (α) έως και (γ), παρουσιάζονται οι ιδιότητες της κλάσης Person ενώ στην εικόνα 4-10, (α) έως και (ζ), παρουσιάζονται τα δομικά στοιχεία που συνθέτουν το διάγραμμα καταστάσεων και οι ιδιότητες τους κατά στοιχείο.



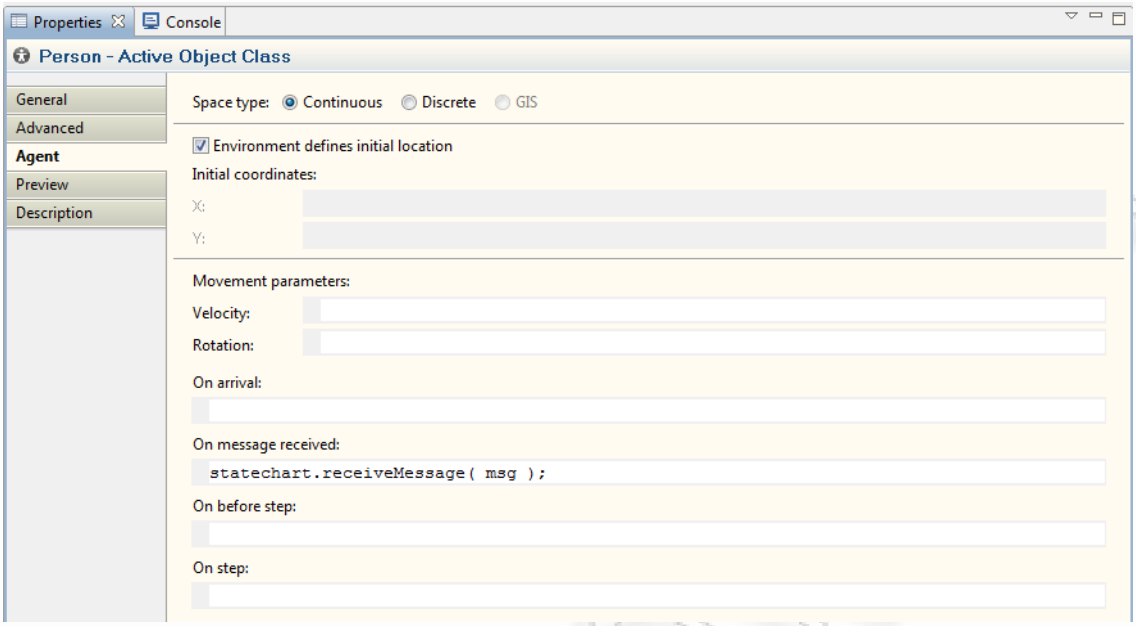
Εικόνα 4-8: Γενικές ιδιότητες της μεταβλητής color.



Πίνακας 4-9(α): Γενικές ιδιότητες της κλάσης Person. Η κλάση είναι υποκλάση της κλάσης Agent που είναι δομική μονάδα του λογισμικού.

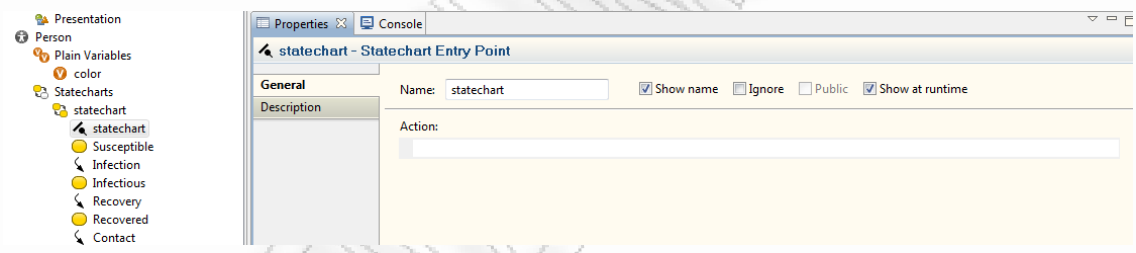


Πίνακας 4-9(β): Προχωρημένες ιδιότητες της κλάσης Person.

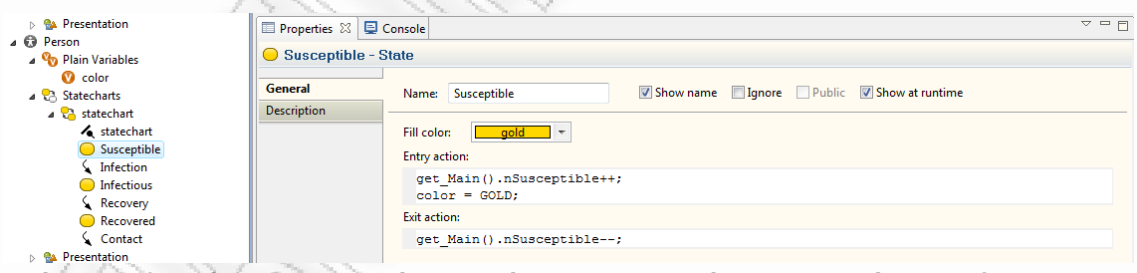


Πίνακας 4-9(γ): Ιδιότητες του πράκτορα της κλάσης Person. Μόλις ο πράκτορας λάβει το μήνυμα msg θα εκτελέσει τον κώδικα που τον συνδέει με το διάγραμμα καταστάσεων statechart.

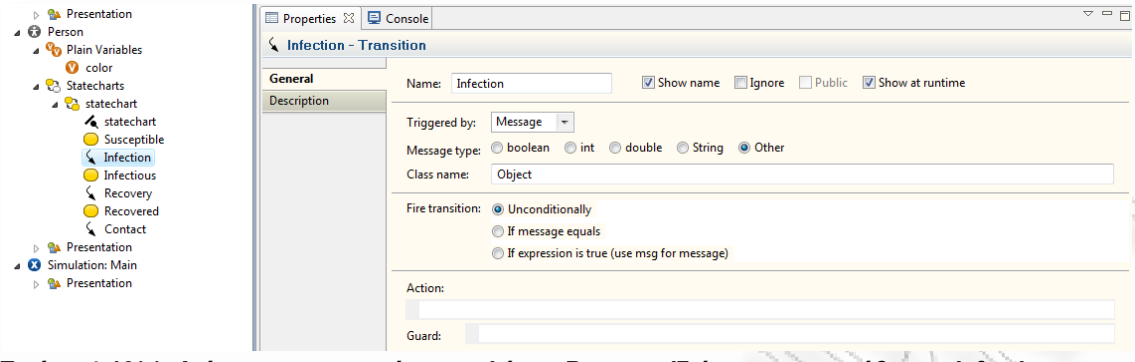
4.2.1 Παρουσίαση των στοιχείων που αποτελούν το διάγραμμα καταστάσεων της κλάσης Person.



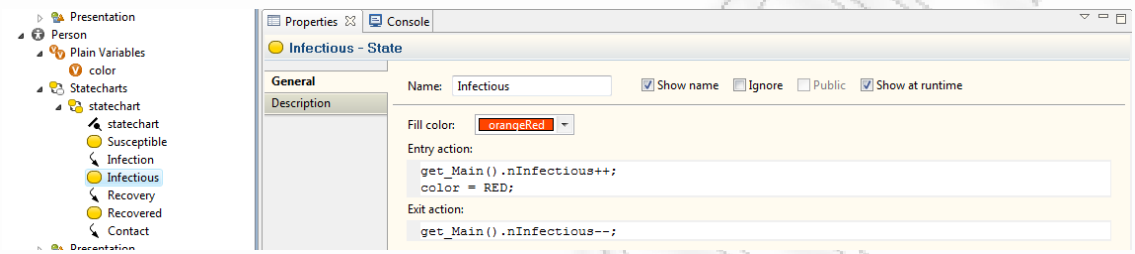
Εικόνα 4-10(α): Διάγραμμα καταστάσεων κλάσης Person – Αρχή.



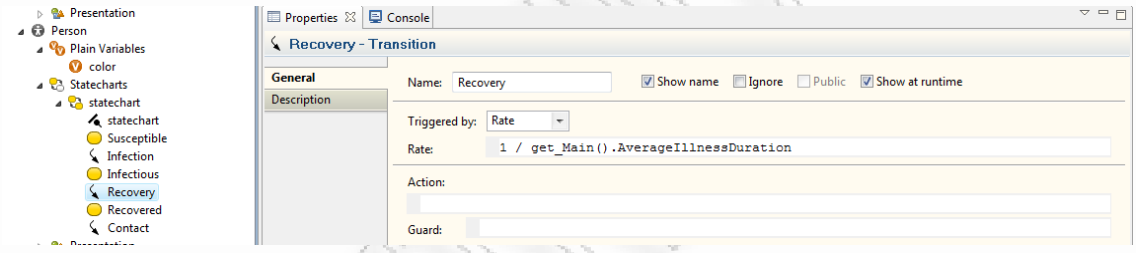
Εικόνα 4-10(β): Διάγραμμα καταστάσεων κλάσης Person – Ιδιότητες της κατάστασης Susceptible.



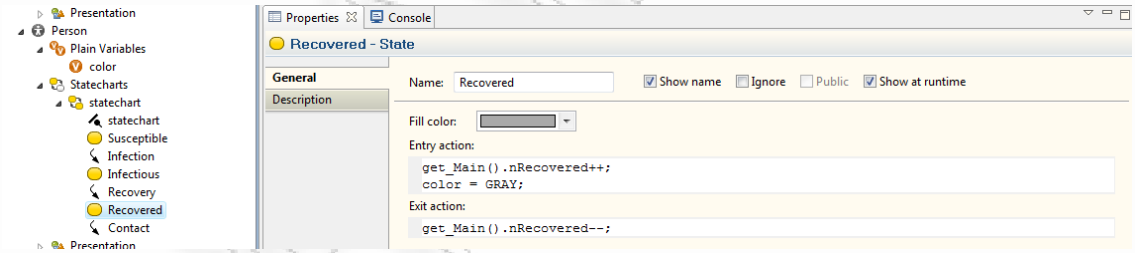
Εικόνα 4-10(γ): Διάγραμμα καταστάσεων κλάσης Person - Ιδιότητες της μετάβασης Infection.



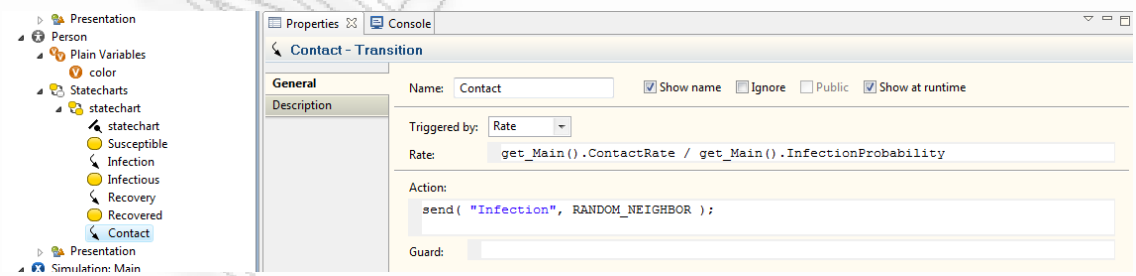
Εικόνα 4-10(δ): Διάγραμμα καταστάσεων κλάσης Person – Ιδιότητες της κατάστασης Infectious.



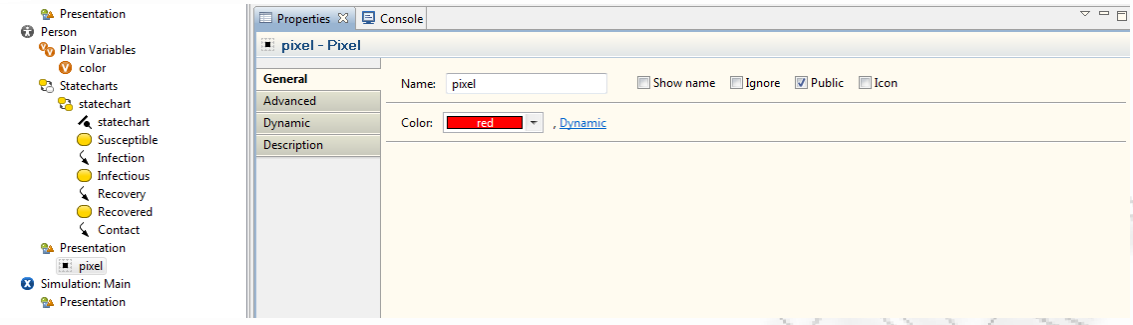
Εικόνα 4-10(ε): Διάγραμμα καταστάσεων κλάσης Person - Ιδιότητες της μετάβασης Recovery.



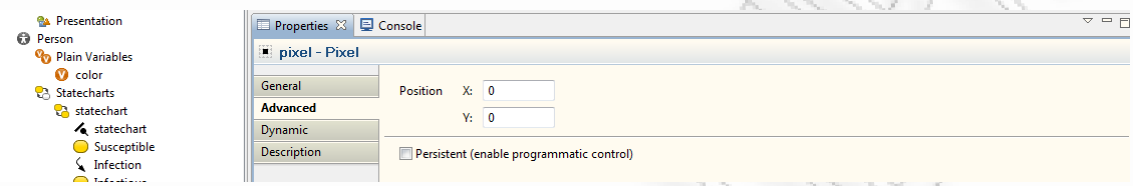
Εικόνα 4-10(στ): Διάγραμμα καταστάσεων κλάσης Person – Ιδιότητες της κατάστασης Recovered.



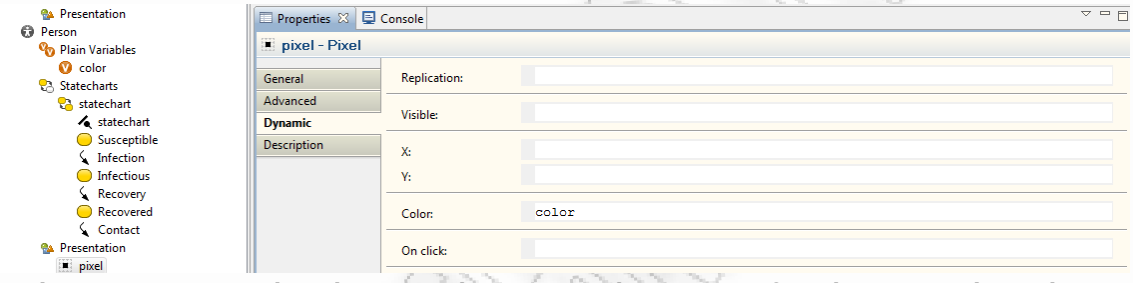
Εικόνα 4-10(ζ): Διάγραμμα καταστάσεων κλάσης Person - Ιδιότητες της μετάβασης Contact.



Εικόνα 4-11(α): Γενικές ιδιότητες εμφάνισης της κλάσης Person.



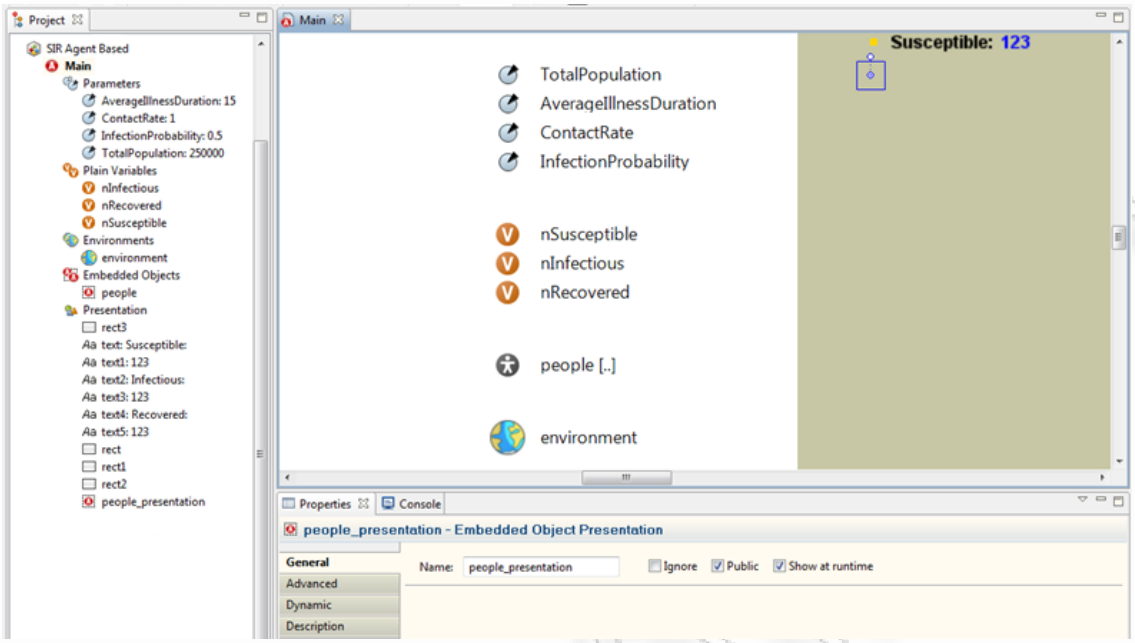
Εικόνα 4-11(β): Προχωρημένες ιδιότητες εμφάνισης της κλάσης Person.



Εικόνα 4-11(γ): Δυναμικές ιδιότητες εμφάνισης της κλάσης Person. Ο πράκτορας κατά τη διάρκεια της προσομοίωσης αλλάζει χρώμα σύμφωνα με την τιμή της μεταβλητής color.

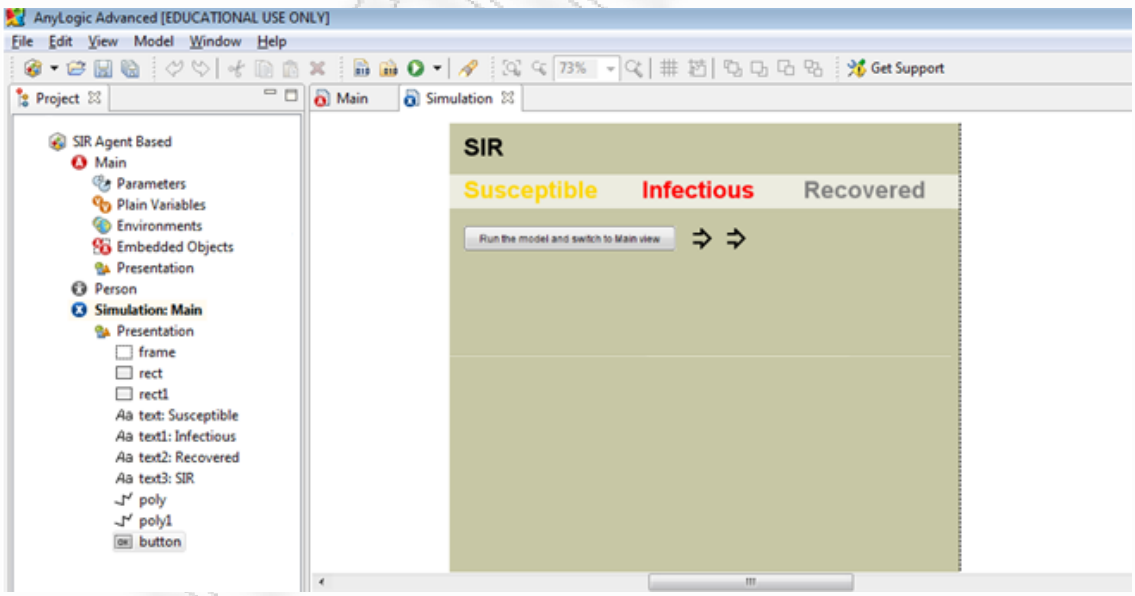
4.3 Η κλάση προσομοίωσης του έργου, Simulation:Main

Το τελευταίο στάδιο που υπολείπεται να υλοποιηθεί, μετά την ανάλυση και τη σχεδίαση της διαδικασίας μοντελοποίησης των κλάσεων του παραδείγματος, για την ολοκλήρωση της προσομοίωσης είναι το γραφικό περιβάλλον στο οποίο θα εμφανιστούν τα αποτελέσματα της εκτέλεσης της προσομοίωσης. Ο πράκτορας λογισμικού σύμφωνα με τις Εικόνες 4-11(α) έως και 4-11(γ) και 4-12 έχει κατά την προσομοίωση ένα απλό σχήμα τύπου Pixel με το οποίο εμφανίζεται στο γραφικό περιβάλλον προσομοίωσης.

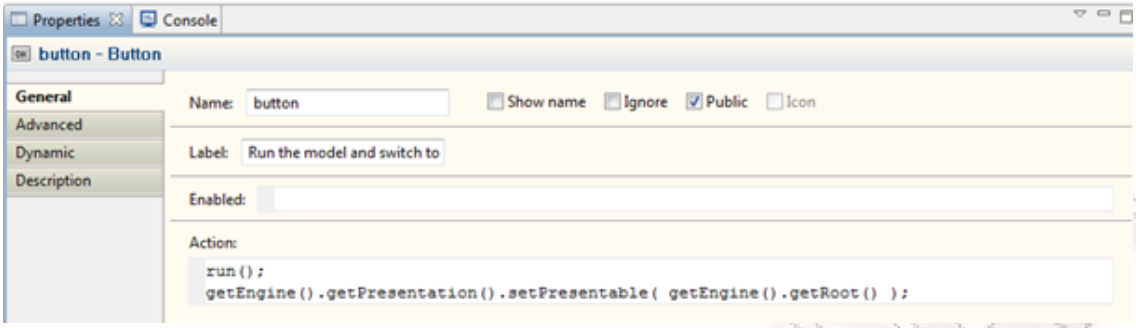


Εικόνα 4-12: Γενικές ιδιότητες παρουσίασης για το ενσωματωμένο αντικείμενο people και η εμφάνιση του στην προσομοίωση.

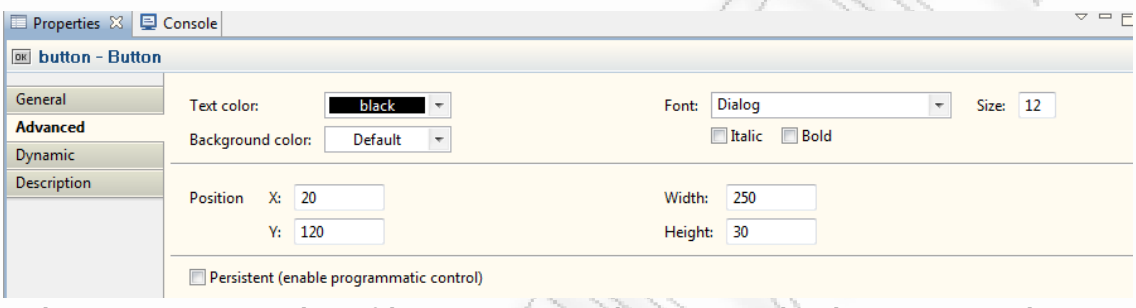
Η αρχική εικόνα του γραφικού περιβάλλοντος της προσομοίωσης παρουσιάζεται στην Εικόνα 4-13, (α) έως και (γ). Όπως φαίνεται, είναι αρκετά λιτή, ένα πολύ απλό πλαίσιο που περιέχει μερικές λέξεις κειμένου, δύο (2) τόξα και ένα κομβίο. Χρησιμοποιείται για την ενσωμάτωση του αντικείμενου ελέγχου «κόμβιο, (Button)» στην προσομοίωση. Μόλις το πατάμε, ξεκινά η προσομοίωση και μεταφερόμαστε στο καθεαυτό γραφικό περιβάλλον της προσομοίωσης.



Εικόνα 4-13(α): Το γραφικό περιβάλλον της προσομοίωσης – Αρχική εικόνα.

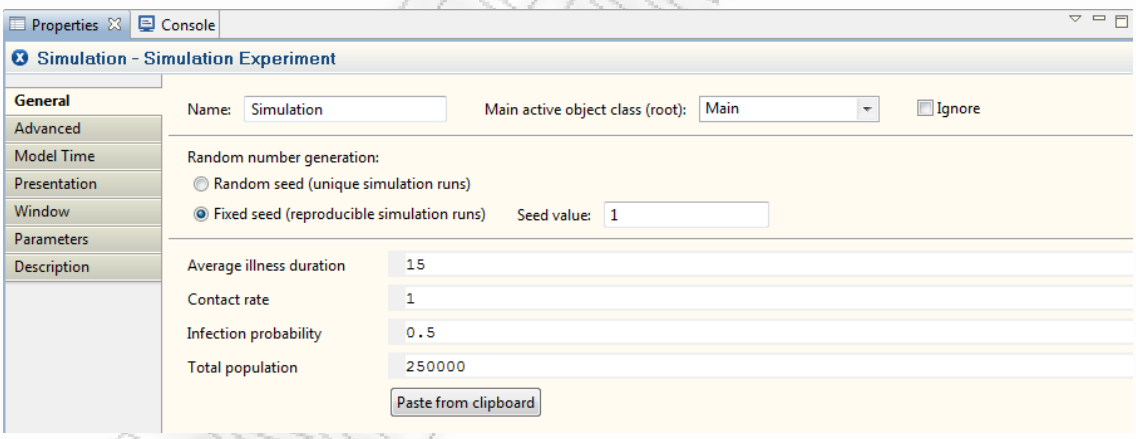


Εικόνα 4-13(β): Το γραφικό περιβάλλον της προσομοίωσης – Αρχική εικόνα – Γενικές ιδιότητες του κομβίου button τύπου Button.

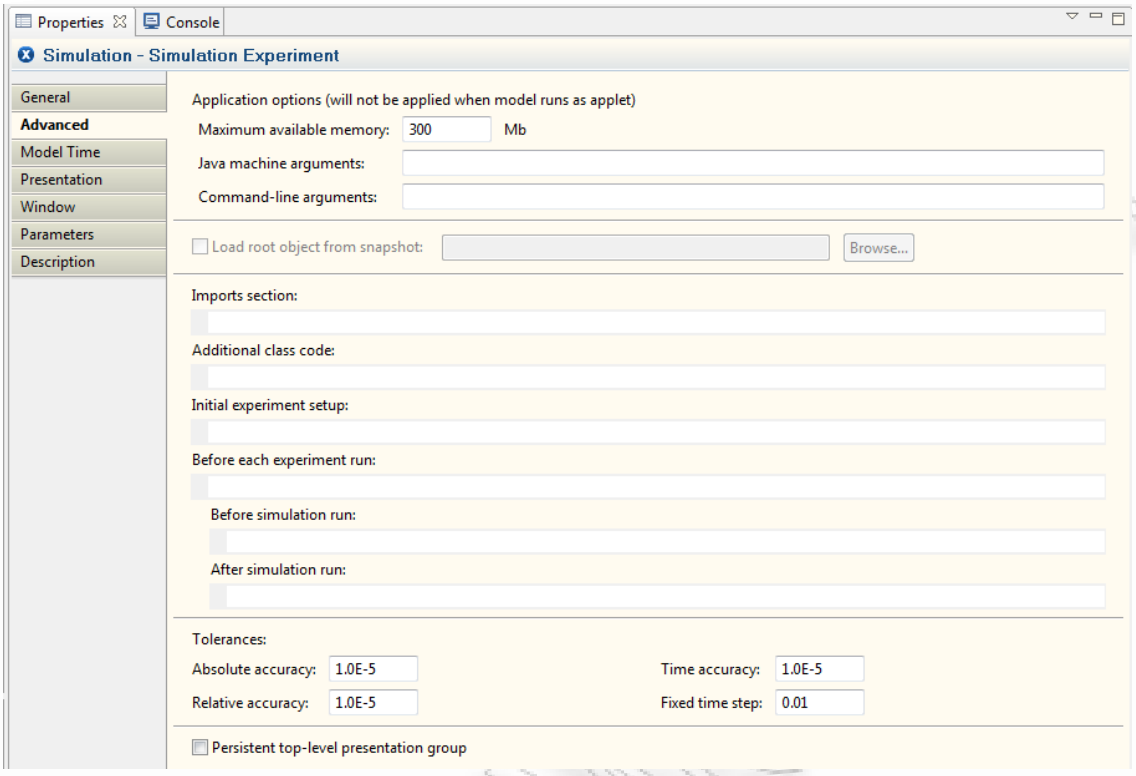


Εικόνα 4-13(γ): Το γραφικό περιβάλλον της προσομοίωσης – Αρχική εικόνα – Προχωρημένες ιδιότητες του κομβίου button τύπου Button.

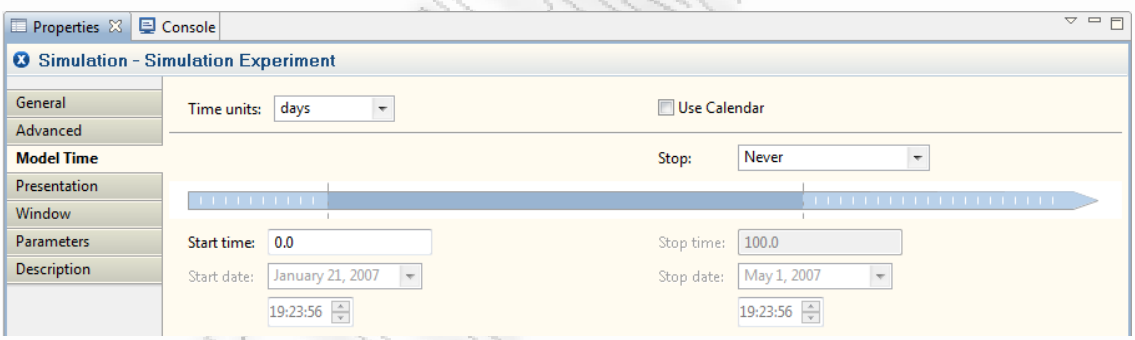
Οι παράμετροι της προσομοίωσης, όπως αυτοί θα χρησιμοποιηθούν από το Λογισμικό παρουσιάζονται στις Εικόνες 4-14(α) έως και 4-14(στ).



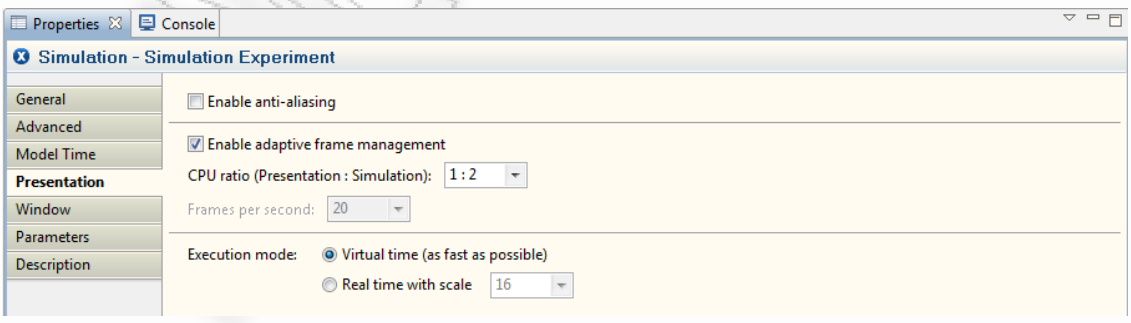
Εικόνα 4-14(α): Προσομοίωση - Γενικά χαρακτηριστικά.



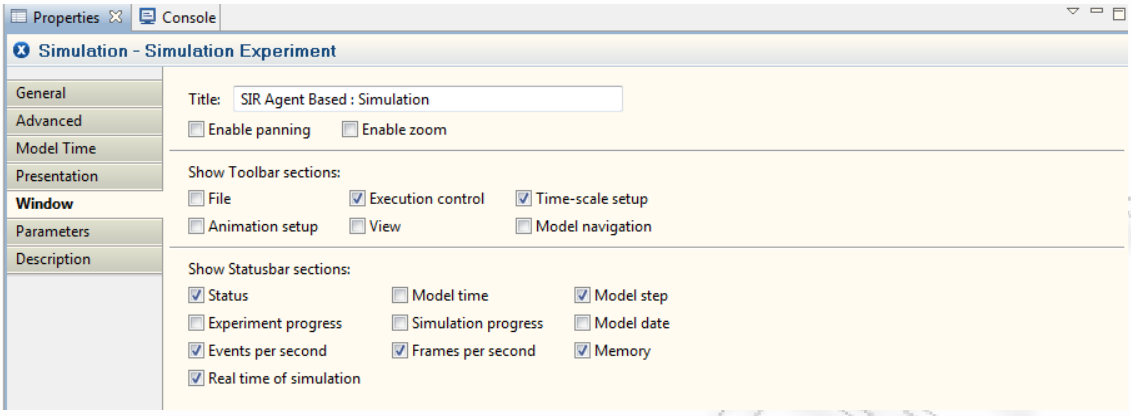
Εικόνα 4-14(β): Προσομοίωση - Προχωρημένα χαρακτηριστικά.



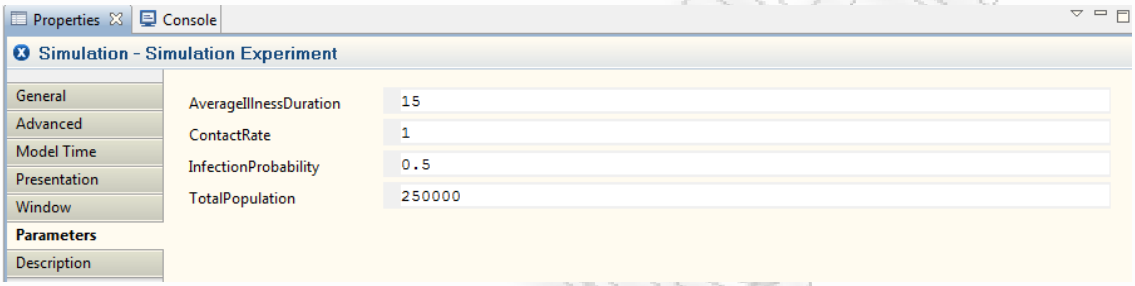
Εικόνα 4-14(γ): Προσομοίωση – Ορισμός χρονικών μεταβλητών.



Εικόνα 4-14(δ): Προσομοίωση – Χαρακτηριστικά παρουσίασης.



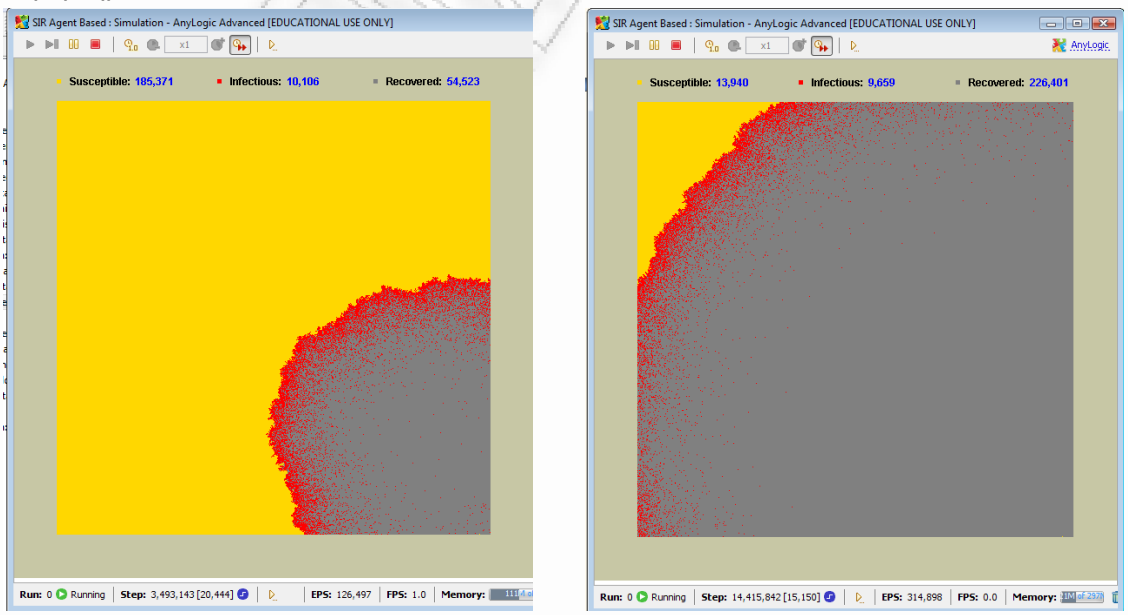
Εικόνα 4-14(ε): Προσομοίωση – Ιδιότητες παραθύρου.



Εικόνα 4-14(στ): Προσομοίωση – Χαρακτηριστικά παραμέτρων.

Κατά την αρχικοποίηση της προσομοίωσης είναι αναγκαίο να οριστεί ο αριθμός των πρακτόρων που θα βρίσκεται σε κάθε μια από τις τρεις καταστάσεις. Η προσομοίωση του μοντέλου από το Λογισμικό για να εκτελεσθεί χρειάζεται ένα γραφικό περιβάλλον για τη σύνδεση της εκτέλεσης της προσομοίωσης με το μοντέλο. Αυτό είναι το τελευταίο στάδιο της μοντελοποίησης και για τη συγκεκριμένη περίπτωση αποτελείται από ένα φόντο και ένα κομβίο που μόλις το πατήσουμε ξεκινά η προσομοίωση. Στην εικόνα 4-15 παρουσιάζονται δύο (2) στιγμιότυπα από την προσομοίωση.

Τα διαγράμματα κλάσεων και ο πλήρης κώδικας του παραδείγματος βρίσκονται στο Παράρτημα 2.



Εικόνα 4-15: Στιγμιότυπα από την προσομοίωση του μοντέλου.

ΚΕΦΑΛΑΙΟ 5

Η μοντελοποίηση της σχέσης θηρευτή και θηράματος (Predator Pray) μέσω της Δυναμικής Συστημάτων

Τα κοινωνικά συστήματα και τα οικοσυστήματα είναι δυνατό να μοντελοποιηθούν και να μας δώσουν μια εικόνα της εξέλιξης τους σε καθορισμένες χρονικές περιόδους με δεδομένες αρχικές συνθήκες. Ήδη από τα τέλη του 18ου αιώνα (1798) ο Malthus υποστήριξε ότι ο ανθρώπινος πληθυσμός αυξάνεται πολύ γρήγορα σε σχέση με τα διαθέσιμα τρόφιμα και η παγκόσμια φτώχια θα είναι το αποτέλεσμα του υπερπληθυσμού. Στοιχειοθέτησε την άποψη του με τη χρήση διαφορικών εξισώσεων αλλά δεν προέβλεψε την εξάρτηση των εξισώσεων αυτών από όρους δεύτερης τάξης. Φτάσαμε στον 20ο αιώνα (Robert May (1976), Feigenbaum (1978)) για να μπορέσουμε να αντιληφθούμε την επίδραση των μη γραμμικών φαινομένων στην εξέλιξη ενός κοινωνικού ή βιολογικού συστήματος, (θεωρία του χάους, ασαφής λογική).

5.1 Θεωρητικό μοντέλο

Οι τρεις (3) βασικές σχέσεις μεταξύ δύο (2) τουλάχιστον ειδών που παρουσιάζονται στη φύση είναι η συμβίωση, ο ανταγωνισμός και η σχέση θύτη και θύματος. Εμείς θα ασχοληθούμε με τη σχέση θύτη και θύματος μοντελοποιώντας τη σχέση του σαρκοβόρου και της λείας του, (predator and prey). Το πληθυσμιακό μοντέλο που θα μας απασχολήσει σε αυτό το κεφάλαιο, (αφορά την οικολογία και έχει σχέση με τη δυναμική των οικοσυστημάτων όπως προείπαμε), θα εξετασθεί μέσω της μεθοδολογίας της Δυναμικής Συστημάτων. Το ερώτημα που πρέπει να απαντηθεί είναι: «Πως δύο (2) οργανισμοί (με την ευρύτερη έννοια του όρου) με συγκρουόμενα συμφέροντα θα καταφέρουν να κινηθούν, να αλληλοεπιδράσουν και να διαμορφώσουν συνθήκες τέτοιες ώστε να μην καταστραφούν σε ένα πεπερασμένο χώρο ευκαιριών;»

Αν η περιγραφή του προβλήματος δεν μοιάζει απλή, η επίλυση του ακόμη και σήμερα δεν είναι εύκολη και γίνεται ακόμη δυσκολότερη αν οι οργανισμοί που θα αλληλεπιδράσουν αυξηθούν σε περισσότερους από δύο (2). Το μοντέλο των πληθυσμών που θα εξετάσουμε βασίζεται στο κλασσικό μοντέλο των Lotka-Voltera και αφορά στη μεταβολή των πληθυσμών δύο (2) ανταγωνιστικών ειδών, του θηρευτή και του θηράματος, που παλεύουν για την επιβίωση τους μέσα σε ένα χωρικό και χρονικό πλαίσιο. Στο παράδειγμα μας ο θηρευτής, το σαρκοβόρο, είναι ο λύγκας (lynx) και το θήραμα, η λεία, είναι ο λαγός (hare). Η εξέλιξη του πληθυσμού των λυγκών σε συνάρτηση με τον αριθμό των λαγών είναι το ζητούμενο από την προσομοίωση.

Η μαθηματική έκφραση του παραπάνω προβλήματος εκφράζεται μέσω του συστήματος των δύο (2) κανονικών διαφορικών εξισώσεων 1ου βαθμού:

$$\frac{dx}{dt} = \alpha x - \beta xy$$
$$\frac{dy}{dt} = \gamma x - \delta xy$$

όπου x,y οι πληθυσμοί των λαγών και των λυγκών αντίστοιχα και α,β,γ και δ οι συντελεστές των εξισώσεων. Οι όροι βxy και δxy είναι όροι πεπλεγμένοι και ορίζονται με αυτό τον τρόπο διότι η θνησιμότητα των λυγκών εξαρτάται από τους λαγούς και οι γεννήσεις των λαγών εξαρτώνται από τους λύγκες.

Στα μοντέλα θύτη και θύματος, (όπως επίσης αυτά ονομάζονται στη βιβλιογραφία) θεωρούμε μερικές βασικές προϋποθέσεις για την κατάστρωση των μοντέλων που είναι οι παρακάτω:

α) Η λεία, το θήραμα είναι το μόνο είδος με το οποίο μπορεί να τραφεί το σαρκοβόρο, ο θηρευτής, επομένως το σαρκοβόρο είναι απόλυτα εξαρτώμενο από τη λεία του.

β) Η λεία τρέφει μόνο ένα είδος σαρκοβόρων.

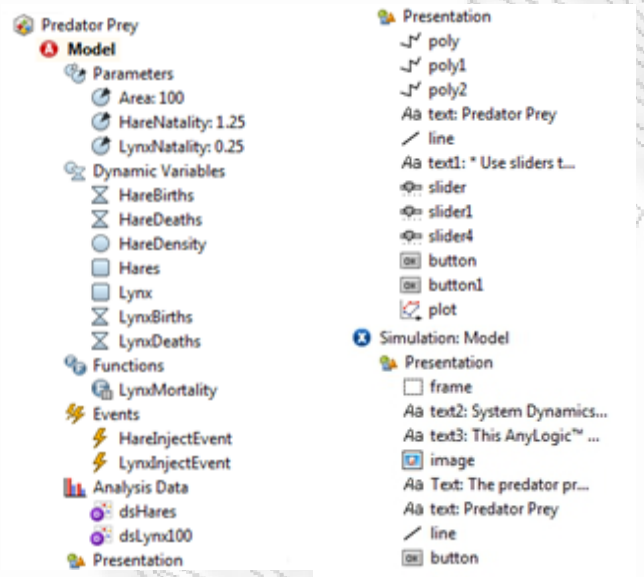
γ) Η λεία δυνητικά είναι απεριόριστη.

δ) Ο χώρος που κινούνται και δρουν τα δύο είδη είναι απομονωμένος και πεπερασμένος.

ε) Η τροφή της λείας είναι απεριόριστη.

Αν και τα παραπάνω είναι μη ρεαλιστικές και δεσμευτικές παραδοχές που δεν ισχύουν στη φύση, τα μοντέλα συνεχίζουν να χρησιμοποιούνται από τους ερευνητές λόγω ακριβώς της απλότητας τους και της μικρής τους μαθηματικής πολυπλοκότητας. Δίνουν ικανοποιητικά αποτελέσματα για μερικούς κύκλους ζωής των ειδών και αποτελούν τη βάση σύγκρισης με την πραγματική ζωή και τη δυναμική της φύσης. Στην περίπτωση μας, ανάλογα με τα «εγγενή» χαρακτηριστικά των λυγκών (εκτιμώμενος χρόνος ζωής, ικανότητα αναπαραγωγής, αριθμός γεννήσεων ανά θηλυκό ζώο, αριθμός μικρών ανά γέννα) και της δυνατότητας εύρεσης τροφής είναι δυνατό να εκτιμήσουμε την μελλοντική πληθυσμιακή εξέλιξη ενός είδους στην συγκεκριμένη περιοχή. Καθώς ο πληθυσμός των λυγκών αυξάνεται, αυξάνεται και η κατανάλωση των λαγών με αποτέλεσμα τη μείωση του πληθυσμού των λαγών. Η μείωση του πληθυσμού των λαγών προκαλεί την αύξηση της θνησιμότητας των λυγκών λόγω της μείωσης και της έλλειψης τροφής. Αυτό, οδηγεί στην μείωση της θνησιμότητας των λαγών και στην αύξηση του πληθυσμού τους. Η αύξηση του πληθυσμού των λαγών οδηγεί στη αύξηση του πληθυσμού των λυγκών και ο κύκλος επαναλαμβάνεται.

5.2 Η κλάση Model



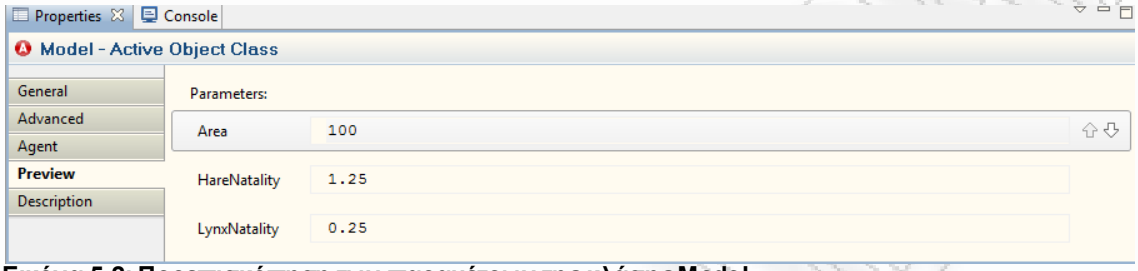
Εικόνα 5-1: Η ιεραρχία του μοντέλου και η δενδρική δομή του.

Το έργο Predator Prey, όπως φαίνεται στην εικόνα 5-1, αποτελείται από την κλάση παρουσίασης του μοντέλου και από την ενεργή κλάση αντικειμένου Model η οποία έχει ως παραμέτρους τις Area, HareNatality και LynxNatality. Οι δυναμικές μεταβλητές αποτελούνται από τις τέσσερις (4) ροές, (flow variables), HareBirths, HareDeaths, LynxBirths και LynxDeaths, τα δύο (2) σημεία συσσώρευσης, (stock variables), Hares και Lynx και τη βοηθητική μεταβλητή HareDensity.

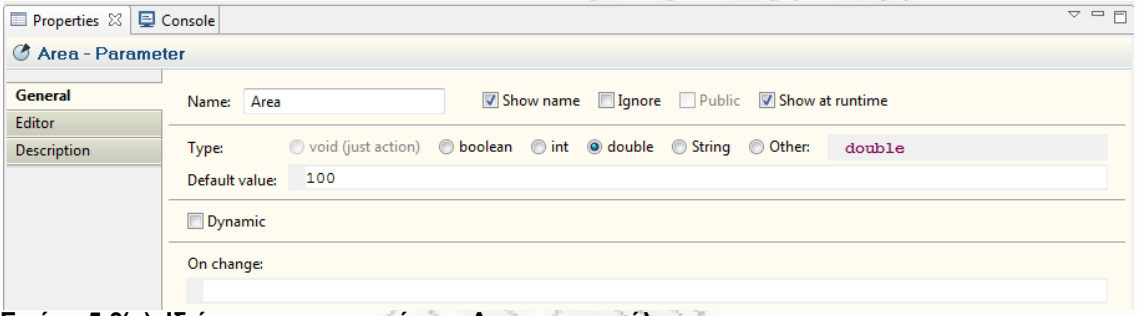
Οι παράμετροι του μοντέλου, (βλέπε πίνακα 5-1, εικόνες 5-2 και 5-3), είναι τύπου double και είναι οι εξής: (α) Area, η έκταση στην οποία ζουν οι θηρευτές και τα θηράματα και όπου το μοντέλο θα προσομοιωθεί, με αρχική τιμή τις 100 τετραγωνικές μονάδες, (β) HareNatality, η γεννητικότητα, ο ρυθμός των γεννήσεων των θηραμάτων (των λαγών) με αρχική τιμή 1.25 και (γ) LynxNatality, ο ρυθμός των γεννήσεων των θηρευτών (των λυγκών) με αρχική τιμή 0.25. Οι παράμετροι παρουσιάζονται στον πίνακα 5-1.

Παράμετροι του μοντέλου		
Όνομα παραμέτρου	Είδος Παραμέτρου	Αρχική τιμή παραμέτρου
Area	double	100
HareNatality	double	1.25
LynxNatality	double	0.25

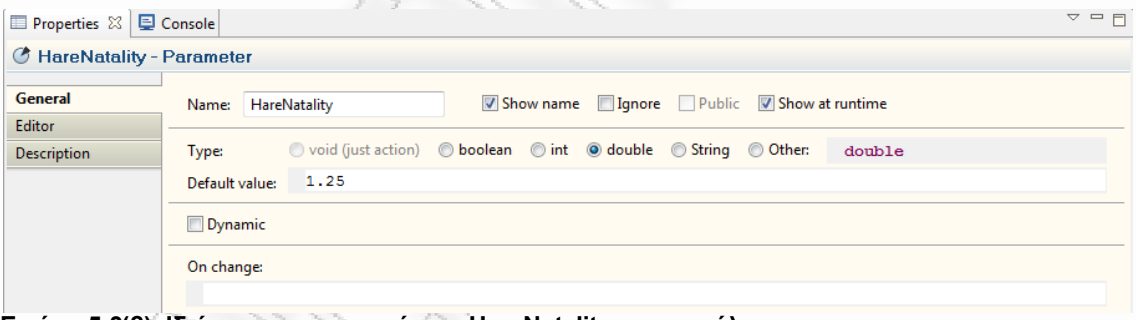
Πίνακας 5-1: Παράμετροι της κλάσης Model.



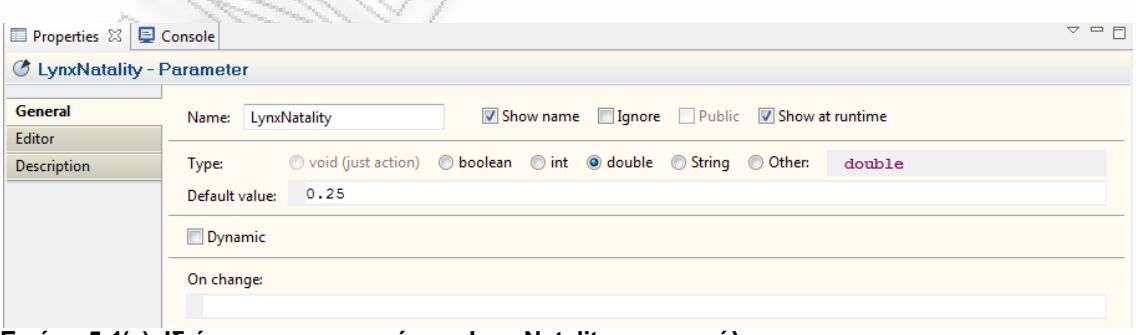
Εικόνα 5-2: Προεπισκόπηση των παραμέτρων της κλάσης Model.



Εικόνα 5-3(α): Ιδιότητες της παραμέτρου Area του μοντέλου.



Εικόνα 5-3(β): Ιδιότητες της παραμέτρου HareNatality του μοντέλου.

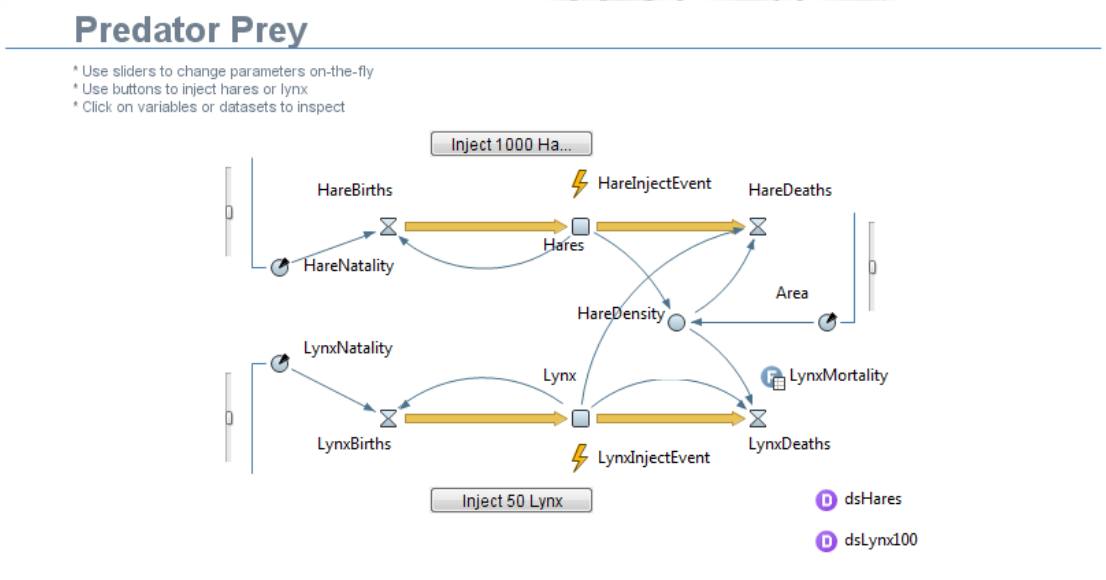


Εικόνα 5-1(γ): Ιδιότητες της παραμέτρου LynxNatality του μοντέλου.

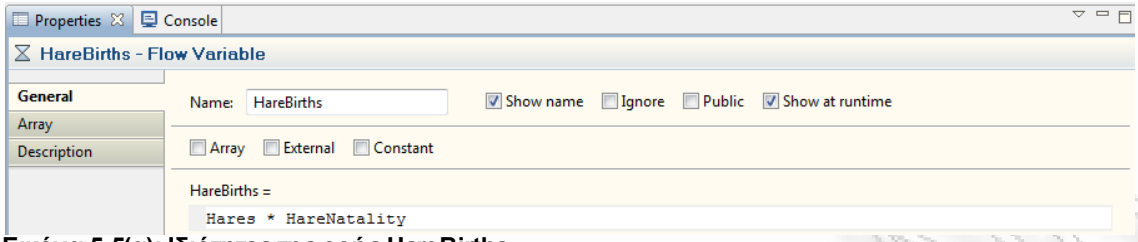
Οι δυναμικές παράμετροι του μοντέλου, όπως διακρίνονται στην εικόνα 5-1 (ιεραρχία του μοντέλου) και στην εικόνα 5-4 (γραφικό περιβάλλον δημιουργίας του μοντέλου), χωρίζονται στις ροές HareBirths και HareDeaths, (γεννήσεις και θάνατοι αντίστοιχα των θηραμάτων, των

λαγών) και LynxBirths και LynxDeaths, (γεννήσεις και θάνατοι αντίστοιχα των θηρευτών, των λυγκών), στα σημεία συσσώρευσης Hares (θηράματα, λαγοί) και Lynx (θηρευτές, λύγκες), με αρχικές τιμές 6000 και 125 μονάδες αντίστοιχα και στη βοηθητική μεταβλητή HareDensity που εκφράζει τη πυκνότητα του πληθυσμού των θηραμάτων στην έκταση του ζωτικού χώρου των ανταγωνιστικών ειδών. Ακολουθούν οι εξισώσεις των δυναμικών παραμέτρων του μοντέλου και στις εικόνες 5-5, 5-6 και 5-7 παρουσιάζονται συνοπτικά οι ιδιότητες των μεταβλητών.

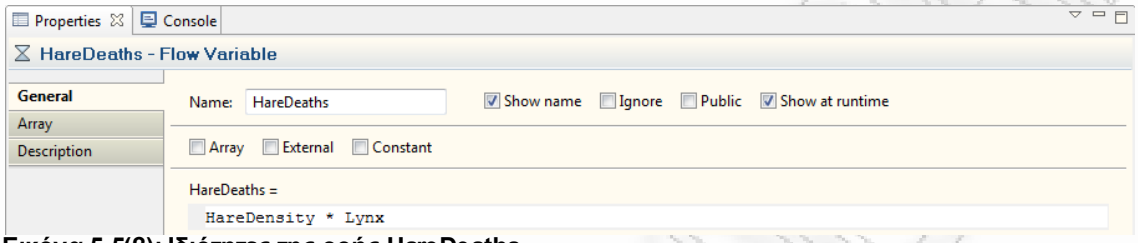
$$\boldsymbol{HareBirths = Hares * HareNatality}$$
$$\boldsymbol{HareDeaths = HareDensity * Lynx}$$
$$\boldsymbol{LynxBirths = Lynx * LynxNatality}$$
$$\boldsymbol{LynxDeaths = Lynx * LynxMortality(HareDensity)}$$
$$\boldsymbol{\frac{d(Hares)}{dt} = HareBirths - HareDeaths}$$
$$\boldsymbol{\frac{d(Lynx)}{dt} = LynxBirths - LynxDeaths}$$
$$\boldsymbol{HareDensity = \frac{Hares}{Area}}$$



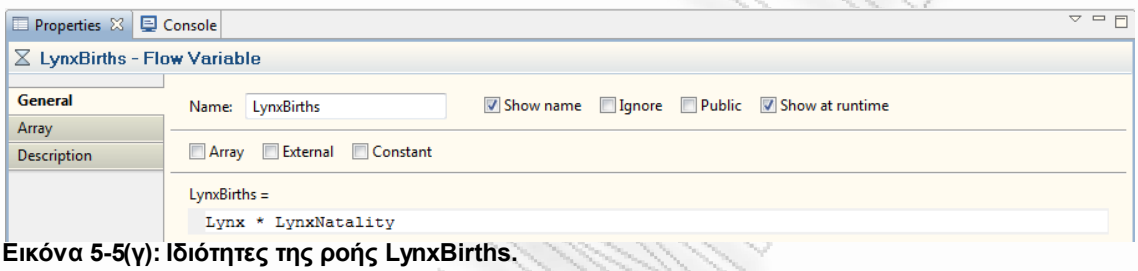
Εικόνα 5-4: Το γραφικό περιβάλλον του μοντέλου, όπως εμφανίζεται ολοκληρωμένα.



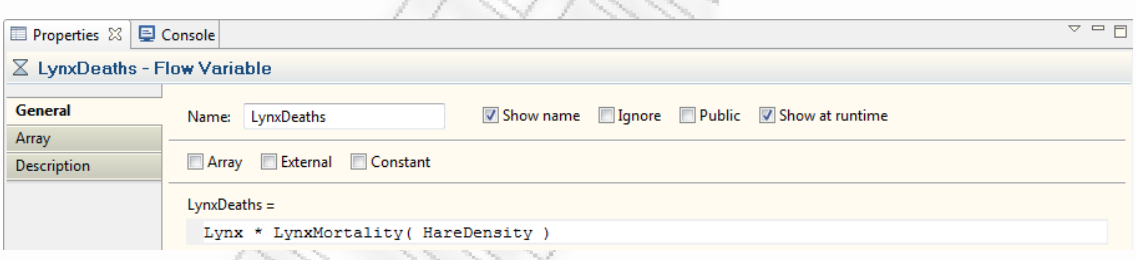
Εικόνα 5-5(α): Ιδιότητες της ροής HareBirths.



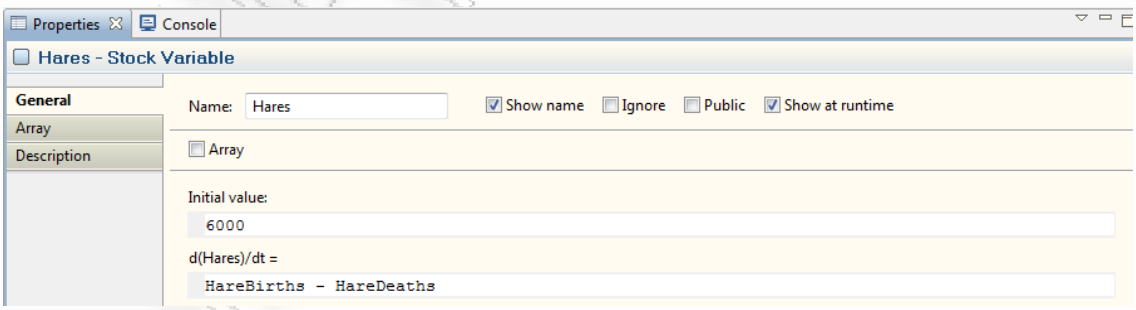
Εικόνα 5-5(β): Ιδιότητες της ροής HareDeaths.



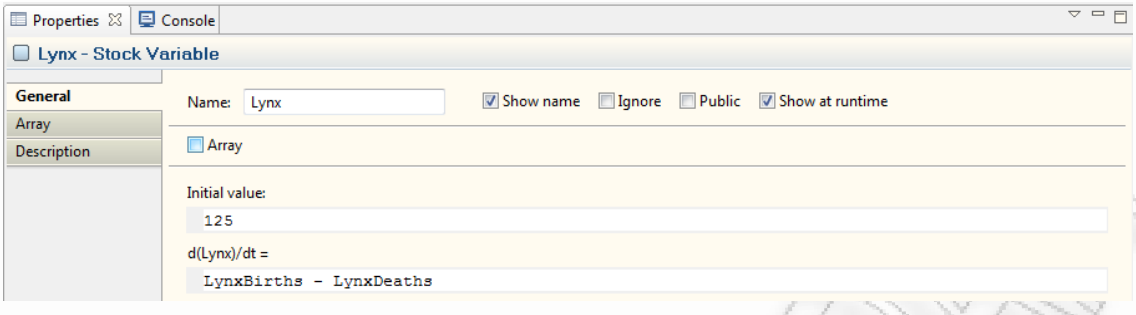
Εικόνα 5-5(γ): Ιδιότητες της ροής LynxBirths.



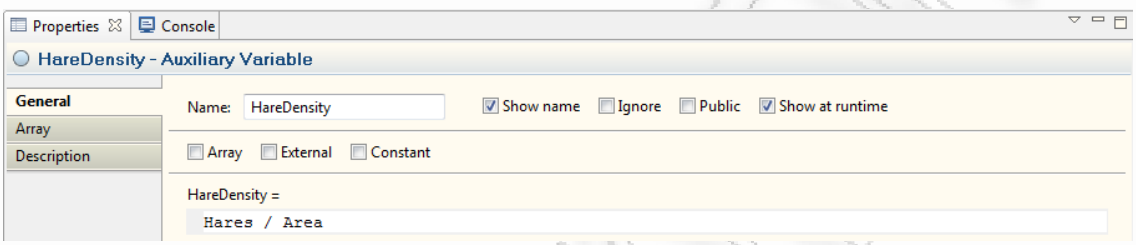
Εικόνα 5-5(δ): Ιδιότητες της ροής LynxDeaths.



Εικόνα 5-6(α): Ιδιότητες του σημείου συσσώρευσης Hares.



Εικόνα 5-6(β): Ιδιότητες του σημείου συσσώρευσης Lynx.

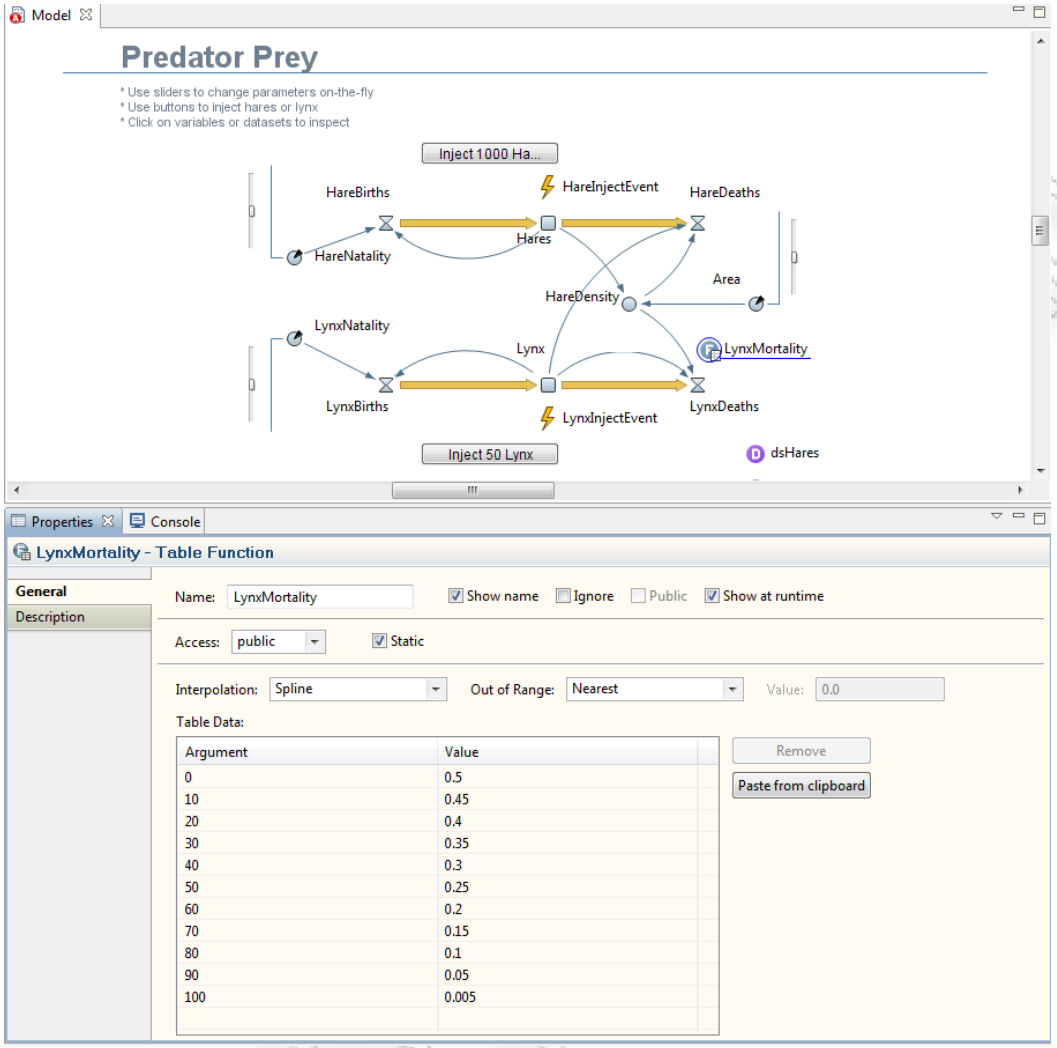


Εικόνα 5-7: Ιδιότητες της βοηθητικής μεταβλητής HareDensity.

Η μεταβλητή LynxMortality (θνησιμότητα λυγκών) είναι τύπου συνάρτησης πίνακα και είναι μια από τις συναρτήσεις ειδικού τύπου που το Λογισμικό υποστηρίζει. Η μεταβλητή ορίζεται με αυτή τη μορφή του πίνακα τιμών ώστε η μη-γραμμική σχέση μεταξύ του αριθμού των λυγκών και της θνησιμότητας τους να μπορεί να γραμμικοποιηθεί στο εύρος των τιμών που μας ενδιαφέρει. Στην περίπτωση που ο αριθμός των λυγκών ξεπεράσει το εύρος των ορισμένων τιμών τότε η τιμή της μεταβλητής θα είναι η πλέον πλησιέστερη. Σε κάθε άλλη περίπτωση οι τιμές καθορίζονται μέσω μη-γραμμικής παρεμβολής πολυωνυμικού χαρακτήρα. Το σύνολο τιμών των ορισμάτων της συνάρτησης παρουσιάζονται στον πίνακα 5-2 και στην εικόνα 5-8.

Όρισμα (αριθμός λυγκών)	Τιμή της ειδικής μεταβλητής LynxMortality
≤0	0,5
10	0,45
20	0,4
30	0,35
40	0,3
50	0,25
60	0,2
70	0,15
80	0,1
90	0,05
≥100	0,005

Πίνακας 5-2: Ο πίνακας τιμών των ορισμάτων της συνάρτησης πίνακα LynxMortality.



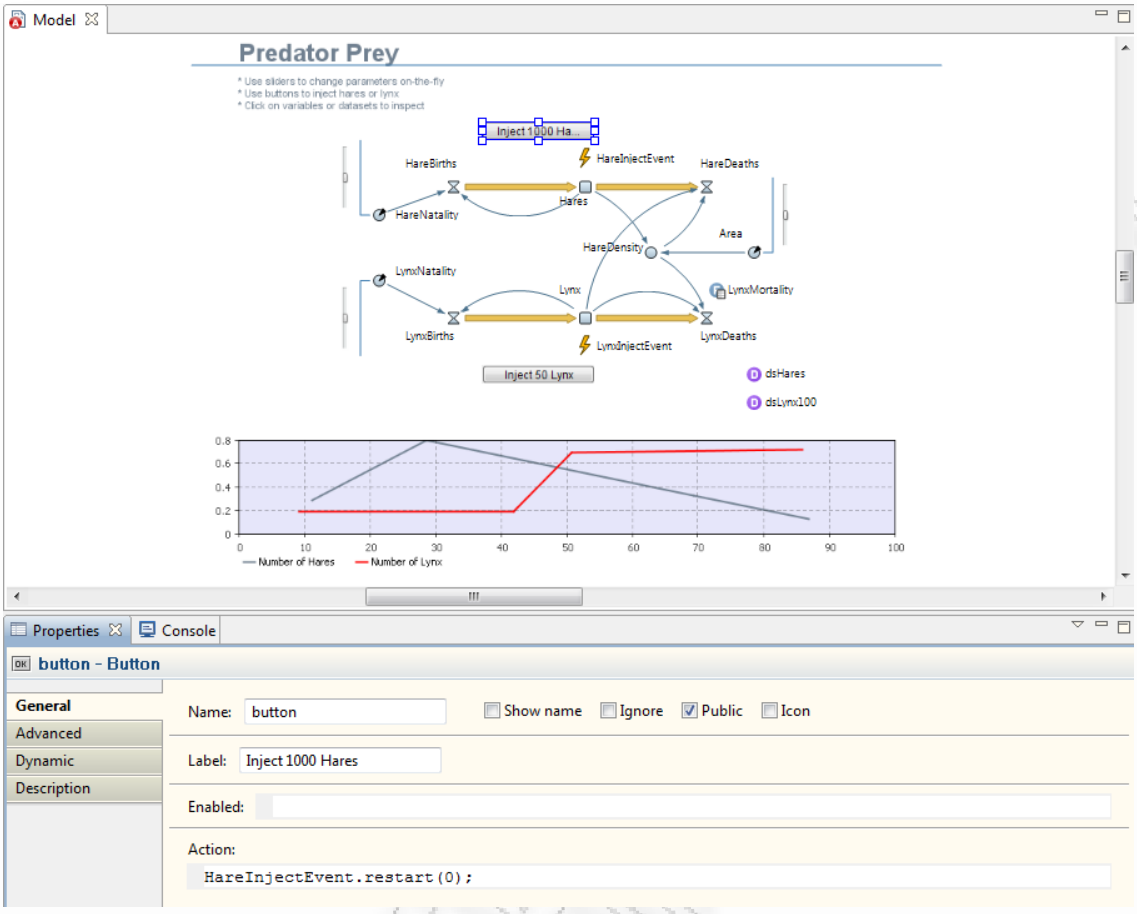
Εικόνα 5-8: Ιδιότητες της ειδικής συνάρτησης πίνακα LynxMortality.

Στην εικόνα 5-2 και στην εικόνα 5-8, εμφανίζονται δύο (2) κομβία, το button και το button1, με τα οποία γίνεται εισαγωγή λαγών και λυγκών αντίστοιχα στο απομονωμένο περιβάλλον της προσομοίωσης. Οι ιδιότητες τους παρουσιάζονται στις εικόνες 5-9 (α) και (β). Τα κομβία έχουν άμεση σχέση, συνεργάζονται και συνδυάζονται με τα δύο (2) γεγονότα (Event), τα οποία είναι ο απλούστερος τρόπος εισαγωγής καταστάσεων, από το Λογισμικό, στο μοντέλο:

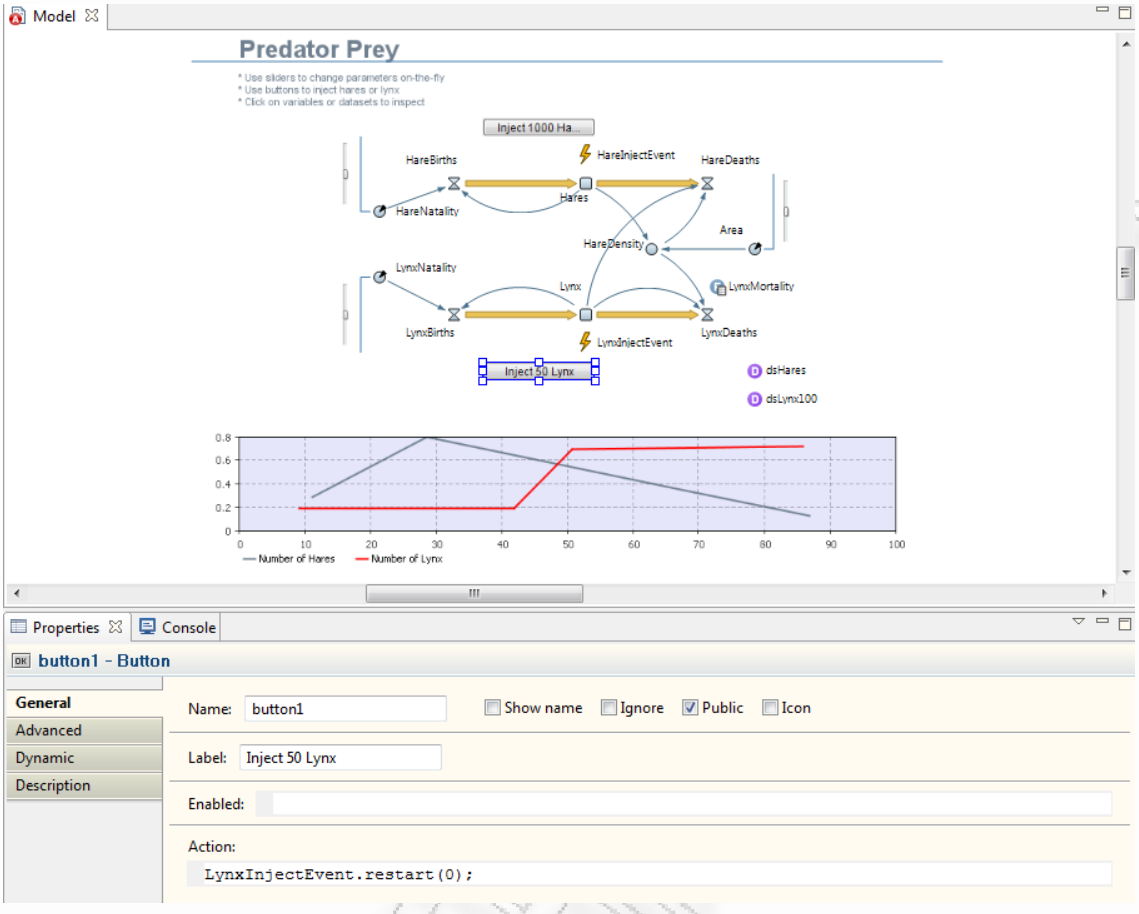
(α) HareInjectEvent, το γεγονός όπου εισάγεται πληθυσμός 1000 λαγών στο μοντέλο κατά την εκτέλεση της προσομοίωσης και μαθηματικά εκφράζεται ως $\mathrm{Hares} = \mathrm{Hares} + 1000$,

(β) LynxInjectEvent, το γεγονός όπου εισάγεται πληθυσμός 50 λυγκών στο μοντέλο κατά την εκτέλεση της προσομοίωσης και μαθηματικά εκφράζεται ως $\mathrm{Lynx} = \mathrm{Lynx} + 50$

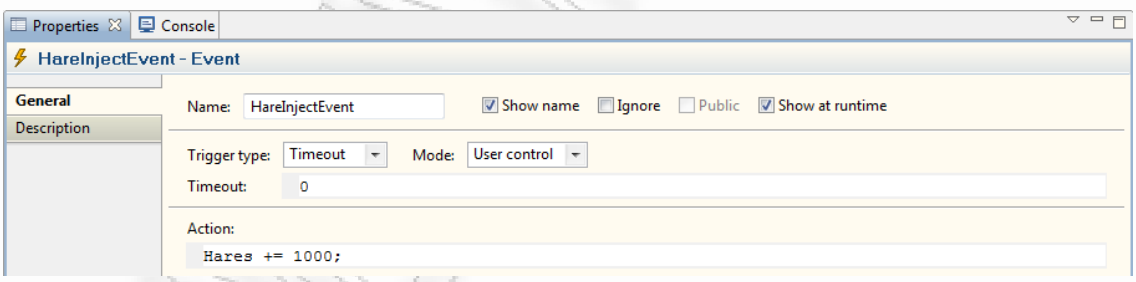
Οι ιδιότητες τους παρουσιάζονται στις εικόνες 5-10(α) και 5-10(β).



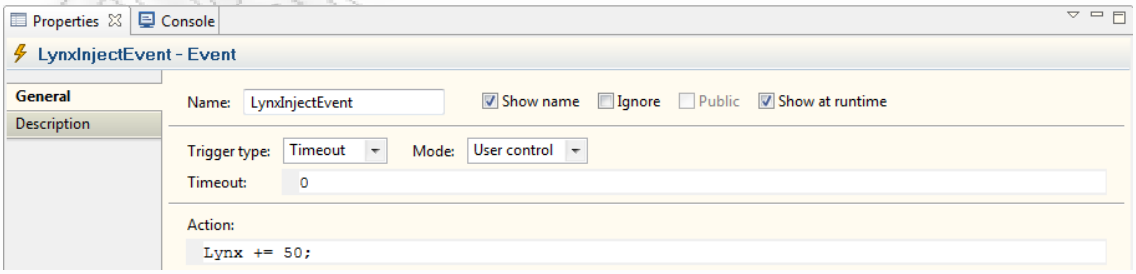
Εικόνα 5-9(α): Ιδιότητες του κομβίου button.



Εικόνα 5-9(β): Ιδιότητες του κομβίου button1.

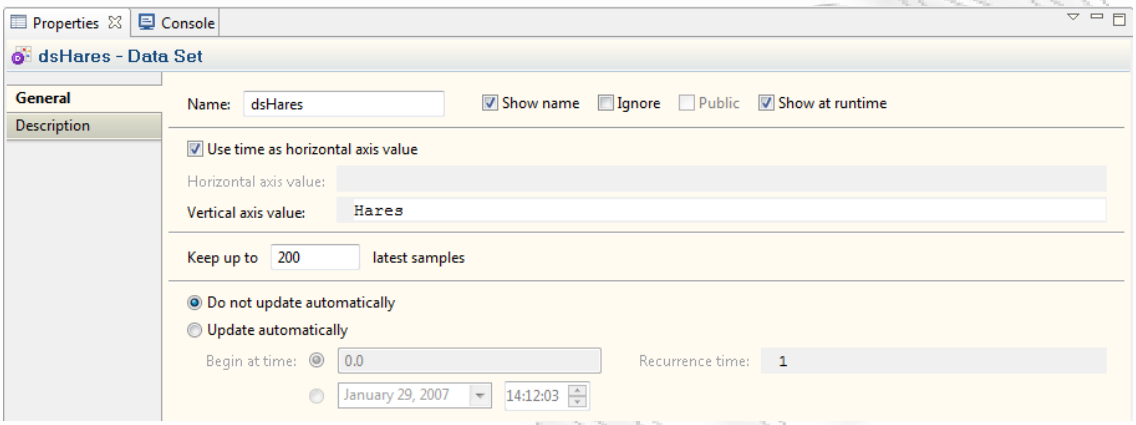


Εικόνα 5-10(α): Ιδιότητες του γεγονότος HareInjectEvent.

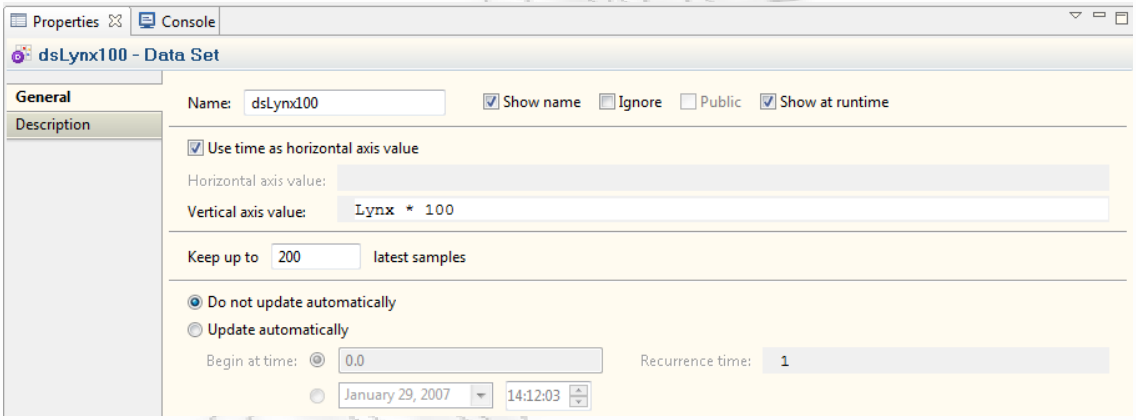


Εικόνα 5-10(β): Ιδιότητες του γεγονότος LynxInjectEvent.

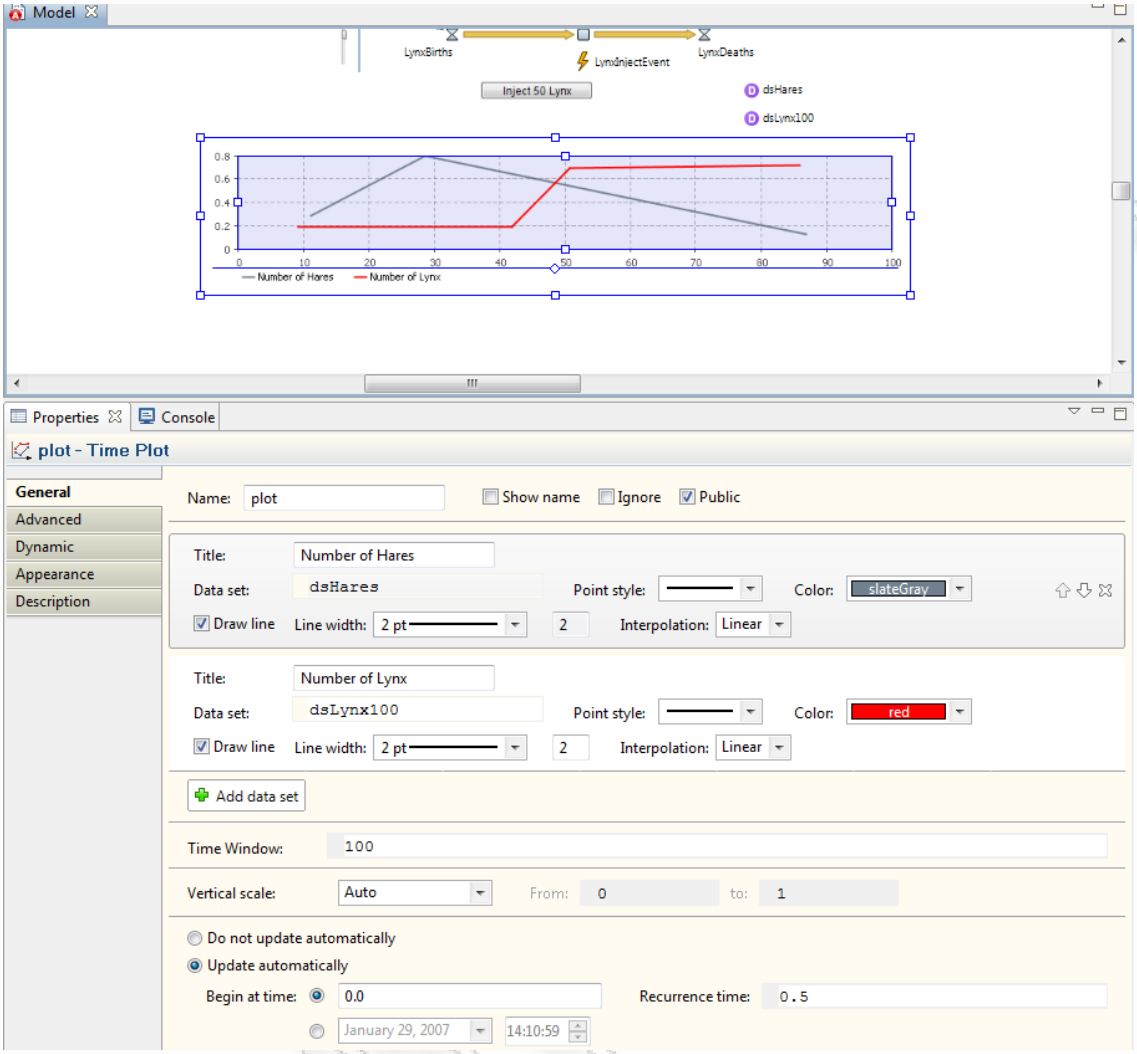
Τα σύνολα δεδομένων dsHares και dsLynx100 χρησιμεύουν στη προσωρινή αποθήκευση των δεδομένων που αναφέρονται στους πληθυσμούς των λαγών και των λυγκών αντίστοιχα. Μέσω του εσωτερικού μηχανισμού διασύνδεσης δεδομένων του Λογισμικού, εισάγονται στο διάγραμμα χρόνου (time plot), όπου και τελικώς εμφανίζονται οι πληθυσμοί των λυγκών (πολλαπλασιασμένες οι τιμές επί 100 ώστε να εμφανίζονται ευδιάκριτα) και των λαγών στην πορεία του χρόνου κατά τη διάρκεια της προσομοίωσης. Τα παραπάνω παρουσιάζονται στις εικόνες 5-11(α), 5-11(β) και 5-11(γ).



Εικόνα 5-11(α): Ιδιότητες του συνόλου δεδομένων dsHares.



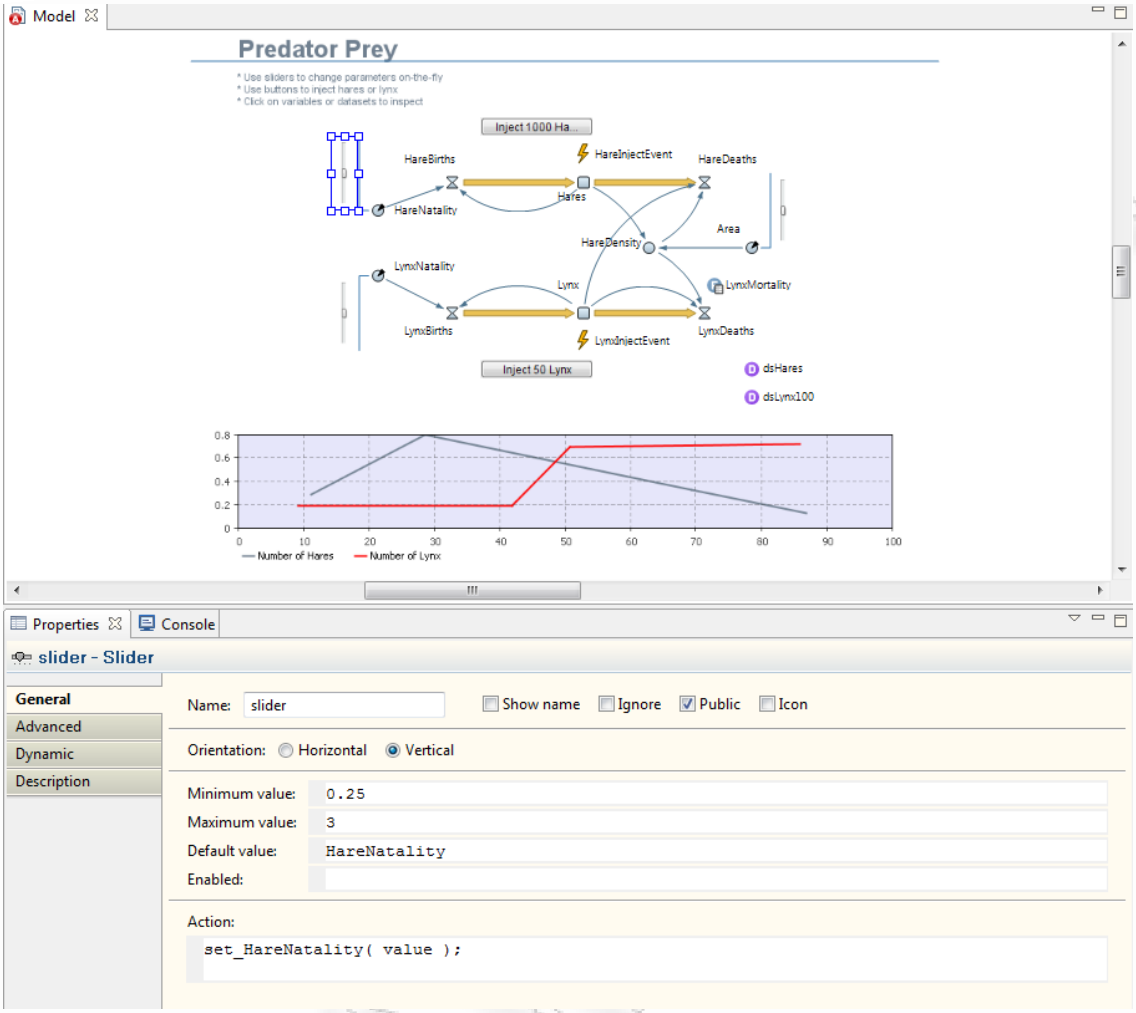
Εικόνα 5-11(β): Ιδιότητες του συνόλου δεδομένων dsLynx100.



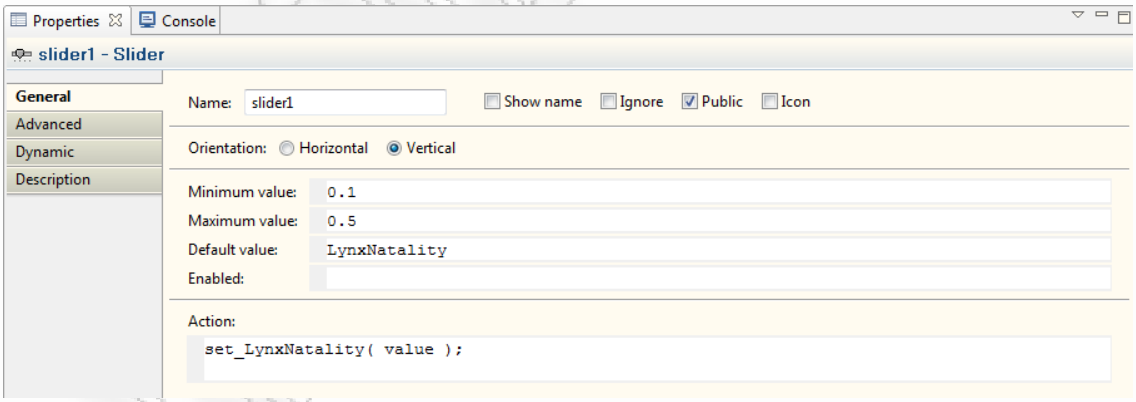
Εικόνα 5-11(γ): Ιδιότητες του διαγράμματος χρόνου plot.

Το μοντέλο ολοκληρώνεται με την εισαγωγή τριών (3) ράβδων κύλισης οι οποίες επιτρέπουν τη μεταβολή των τιμών των μεταβλητών. Στο παράδειγμα μας, συνδέονται άμεσα με τις προσδιδόμενες τιμές στις τρεις (3) μεταβλητές του μοντέλου Area, LynxNatality και HareNatality. Στην εικόνα 5-12 (α), (β) και (γ) παρουσιάζονται οι ιδιότητες των ράβδων κύλισης και η συσχέτιση τους με τις μεταβλητές.

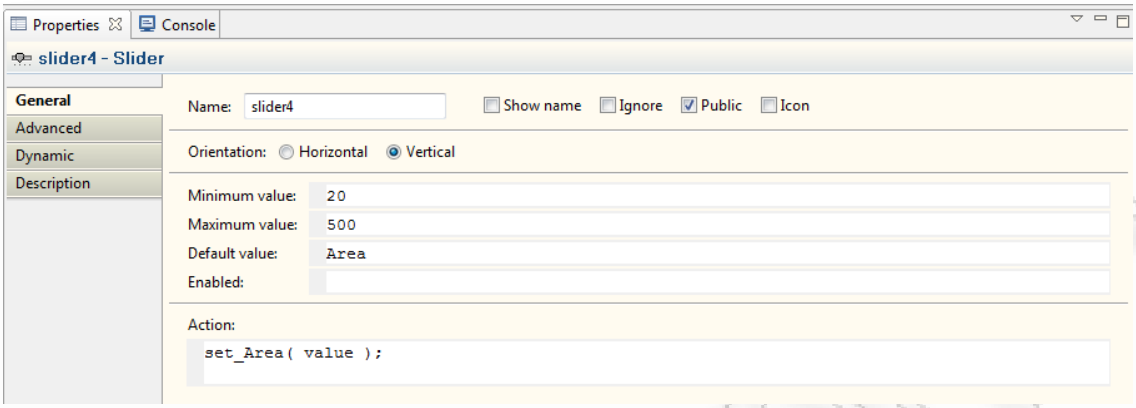
Το μοντέλο έχει ολοκληρωθεί. Το κύριο χαρακτηριστικό του είναι ότι μπορούμε να μεταβάλουμε τις τιμές των παραμέτρων Area, HareNatality και LynxNatality κατά τη διαδικασία της προσομοίωσης (μέσω των ράβδων κύλισης) και να παρατηρούμε τις αλλαγές που εμφανίζονται σε «πραγματικό» χρόνο. Αυτό είναι πρακτικό διότι με αυτό τον τρόπο μπορούμε να δημιουργούμε σενάρια στη στιγμή χωρίς να χρειάζεται να ξεκινάμε την προσομοίωση από την αρχή.



Εικόνα 5-12(α): Γενικές ιδιότητες της ράβδου κύλισης slider που μεταβάλει τις τιμές της μεταβλητής HareNatality.



Εικόνα 5-12(β): Γενικές ιδιότητες της ράβδου κύλισης slider1 που μεταβάλει τις τιμές της μεταβλητής LynxNatality.

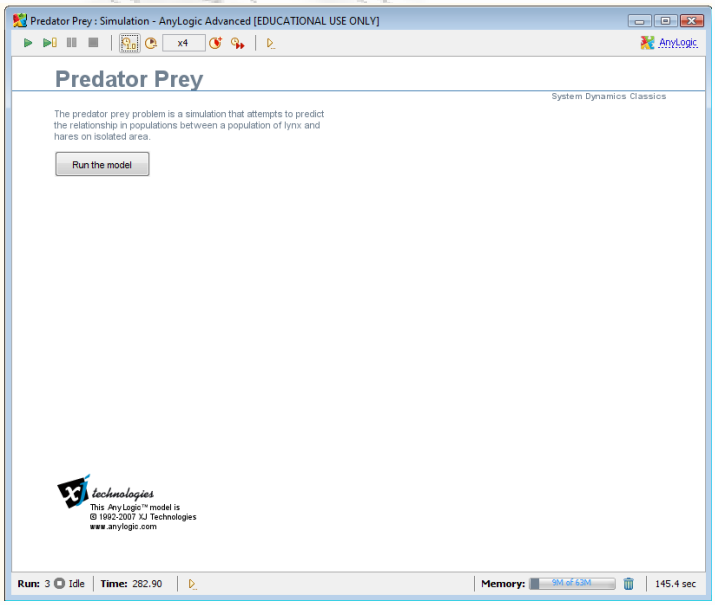


Εικόνα 5-12(γ): Γενικές ιδιότητες της ράβδου κύλισης slider4 που μεταβάλει τις τιμές της μεταβλητής Area.

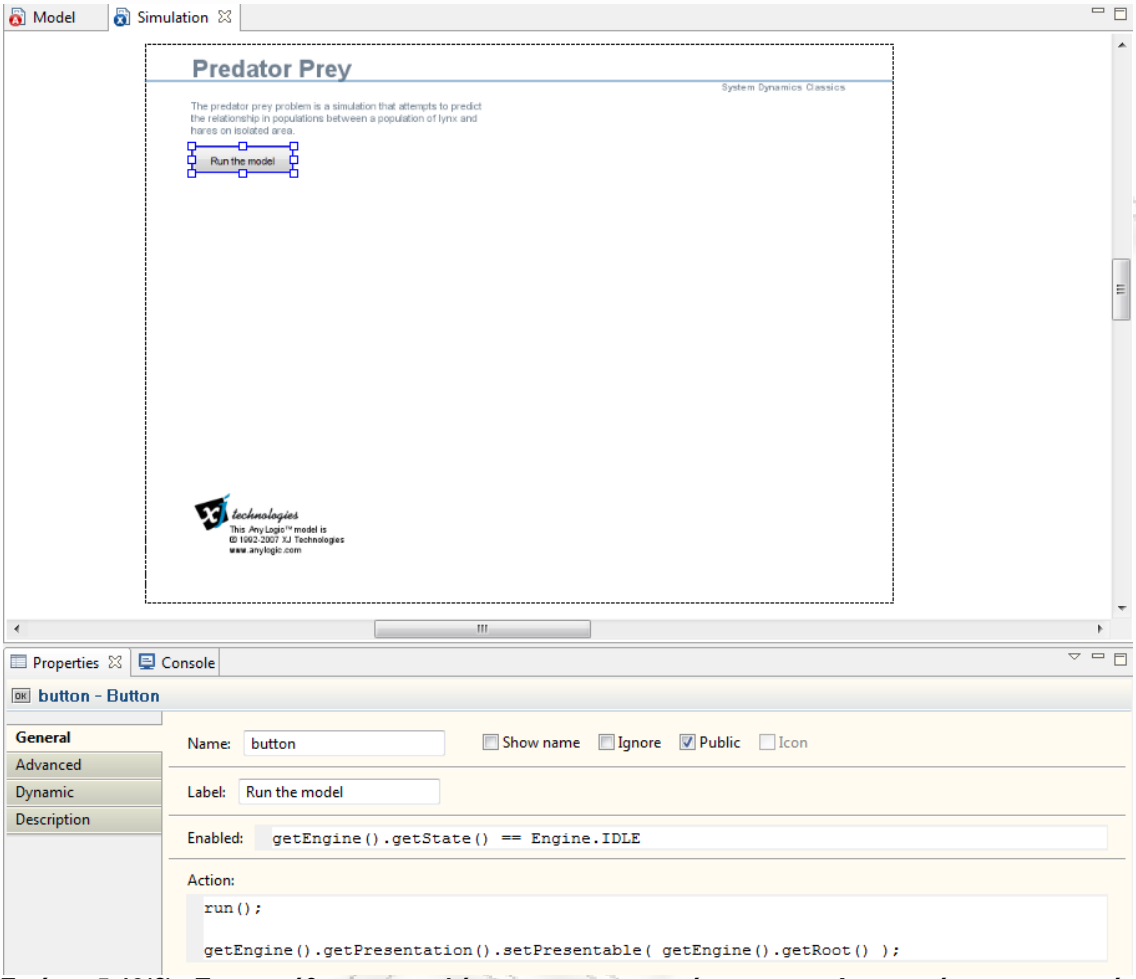
5.3 Η κλάση προσομοίωσης του έργου, Simulation:Main

Το έργο Predator Prey συμπεριλαμβάνει την κλάση Simulation:Model που υλοποιεί το τμήμα που αφορά στην παρουσίαση της προσομοίωσης του μοντέλου, (εικόνα 5-1). Με την κλάση αυτή δημιουργείται μια αρχική εικόνα αρκετά λιτή, εικόνα 5-13(α) και 5-13(β), που αποτελείται μόνο από ένα παράθυρο που περιέχει μερικά αρχικά μηνύματα, το λογότυπο της εταιρείας κατασκευής του λογισμικού και ένα μόνο κόμβιο για την συσχέτιση της παρουσίασης της προσομοίωσης με την κλάση του μοντέλου.

Η προσομοίωση του παραδείγματος ξεκινά με το πάτημα του κομβίου. Αμέσως μετά εμφανίζεται το παράθυρο με το μοντέλο, όπως το αναλύσαμε προηγούμενα. Σε «πραγματικό χρόνο» γίνεται η ανάλυση του μοντέλου. Είναι ορατές από τον χρήστη οι ενδιάμεσες τιμές των μεταβλητών ενώ ταυτόχρονα απεικονίζονται και στο διάγραμμα χρόνου οι τιμές των πληθυσμών των θηρευτών και των θηραμάτων. Το αρχικό παράθυρο της προσομοίωσης μαζί με δύο (2) στιγμιότυπα κατά την εκτέλεση της παρουσιάζονται στην εικόνα 5-14(α) και 5-14(β).

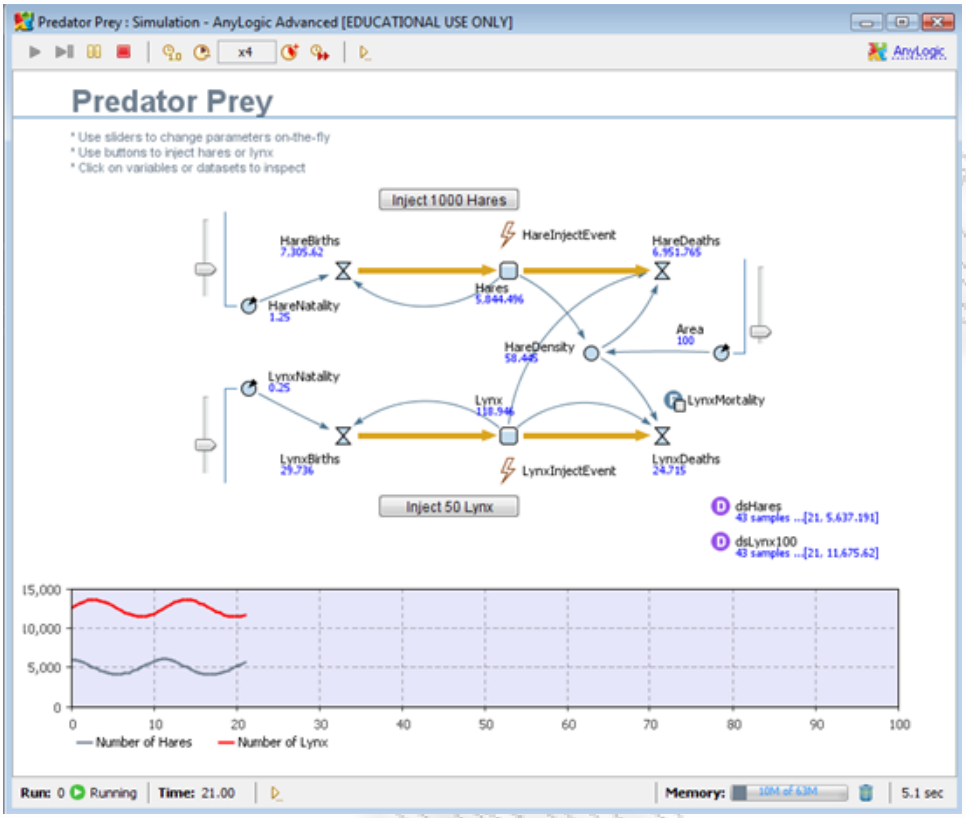


Εικόνα 5-13(α): Το αρχικό παράθυρο της προσομοίωσης.

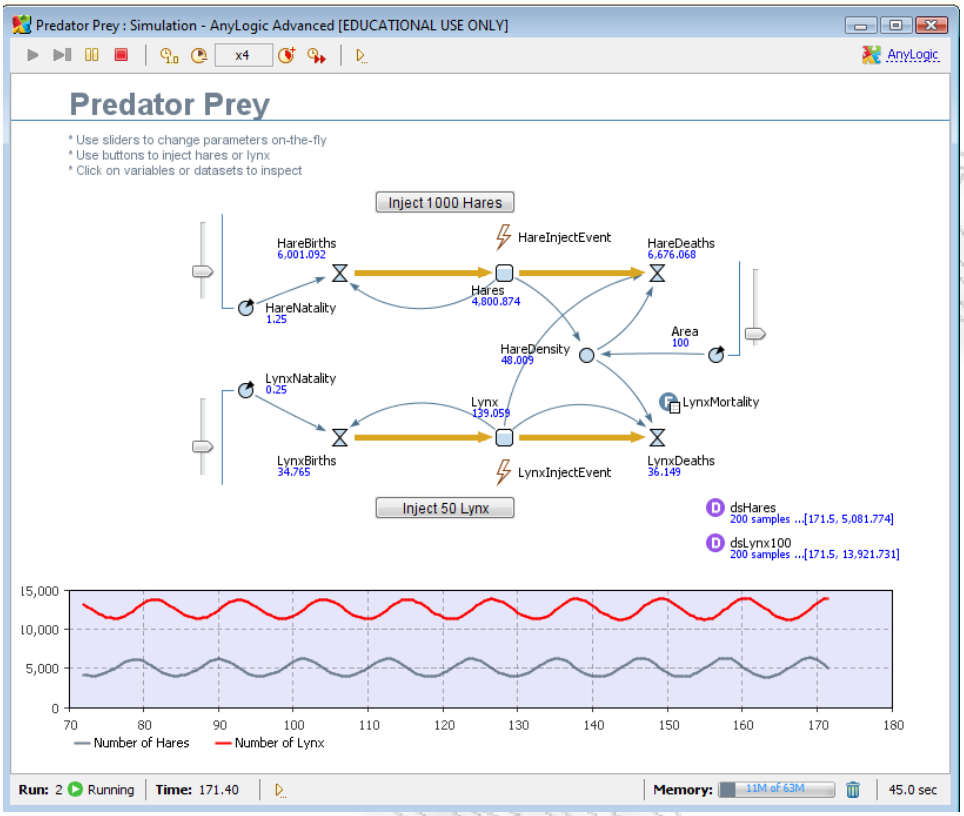


Εικόνα 5-13(β): Το παράθυρο της κλάσης της προσομοίωσης στο Λογισμικό και οι γενικές ιδιότητες του κομβίου button.

Τα διαγράμματα κλάσεων και ο πλήρης κώδικας του παραδείγματος βρίσκονται στο Παράρτημα 3.



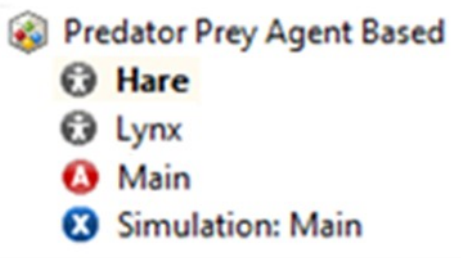
Εικόνα 5-14(α): Στιγμιότυπο από την προσομοίωση.



Εικόνα 5-14(β): Δεύτερο στιγμιότυπο από την προσομοίωση.

ΚΕΦΑΛΑΙΟ 6

Μοντελοποίηση της αλληλεπίδρασης δύο πληθυσμών, (θηρευτής και θήραμα), με τη μέθοδο των Πρακτόρων Λογισμικού



Εικόνα 6-1: Η ιεραρχία του μοντέλου.

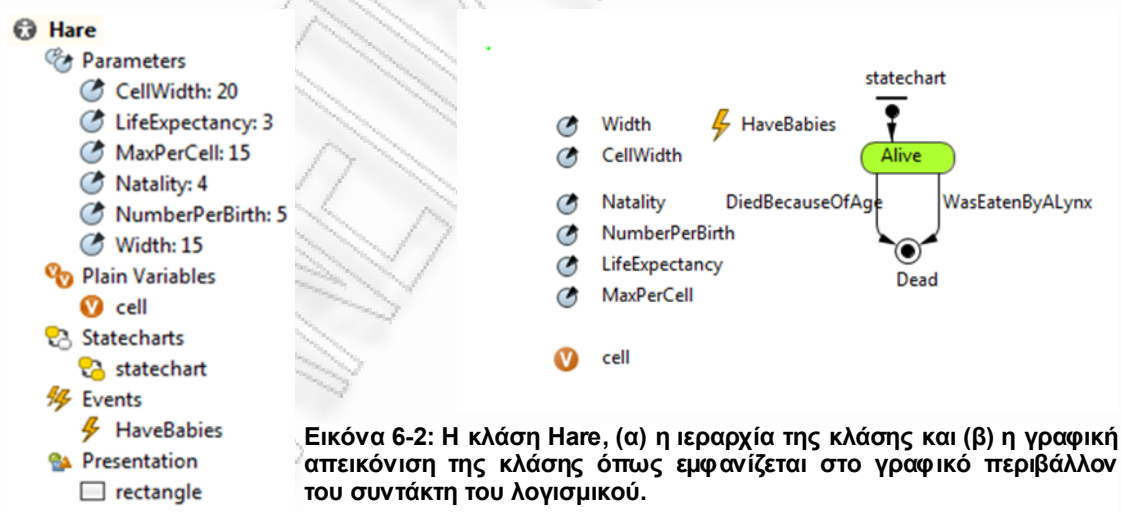
Το παράδειγμα αυτού του κεφαλαίου εξετάζει το μοντέλο του θηρευτή και του θηράματος, εξετάζει την αλληλεπίδραση μεταξύ των πληθυσμών δύο (2) ανταγωνιστικών ειδών, των λυγκών και των λαγών αντίστοιχα σε μια ορισμένη ως προς την έκταση της περιοχή, η οποία είναι απομονωμένη από άλλες εξωτερικές επιδράσεις, όπως περιγράφηκε ήδη στο κεφάλαιο 5.

Σε αυτό το κεφάλαιο όμως, η μέθοδος μοντελοποίησης του μοντέλου του παραδείγματος είναι αυτή των πρακτόρων λογισμικού, οι πληθυσμοί μοντελοποιούνται μέσω πρακτόρων ενώ η μεταβολή του πλήθους των πρακτόρων εξαρτάται από στοχαστικές διαδικασίες.

Το έργο ονομάζεται Predator Prey Agent Based και η ιεραρχική δομή του παρουσιάζεται στην εικόνα 6-1. Αποτελείται από τις ενεργές κλάσεις Hare και Lynx που υλοποιούν τους πράκτορες λογισμικού των θηρευτών και των θηραμάτων αντίστοιχα, την κλάση Main που υλοποιεί το μοντέλο και την κλάση Simulation:Main που ολοκληρώνει την παρουσίαση του μοντέλου κατά την προσομοίωση.

6.1 Η κλάση Hare

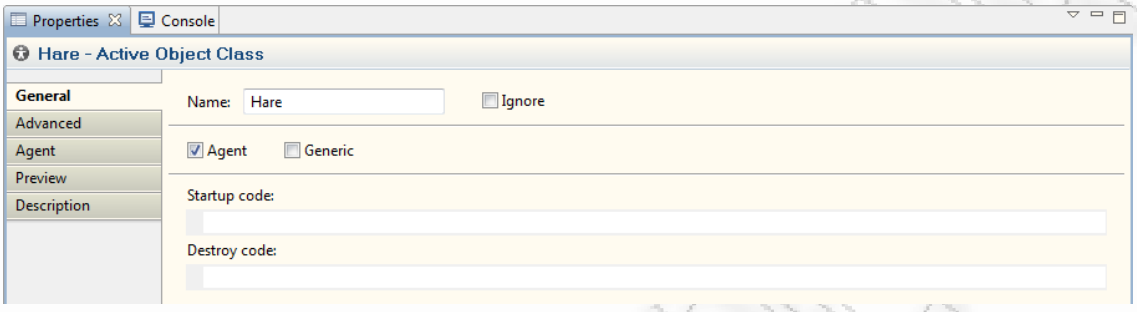
Η κλάση Hare υλοποιείται μέσω ενός διαγράμματος καταστάσεων, εικόνα 6-2(β) και στην εικόνα 6-2(α) παρουσιάζεται η δενδρική δομή της ιεραρχίας της κλάσης.



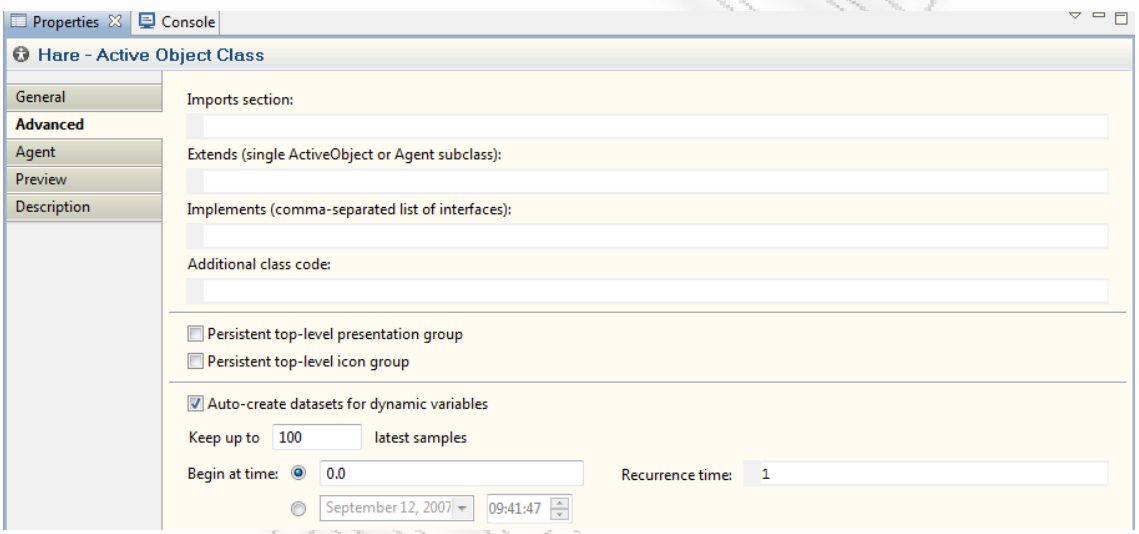
Εικόνα 6-2: Η κλάση Hare, (α) η ιεραρχία της κλάσης και (β) η γραφική απεικόνιση της κλάσης όπως εμφανίζεται στο γραφικό περιβάλλον του συντάκτη του λογισμικού.

Το θήραμα, (λαγός, hare), μετά τη γέννηση του και κατά τη διάρκεια της ζωής του (κατάσταση Alive στο διάγραμμα καταστάσεων), μπορεί είτε (α) να πεθαίνει από φυσικά αίτια, να πεθάνει δηλαδή από γηρατειά, είτε (β) να θανατωθεί από τον θηρευτή (λύγκα, lynx) για να γίνει η τροφή του, (οι δύο μεταβάσεις του διαγράμματος). Η πορεία της ζωής του θηράματος

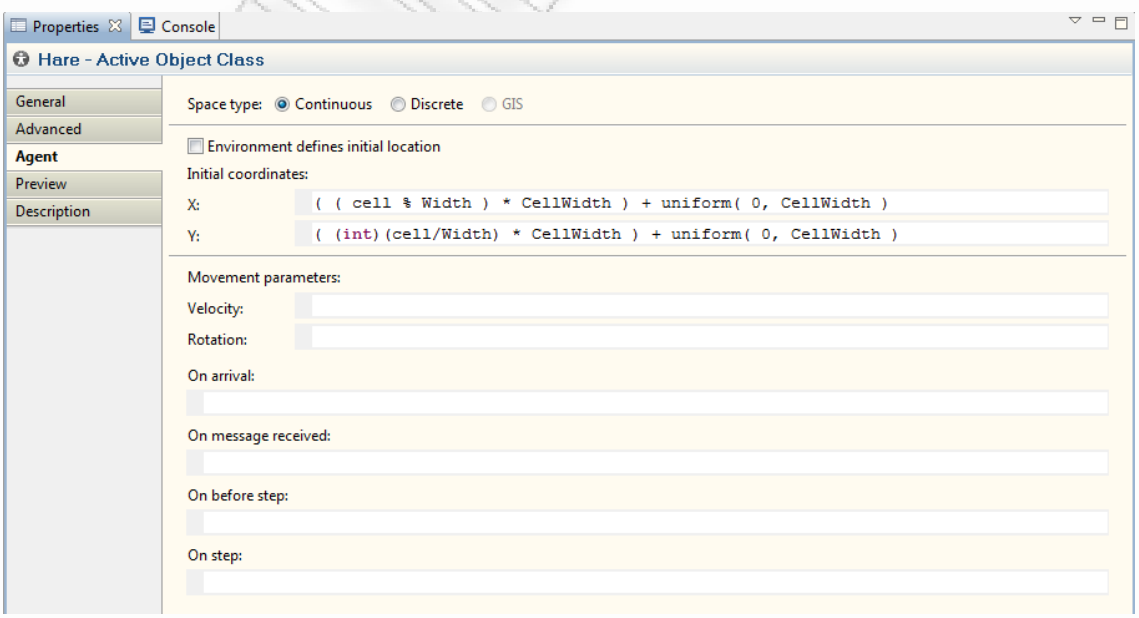
στην περιοχή που μας ενδιαφέρει εξαρτάται από τις παραμέτρους της κλάσης Width, CellWidth, Natality, NumberPerBirth, LifeExpectancy και MaxPerCell. Το θήραμα αποκτά απογόνους μέσω του γεγονότος HaveBabies. Στην εικόνα 6-2 παρουσιάζεται η κλάση και στην εικόνα 6-3 παρουσιάζονται οι ιδιότητες της κλάσης ενεργού αντικειμένου Hare όπως αυτές εμφανίζονται στο περιβάλλον του Λογισμικού.



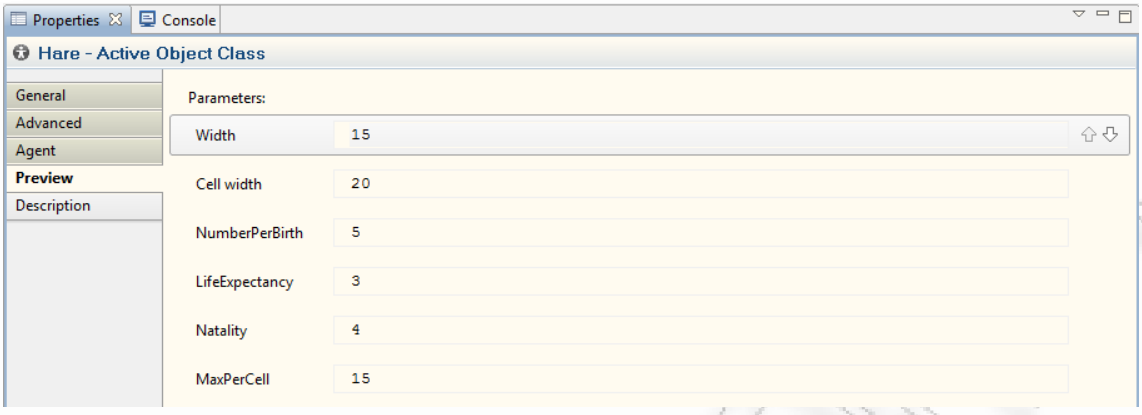
Εικόνα 6-3(α): Γενικές ιδιότητες της κλάσης Hare.



Εικόνα 6-3(β): Προχωρημένες ιδιότητες της κλάσης Hare.



Εικόνα 6-3(γ): Ιδιότητες του πράκτορα της κλάσης Hare.



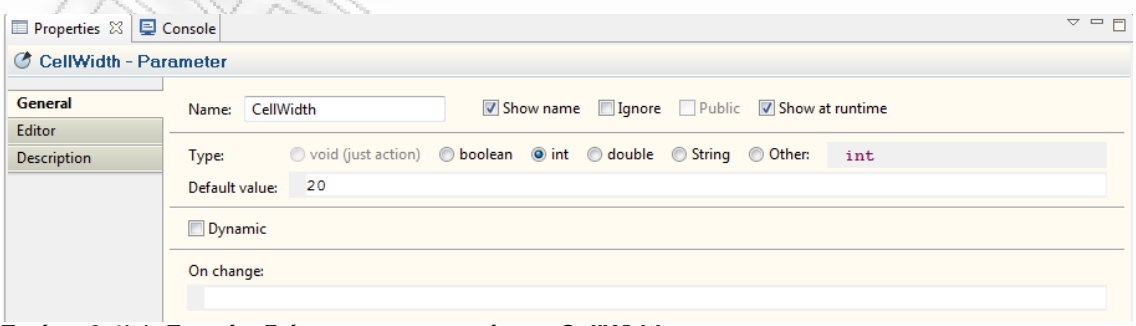
Εικόνα 6-3(δ): Οι παράμετροι της κλάσης Hare, συγκεντρωτικά.

Ο πίνακας 6-1 παρουσιάζει τις παραμέτρους της κλάσης. Η παράμετρος Natality ορίζει τη γεννητικότητα, τον ρυθμό γεννήσεων, τις γέννες του θηράματος ενώ η παράμετρος NumberPerBirth ορίζει τον αριθμό των νέων λαγών που γεννιούνται ανά γέννα. Η παράμετρος LifeExpectancy ορίζει την προσδοκώμενη διάρκεια ζωής των θηραμάτων.

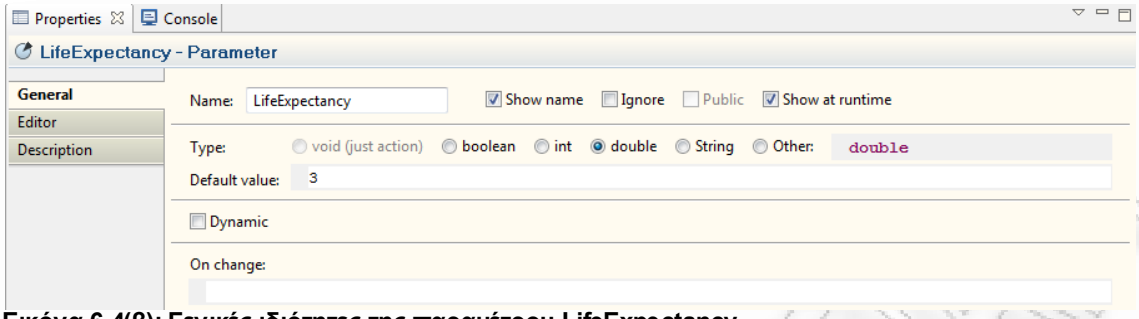
Παράμετροι της κλάσης Hare		
Όνομα παραμέτρου	Είδος Παραμέτρου	Αρχική τιμή παραμέτρου
Width	int	15
CellWidth	int	20
Natality	double	4
NumberPerBirth	int	5
LifeExpectancy	double	3
MaxPerCell	int	15

Πίνακας 6-1: Παράμετροι της κλάσης Hare.

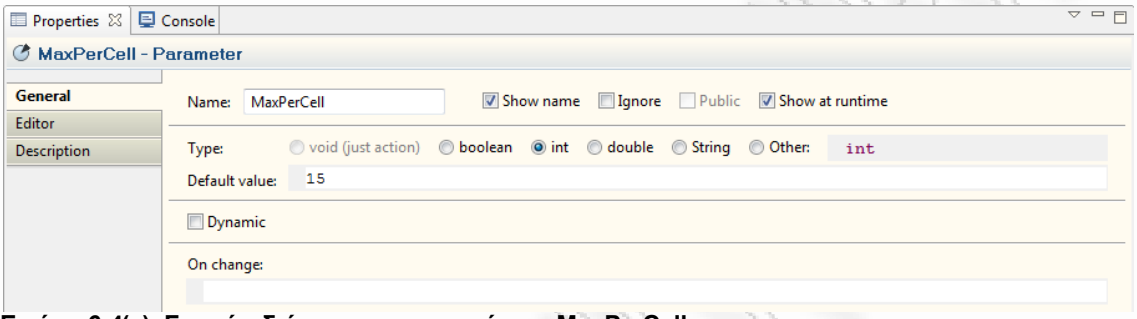
Η παράμετρος MaxPerCell ορίζει τον μέγιστο αριθμό των θηραμάτων που μπορούν να βρίσκονται ανά κελί στη διάρκεια της προσομοίωσης. Η απομονωμένη περιοχή στην οποία θα γίνει η προσομοίωση χωρίζεται σε ελαχίστου μήκους και πλάτους εμβαδά, CellWidth, τα οποία ονομάζονται κελιά. Πρέπει να είναι ξεκάθαρο κατά την προσομοίωση σε ποιά υποπεριοχή, σε ποιά κελιά ανήκουν οι πράκτορες λογισμικού. Ειδικά για αυτούς τους πράκτορες που βρίσκονται στα σύνορα των κελιών θέλουμε να είναι ξεκάθαρη η αντιστοίχιση τους ώστε να μην υπάρχουν φαινόμενα αλληλοκαλύψεων. Έτσι, κατά τη μοντελοποίηση και για λόγους προγραμματιστικής ευκολίας μειώνονται οι διαστάσεις της ελάχιστης περιοχής από CellWidth σε Width. Στις εικόνες από 6-4(α) έως και 6-4(στ) παρουσιάζονται οι ιδιότητες των παραμέτρων, όπως αυτές ορίζονται στο Λογισμικό.



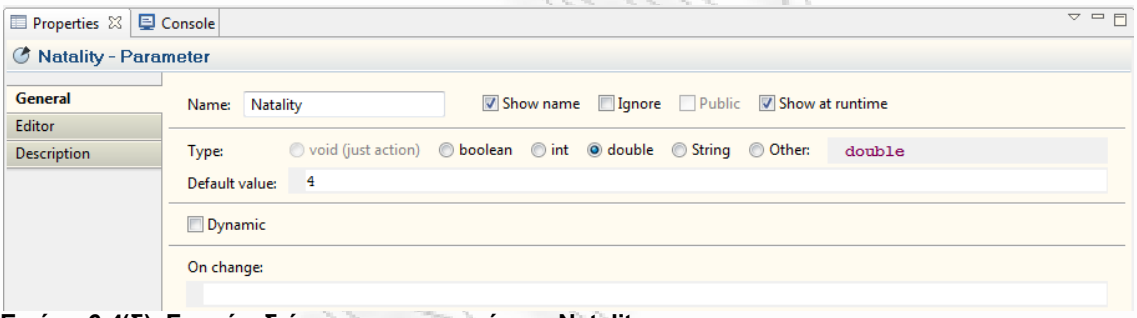
Εικόνα 6-4(α): Γενικές ιδιότητες της παραμέτρου CellWidth.



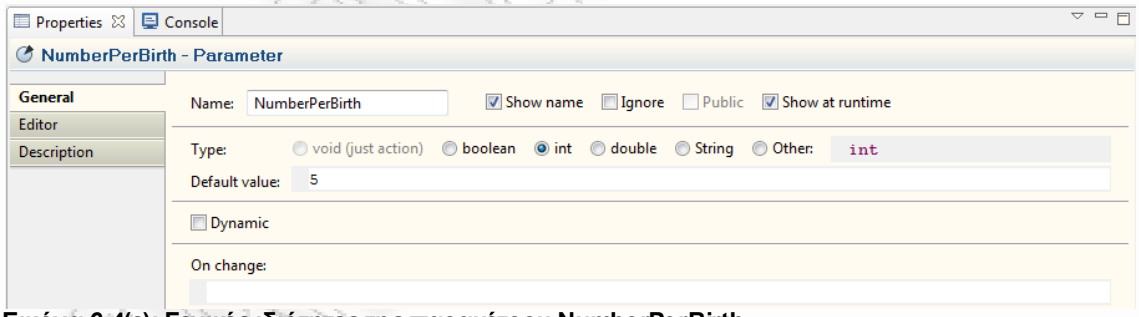
Εικόνα 6-4(β): Γενικές ιδιότητες της παραμέτρου LifeExpectancy.



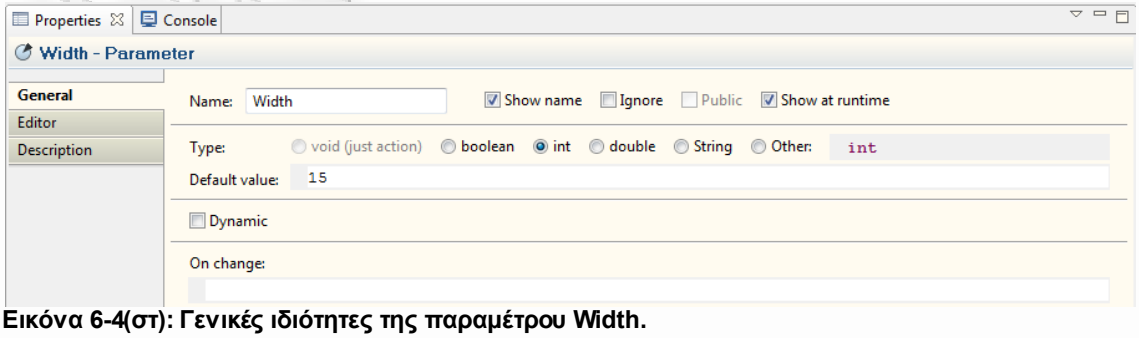
Εικόνα 6-4(γ): Γενικές ιδιότητες της παραμέτρου MaxPerCell.



Εικόνα 6-4(δ): Γενικές ιδιότητες της παραμέτρου Natality.

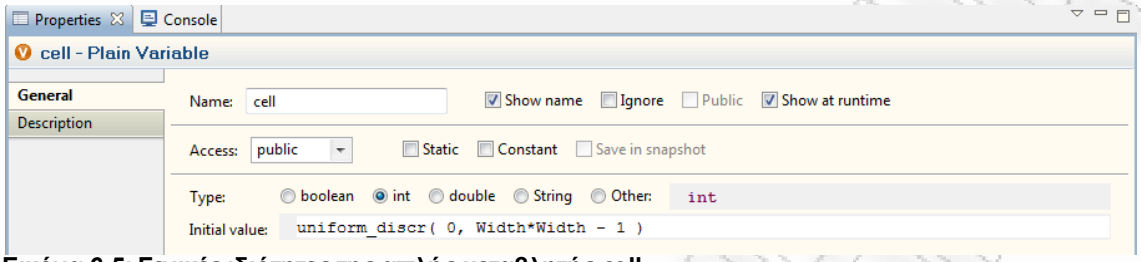


Εικόνα 6-4(ε): Γενικές ιδιότητες της παραμέτρου NumberPerBirth.



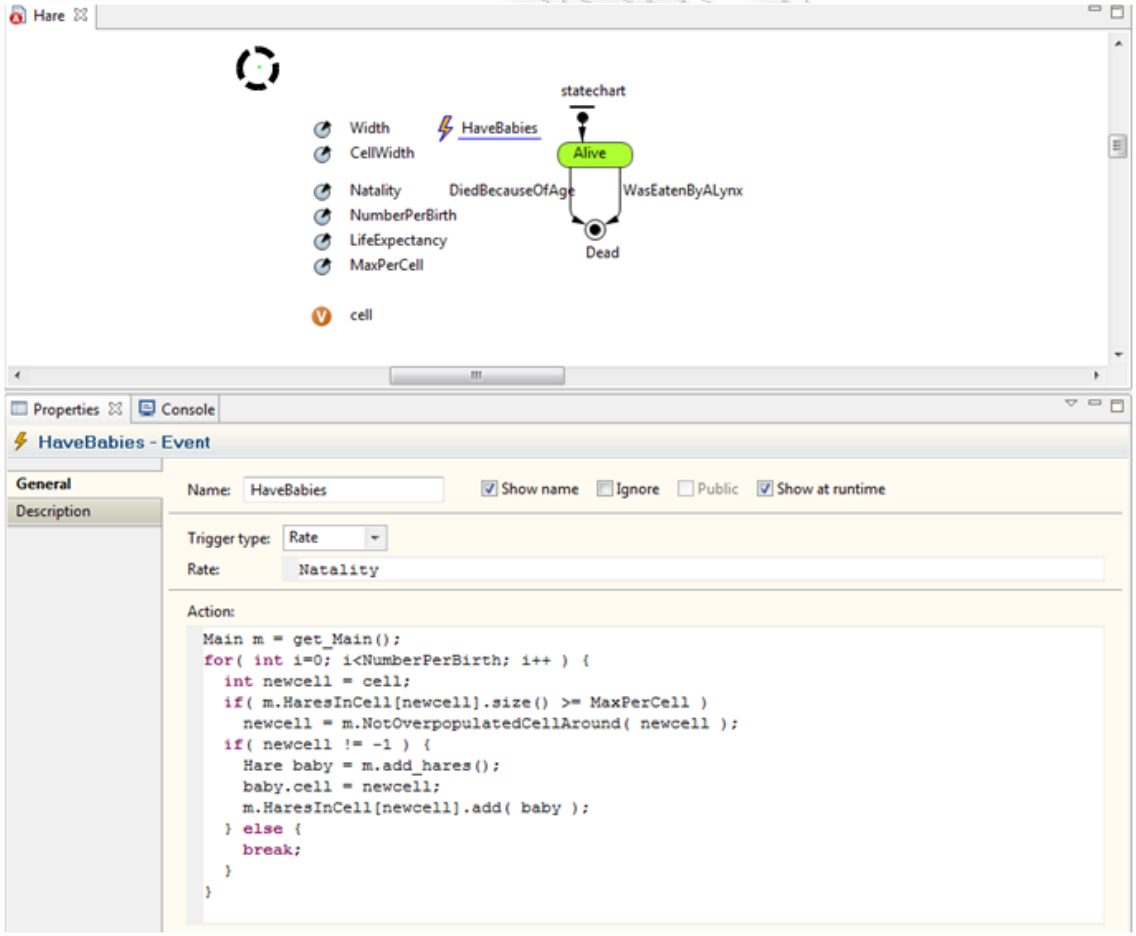
Εικόνα 6-4(στ): Γενικές ιδιότητες της παραμέτρου Width.

Η απλή μεταβλητή cell είναι αυτή που αναπαριστά την ελάχιστη περιοχή και ορίζει τον αριθμό των κελιών στα οποία χωρίζεται η απομονωμένη περιοχή της προσομοίωσης. Τα κελιά δημιουργούνται με την χρήση της διακριτής συνάρτησης ομοιόμορφης κατανομής, uniform_discr(), μιας από τις εσωτερικές συναρτήσεις του Λογισμικού, ενώ στην εικόνα 6-5 παρουσιάζονται οι ιδιότητες της απλής μεταβλητής, όπως αυτή ορίζεται στο Λογισμικό.



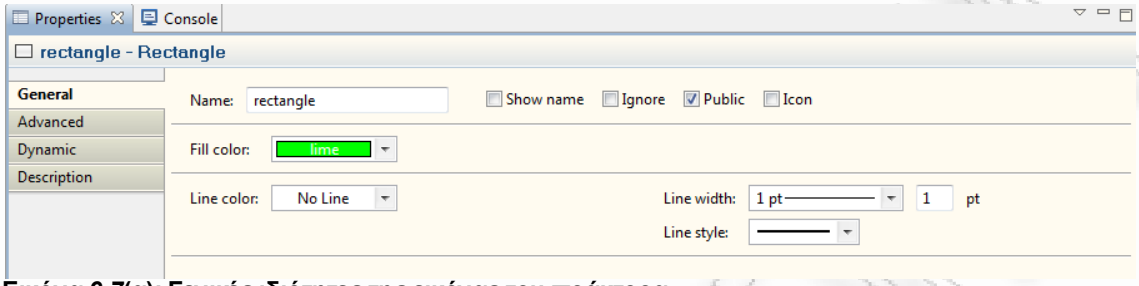
Εικόνα 6-5: Γενικές ιδιότητες της απλής μεταβλητής cell.

Το γεγονός HaveBabies ελέγχεται από τη μεταβλητή Natality, οπότε σε κάθε χρονική περίοδο το γεγονός θα επαναληφθεί κατά μέσο όρο τέσσερις (4) φορές. Με τη γέννηση των νέων θηραμάτων, αυτά θα πρέπει να κατανεμηθούν στα ήδη ορισμένα κελιά. Αν ο αριθμός των θηραμάτων ανά κελί ξεπεράσει τα δεκαπέντε (15), την τιμή της παραμέτρου MaxPerCell, τότε δημιουργούνται νέα κελιά.

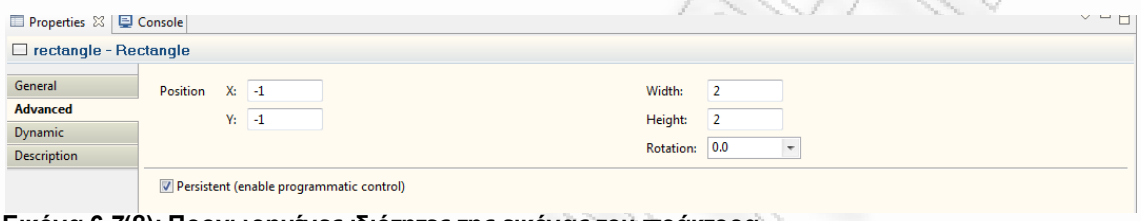


Εικόνα 6-6: Οι γενικές ιδιότητες του γεγονότος HaveBabies και ο κώδικας που το συνοδεύει.

Στην εικόνα 6-6 παρουσιάζονται οι γενικές ιδιότητες του γεγονότος HaveBabies. Μέσα στον διακεκομμένο κύκλο βρίσκεται το σχήμα του πράκτορα του θηράματος, ένα τετράγωνο πράσινου χρώματος, εικόνα 6-7(α) και 6-7(β).

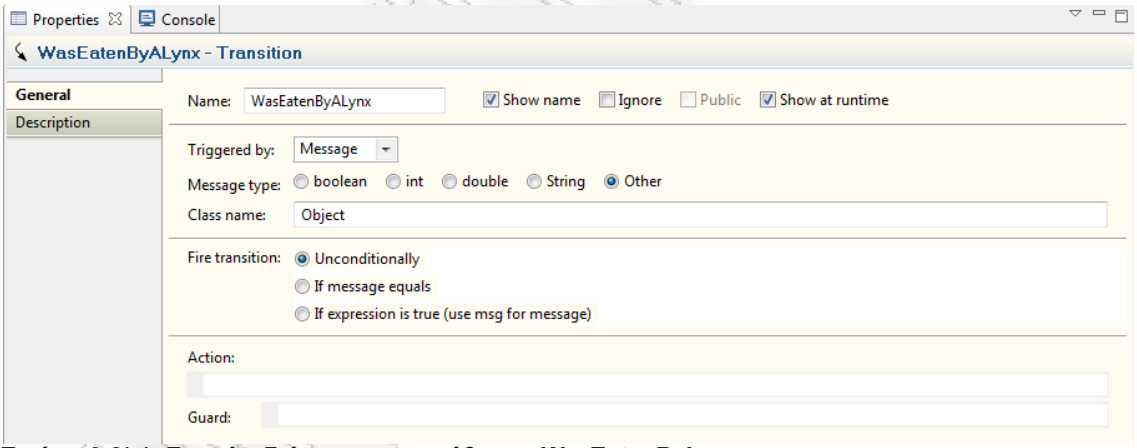


Εικόνα 6-7(α): Γενικές ιδιότητες της εικόνας του πράκτορα.



Εικόνα 6-7(β): Προχωρημένες ιδιότητες της εικόνας του πράκτορα.

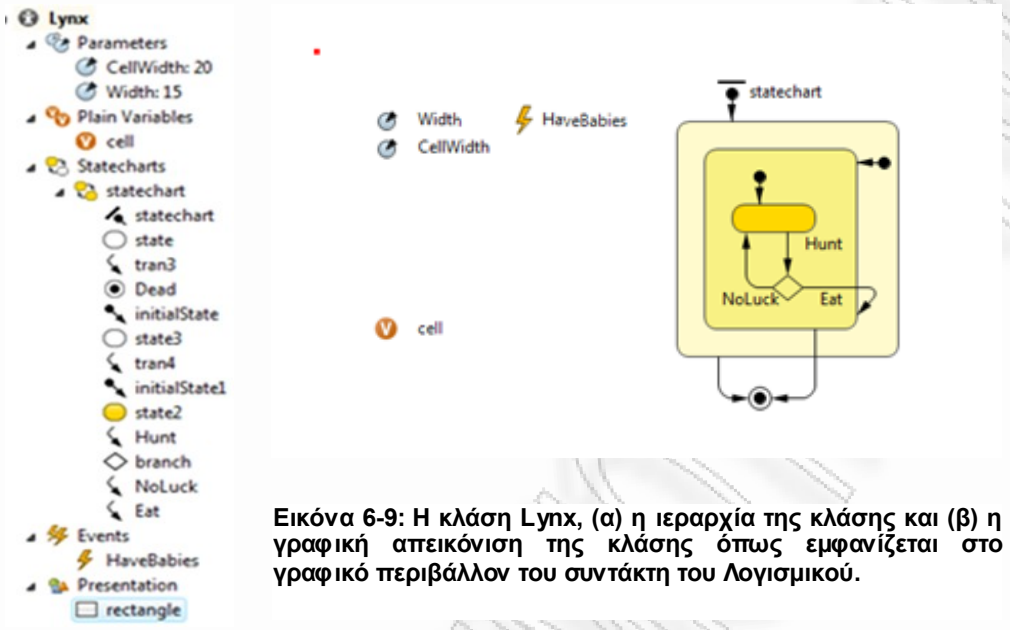
Ακολουθεί στην εικόνα 6-8(α), η παρουσίαση των ιδιοτήτων της μετάβασης WasEatenByALynx του διαγράμματος καταστάσεων της κλάσης όπως αυτές έχουν ορισθεί στο Λογισμικό. Οι ιδιότητες της μετάβασης DiedBecauseOfAge είναι ακριβώς οι ίδιες και δεν τις παρουσιάζουμε. Τέλος, στην εικόνα 6-8(β), δίνονται οι ιδιότητες της τελικής κατάστασης του διαγράμματος, Dead.



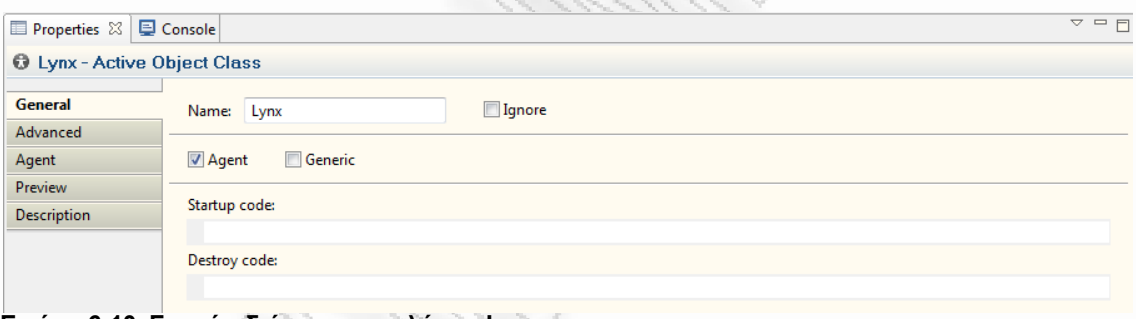
Εικόνα 6-8(α): Γενικές ιδιότητες της μετάβασης WasEatenByLynx.

Εικόνα 6-8(β): Τελική κατάσταση του διαγράμματος κατάστασης.

6.2 Η κλάση Lynx



Εικόνα 6-9: Η κλάση Lynx, (α) η ιεραρχία της κλάσης και (β) η γραφική απεικόνιση της κλάσης όπως εμφανίζεται στο γραφικό περιβάλλον του συντάκτη του Λογισμικού.



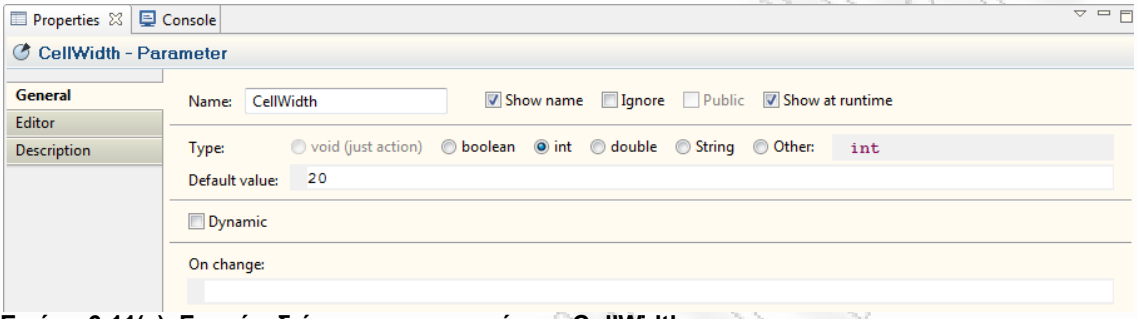
Εικόνα 6-10: Γενικές ιδιότητες της κλάσης Lynx.

Το διάγραμμα καταστάσεων της κλάσης Lynx παρουσιάζεται στην εικόνα 6-9. Το χαρακτηριστικό του διαγράμματος είναι το ότι αποτελείται από δύο (2) βρόχους, τον εσωτερικό και τον εξωτερικό, όπου ο ένας περιέχει τον άλλο και που συνδέονται μεταξύ τους. Η ύπαρξη, η υπόσταση του θηρευτή, (λύγκα, lynx), μοντελοποιείται από τον εξωτερικό βρόχο ενώ από τον εσωτερικό βρόχο μοντελοποιείται η κατάσταση του ως κυνηγού. Όσο βρίσκεται σε περίοδο κυνηγιού ο θηρευτής ψάχνει τη λεία του. Αν έχει τύχη βρίσκει το θήραμα, τρέφεται και συνεχίζει να ζει και να αναζητά την τροφής του. Στην περίπτωση που δεν καταφέρνει να βρει τη λεία του, ο θηρευτής συνεχίζει το κυνήγι και την αναζήτηση τροφής. Υπάρχει ένα χρονικό κατώφλι, στην αναζήτηση της τροφής, που αν για ορισμένο χρονικό διάστημα δεν τραφεί πεθαίνει από πείνα. Επίσης, ο θηρευτής πεθαίνει αν ξεπεραστεί το προσδόκιμο της ζωής του, (εξωτερικός βρόχος).

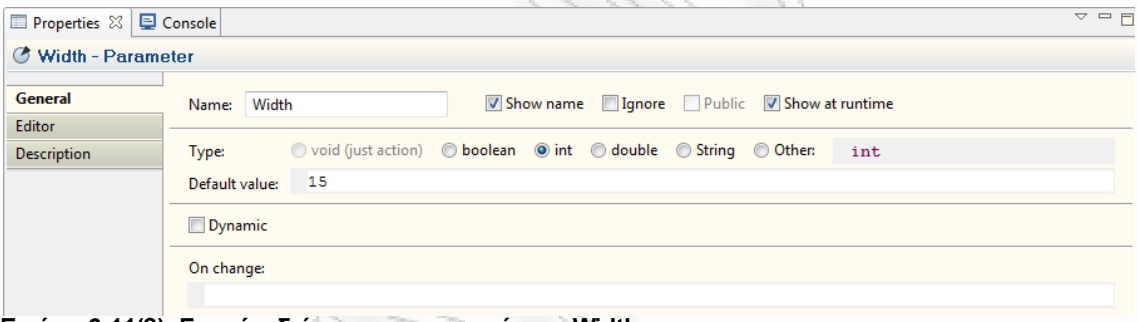
Παράμετροι της κλάσης Lynx		
Όνομα παραμέτρου	Είδος Παραμέτρου	Αρχική τιμή παραμέτρου
Width	int	15
CellWidth	int	20

Πίνακας 6-2: Παράμετροι της κλάσης Lynx

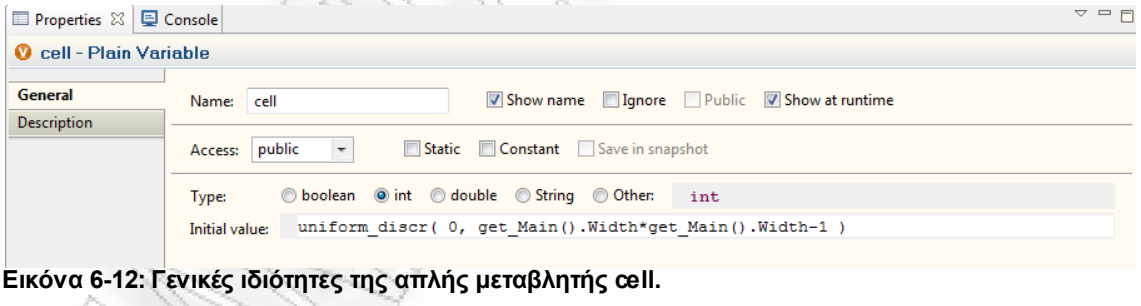
Οι παράμετροι Width και CellWidth που παρουσιάζονται στην εικόνα 6-11(α) και 6-11(β) και στον πίνακα 6-2, έχουν ακριβώς τις ίδιες ιδιότητες και εκτελούν την ίδια λειτουργία όπως και στην περίπτωση του πράκτορα του θηράματος, προηγούμενα. Επίσης, το ίδιο συμβαίνει και για την απλή μεταβλητή cell, εικόνα 6-12 αλλά και για το γεγονός HaveBabies, εικόνα 6-13, το οποίο βέβαια στην εδώ κλάση εξαρτάται από την παράμετρο LynxNatality. Στην εικόνα 6-14 παρουσιάζονται οι ιδιότητες μερικών στοιχείων του διαγράμματος των καταστάσεων και στην εικόνα 6-15 παρουσιάζονται οι ιδιότητες της εικόνας του πράκτορα κατά την παρουσίαση του στην προσομοίωση.



Εικόνα 6-11(α): Γενικές ιδιότητες της παραμέτρου CellWidth.

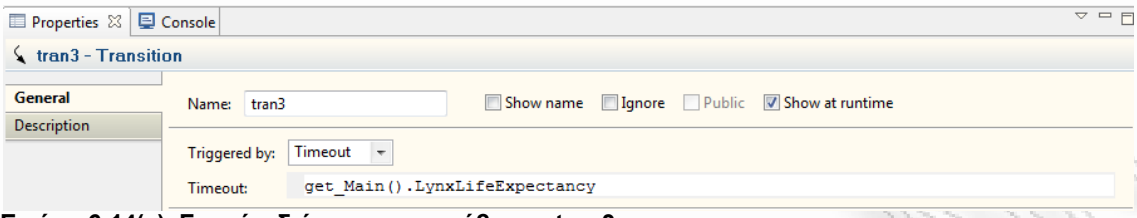


Εικόνα 6-11(β): Γενικές ιδιότητες της παραμέτρου Width.

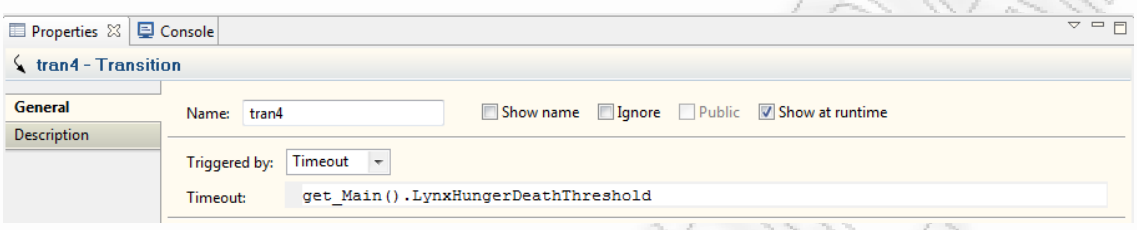


Εικόνα 6-12: Γενικές ιδιότητες της απλής μεταβλητής cell.

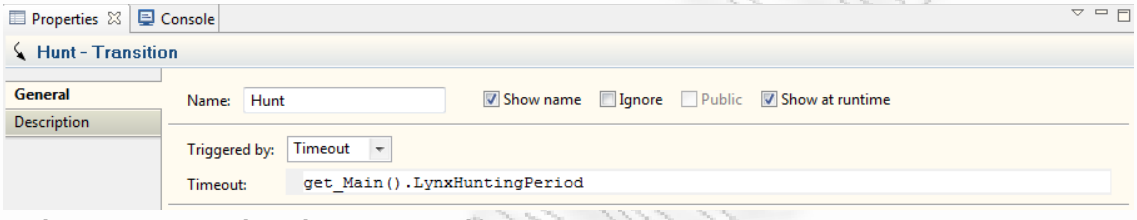
Εικόνα 6-13: Γενικές ιδιότητες του γεγονότος HaveBabies της κλάσης Lynx.



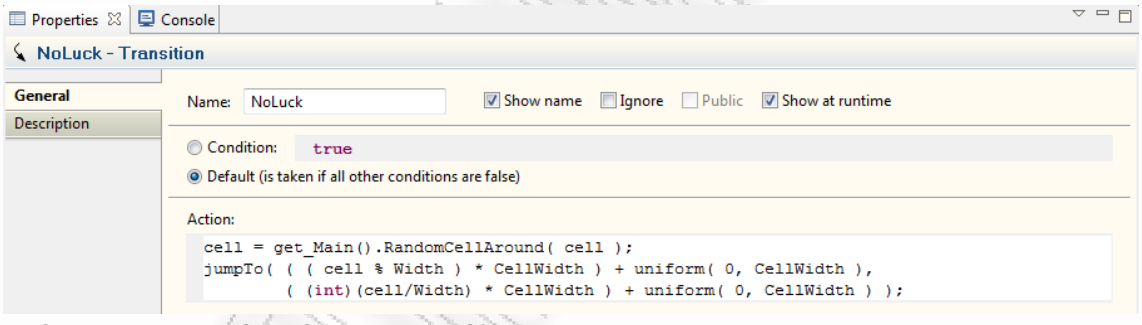
Εικόνα 6-14(α): Γενικές ιδιότητες της μετάβασης tran3.



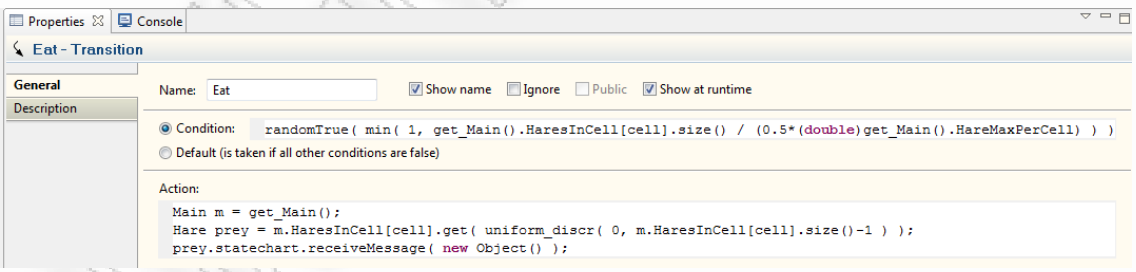
Εικόνα 6-14(β): Γενικές ιδιότητες της μετάβασης tran4.



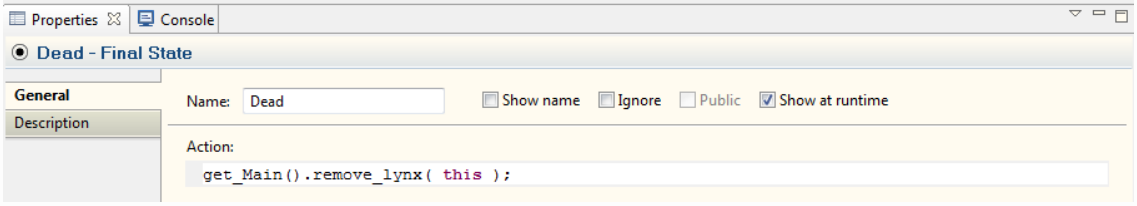
Εικόνα 6-14(γ): Γενικές ιδιότητες της μετάβασης Hunt.



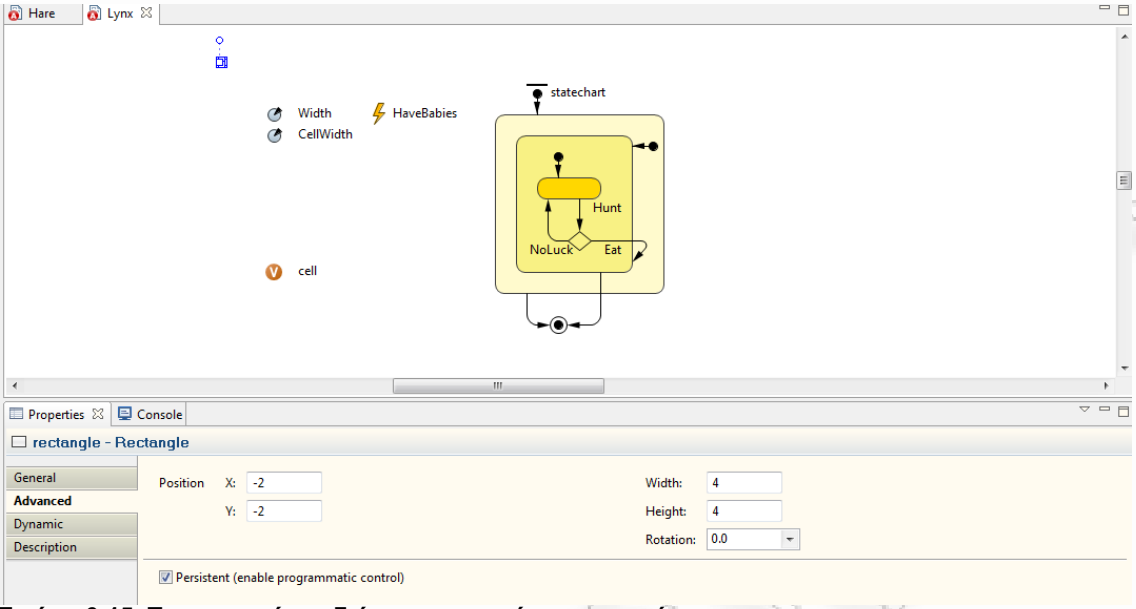
Εικόνα 6-14(δ): Γενικές ιδιότητες της μετάβασης NoLuck.



Εικόνα 6-14(ε): Γενικές ιδιότητες της μετάβασης Eat.



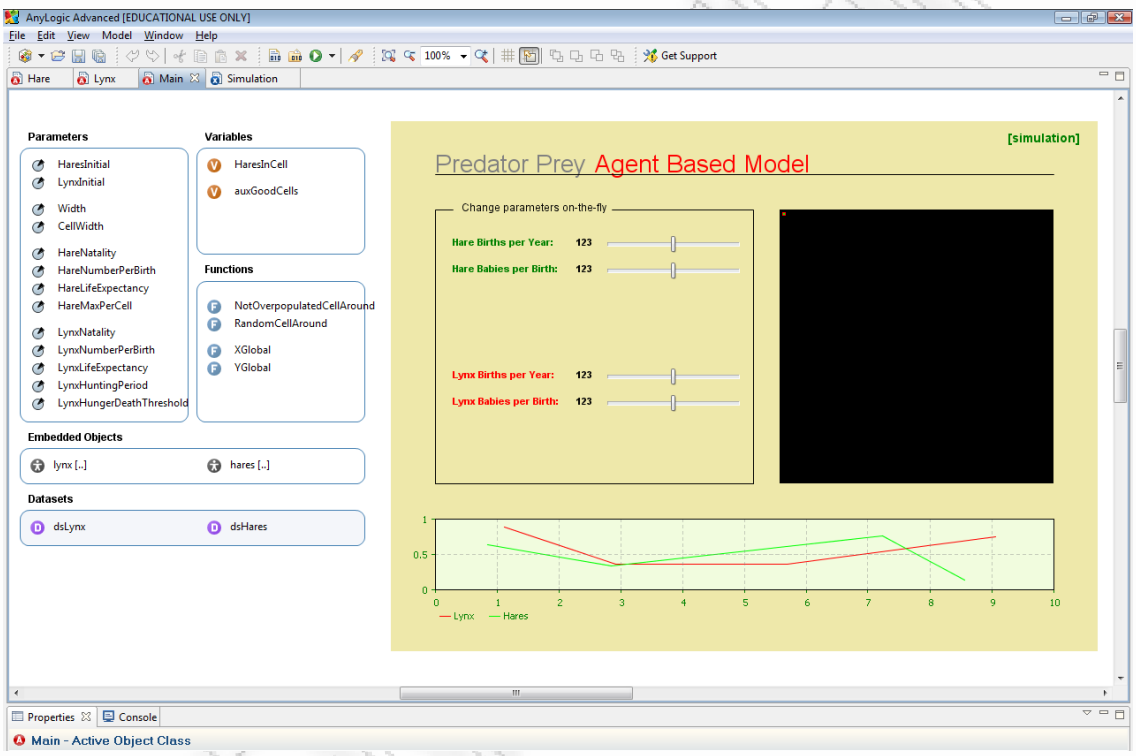
Εικόνα 6-14(στ): Γενικές ιδιότητες της τελικής κατάστασης Dead.



Εικόνα 6-15: Προχωρημένες ιδιότητες της εικόνας του πράκτορα.

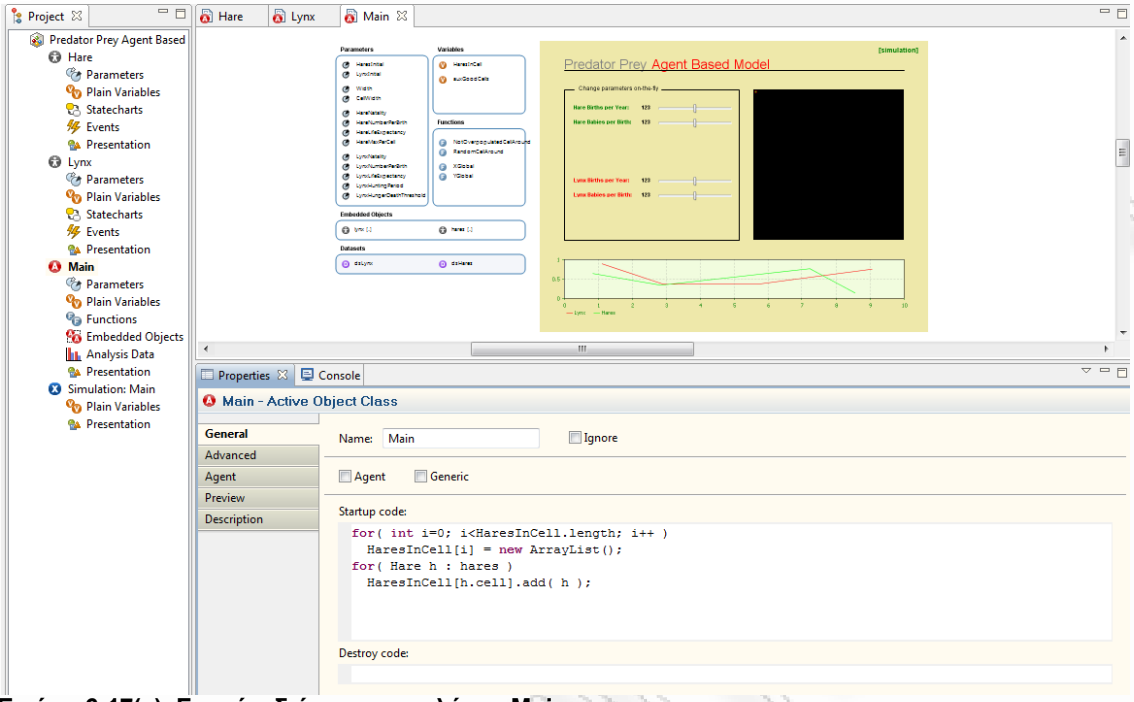
6.3 Η κλάση Main

Η κλάση ενεργών αντικειμένων Main, εικόνα 6-16, είναι αυτή που συνεργάζεται με τις άλλες κλάσεις του έργου για την ολοκλήρωση του μοντέλου. Αποτελείται από αριθμό παραμέτρων, απλές μεταβλητές, συναρτήσεις, ενσωματωμένα αντικείμενα και σύνολα δεδομένων. Επίσης, το γραφικό περιβάλλον αποτελείται από τέσσερις (4) ράβδους κύλισης, ένα (1) παράθυρο εμφάνισης γραφικών και ένα (1) διάγραμμα χρόνου. Στις εικόνες 6-17(α)-(γ), 6-18(α)-(ιγ) και 6-19(α)-(β) παρουσιάζονται οι ιδιότητες της κλάσης, οι γενικές ιδιότητες των παραμέτρων και οι γενικές ιδιότητες των απλών μεταβλητών αντίστοιχα. Στον πίνακα 6-3 βρίσκονται συγκεντρωμένες οι αρχικές τιμές των παραμέτρων και οι τύποι τους.

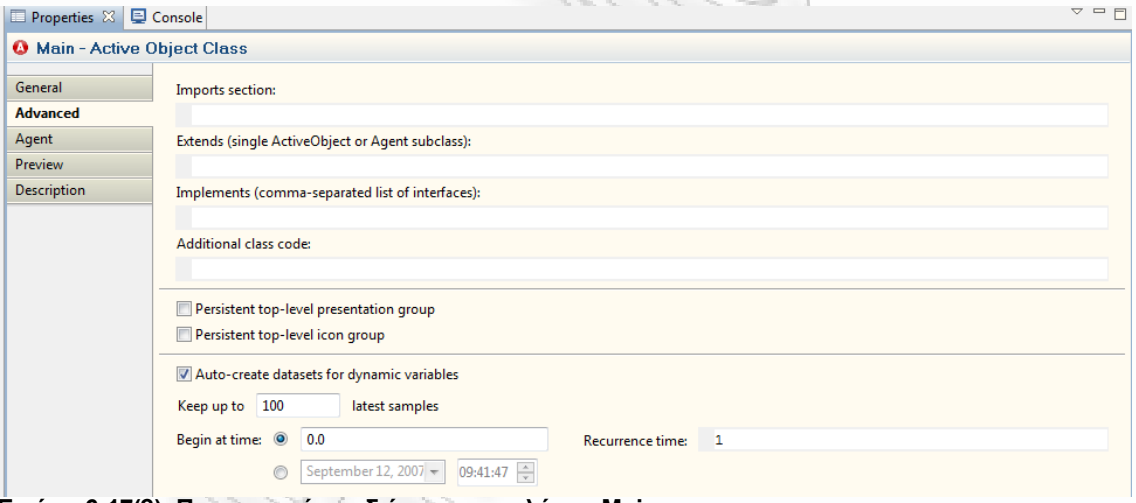


Εικόνα 6-16: Η κλάση Main όπως εμφανίζεται στον γραφικό συντάκτη του λογισμικού μετά την ολοκλήρωση της.

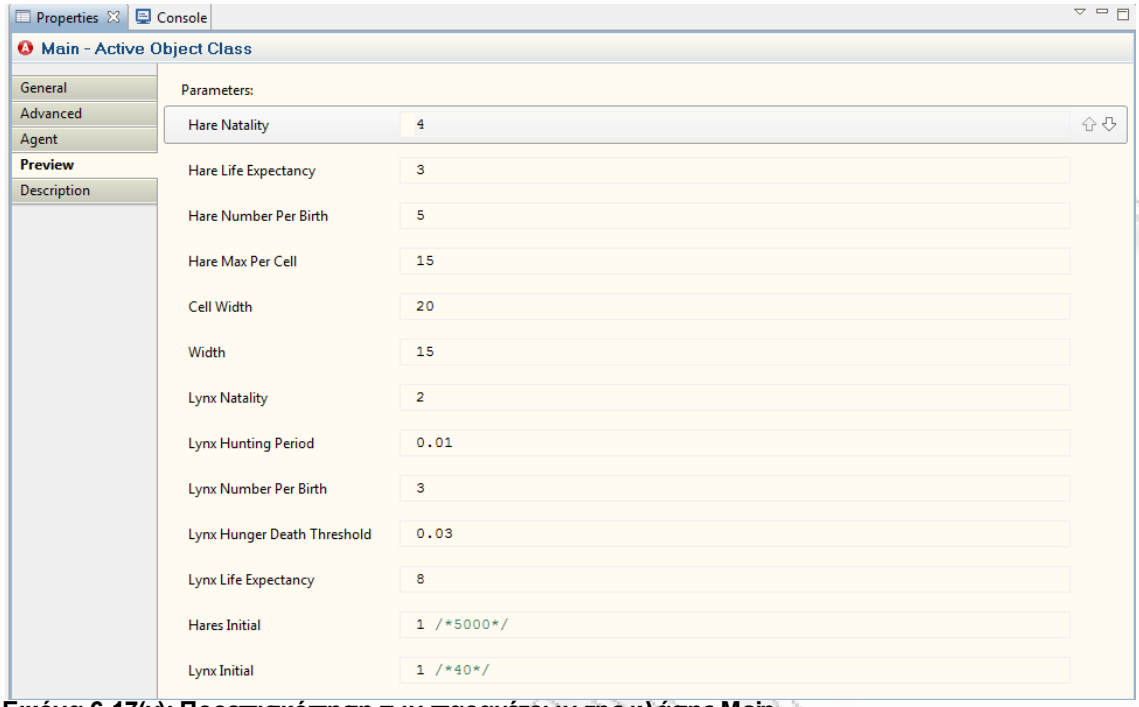
Η παράμετρος CellWidth και η Width έχει τα ίδια χαρακτηριστικά με αυτά των κλάσεων Hare και Lynx. Οι παράμετροι HareLifeExpectancy και LynxLifeExpectancy αφορούν στο ευδόκιμο των λαγών και των λυγκών αντίστοιχα. Η παράμετρος HareMaxPerCell καθορίζει το μέγιστο αριθμό λαγών σε κάθε κελί. Οι παράμετροι HareNatality, HareNumberPerBirth, LynxNatality και LynxNumberPerBirth έχουν σχέση με το ρυθμό γεννήσεων και τον αριθμό των γεννήσεων σε θηράματα και θηρευτές, αντίστοιχα. Η παράμετρος LynxHungerDeathThreshold ορίζει το χρόνο μετά τον οποίο ο θηρευτής πεθαίνει από λοιμό ενώ η παράμετρος LynxHuntingPeriod αφορά στον καθορισμό της κυνηγετικής περιόδου, στον ορισμό του χρόνου κατά τον οποίο ο θηρευτής αναζητά τη λεία του. Οι παράμετροι HaresInitial και LynxInitial ορίζουν τον αρχικό αριθμό των πρακτόρων λογισμικού των θηραμάτων και των θηρευτών αντίστοιχα.



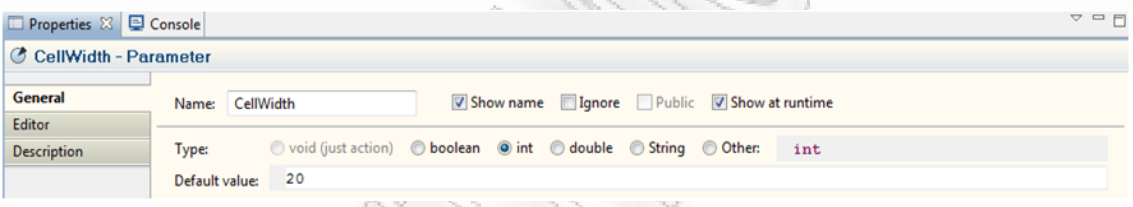
Εικόνα 6-17(α): Γενικές ιδιότητες της κλάσης Main.



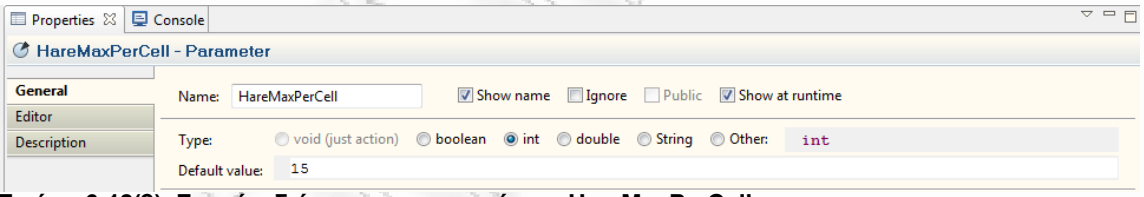
Εικόνα 6-17(β): Προχωρημένες ιδιότητες της κλάσης Main.



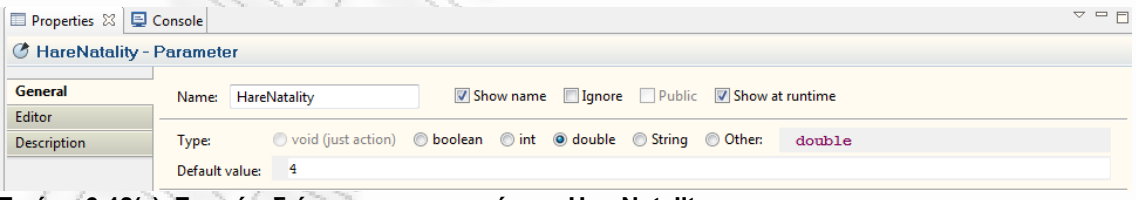
Εικόνα 6-17(γ): Προεπισκόπηση των παραμέτρων της κλάσης Main.



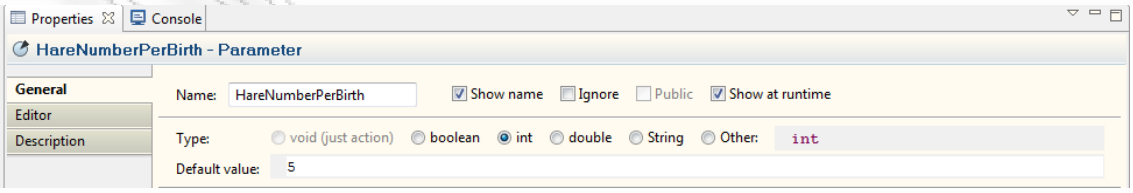
Εικόνα 6-18(α): Γενικές ιδιότητες της παραμέτρου CellWidth.



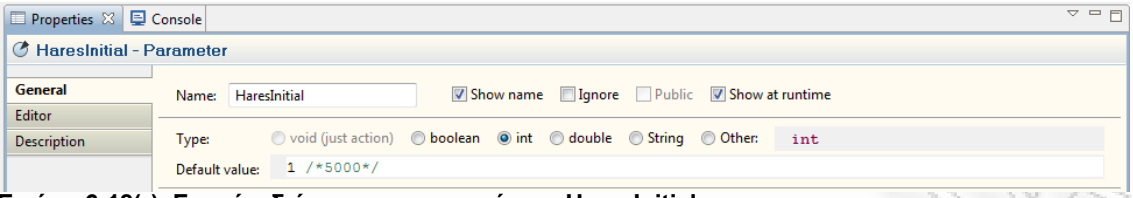
Εικόνα 6-18(β): Γενικές ιδιότητες της παραμέτρου HareMaxPerCell.



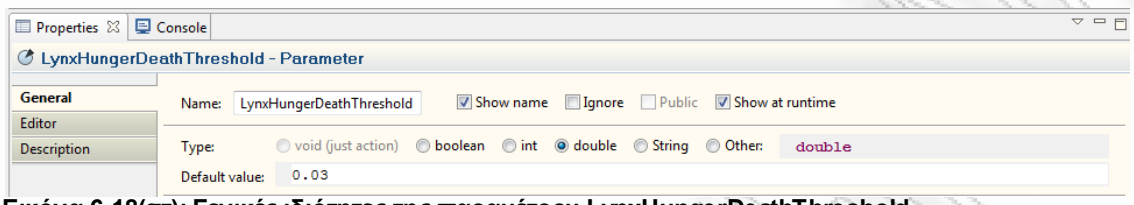
Εικόνα 6-18(γ): Γενικές ιδιότητες της παραμέτρου HareNatality.



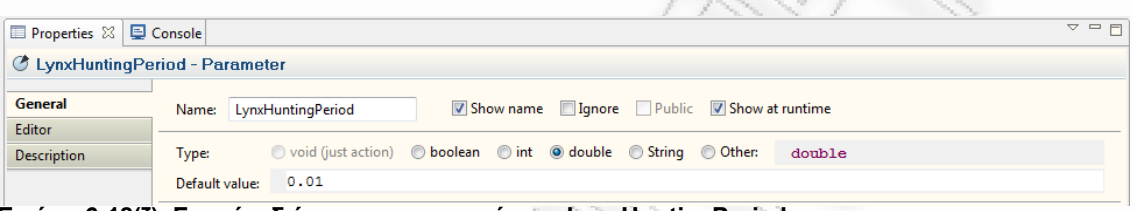
Εικόνα 6-18(δ): Γενικές ιδιότητες της παραμέτρου HareNumberPerBirth.



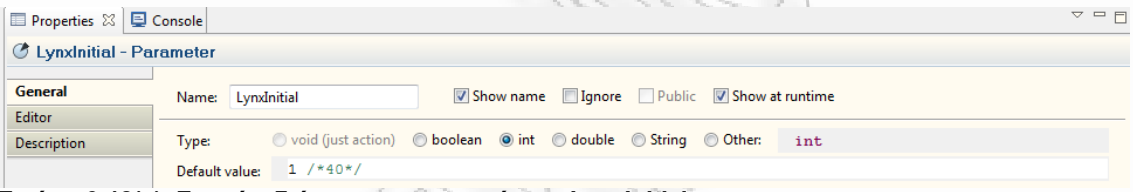
Εικόνα 6-18(ε): Γενικές ιδιότητες της παραμέτρου HaresInitial.



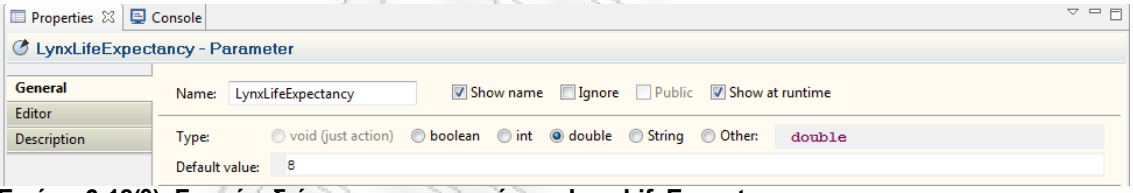
Εικόνα 6-18(στ): Γενικές ιδιότητες της παραμέτρου LynxHungerDeathThreshold.



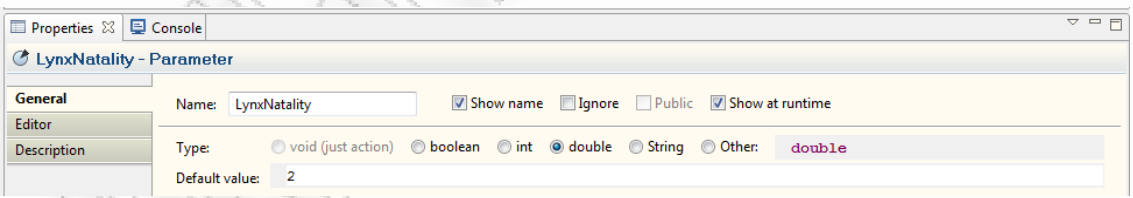
Εικόνα 6-18(ζ): Γενικές ιδιότητες της παραμέτρου LynxHuntingPeriod.



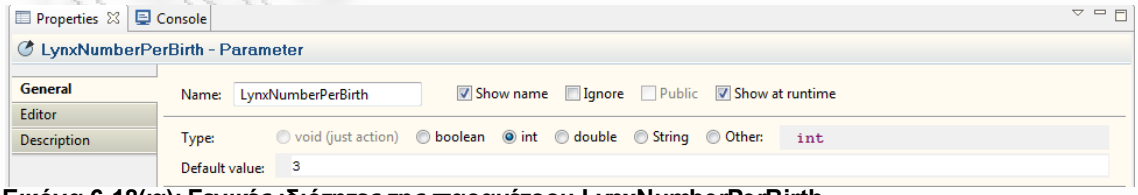
Εικόνα 6-18(η): Γενικές ιδιότητες της παραμέτρου LynxInitial.



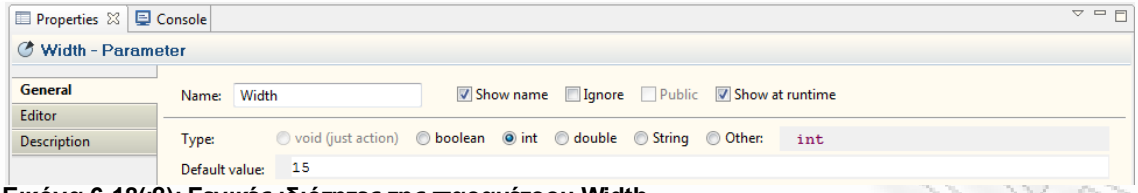
Εικόνα 6-18(θ): Γενικές ιδιότητες της παραμέτρου LynxLifeExpectancy.



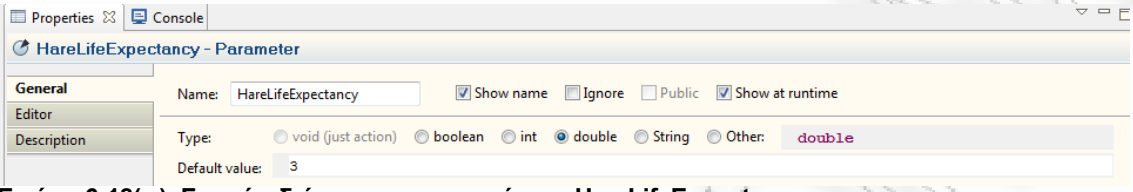
Εικόνα 6-18(ι): Γενικές ιδιότητες της παραμέτρου LynxNatality.



Εικόνα 6-18(ια): Γενικές ιδιότητες της παραμέτρου LynxNumberPerBirth.



Εικόνα 6-18(ιβ): Γενικές ιδιότητες της παραμέτρου Width.

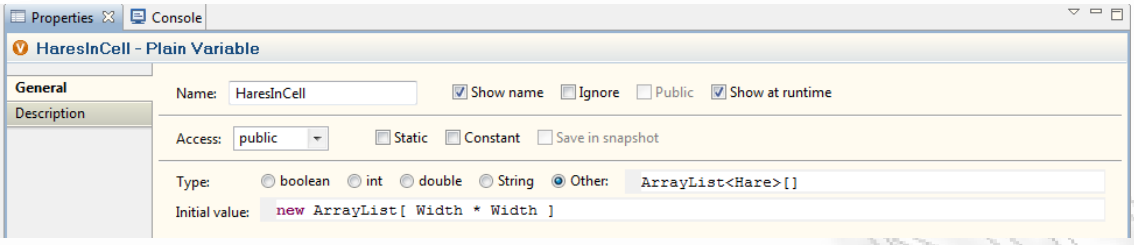


Εικόνα 6-18(ιγ): Γενικές ιδιότητες της παραμέτρου HareLifeExpectancy.

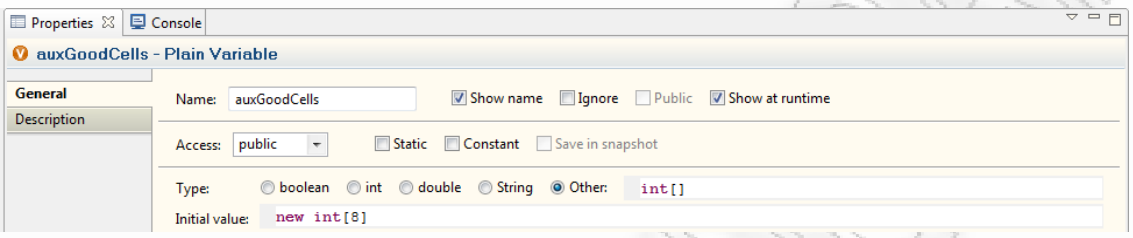
Παράμετροι του μοντέλου		
Όνομα παραμέτρου	Είδος Παραμέτρου	Αρχική τιμή παραμέτρου
CellWidth	int	20
Width	Int	15
HareLifeExpectancy	Double	3
HareMaxPerCell	Int	15
HareNatality	Double	4
HareNumberPerBirth	Int	5
HaresInitial	Int	1 (αντιστοιχεί σε 5000)
LynxHungerDeathThreshold	Double	0.03
LynxHuntingPeriod	Double	0.01
LynxNatality	Double	2
LynxInitial	Int	1 (αντιστοιχεί σε 40)
LynxLifeExpectancy	Double	8
LynxNumberPerBirth	Int	3

Πίνακας 6-3: Οι παράμετροι της κλάσης Main.

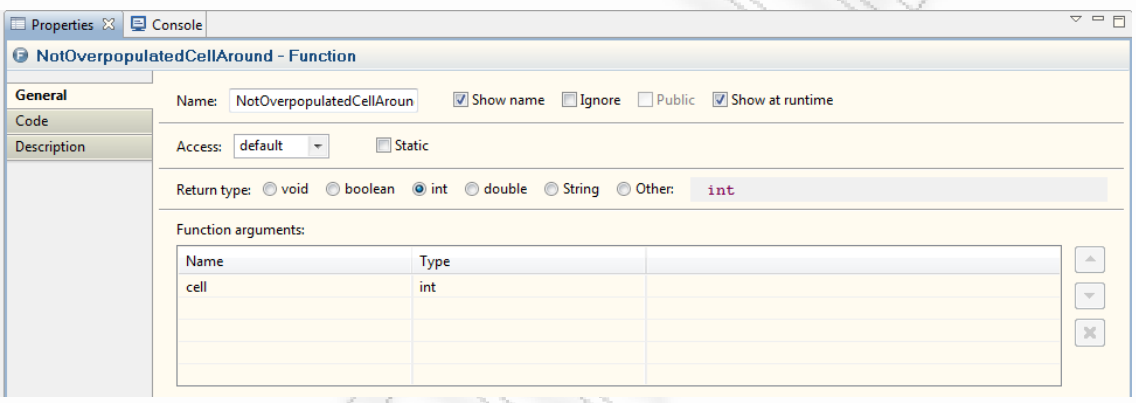
Στην εικόνα 6-19(α) η απλή μεταβλητή HaresInCell είναι τύπου λίστας, ArrayList και σε αυτή καταχωρείται ο αριθμός των θηραμάτων σε κάθε κελί. Στην εικόνα 6-19(β) η απλή μεταβλητή auxGoodCells είναι βοηθητική μεταβλητή τύπου πίνακα ακεραίων της συνάρτησης NotOverpopulatedCellAround. Η συνάρτηση NotOverpopulatedCellAround, εικόνα 6-20(α) και (β), ελέγχει αν ο αριθμός των θηραμάτων ανά κελί υπερβαίνει τον αριθμό που έχουμε θέσει ως μέγιστο. Στην περίπτωση της υπέρβασης δημιουργούνται νέα κελιά. Η συνάρτηση RandomCellAround, εικόνα 6-21(α) και (β), είναι τύπου int και είναι αυτή που καθορίζει τις διαστάσεις των νέων κελιών. Οι συναρτήσεις XGlobal και YGlobal, 6-22(α) και (β) και 6-23(α) και (β) αντίστοιχα, τις συντεταγμένες των κελιών κατά την προσομοίωση.



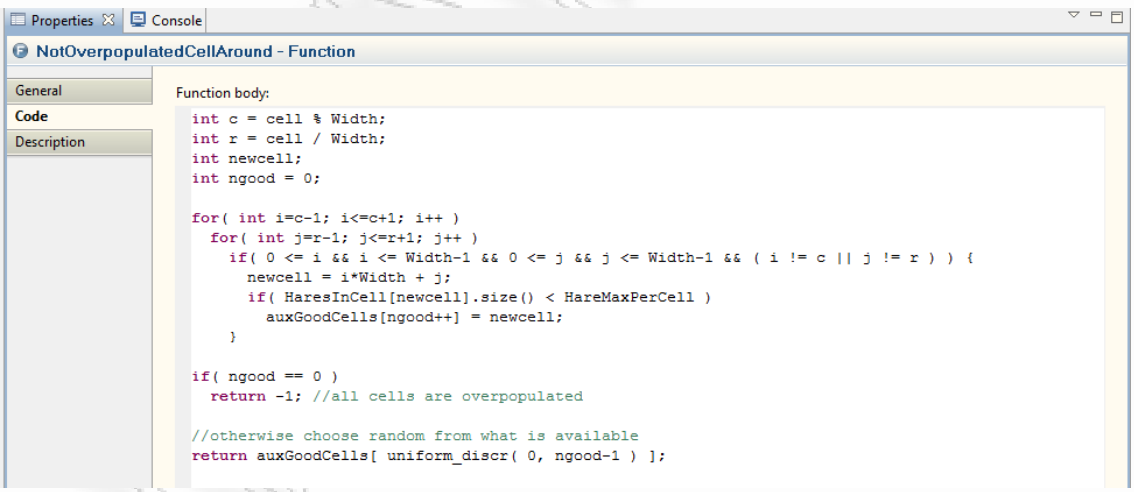
Εικόνα 6-19(α): Γενικές ιδιότητες της απλής μεταβλητής HaresInCell.



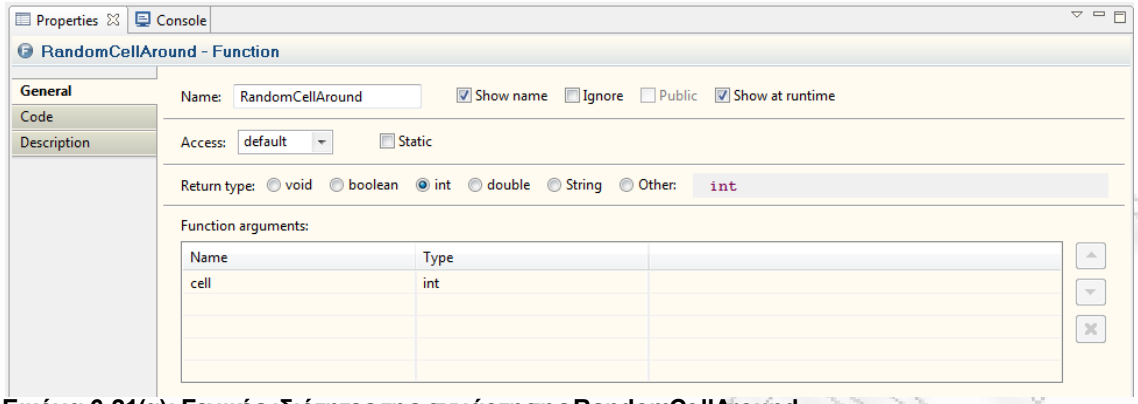
Εικόνα 6-19(β): Γενικές ιδιότητες της απλής μεταβλητής auxGoodCells.



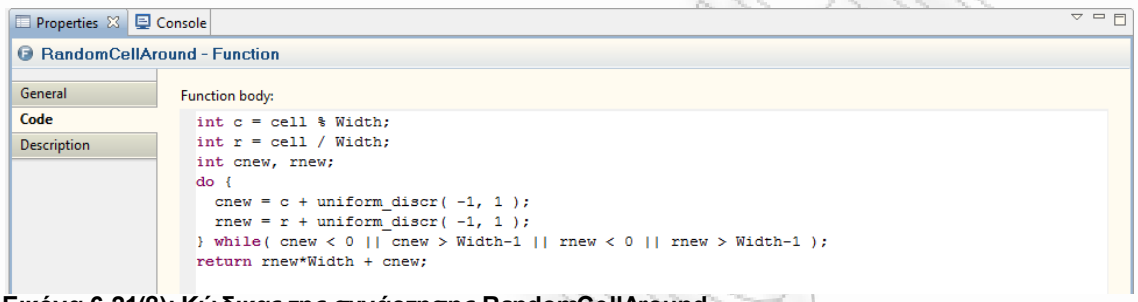
Εικόνα 6-20(α): Γενικές ιδιότητες της συνάρτησης NotOverpopulatedCellAround.



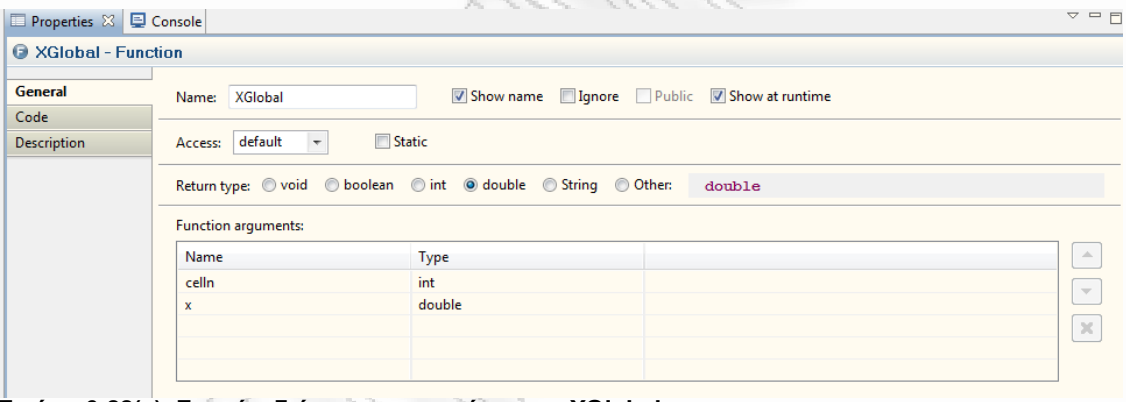
Εικόνα 6-20(β): Κώδικας της συνάρτησης NotOverpopulatedCellAround.



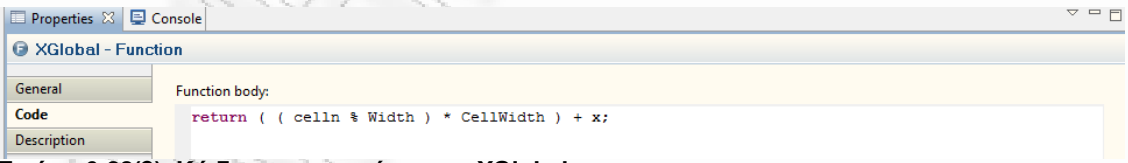
Εικόνα 6-21(α): Γενικές ιδιότητες της συνάρτησης RandomCellAround.



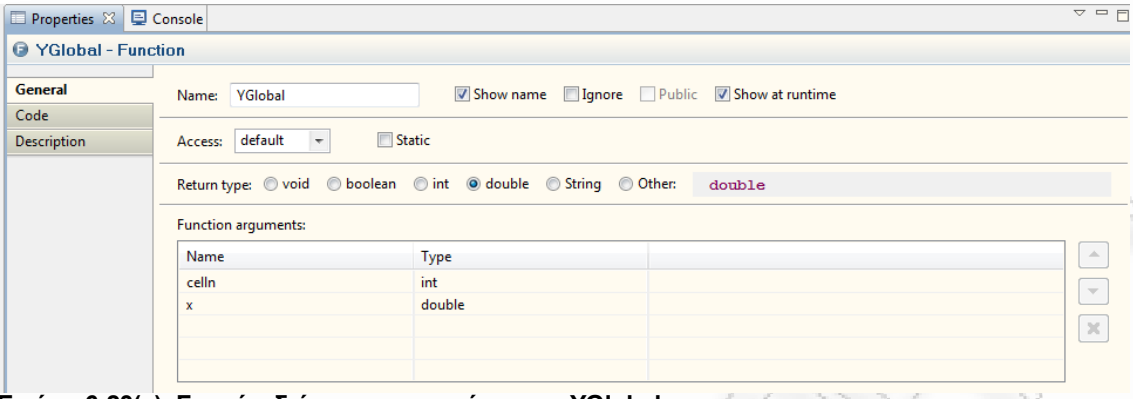
Εικόνα 6-21(β): Κώδικας της συνάρτησης RandomCellAround.



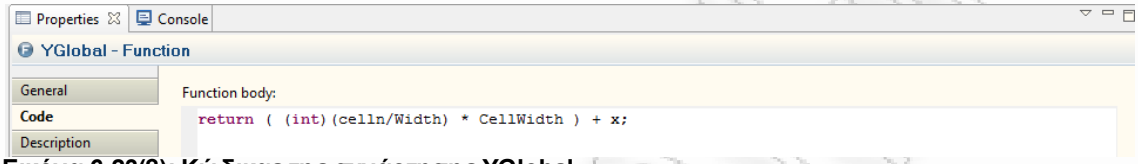
Εικόνα 6-22(α): Γενικές ιδιότητες της συνάρτησης XGlobal.



Εικόνα 6-22(β): Κώδικας της συνάρτησης XGlobal.

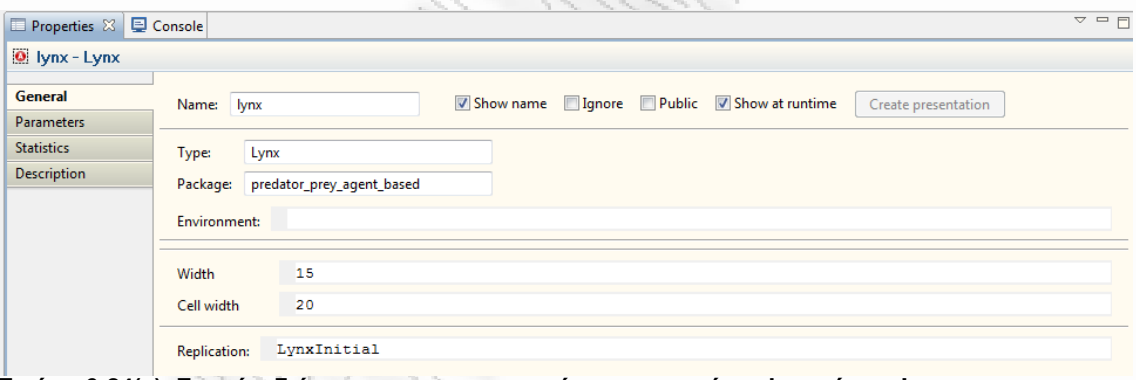


Εικόνα 6-23(α): Γενικές ιδιότητες της συνάρτησης YGlobal.

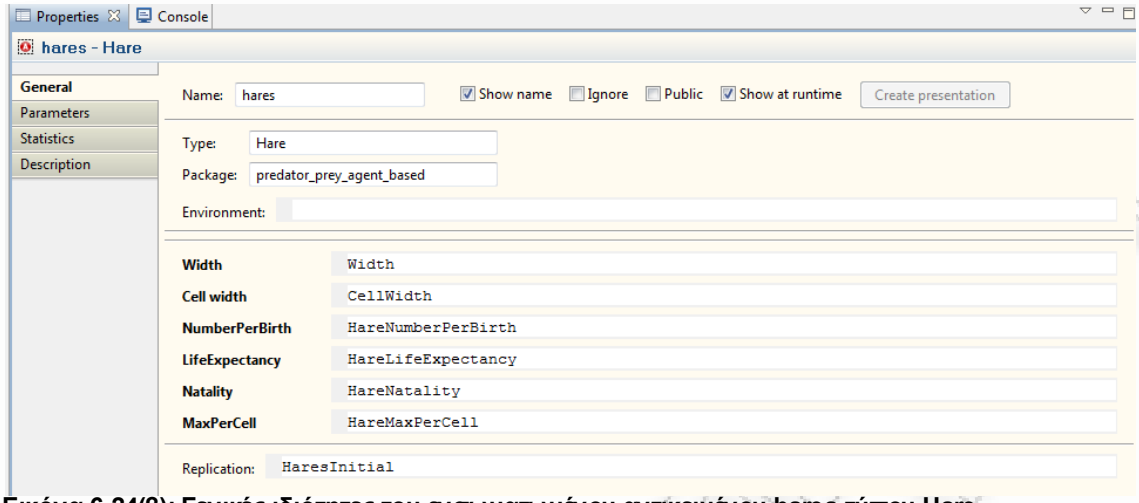


Εικόνα 6-23(β): Κώδικας της συνάρτησης YGlobal.

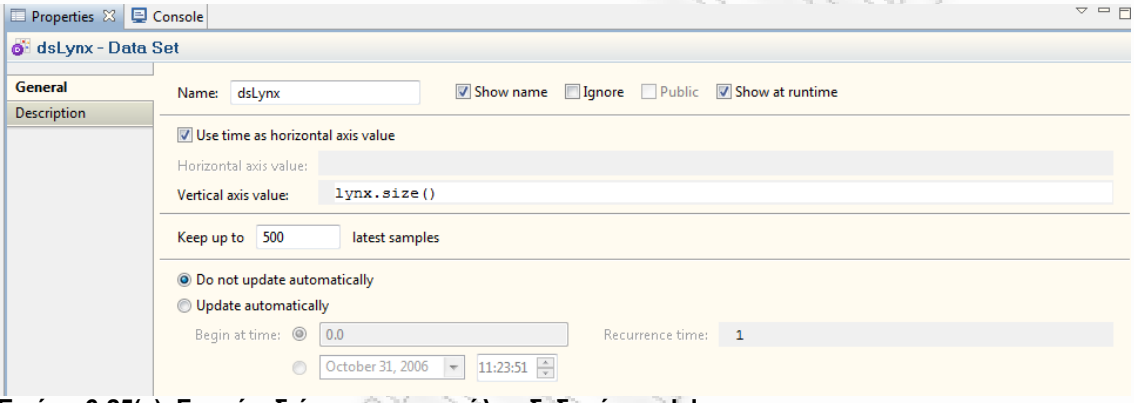
Η εικόνα 6-24(α) και η 6-24(β) παρουσιάζουν τη διασύνδεση των στιγμιότυπων των πρακτόρων λογισμικού που θα χρησιμοποιηθούν από την κλάση Main στο μοντέλο με τις κλάσεις των θηρευτών και των θηραμάτων αντίστοιχα. Στην εικόνα 6-25(α) και (β) παρουσιάζονται οι ιδιότητες των συνόλων δεδομένων που θα χρησιμοποιηθούν για τη γραφική αναπαράσταση των αριθμών των πληθυσμών των λυγκών και των λαγών αντίστοιχα.



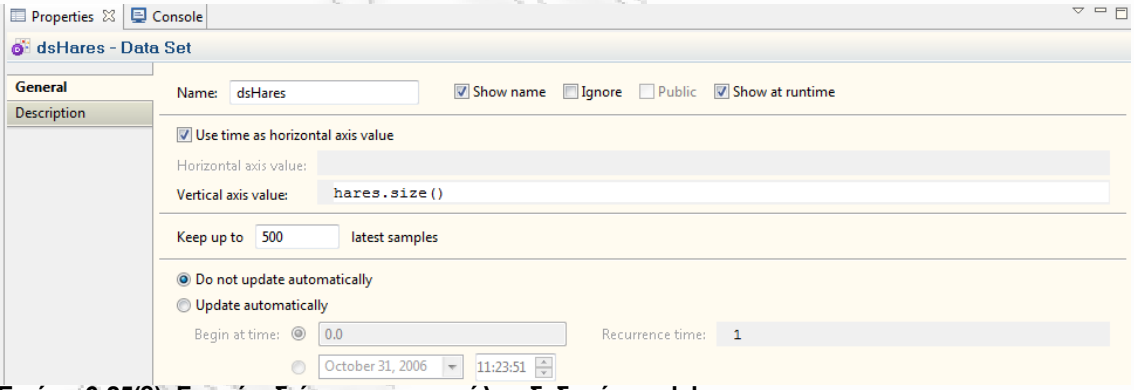
Εικόνα 6-24(α): Γενικές ιδιότητες του ενσωματωμένου αντικειμένου lynx τύπου Lynx.



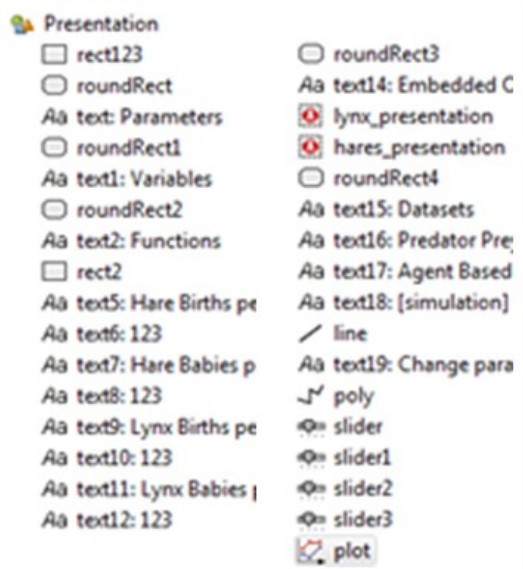
Εικόνα 6-24(β): Γενικές ιδιότητες του ενσωματωμένου αντικειμένου hares τύπου Hare.



Εικόνα 6-25(α): Γενικές ιδιότητες του συνόλου δεδομένων dslynx.



Εικόνα 6-25(β): Γενικές ιδιότητες του συνόλου δεδομένων dshares.

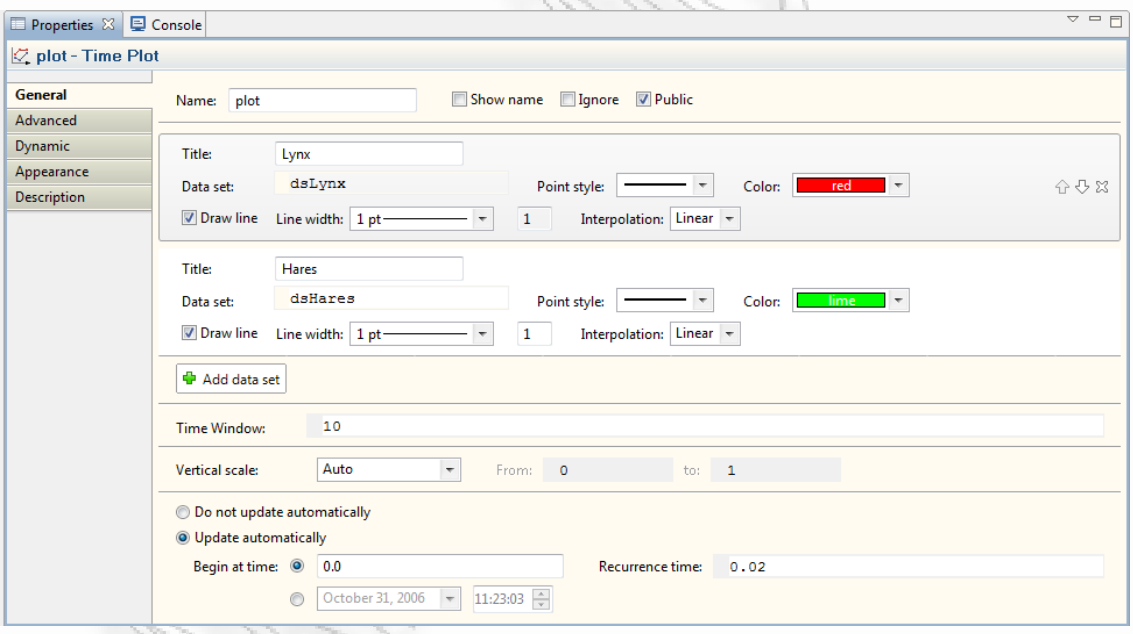


Εικόνα 6-26: Η ιεραρχική δομή των στοιχείων παρουσίασης της κλάσης Main.

Η κλάση Main ολοκληρώνεται με την δημιουργία της γραφικής οπτικοποίησης της. Η ιεραρχική δομή των στοιχείων παρουσίασης που συνθέτουν την κλάση της παρουσίασης παρουσιάζονται στην εικόνα 6-26.

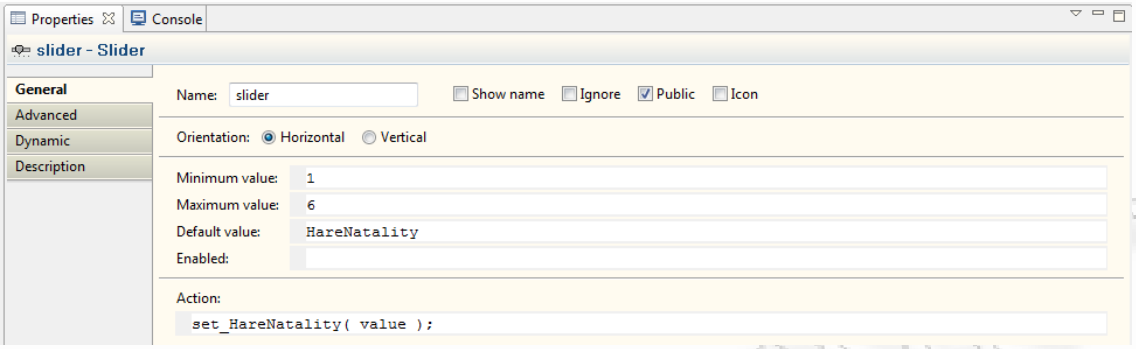
Στην εικόνα 6-27(α) - (β) δίνονται οι γενικές και οι προχωρημένες ιδιότητες του διαγράμματος χρόνου όπου εμφανίζονται οι πληθυσμοί των θηρευτών και των θηραμάτων στην πορεία της προσομοίωσης.

Στη συνέχεια, στην εικόνα 6-28(α) - (δ) παρουσιάζονται οι γενικές ιδιότητες των ράβδων κύλισης με τις οποίες οι τιμές των παραμέτρων HareNatality, HareNumberPerBirth, LynxNatality και LynxNumberPerBirth μεταβάλλονται μέσα σε ένα εύρος τιμών που καθορίζεται από τα πεδία minimum και maximum value.

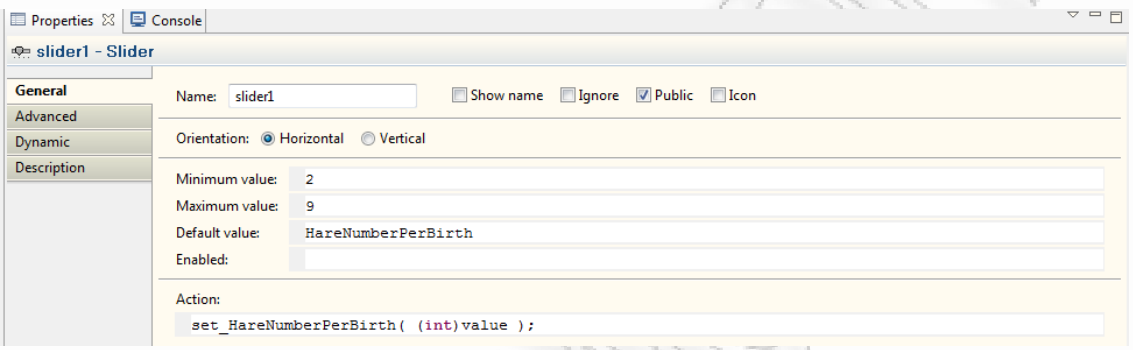


Εικόνα 6-27(α): Γενικές ιδιότητες του διαγράμματος χρόνου.

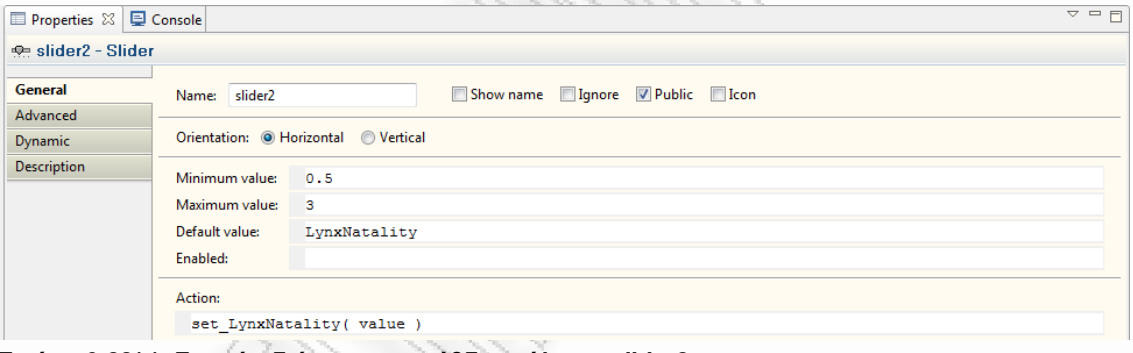
Εικόνα 6-27(β): Προχωρημένες ιδιότητες του διαγράμματος χρόνου.



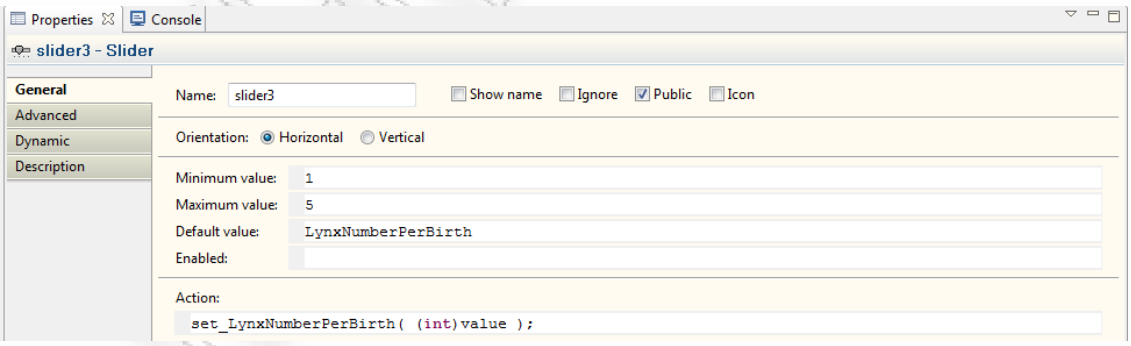
Εικόνα 6-28(α): Γενικές ιδιότητες της ράβδου κύλισης slider.



Εικόνα 6-28(β): Γενικές ιδιότητες της ράβδου κύλισης slider1.

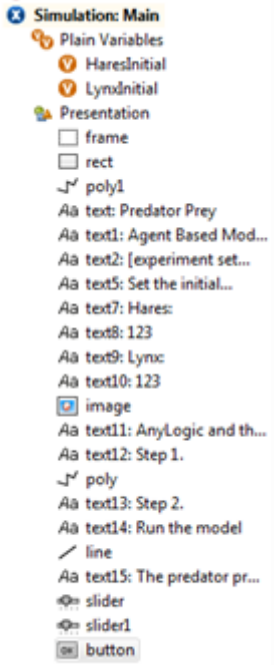


Εικόνα 6-28(γ): Γενικές ιδιότητες της ράβδου κύλισης slider2.



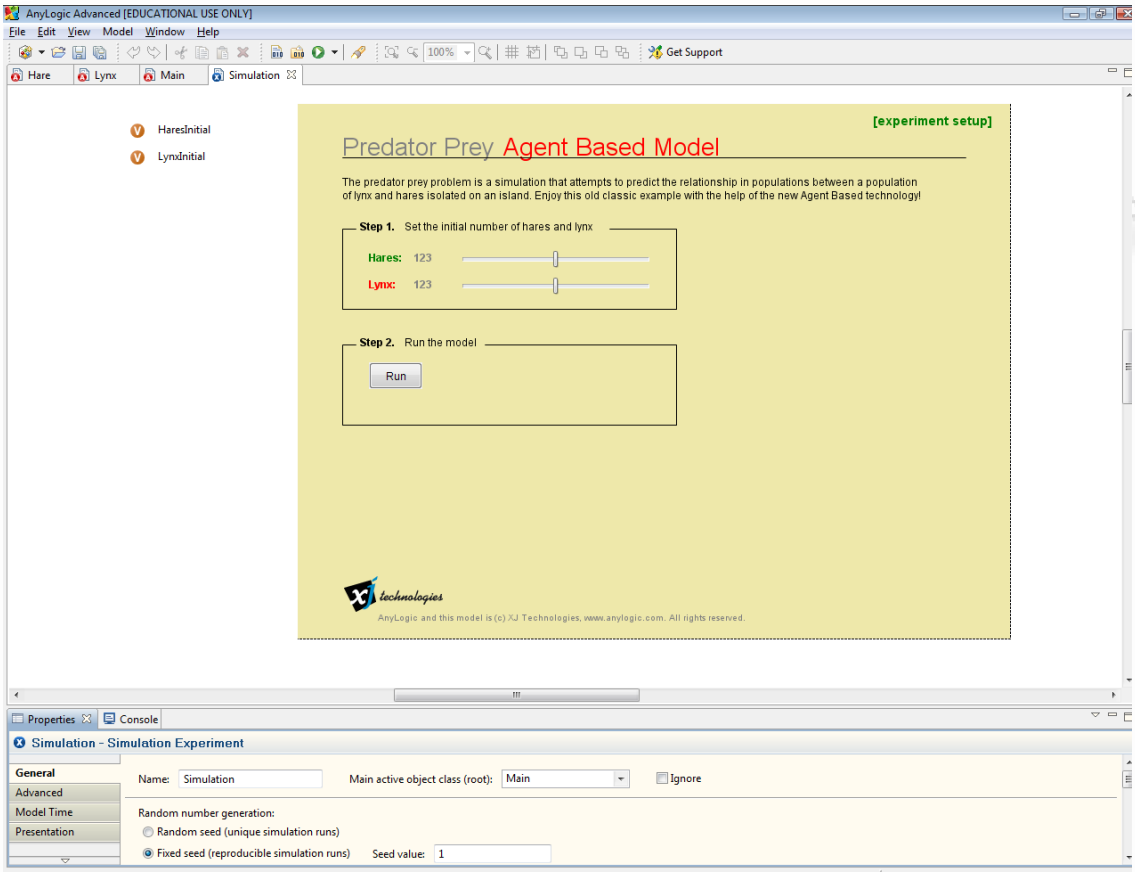
Εικόνα 6-28(δ): Γενικές ιδιότητες της ράβδου κύλισης slider3.

6.4 Η κλάση προσομοίωσης του έργου, Simulation:Main

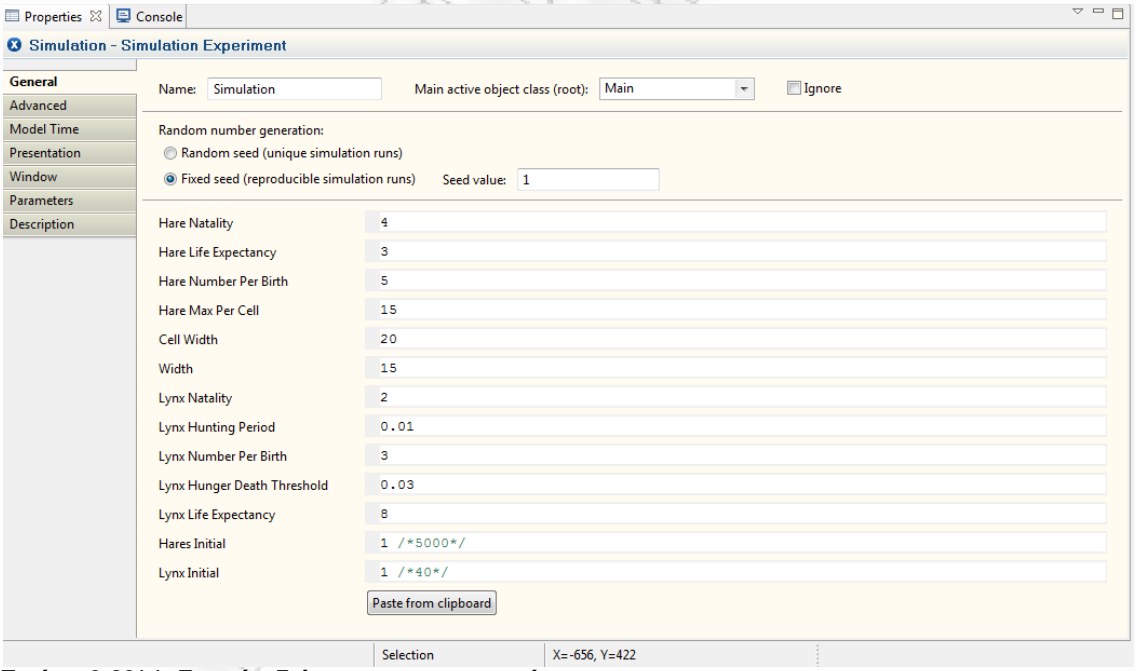


Η κλάση Simulation:Main για τη δημιουργία της προσομοίωσης του μοντέλου αποτελείται από τις δυο (2) απλές μεταβλητές HaresInitial και LynxInitial, τις δύο (2) ράβδους κύλισης, (slider και slider1), με τις οποίες επιλέγουμε τις αρχικές τιμές των θηραμάτων και των θηρευτών, αντίστοιχα και το κομβίο, (button), έναρξης της προσομοίωσης. Τα υπόλοιπα στοιχεία που συνθέτουν την κλάση είναι τα γραφικά στοιχεία παρουσίασης της κλάσης που εμφανίζονται στην εικόνα 6-29(α). Στην εικόνα 6-29(β) εμφανίζεται η ολοκληρωμένη μορφή της κλάσης όπως αυτή παρουσιάζεται στο συντάκτη γραφικών του Λογισμικού, ενώ στην εικόνα 6-29(γ) εμφανίζονται οι γενικές ιδιότητες της κλάσης της προσομοίωσης και στην εικόνα 6-29(δ) οι προχωρημένες ιδιότητες της. Οι εικόνες 6-30 έως και 6-32 παρουσιάζουν τις ιδιότητες των απλών μεταβλητών, των ράβδων κύλισης και του κομβίου αντίστοιχα ενώ στην εικόνα 6-33 φαίνονται μερικά στιγμιότυπα της προσομοίωσης.

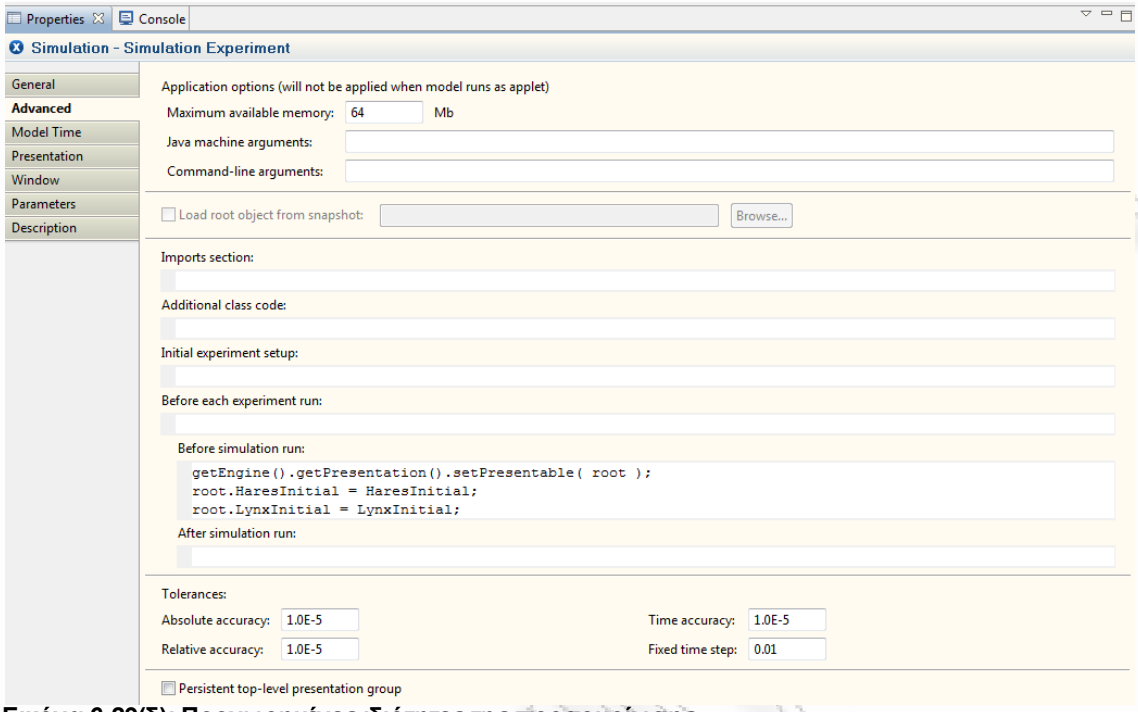
Εικόνα 6-29(α): Η ιεραρχική δομή της κλάσης της προσομοίωσης.



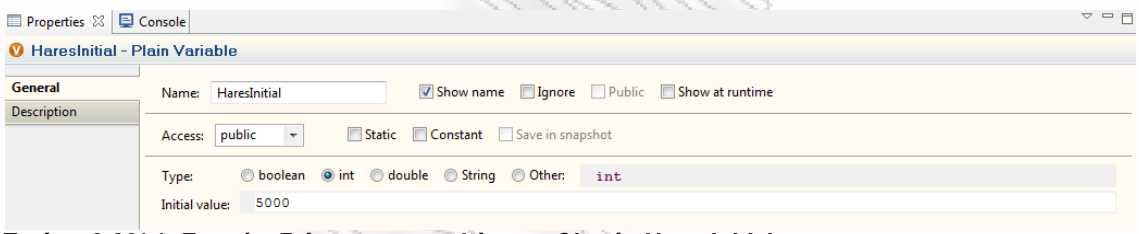
Εικόνα 6-29(β): Η κλάση γραφικών για τη δημιουργία του παράθυρου της προσομοίωσης.



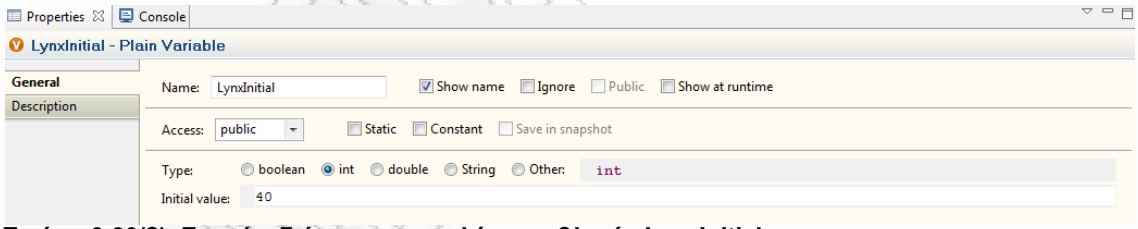
Εικόνα 6-29(γ): Γενικές ιδιότητες της προσομοίωσης.



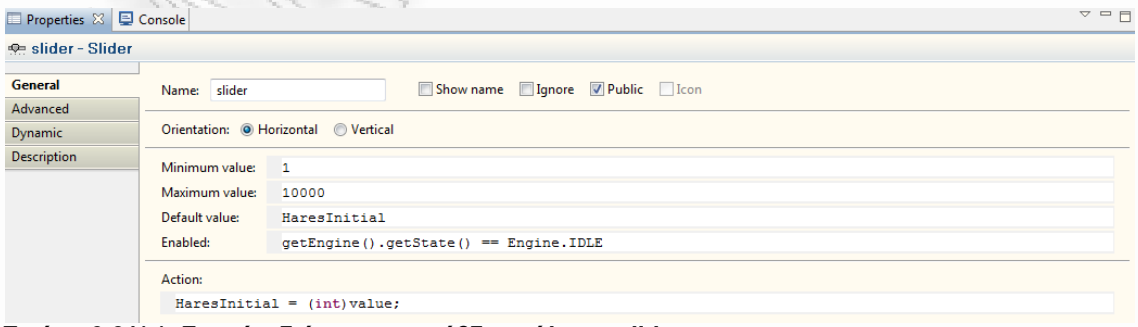
Εικόνα 6-29(δ): Προχωρημένες ιδιότητες της προσομοίωσης.



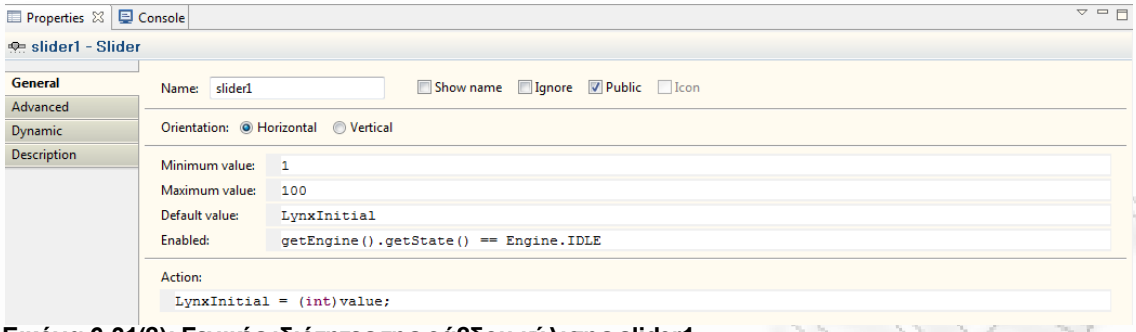
Εικόνα 6-30(α): Γενικές ιδιότητες της απλής μεταβλητής HaresInitial.



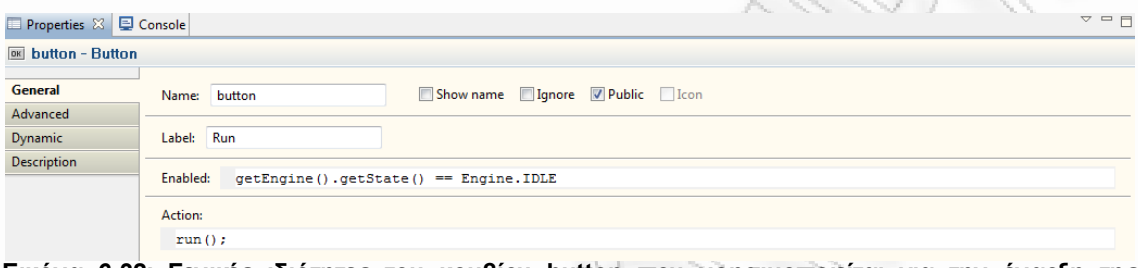
Εικόνα 6-30(β): Γενικές ιδιότητες της απλής μεταβλητής LynxInitial.



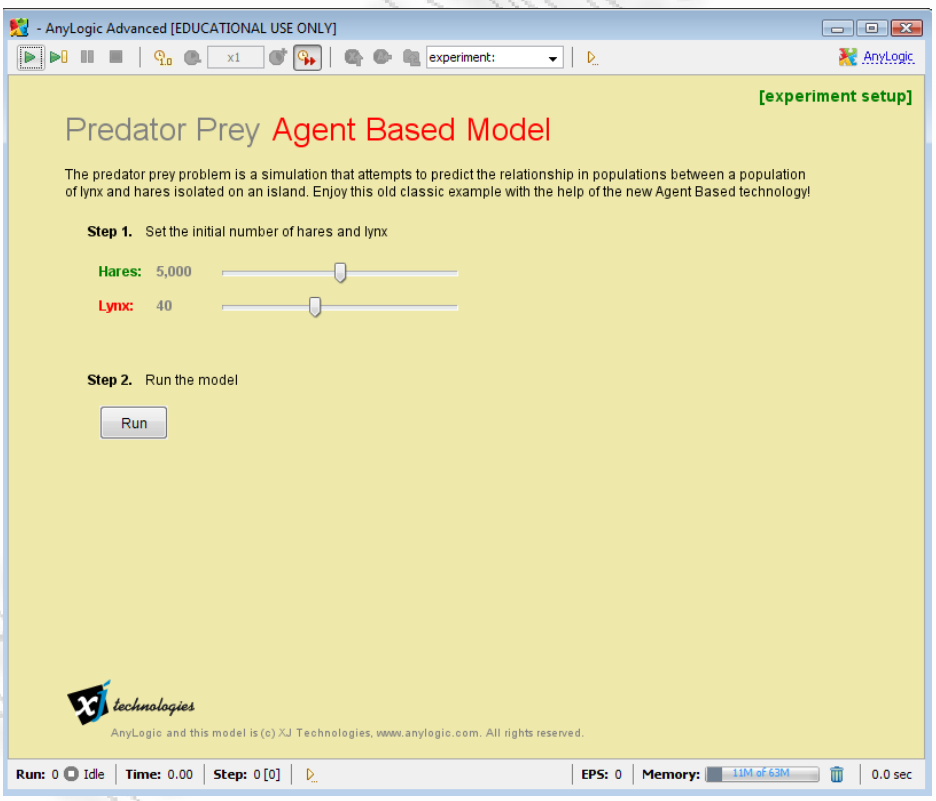
Εικόνα 6-31(α): Γενικές ιδιότητες της ράβδου κύλισης slider.



Εικόνα 6-31(β): Γενικές ιδιότητες της ράβδου κύλισης slider1.

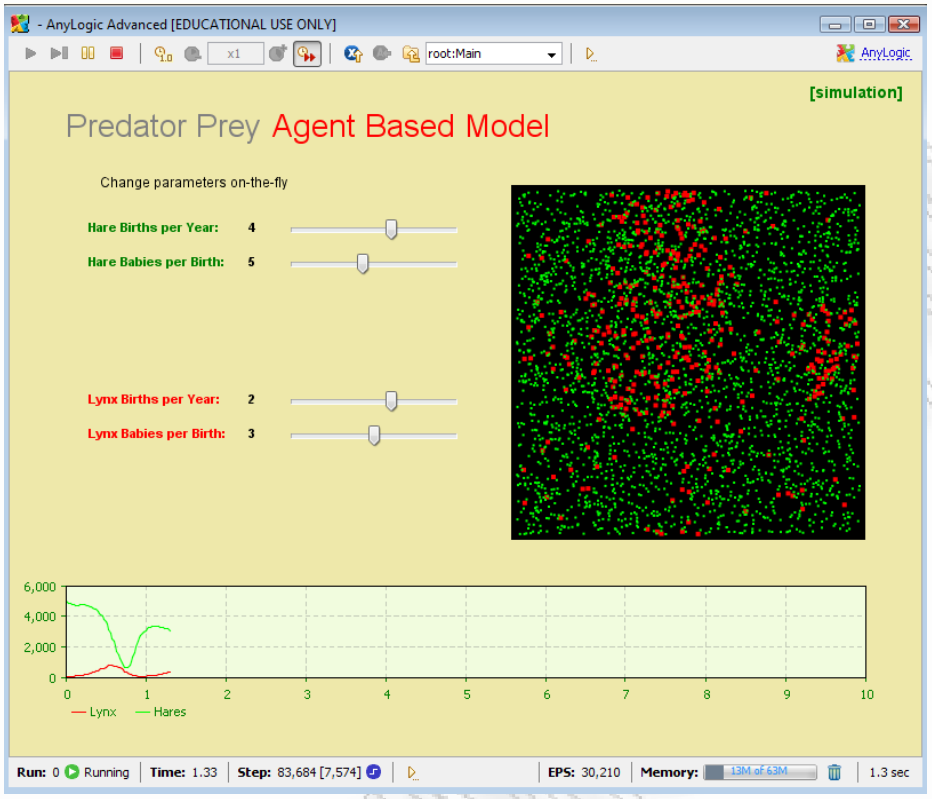


Εικόνα 6-32: Γενικές ιδιότητες του κομβίου button που χρησιμοποιείται για την έναρξη της προσομοίωσης.

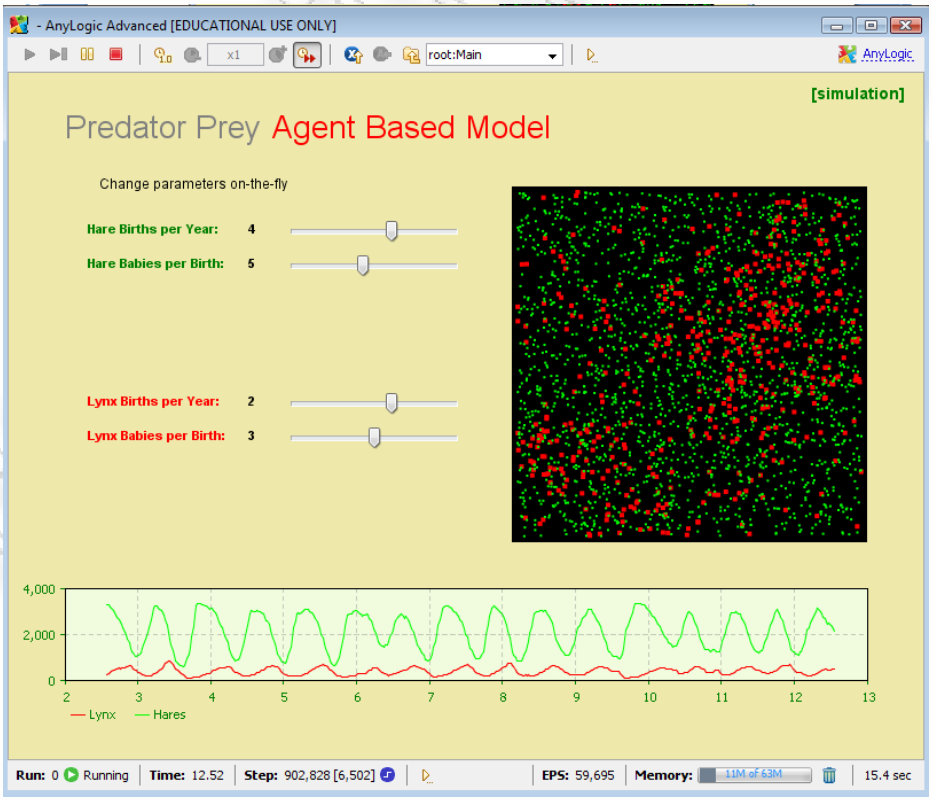


Εικόνα 6-33(α): Το παράθυρο της προσομοίωσης.

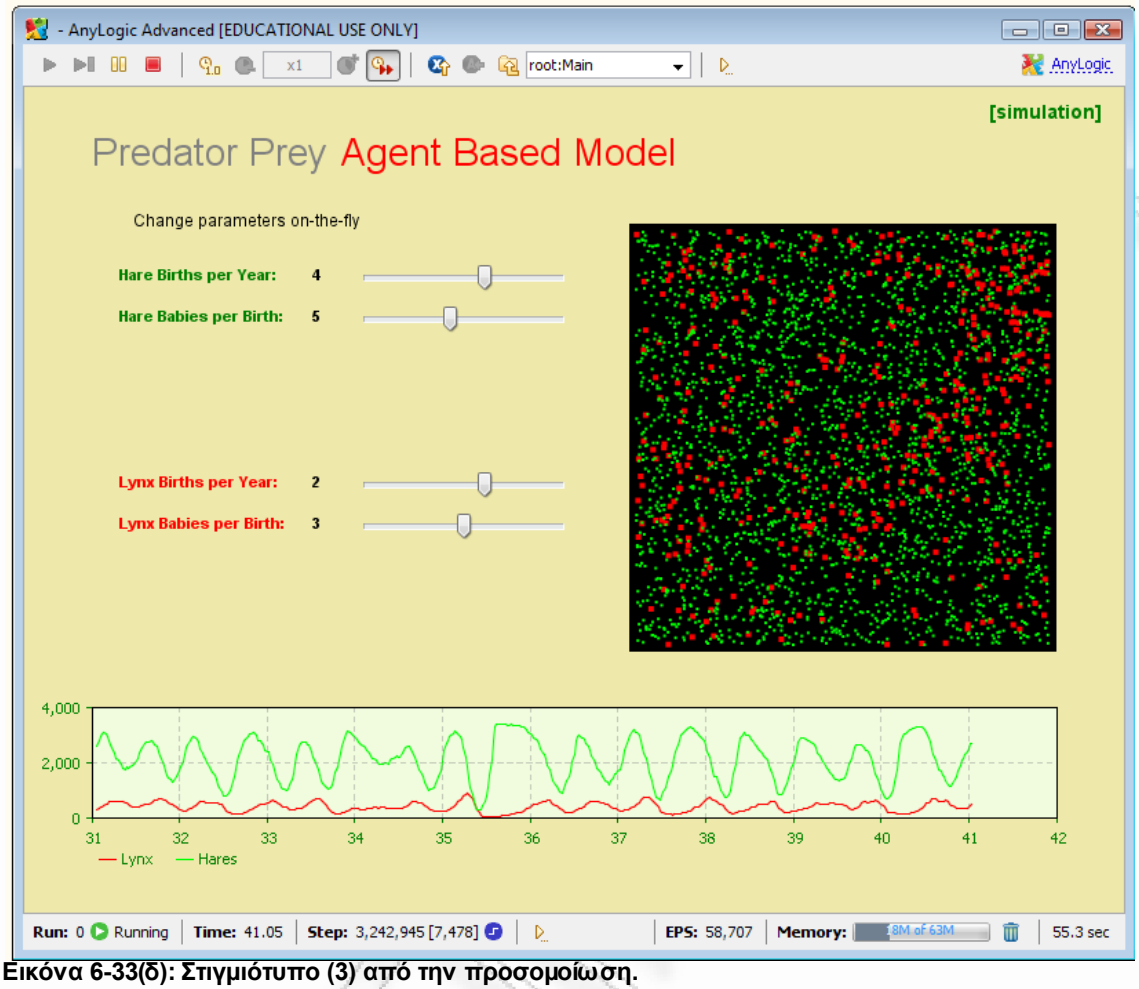
Μετά την επιλογή του κομβίου, έχουμε την έναρξη της προσομοίωσης και παρατηρείται ότι αλλάζει το παράθυρο της προσομοίωσης σε άλλο νέο. Το νέο γραφικό περιβάλλον έχει ήδη περιγραφεί προηγούμενα.



Εικόνα 6-33(β): Στιγμιότυπο (1) από την προσομοίωση.



Εικόνα 6-33(γ): Στιγμιότυπο (2) από την προσομοίωση.



Εικόνα 6-33(δ): Στιγμιότυπο (3) από την προσομοίωση.

Τα διαγράμματα κλάσεων και ο πλήρης κώδικας του παραδείγματος βρίσκονται στο Παράρτημα 4.

ΚΕΦΑΛΑΙΟ 7

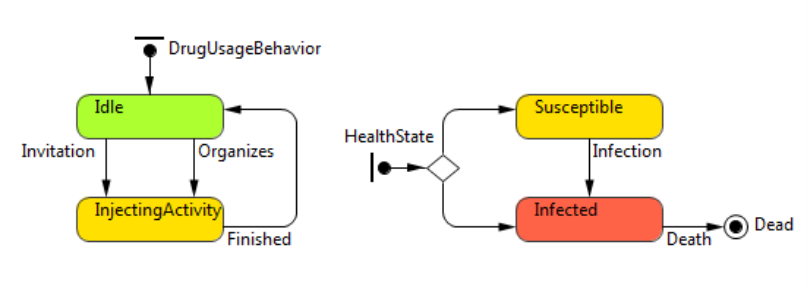
Μοντελοποίηση με τη μέθοδο των πρακτόρων λογισμικού και προσομοίωση της διασποράς του ιού HIV λόγω της χρήσης μολυσμένων συριγγών από χρήστες ναρκωτικών ουσιών.

Στο παράδειγμα αυτού του κεφαλαίου υλοποιείται ένα μοντέλο που διερευνά (α) την συσχέτιση μεταξύ της εμπειρίας στη χρήση ναρκωτικών και της πιθανότητας μόλυνσης από τον ιό HIV και (β) τη σχέση μεταξύ της χρήσης ασφαλών συριγγών και τη διάδοση του ιού HIV. Για την απάντηση στα παραπάνω, θα μοντελοποιηθεί ένα δίκτυο χρηστών ναρκωτικών ουσιών που εκτείνεται σε μια ορισμένη περιοχή.

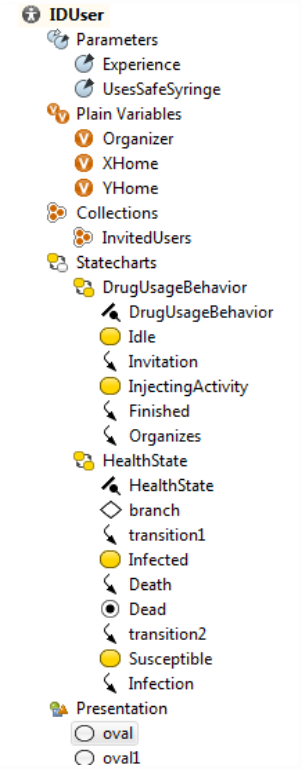
Όταν πρωτοεμφανίστηκε ο ιός HIV στις ΗΠΑ παρατηρήθηκε ότι οι χρήστες ηρωίνης που έκαναν ενέσιμη χρήση για μεγάλο χρονικό διάστημα είχαν μικρότερα ποσοστά μολύνσεων από τον ιό σε σχέση με τους χρήστες που έκαναν χρήση για μικρότερο χρονικό διάστημα. Το γεγονός αυτό, αν και φαίνεται περίεργο, εξηγείται πολύ εύκολα παρατηρώντας την ιεραρχική δομή των κοινοτήτων των χρηστών ναρκωτικών. Οι χρήστες εκείνη την εποχή (α) σε τακτά χρονικά διαστήματα συναντούσαν σε διαφορετικά μέρη κάθε φορά τους φίλους τους που ήταν και αυτοί χρήστες και έκαναν ενέσιμη ηρωίνη χρησιμοποιώντας τις ίδιες σύριγγες και (β) κατά την αγορά από τον έμπορο, τον προμηθευτή ναρκωτικών γίνονταν η χρήση με σύριγγες του εμπόρου. Ο χρόνιος χρήστης βρισκόταν υψηλά στην πυραμίδα των χρηστών όπου η ιεραρχία του έδινε το «δικαίωμα» της ευκολότερης και γρηγορότερης εύρεσης ηρωίνης. Αποτέλεσμα αυτού ήταν ότι ο χρόνιος χρήστης χρησιμοποιούσε πρώτος της σύριγγες και η πιθανότητα οι σύριγγες να ήταν μολυσμένες από τον ιό ήταν πολύ μικρή. Αντίθετα, όσο νεότερος ήταν ο χρήστης τόσο αυξανόταν η πιθανότητα η σύριγγες που χρησιμοποιούσε να ήταν μολυσμένες με αποτέλεσμα την μόλυνση του από τον ιό. Όσο αυξάνονταν οι φορείς του ιού, τόσο αυξάνονταν οι πιθανότητες οι σύριγγες να είχαν μολυνθεί και άρα ο κύκλος της μόλυνσης των χρηστών άνοιγε περισσότερο.

7.1 Η κλάση IDUser

Η ιεραρχική δομή της κλάσης IDUser παρουσιάζεται στην εικόνα 7-2. Το χαρακτηριστικό αυτής της κλάσης είναι ότι ο πράκτορας λογισμικού IDUser υλοποιείται συνδυάζοντας δύο (2) διαγράμματα καταστάσεων, του DrugUsageBehavior και του HealthState όπως φαίνεται στην εικόνα 7-1.



Εικόνα 7-1: Τα διαγράμματα καταστάσεων της κλασης IDUser.



Εικόνα 7-2: Η ιεραρχική δομή της κλασης IDUser.

Ο πράκτορας λογισμικού IDUser, (ο χρήστης των ενέσιμων ναρκωτικών που ορίζει την κλάση ενεργού αντικειμένου), ελέγχεται ως προς την κατάσταση της υγείας του, HealthState και ως προς τις συνήθειες του κατά τη χρήση ναρκωτικών, DrugUsageBehavior.

Στο διάγραμμα καταστάσεων DrugUsageBehavior, ο χρήστης ναρκωτικών ξεκινά από την κατάσταση αδράνειας Idle και είτε σε ορισμένη στιγμή σύμφωνα με τη μετάβαση Invitation δέχεται πρόσκληση από τους φίλους του να συμμετάσχει σε κύκλο χρήσης ενέσιμων ναρκωτικών είτε οργανώνει ο ίδιος τη συνάντηση για τη χρήση ναρκωτικών, μετάβαση Organizes. Η κατάσταση Injecting Activity προσομοιώνει τη χρήση η οποία κάποια στιγμή ολοκληρώνεται και ο χρήστης επανέρχεται στην κατάσταση αδράνειας μέχρι την επόμενη χρήση.

Ταυτόχρονα, ο πράκτορας λογισμικού από πλευράς κατάστασης της υγείας του, HealthState, μπορεί να βρίσκεται σε μια από τις δύο καταστάσεις, την «Ευπαθής» (Susceptible) ή την «Μολυσμένος» (Infected). Η μετάβαση από τη μια κατάσταση στην άλλη, συμβαίνει μέσω της μετάβασης Infection. Η προσομοίωση ξεκινά θεωρώντας ένα ποσοστό χρηστών ως μολυσμένους από τον ιό. Οι χρήστες αλλάζουν κατάσταση από «Ευπαθής» σε «Μολυσμένος» όταν συμβεί το γεγονός της χρήσης μιας μολυσμένης σύριγγας. Οι μολυσμένοι χρήστες συνεχίζουν να λαμβάνουν μέρος στην προσομοίωση μέχρι τη στιγμή που πεθαίνουν (Dead) οπότε και διαγράφονται από την προσομοίωση. Ο χρόνος ζωής των μολυσμένων από τον ιό χρηστών μέχρι τον θάνατο (Death) τους εξαρτάται από τη μαθηματική κατανομή που χρησιμοποιείται.

Η κύρια παραδοχή του μοντέλου είναι ότι ο χρήστης κάνει χρήση ναρκωτικών (InjectingActivity) πάντα με παρέα, (είτε καλείται από, είτε καλεί φίλους) σε ορισμένο χώρο. Η σύριγγα θεωρείται ότι αρχικά δεν είναι μολυσμένη από τον ιό HIV. Στην πορεία της χρήσης όμως αυτή η κατάσταση μπορεί να αλλάζει. Στις εικόνες 7-3(α) έως και 7-3(δ) παρουσιάζονται οι ιδιότητες της κλάσης.

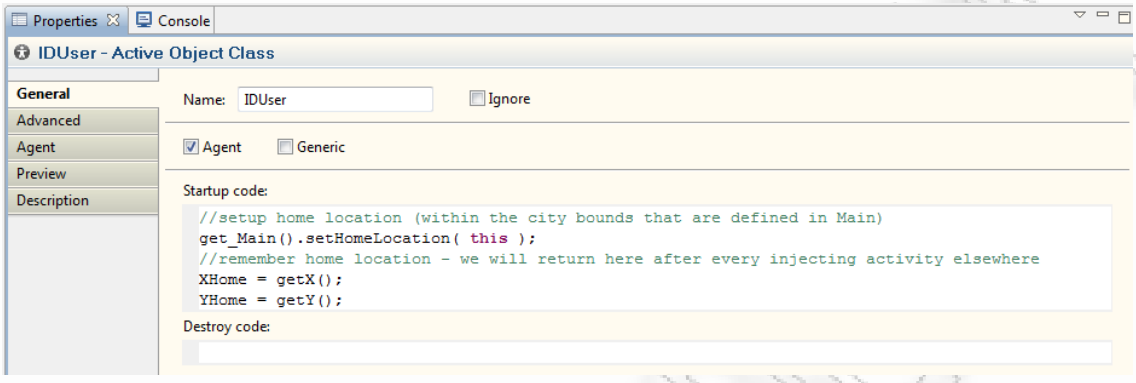
Η κλάση IDUser αποτελείται από τις παραμέτρους, πίνακας 7-1, Experience και UsesSafeSyringe. Η παράμετρος Experience ορίζει την εμπειρία, η οποία μετράται σε χρόνια χρήσης ναρκωτικών, των χρηστών και είναι το μέτρο σύγκρισης του σεβασμού και της θέσης στην ιεραρχία των χρηστών. Η παράμετρος UsesSafeSyringe είναι μια λογική μεταβλητή, μπορεί να πάρει τις τιμές TRUE ή FALSE, που υποδηλώνει την μόλυνση ή όχι της σύριγγας από τον ιό HIV. Στην εικόνα 7-4 παρουσιάζεται ολοκληρωμένα η γραφική αναπαράσταση της κλάσης και στις εικόνες 7-5(α) έως και 7-5(γ) δίνονται οι ιδιότητες των παραμέτρων.

Παράμετροι της κλάσης IDUser		
Όνομα παραμέτρου	Τύπος παραμέτρου	Αρχική τιμή παραμέτρου
Experience	Double	0.0
UsesSafeSyringe	Boolean	FALSE

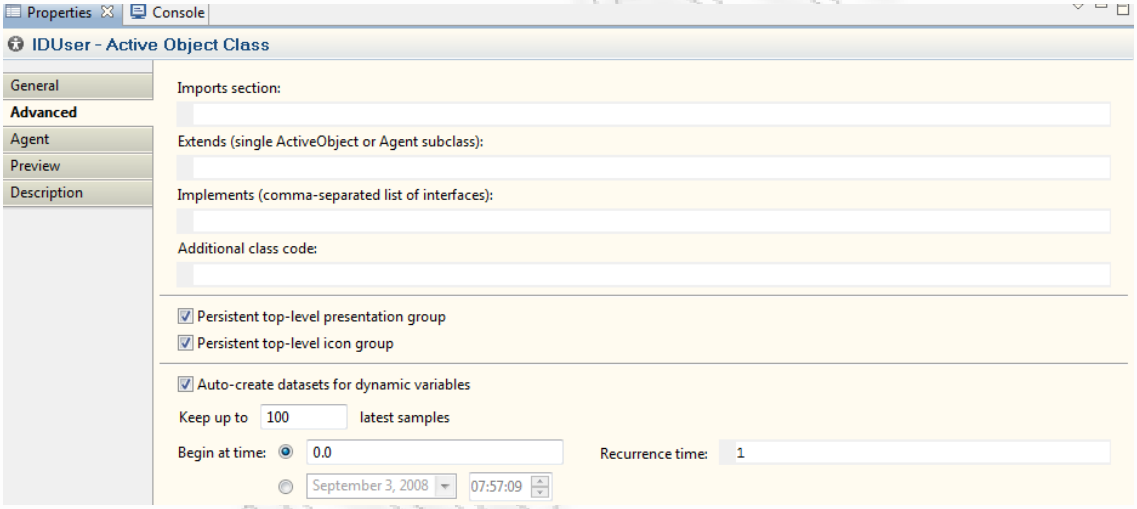
Πίνακας 7-1: Παράμετροι της κλάσης IDUser

Στον πίνακα 7-2 βρίσκονται συγκεντρωμένες οι απλές μεταβλητές. Η κλάση αποτελείται από τις απλές μεταβλητές XHome και YHome οι οποίες περιγράφουν τις καρτεσιανές συντεταγμένες που ορίζουν τη θέση του χρήστη στο γεωγραφικό χώρο που εξετάζουμε κατά την προσομοίωση. Οι συντεταγμένες δηλώνουν ότι ο χρήστης μπορεί να βρίσκεται είτε στον προσωπικό του χώρο είτε σε κάποιο άλλο μέρος στο οποίο έχει προσκληθεί για να προχωρήσει στη χρήση ναρκωτικών. Η απλή μεταβλητή Organizer καθορίζει τον οργανωτή, τον χρήστη που

αναλαμβάνει να οργανώσει τη σύναξη των ατόμων που θα κάνουν χρήση ναρκωτικών στον προσωπικό του χώρο. Είναι τύπου IDUser και ανήκει στη γενική κλάση Agent. Στην εικόνα 7-6 παρουσιάζονται οι ιδιότητες των απλών μεταβλητών.

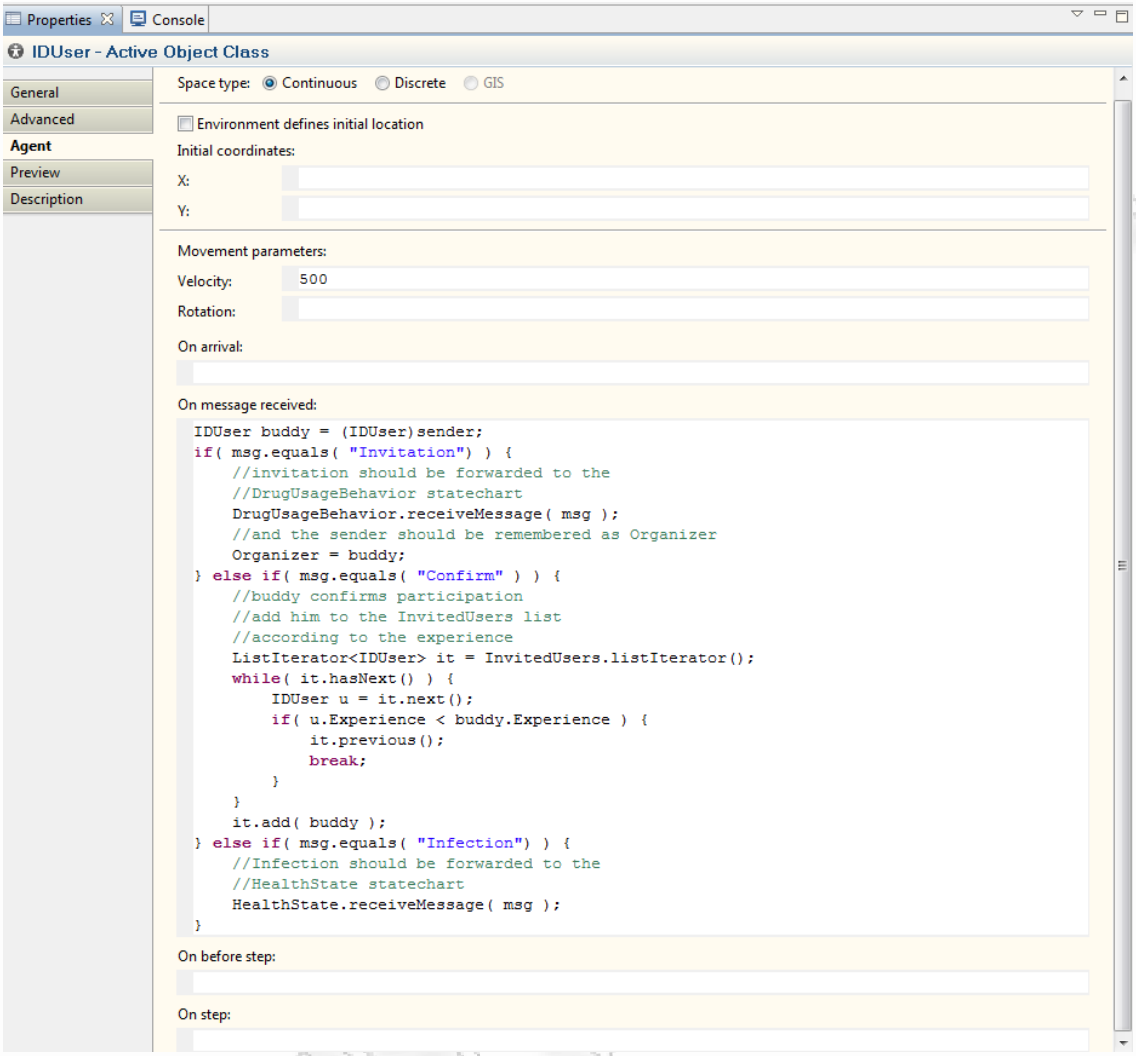


Εικόνα 7-3(α): Οι γενικές ιδιότητες της κλάσης IDUser.

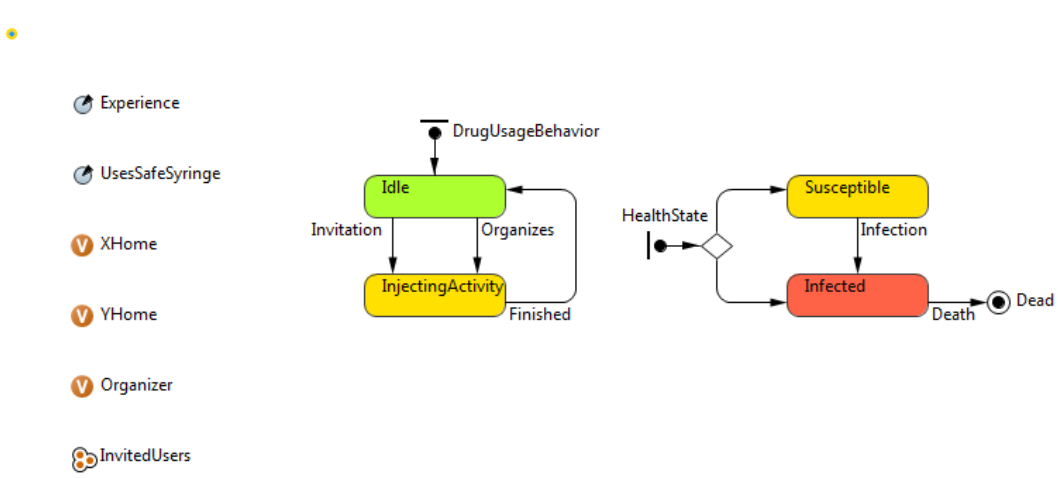


Εικόνα 7-3(β): Οι προχωρημένες ιδιότητες της κλάσης IDUser.

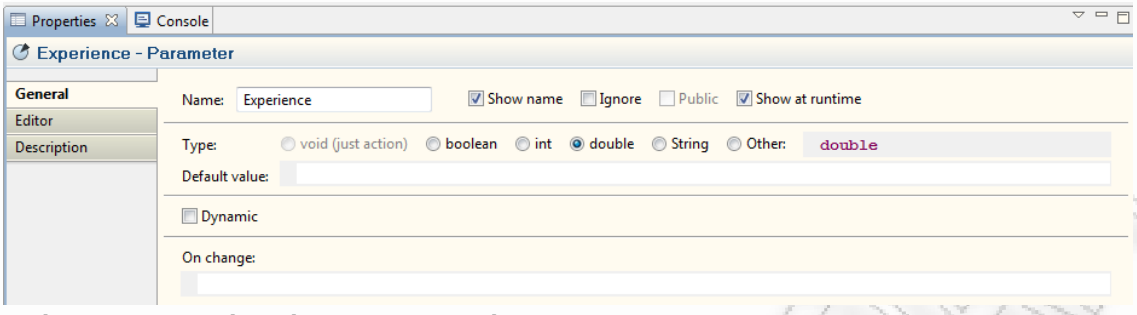
Εικόνα 7-3(γ): Προεπισκόπιση των παραμέτρων της κλάσης IDUser.



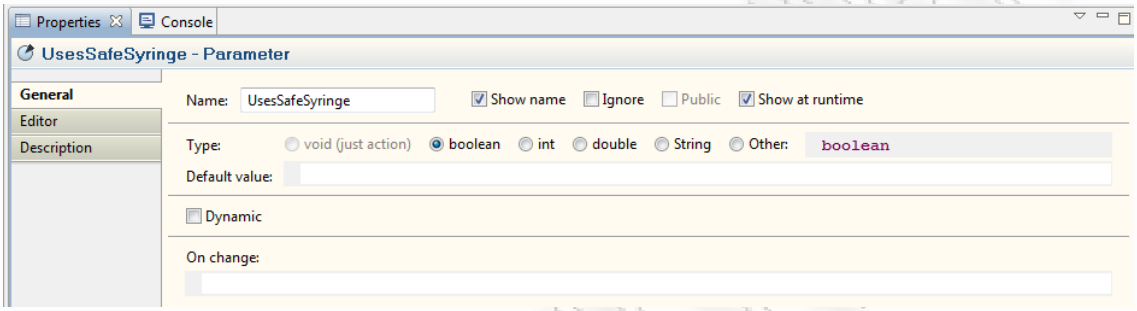
Εικόνα 7-3(δ): Ο κώδικας που συνοδεύει τον πράκτορα λογισμικού της κλάσης IDUser.



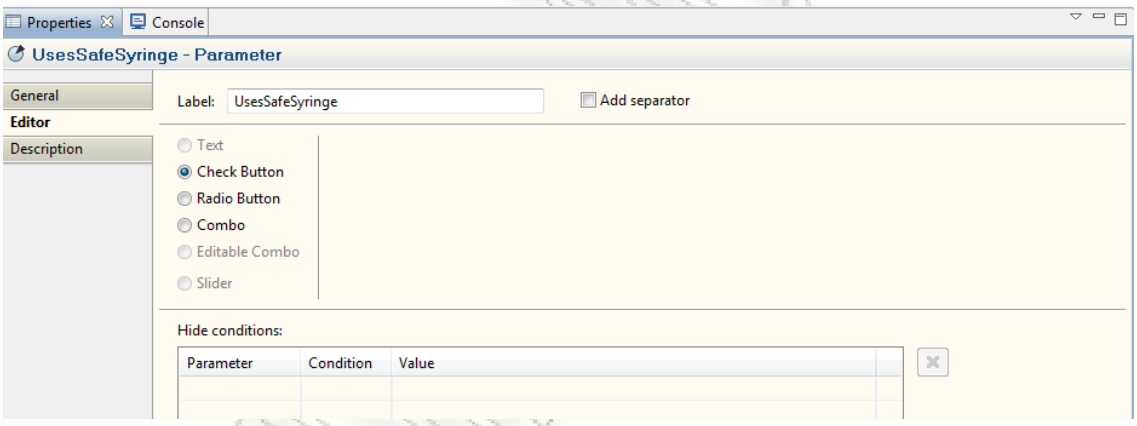
Εικόνα 7-4: Η ολοκληρωμένη γραφική σύνθεση της κλάσης IDUser.



Εικόνα 7-5(α): Γενικές ιδιότητες της παραμέτρου Experience.



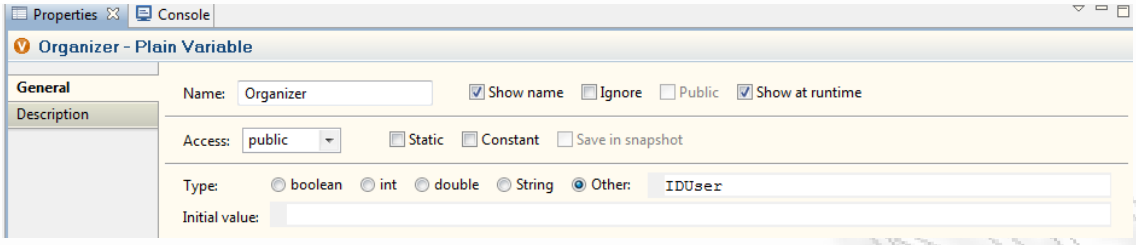
Εικόνα 7-5(β): Γενικές ιδιότητες της παραμέτρου UsesSafeSyringe.



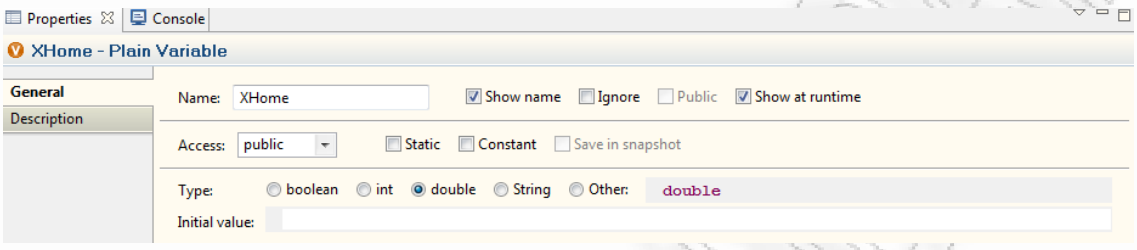
Εικόνα 7-5(γ): Μορφοποίηση του τρόπου επιλογής της παραμέτρου UsesSafeSyringe.

Απλές μεταβλητές της κλάσης IDUser		
Όνομα μεταβλητής	Τύπος Μεταβλητής	Αρχική τιμή Μεταβλητής
XHome	Double	Δεν είναι γνωστή. Εξαρτάται από την πορεία της προσομοίωσης.
YHome	Double	
Organizer	IDUser	

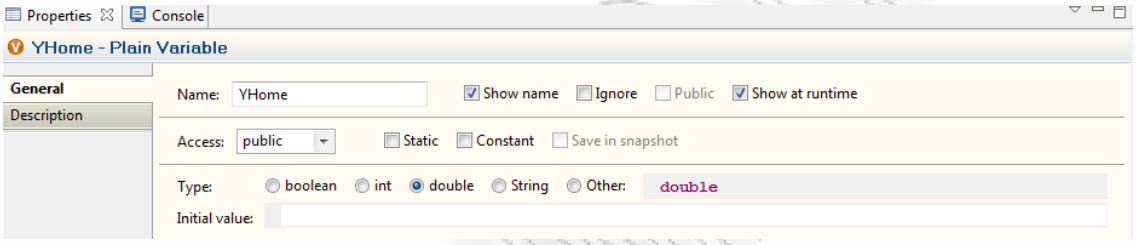
Πίνακας 7-2: Οι απλές μεταβλητές της κλάσης IDUser



Εικόνα 7-6(α): Γενικές ιδιότητες της απλής μεταβλητής Organizer

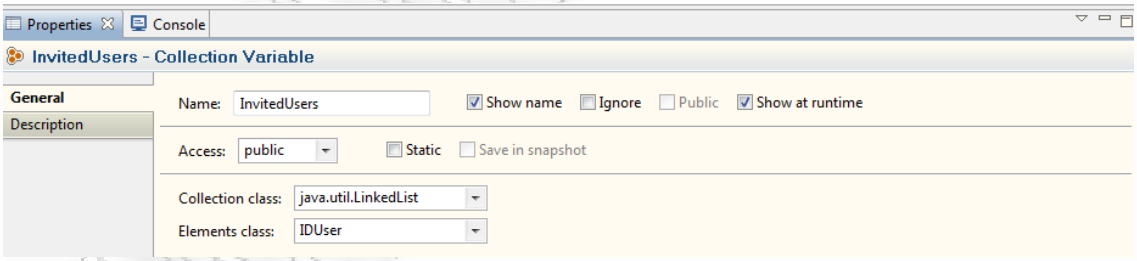


Εικόνα 7-6(β): Γενικές ιδιότητες της απλής μεταβλητής XHome



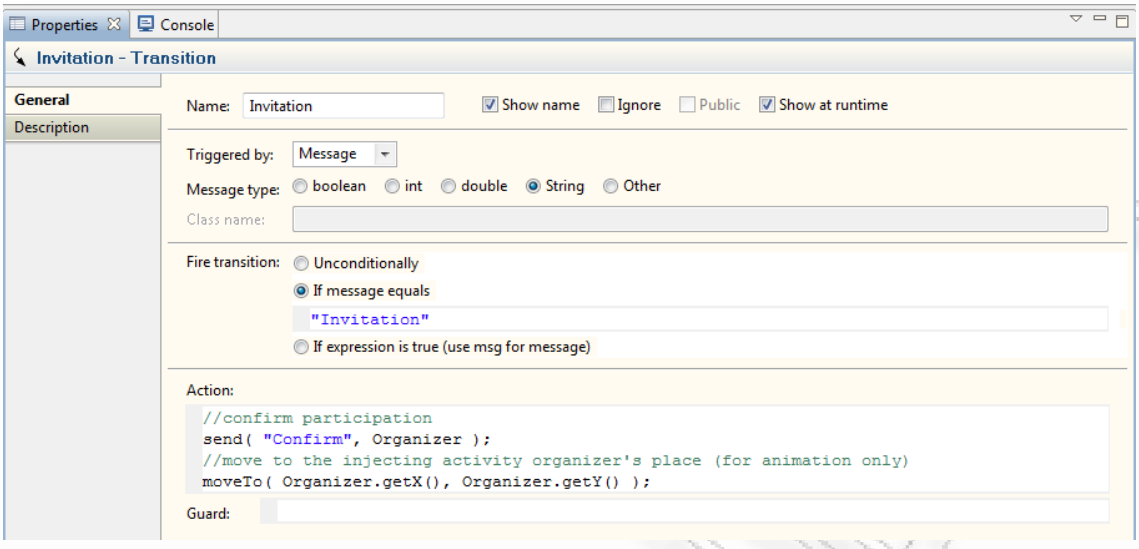
Εικόνα 7-6(γ): Γενικές ιδιότητες της απλής μεταβλητής YHome

Η μεταβλητή InvitedUsers, εικόνα 7-4 και εικόνα 7-7, είναι μια μεταβλητή ειδικού τύπου διασυνδεδεμένη λίστα που αποτελείται από στοιχεία IDUser. Ορίζεται στις συλλογές, (collections), της Java και επεκτείνει την κλάση java.util.LinkedList. Χρησιμοποιείται για την αποθήκευση σε λίστα των ατόμων, των πρακτόρων λογισμικού, που αποδέχονται την πρόσκληση του οργανωτή Organizer και λαμβάνουν μέρος στις συγκεντρώσεις χρήσης ναρκωτικών.

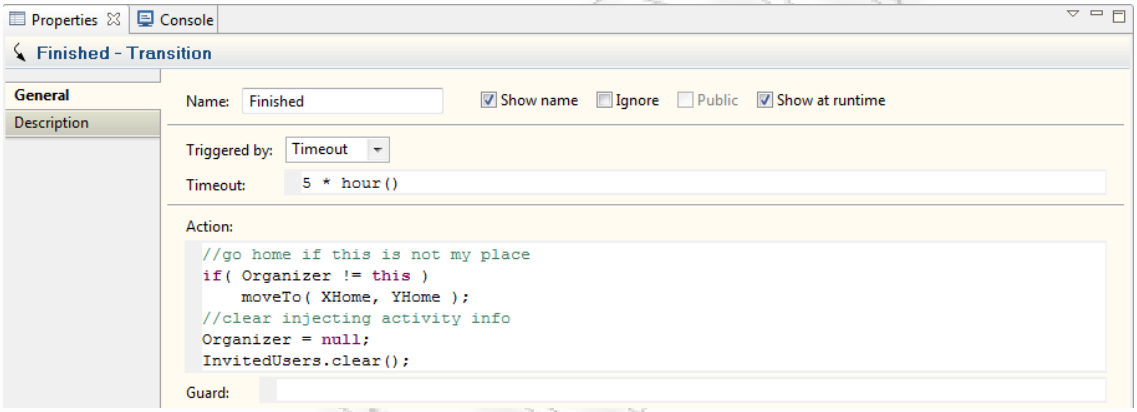


Εικόνα 7-7: Γενικές ιδιότητες της μεταβλητής InvitedUsers.

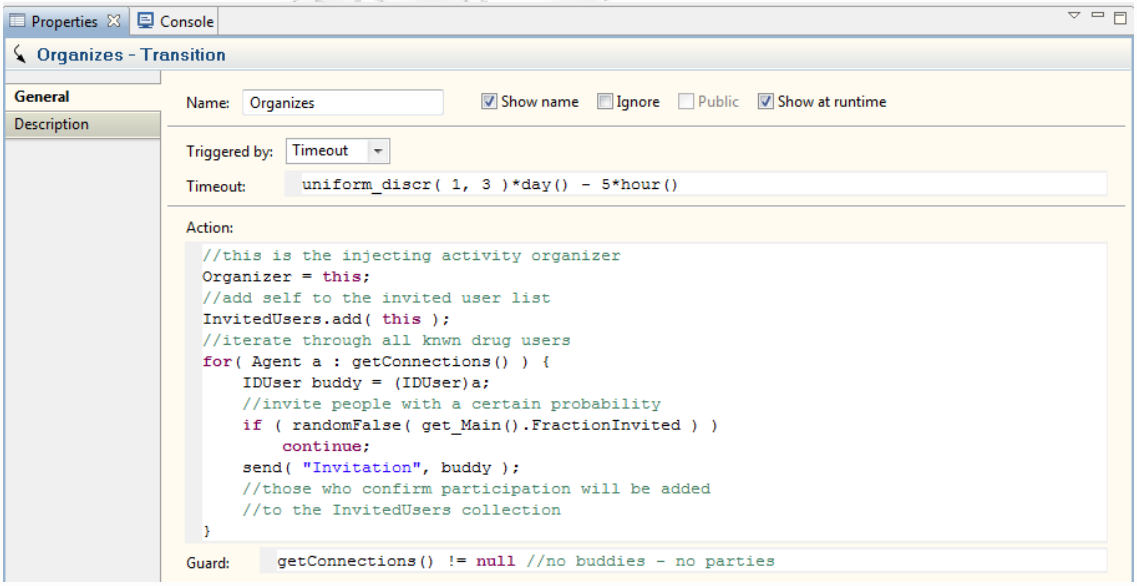
Στη συνέχεια, εικόνες 7-8(α) – 7-8(θ), παρουσιάζονται οι ιδιότητες και ο κώδικας προγραμματισμού σε Java μερικών από τα στοιχεία που συνθέτουν τα διαγράμματα καταστάσεων της κλάσης IDUser. Επίσης, στις εικόνες 7-9(α) και 7-9(β), δίνονται τα στοιχεία που αποτελούν τη γραφική παρουσίαση της κλάσης κατά την προσομοίωση.



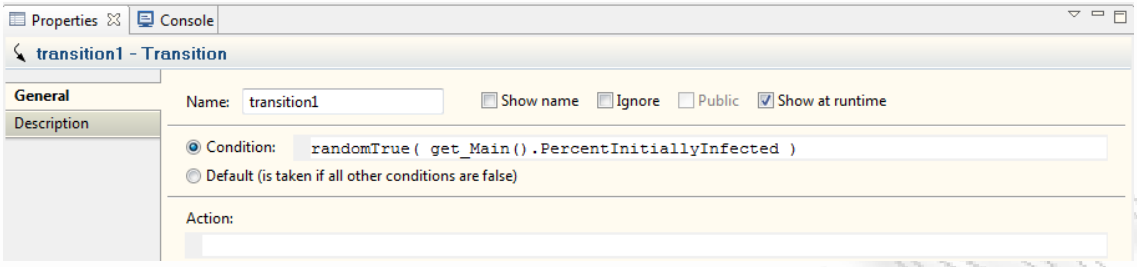
Εικόνα 7-8(α): Γενικές ιδιότητες της μετάβασης Invitation.



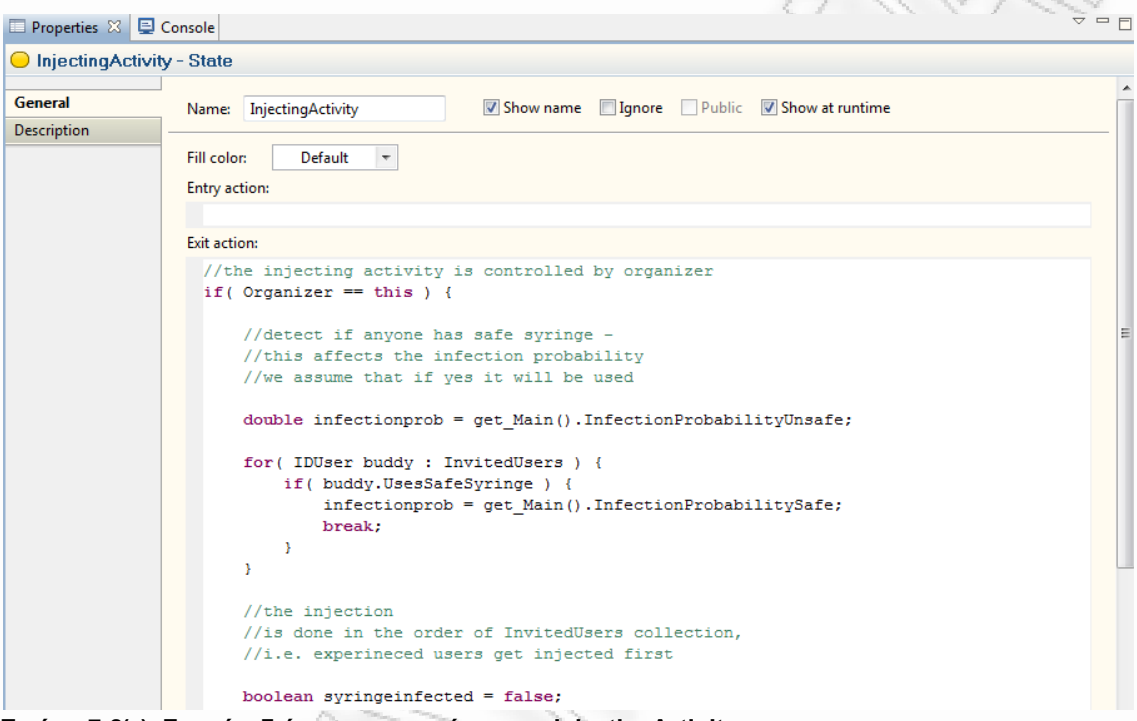
Εικόνα 7-8(β): Γενικές ιδιότητες της μετάβασης Finished.



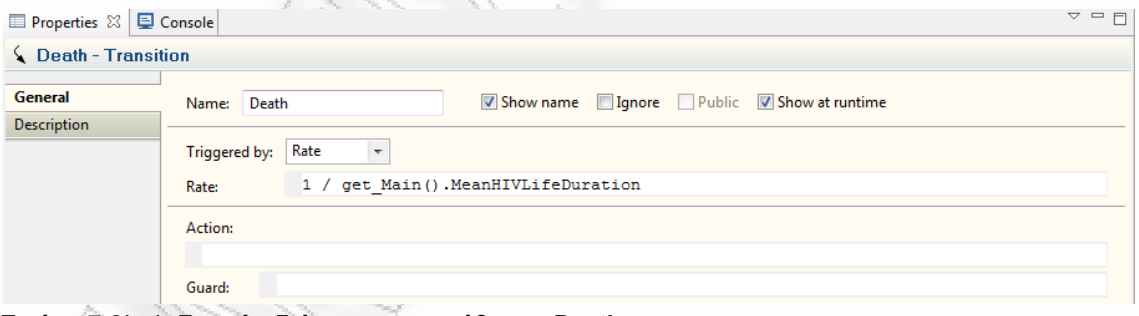
Εικόνα 7-8(γ): Γενικές ιδιότητες της μετάβασης Organizes.



Εικόνα 7-8(δ): Γενικές ιδιότητες της μετάβασης transition1.

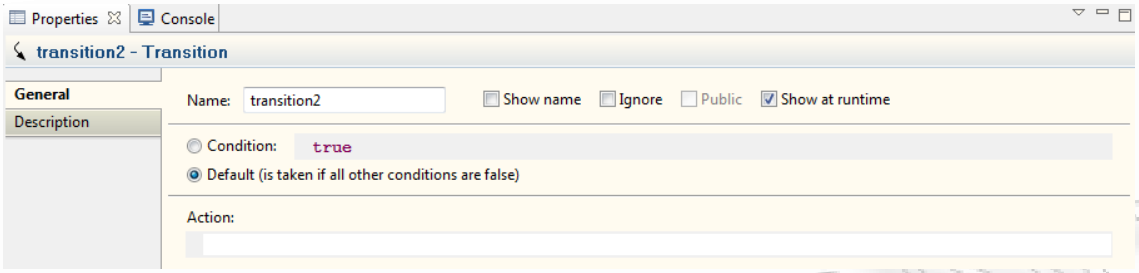


Εικόνα 7-8(ε): Γενικές ιδιότητες της κατάστασης InjectingActivity.

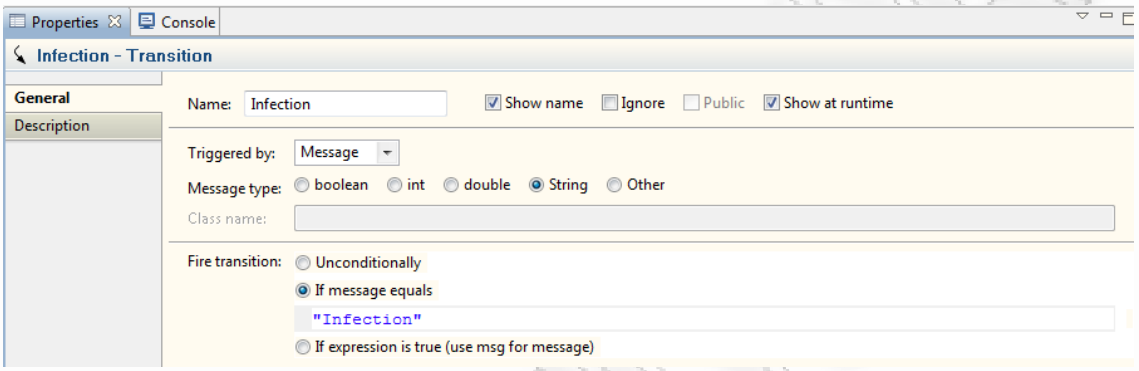


Εικόνα 7-8(στ): Γενικές ιδιότητες της μετάβασης Death.

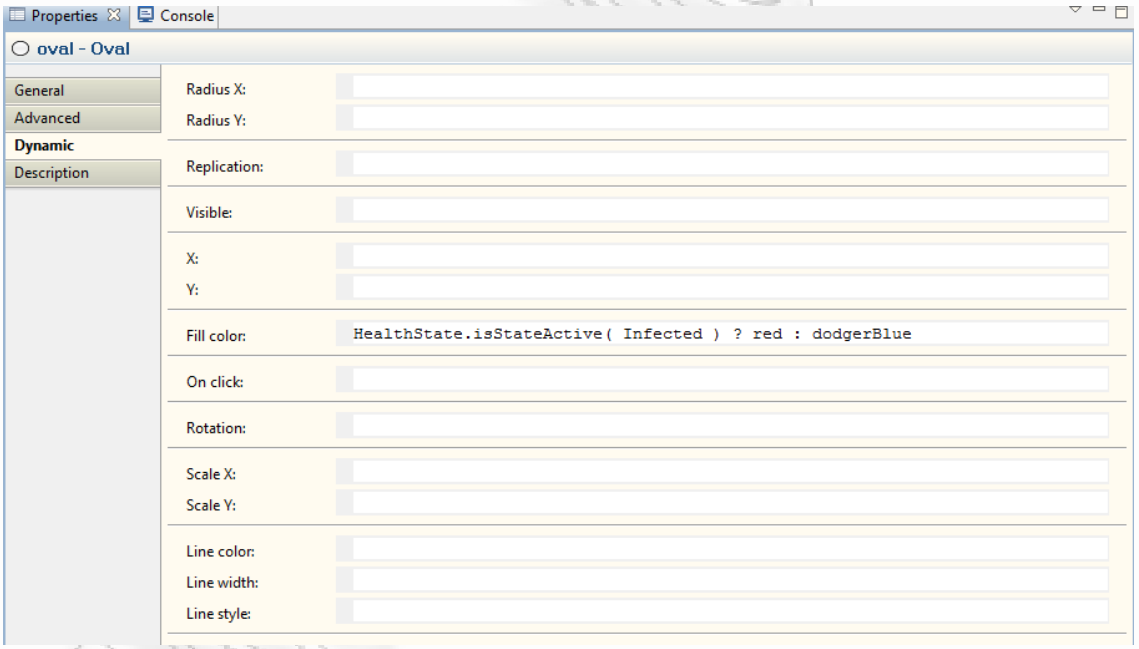
Εικόνα 7-8(ζ): Γενικές ιδιότητες της τελικής κατάστασης Dead.



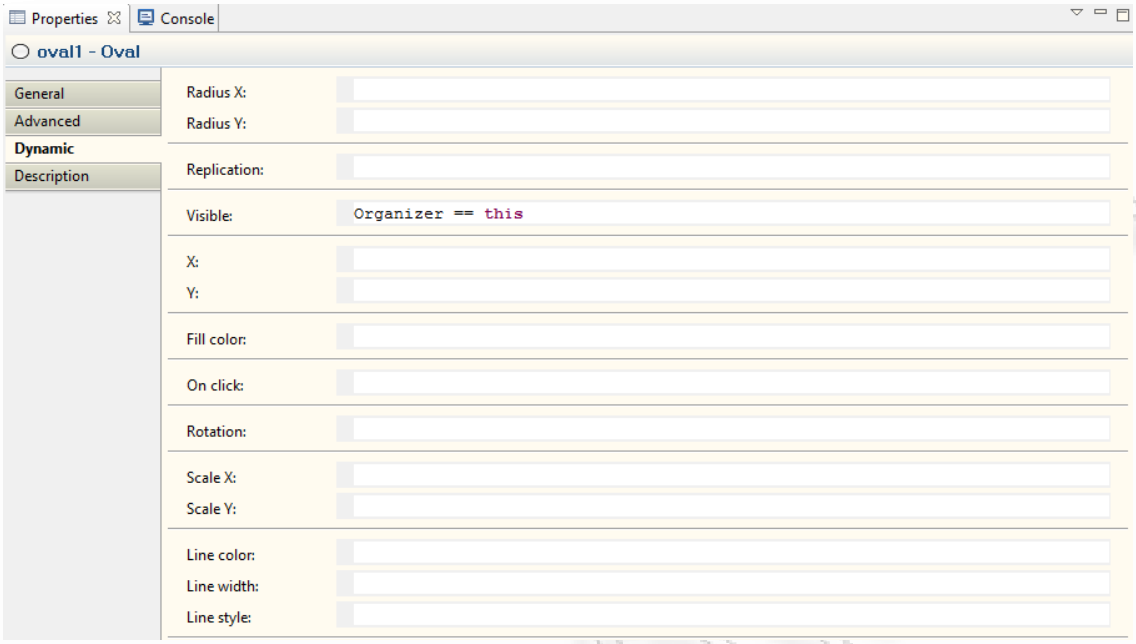
Εικόνα 7-8(η): Γενικές ιδιότητες της μετάβασης transition2.



Εικόνα 7-8(θ): Γενικές ιδιότητες της μετάβασης Infection.



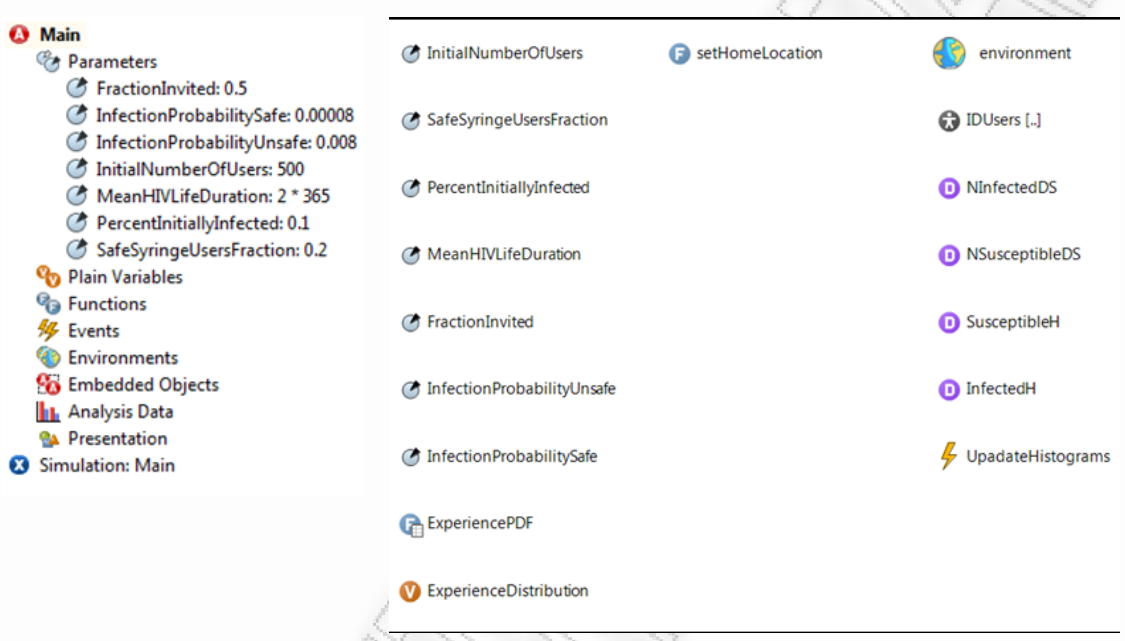
Εικόνα 7-9(α): Δυναμικές ιδιότητες της γραφικής αναπαράστασης του πράκτορα λογισμικού.



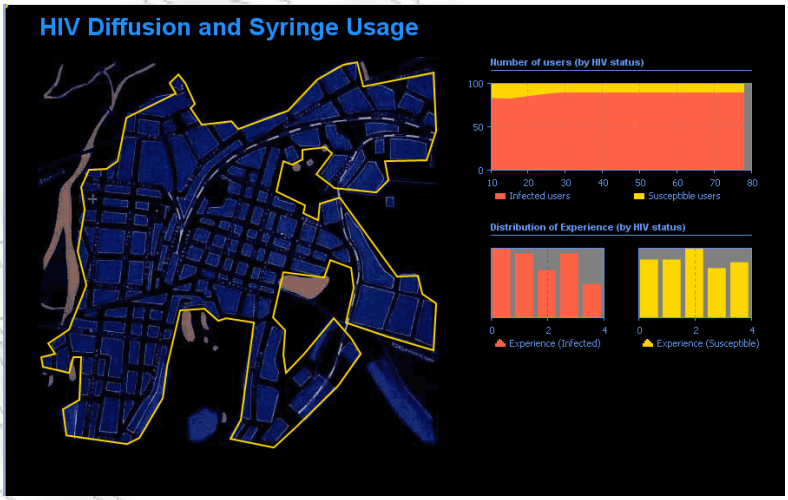
Εικόνα 7-9(β): Δυναμικές ιδιότητες της γραφικής αναπαράστασης του πράκτορα λογισμικού.

7.2 Η κλάση Main

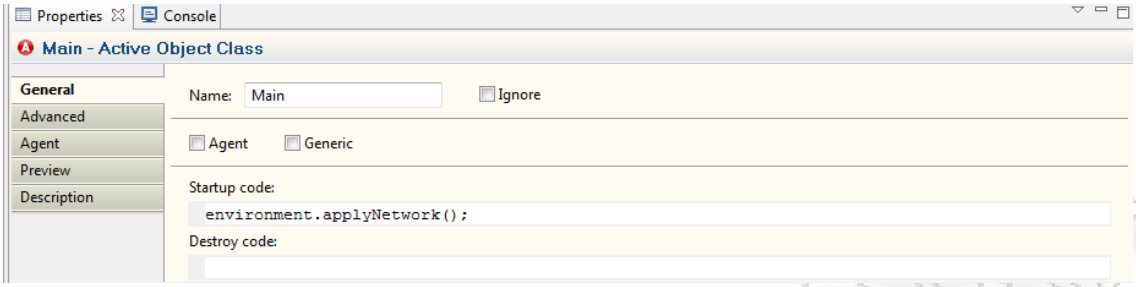
Η κλάση ενεργού αντικειμένου Main, εικόνες 7-10(α) έως και 7-10(γ), καθορίζει την κίνηση και τις ενέργειες των πρακτόρων λογισμικού, των χρηστών ναρκωτικών μέσα σε μια ορισμένη γεωγραφική περιοχή. Κάθε χρήστης εμφανίζεται ως ένα μπλέ στίγμα μέσα στη γεωγραφική περιοχή και στην περίπτωση που έχει μολυνθεί από τον ιό τότε το χρώμα αλλάζει σε κόκκινο.



Εικόνα 7-10(α): Η κλαση Main όπως εμφανίζεται στον συντάκτη του λογισμικού.



Εικόνα 7-10(β): Η κλάση Main όπως εμφανίζεται στον συντάκτη του λογισμικού.

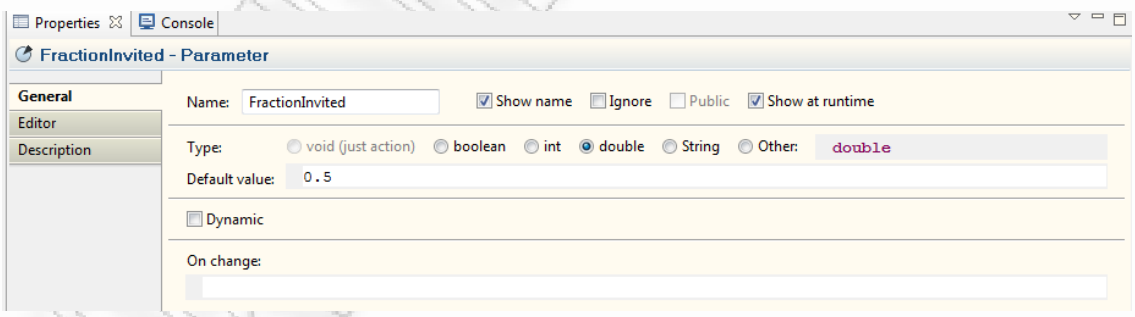


Εικόνα 7-10(γ): Γενικές ιδιότητες της κλάσης Main.

Η κλάση αποτελείται από επτά (7) παραμέτρους, πίνακας 7-3, τις InitialNumberofUsers (αρχικός αριθμός χρηστών), PercentInitiallyInfected (ο αρχικός αριθμός των μολυσμένων χρηστών, εκφρασμένος σε ποσοστό επί τοις εκατό), MeanHIVLifeDuration (ο μέσος χρόνος ζωής μετά την μόλυνση από τον ιό), InfectionProbabilityUnsafe (η πιθανότητα όλοι οι χρήστες της ομάδας να έχουν μολυσμένη σύριγγα), InfectionProbabilitySafe (η πιθανότητα όλοι οι χρήστες της ομάδας να έχουν ασφαλή σύριγγα), SafeSyringeUsersFraction (το κλάσμα των χρηστών που έχουν ασφαλή σύριγγα) και FractionInvited (το κλάσμα των χρηστών που καλούνται να λάβουν μέρος σε ομαδικές χρήσεις).

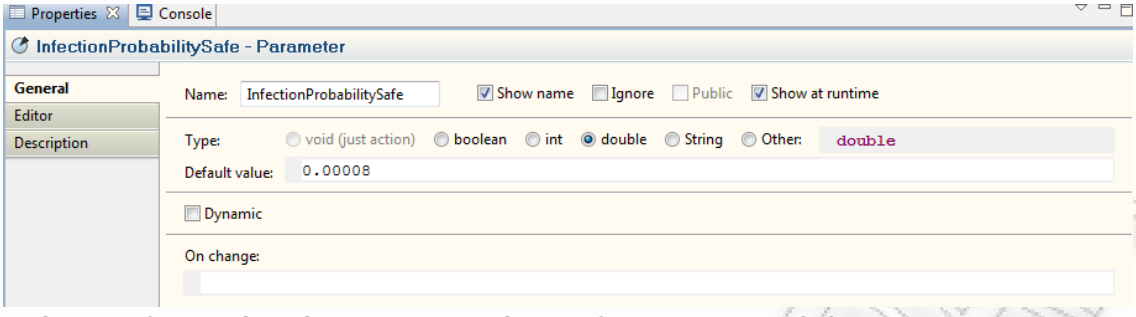
Παράμετροι της κλάσης Main		
Όνομα παραμέτρου	Τύπος παραμέτρου	Αρχική τιμή παραμέτρου
InitialNumberofUsers	Double	500
SafeSyringeUsersFraction	Double	0.2
PercentInitiallyInfected	Double	0.1
MeanHIVLifeDuration	Double	2 * 365
FractionInvited	Double	0.5
InfectionProbabilityUnsafe	Double	0.008
InfectionProbabilitySafe	Double	0.00008

Πίνακας 7-3: Οι παράμετροι της κλάσης Main.

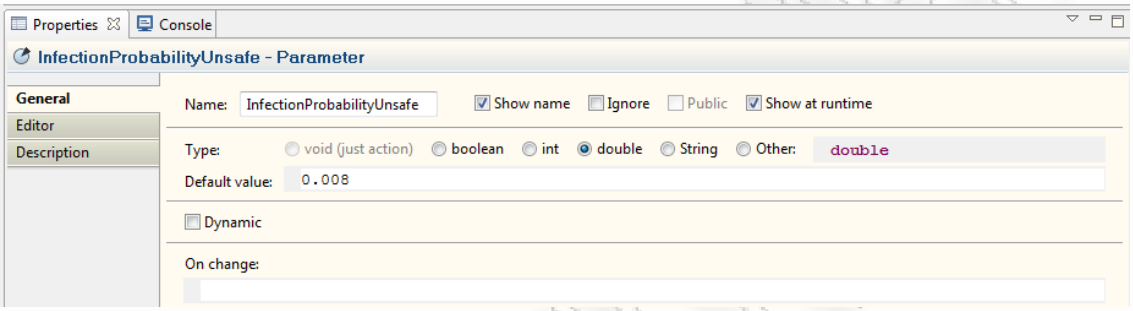


Εικόνα 7-11(α): Γενικές ιδιότητες της παραμέτρου FractionInvited.

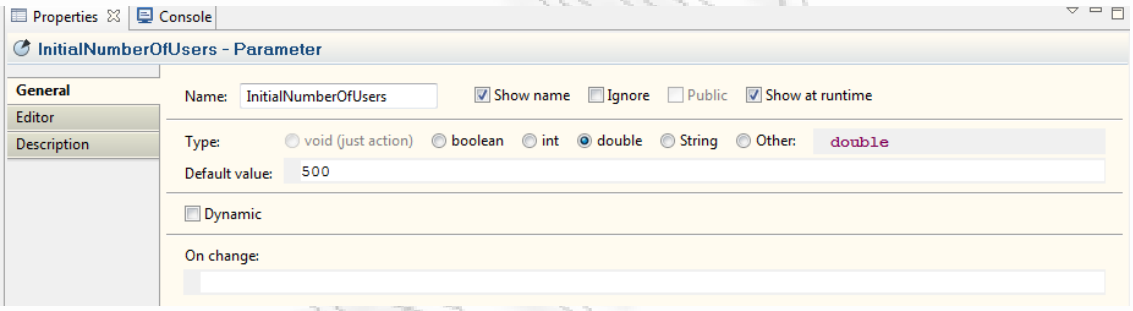
Στις εικόνες 7-11(α) έως και 7-11(ζ) παρουσιάζονται οι γενικές ιδιότητες των παραμέτρων της κλάσης Main.



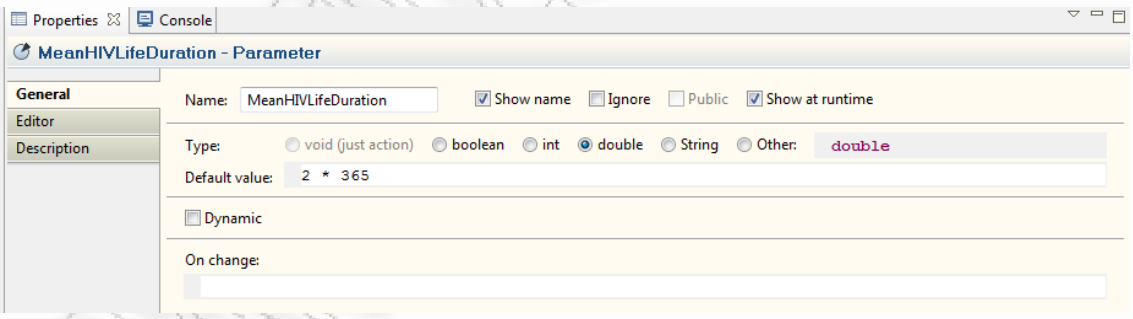
Εικόνα 7-11(β): Γενικές ιδιότητες της παραμέτρου InfectionProbabilitySafe.



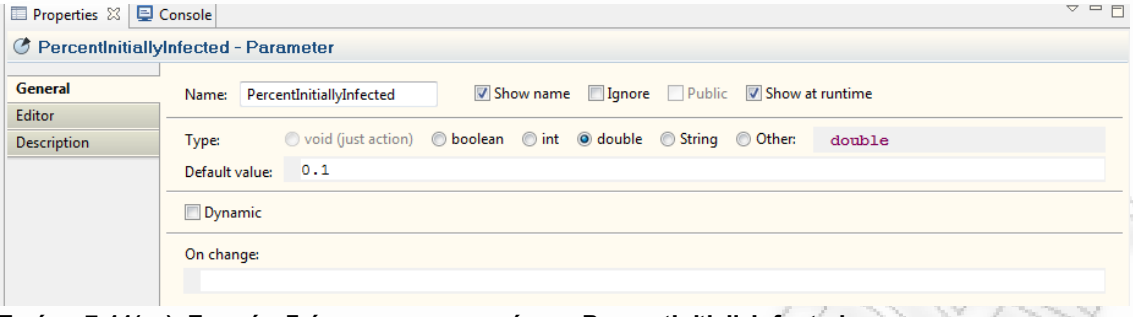
Εικόνα 7-11(γ): Γενικές ιδιότητες της παραμέτρου InfectionProbabilityUnsafe.



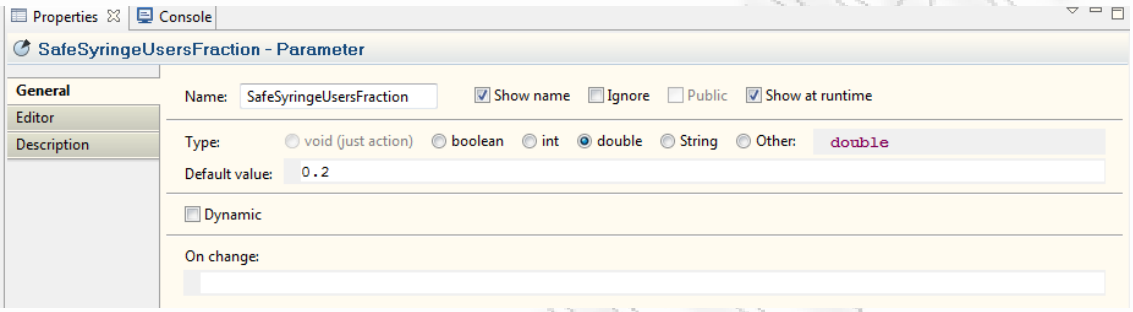
Εικόνα 7-11(δ): Γενικές ιδιότητες της παραμέτρου InitialNumberofUsers.



Εικόνα 7-11(ε): Γενικές ιδιότητες της παραμέτρου MeanHIVLifeDuration.



Εικόνα 7-11(στ): Γενικές ιδιότητες της παραμέτρου PercentInitiallyInfected.

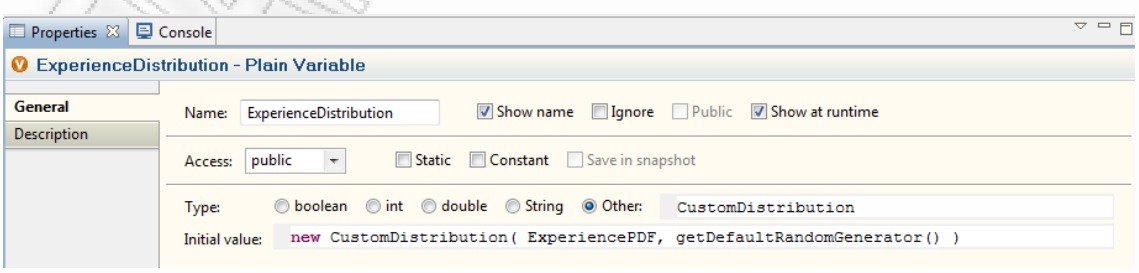


Εικόνα 7-11(ζ): Γενικές ιδιότητες της παραμέτρου SafeSyringeUsersFraction.

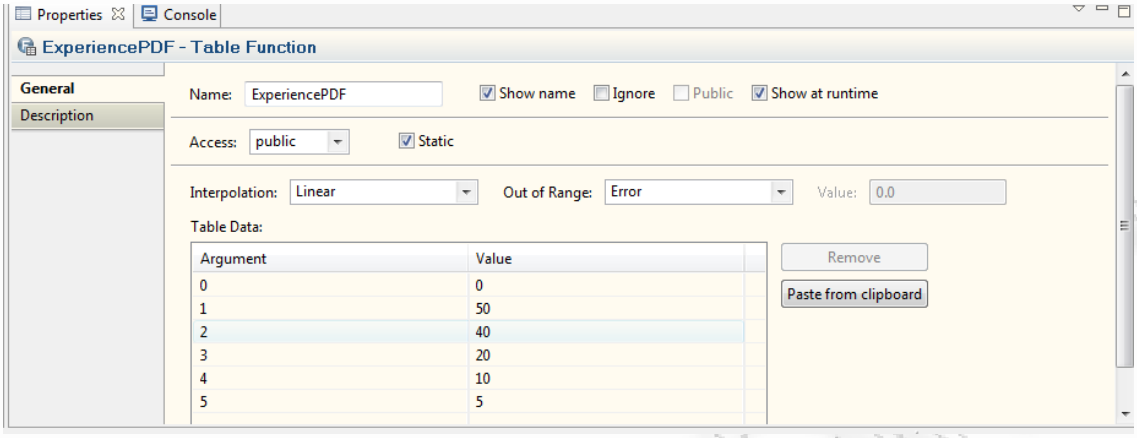
Η απλή μεταβλητή ExperienceDistribution είναι αυτή που καθορίζει τα χρόνια εμπειρίας των χρηστών σε συνδυασμό με τη συνάρτηση πίνακα ExperiencePDF. Η εμπειρία που καθορίζει την θέση του χρήστη στην κοινωνία των χρηστών είναι η μόνη πηγή ετερογένειας του μοντέλου. Το γεγονός UpdateHistograms χρησιμοποιείται για την επικαιροποίηση των ιστογραμμάτων. Η απλή συνάρτηση setHomeLocation καθορίζει τις συντεταγμένες του σημείου όπου έχει μαζευτεί η ομάδα των χρηστών για να κάνουν χρήση ενέσιμων ναρκωτικών, στην οριοθετημένη γεωγραφική περιοχή.

Η κλάση περιέχει σύνολα δεδομένων στα οποία αποθηκεύονται τα ενδιάμεσα αποτελέσματα της προσομοίωσης και τα οποία χρησιμοποιούνται για τη γραφική απεικόνιση των αποτελεσμάτων στα ιστογράμματα που συμπληρώνουν τη γραφική διεπαφή. Τα δύο (2) είναι σύνολα δεδομένων (Data Set), το NInfectedDS (ο αριθμός των μολυσμένων με τον ιό χρηστών) και το NSusceptibleDS (ο αριθμός των υποψήφιων να μολυνθούν χρηστών). Το άθροισμα των χρηστών των συνόλων δίνει το σύνολο των χρηστών της προσομοίωσης. Να σημειωθεί πως κατά τη διάρκεια της προσομοίωσης δεν επιτρέπεται η εισαγωγή νέων χρηστών. Τα υπόλοιπα δύο (2) είναι τα δεδομένα ιστογραμμάτων (Histogram Data), τα SusceptibleH και InfectedH που εμφανίζουν τους χρήστες κατηγοριοποιημένους με βάση την εμπειρία τους, τα χρόνια χρήσης ναρκωτικών.

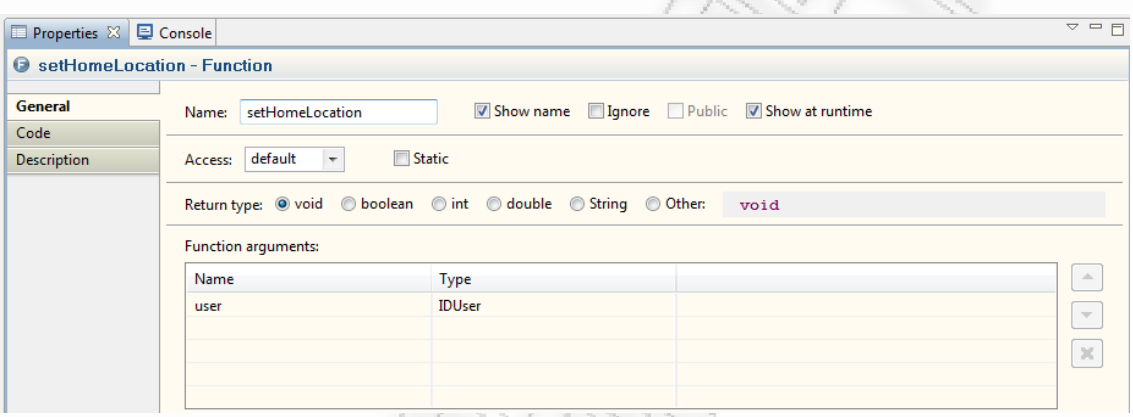
Τέλος, την κλάση ολοκληρώνουν το περιβάλλον environment και η λίστα των ενσωματωμένων αντικειμένων IDUsers. Στις εικόνες 7-12(α) – 7-12(ιη) παρουσιάζονται οι ιδιότητες των όσων αναφέρθηκαν παραπάνω.



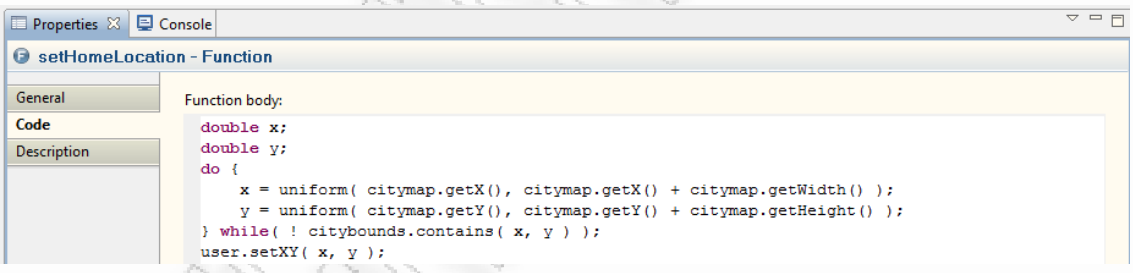
Εικόνα 7-12(α): Γενικές ιδιότητες της απλής μεταβλητής ExperienceDistribution.



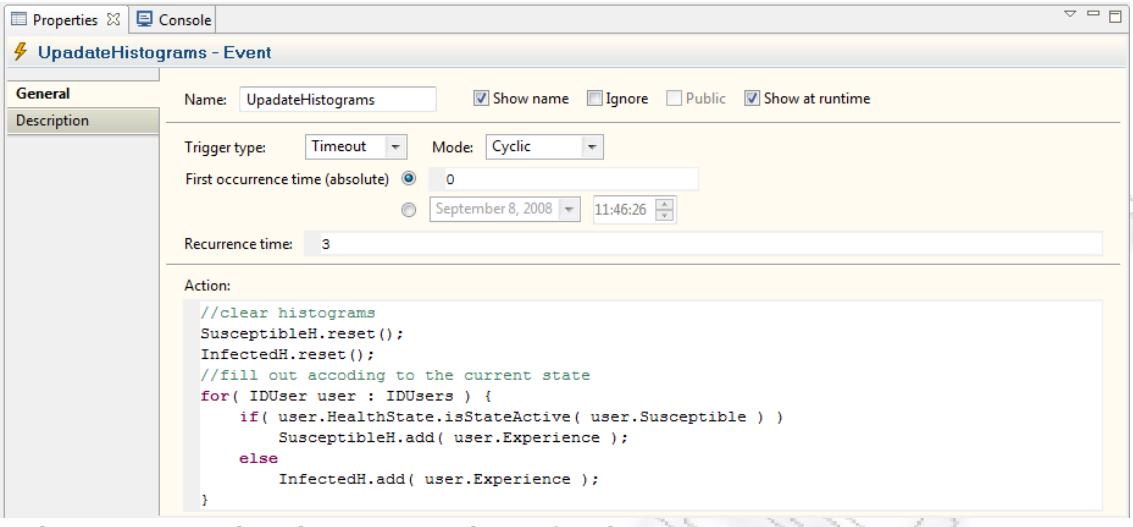
Εικόνα 7-12(β): Γενικές ιδιότητες της συνάρτησης πίνακα ExperiencePDF.



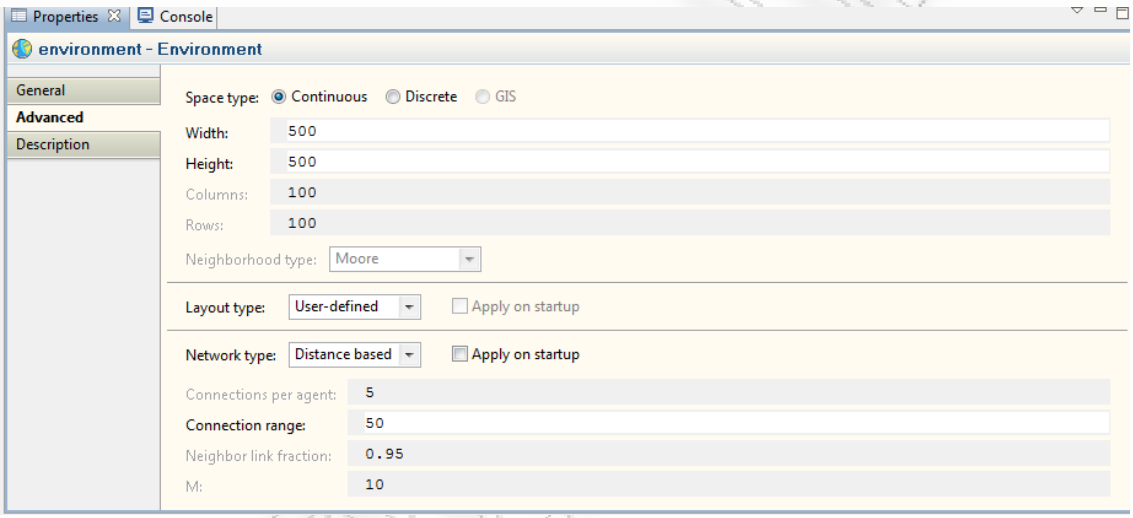
Εικόνα 7-12(γ): Γενικές ιδιότητες της απλής συνάρτησης setHomeLocation.



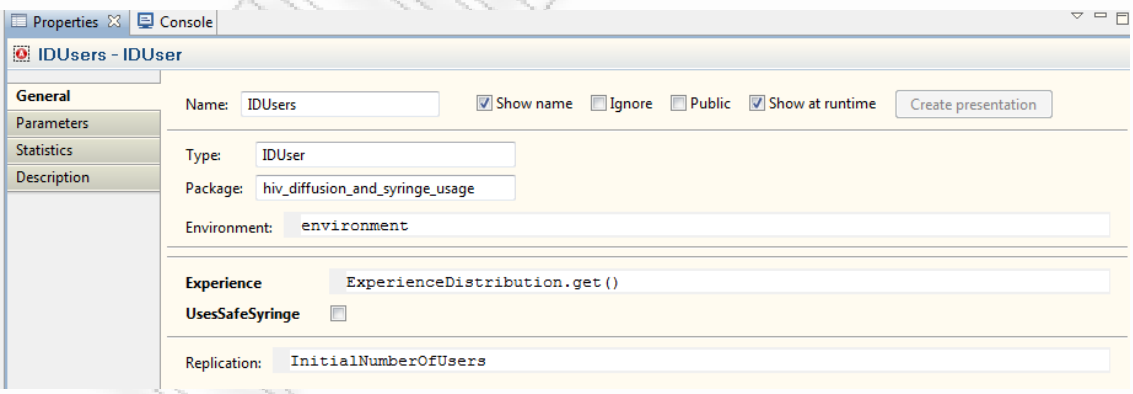
Εικόνα 7-12(δ): Κώδικας υλοποίησης της απλής μεταβλητής setHomeLocation.



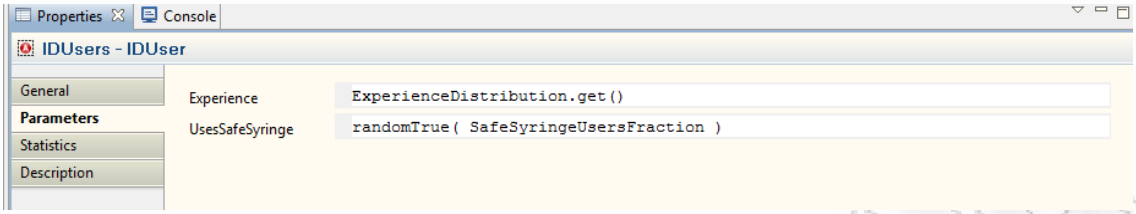
Εικόνα 7-12(ε): Γενικές ιδιότητες της απλής μεταβλητής UpdateHistograms.



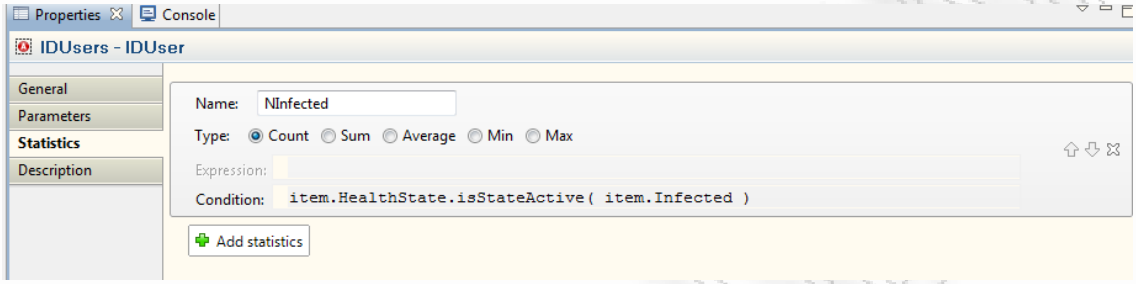
Εικόνα 7-12(στ): Προχωρημένες ιδιότητες του περιβάλλοντος environment.



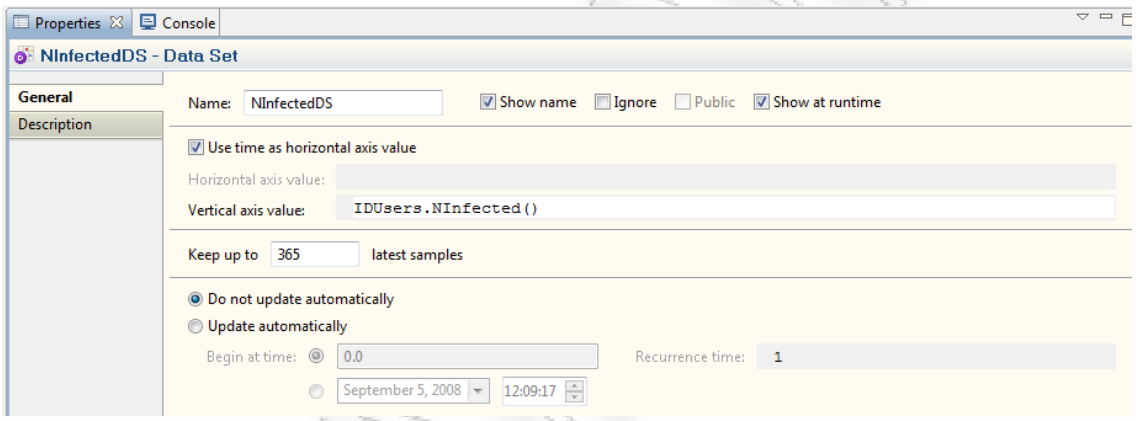
Εικόνα 7-12(ζ): Γενικές ιδιότητες του ενεργού αντικειμένου IDUsers.



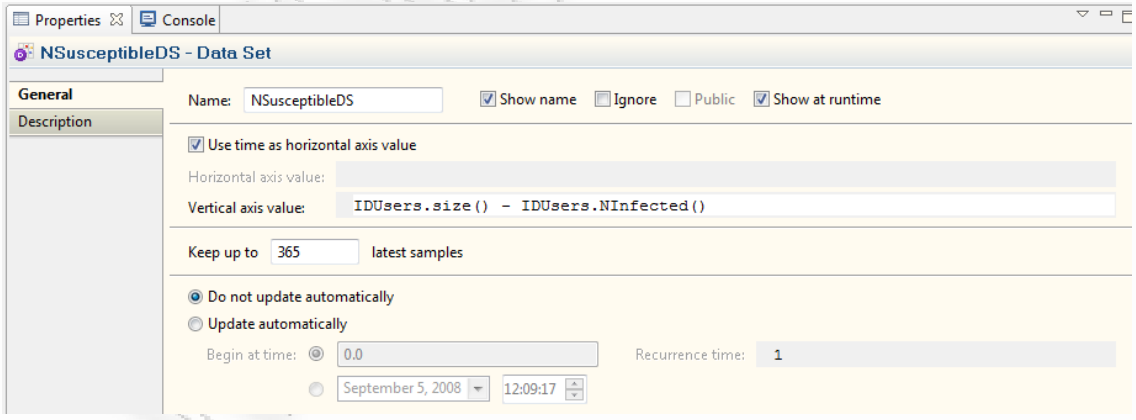
Εικόνα 7-12(η): Παράμετροι του ενεργού αντικειμένου IDUsers.



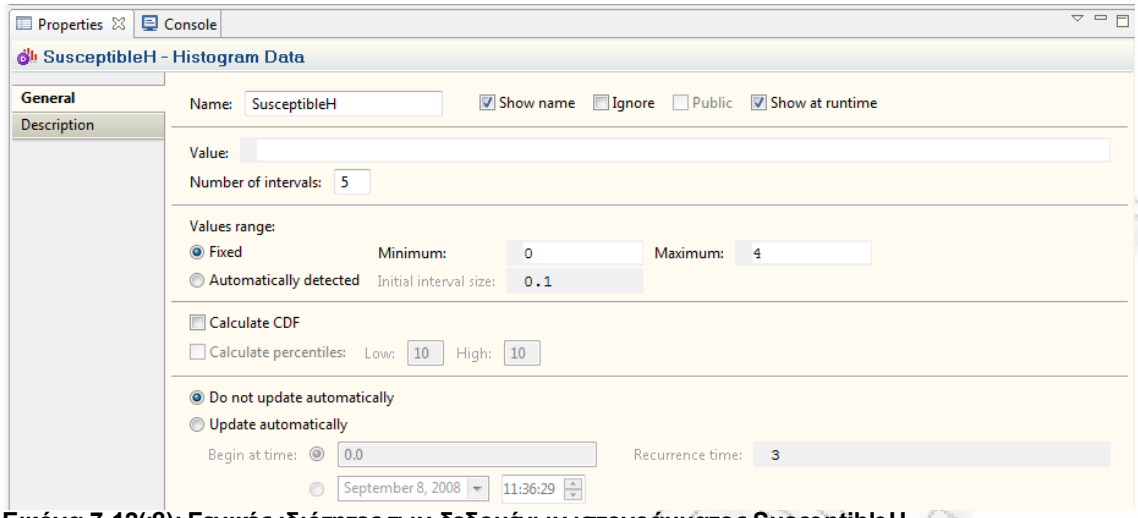
Εικόνα 7-12(θ): Στατιστικά του ενεργού αντικειμένου IDUsers.



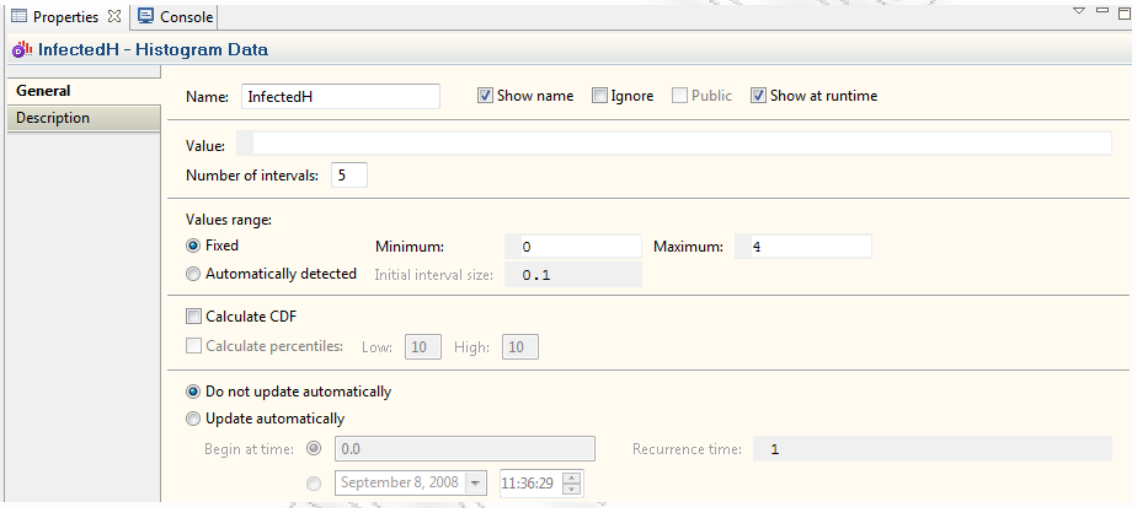
Εικόνα 7-12(ι): Γενικές ιδιότητες του συνόλου δεδομένων NInfectedDS.



Εικόνα 7-12(ια): Γενικές ιδιότητες της του συνόλου δεδομένων NSusceptibleDS.

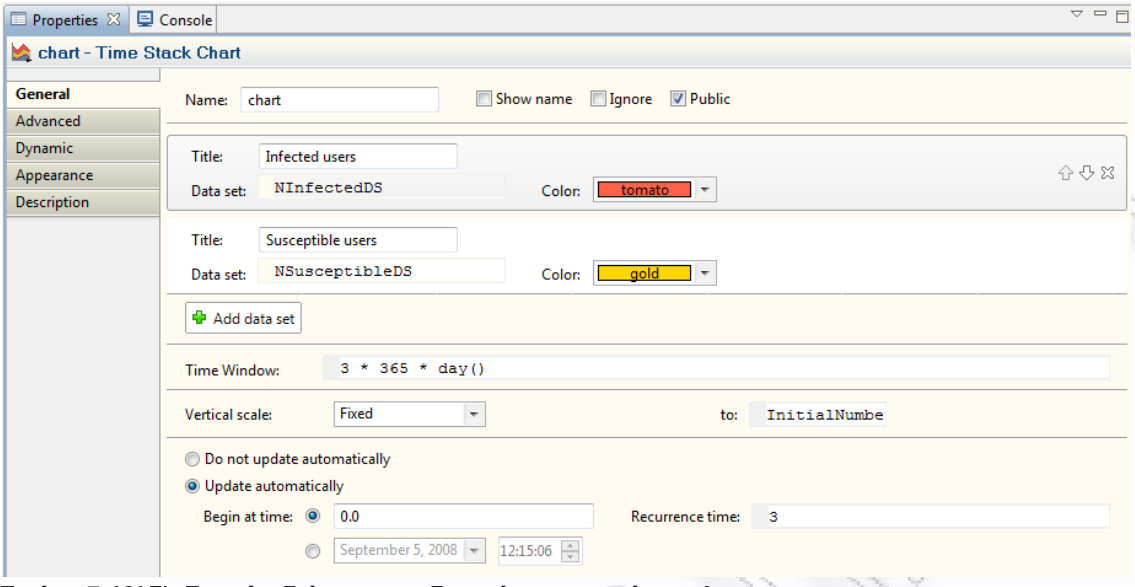


Εικόνα 7-12(ιβ): Γενικές ιδιότητες των δεδομένων ιστογράμματος SusceptibleH.

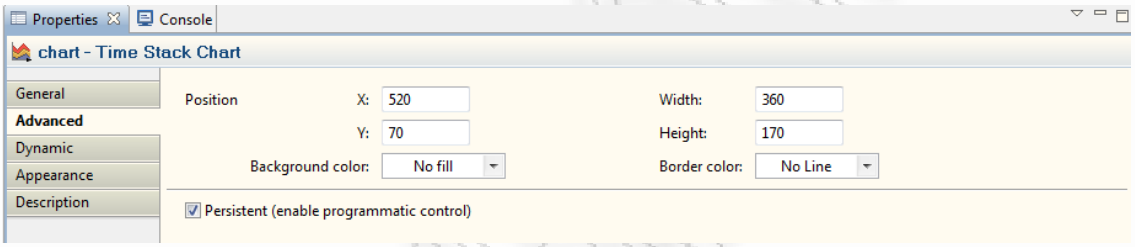


Εικόνα 7-12(ιγ): Γενικές ιδιότητες των δεδομένων ιστογράμματος InfectedH.

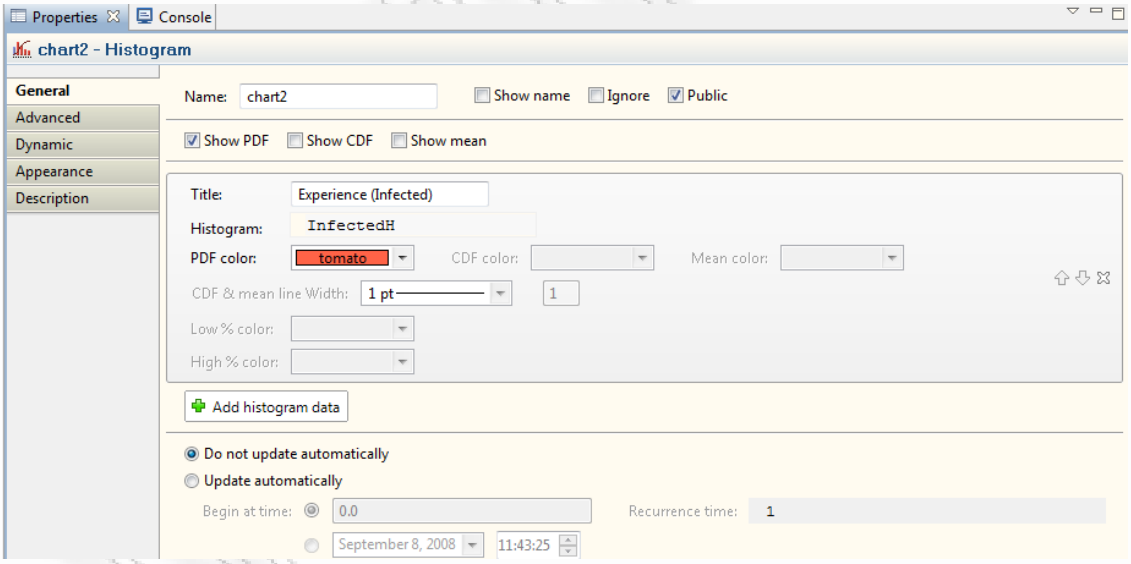
Στο σημείο αυτό θα πρέπει να ξεκαθαρίσουμε τι ακριβώς είναι τα δεδομένα ιστογράμματος. Είναι αντικείμενα τα οποία συλλέγουν στατιστικά στοιχεία κατά τη διάρκεια της προσομοίωσης. Τα αντικείμενα αυτά μπορούν να προχωρήσουν σε βασικές στατιστικές αναλύσεις, στη δημιουργία κατανομών και συναρτήσεων πυκνότητας-πιθανότητας και στον υπολογισμό αθροιστικών συναρτήσεων κατανομής. Τα αποτελέσματα οπτικοποιούνται μέσω των (αντικειμένων) ιστογραμμάτων.



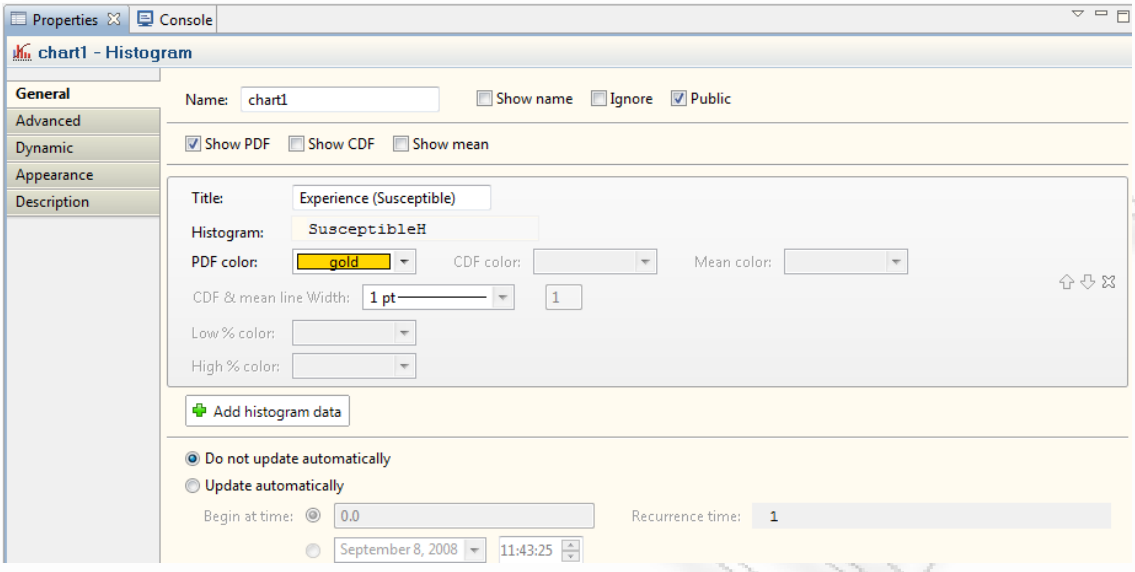
Εικόνα 7-12(ιδ): Γενικές ιδιότητες του διαγράμματος χρόνου chart.



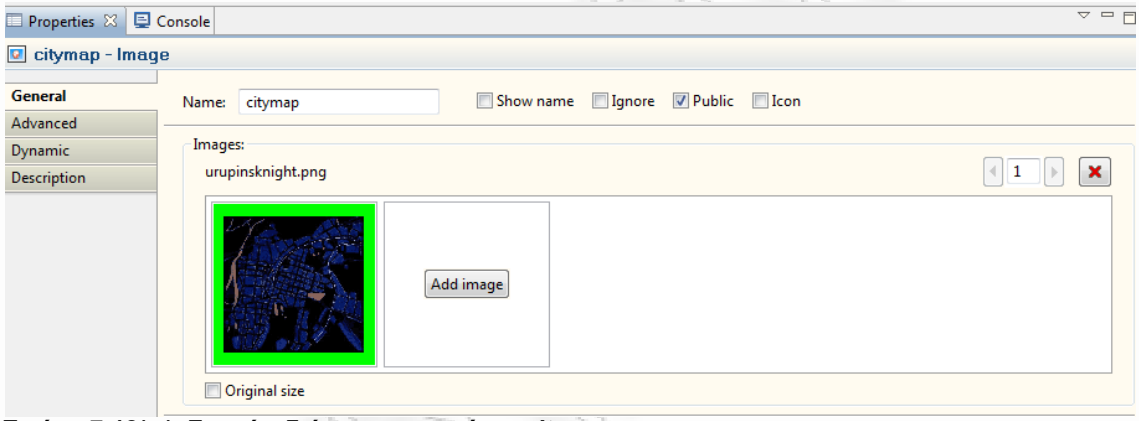
Εικόνα 7-12(ιε): Προχωρημένες ιδιότητες του διαγράμματος χρόνου chart.



Εικόνα 7-12(ιστ): Γενικές ιδιότητες του ιστογράμματος chart2.



Εικόνα 7-12(ιζ): Γενικές ιδιότητες του ιστογράμματος chart1.



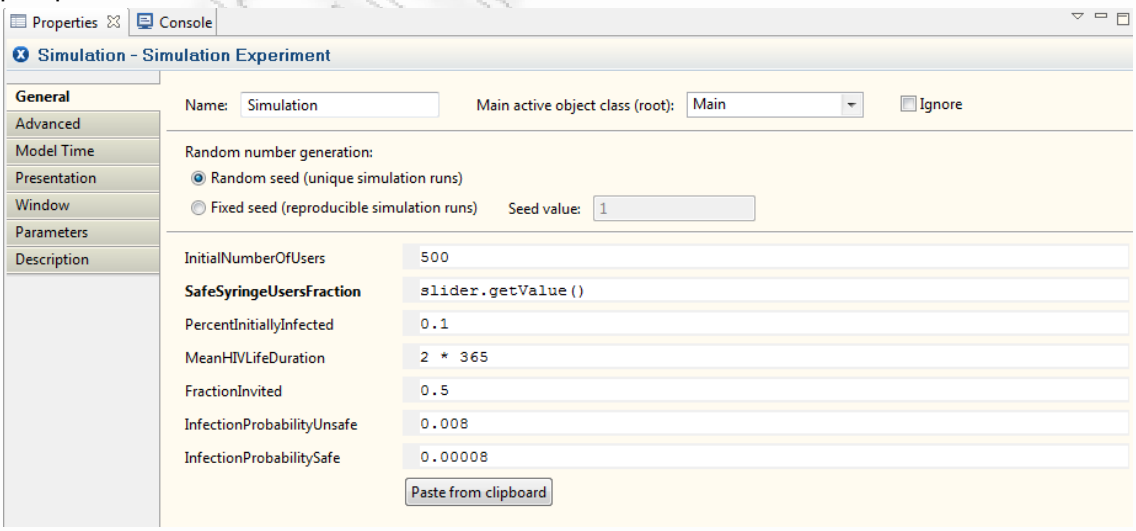
Εικόνα 7-12(ιη): Γενικές ιδιότητες της εικόνας citymap.

7.3 Η κλάση προσομοίωσης του έργου, Simulation:Main

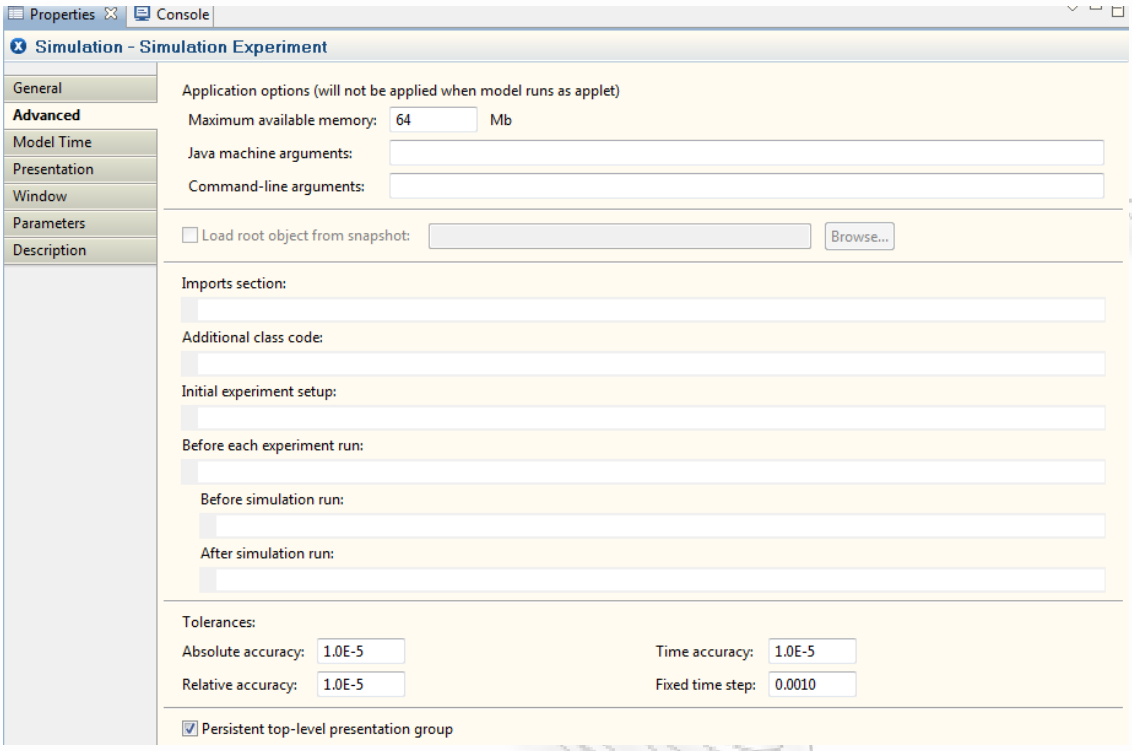
Για τη προσομοίωση έχει δημιουργηθεί μια πρώτη εισαγωγική «καρτέλα» στο γραφικό περιβάλλον διεπαφής του μοντέλου. Πέρα από τα λογότυπα της εταιρείας XJTek και του οργανισμού RTI εμφανίζεται μια εμπεριστατωμένη περιγραφή του μοντέλου, ένα κομβίο έναρξης της προσομοίωσης και μια ράβδος κύλισης για την αρχική εισαγωγή του ποσοστού των χρηστών που χρησιμοποιούν αμόλυντες σύριγγες, εικόνα 7-13.

Εικόνα 7-13: Το πρώτο παράθυρο της προσομοίωσης του έργου HIV Diffusion and Syringe Usage.

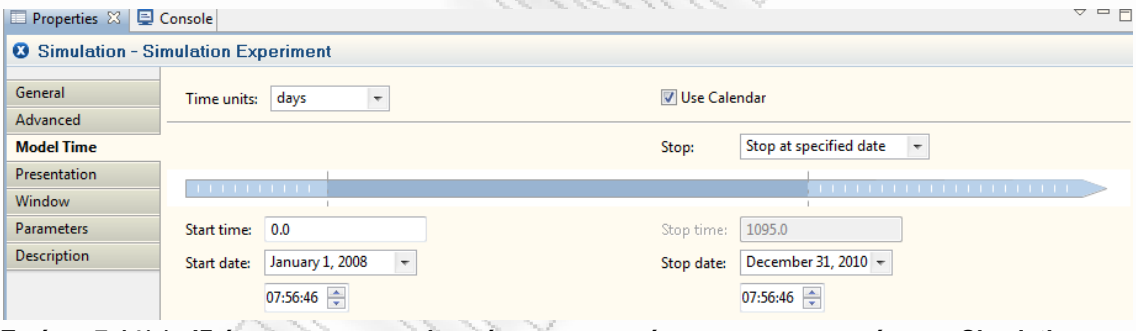
Στις επόμενες εικόνες 7-14(α) έως και 7-14(στ) παρουσιάζονται οι ιδιότητες της κλάσης προσομοίωσης του μοντέλου και οι παράμετροι της προσομοίωσης που χρησιμοποιούνται από το Λογισμικό για την εκτέλεση της προσομοίωσης. Οι περισσότεροι παράμετροι της προσομοίωσης έχουν προκαθορισμένες τιμές από το Λογισμικό, ο χρήστης βέβαια, μπορεί να τις μεταβάλλει.



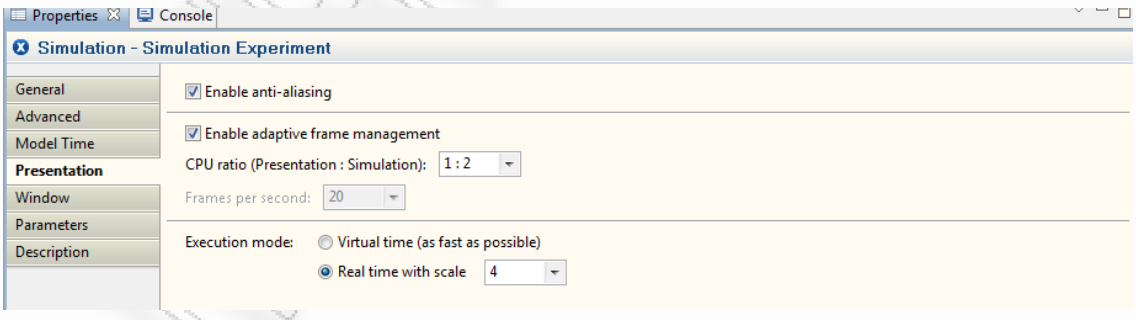
Εικόνα 7-14(α): Γενικές ιδιότητες της προσομοίωσης Simulation του έργου HIV Diffusion and Syringe Usage.



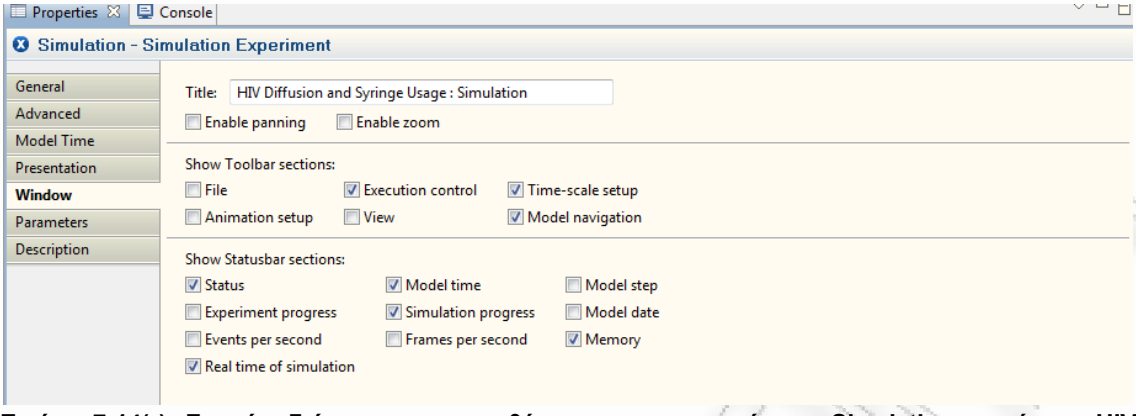
Εικόνα 7-14(β): Προχωρημένες ιδιότητες της προσομοίωσης Simulation του έργου HIV Diffusion and Syringe Usage.



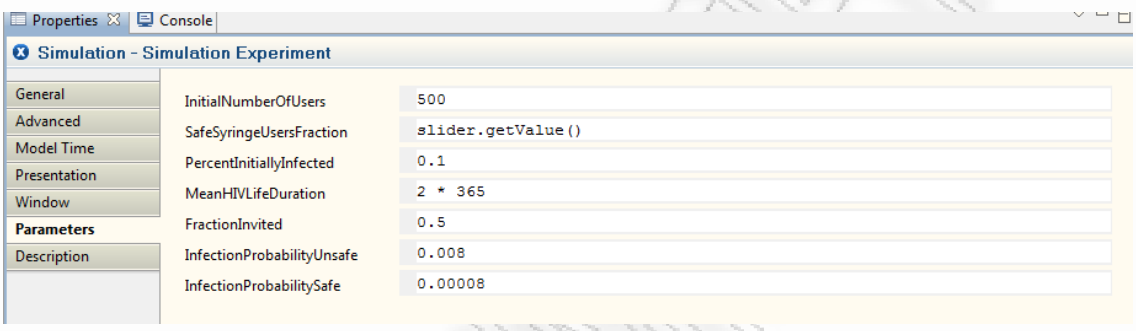
Εικόνα 7-14(γ): Ιδιότητες της μοντελοποίησης του χρόνου της προσομοίωσης Simulation του έργου HIV Diffusion and Syringe Usage.



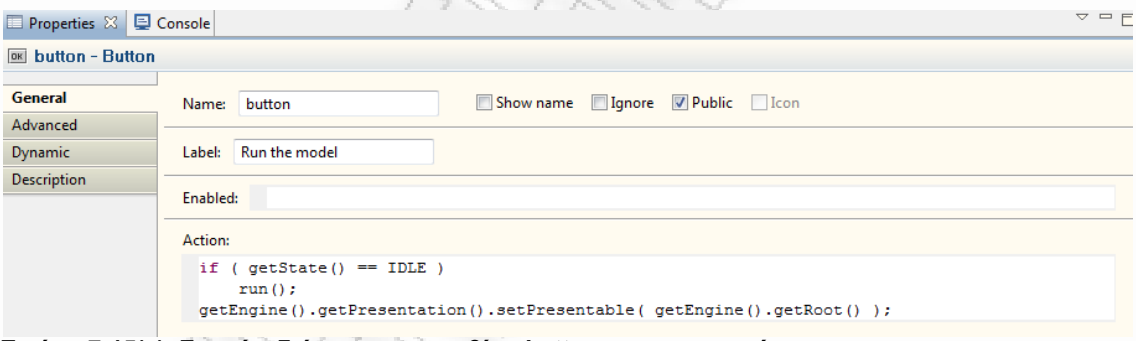
Εικόνα 7-14(δ): Ιδιότητες της παρουσίασης της προσομοίωσης Simulation του έργου HIV Diffusion and Syringe Usage.



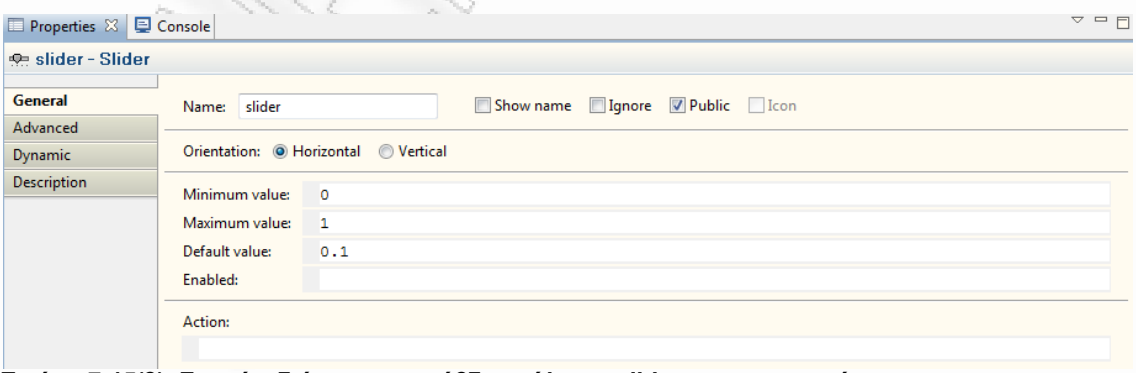
Εικόνα 7-14(ε): Γενικές ιδιότητες του παραθύρου της προσομοίωσης Simulation του έργου HIV Diffusion and Syringe Usage.



Εικόνα 7-14(στ): Προεπισκόπιση των ιδιοτήτων των παραμέτρων της προσομοίωσης Simulation του έργου HIV Diffusion and Syringe Usage.



Εικόνα 7-15(α): Γενικές ιδιότητες του κομβίου button της προσομοίωσης.

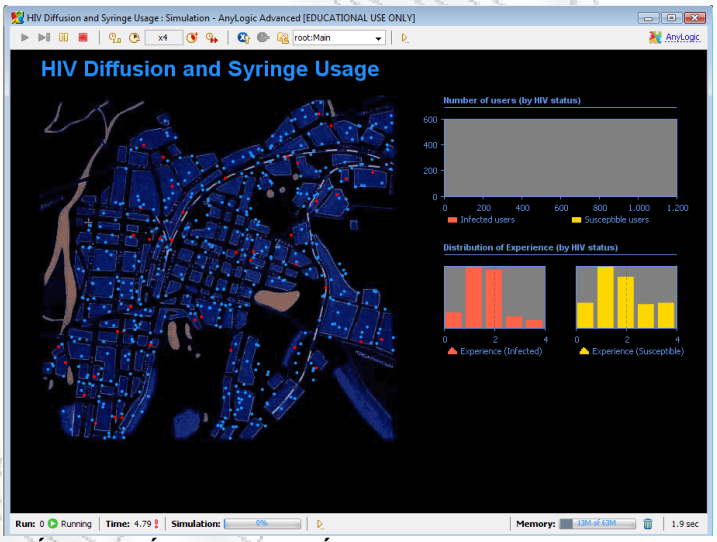


Εικόνα 7-15(β): Γενικές ιδιότητες της ράβδου κύλισης slider της προσομοίωσης.

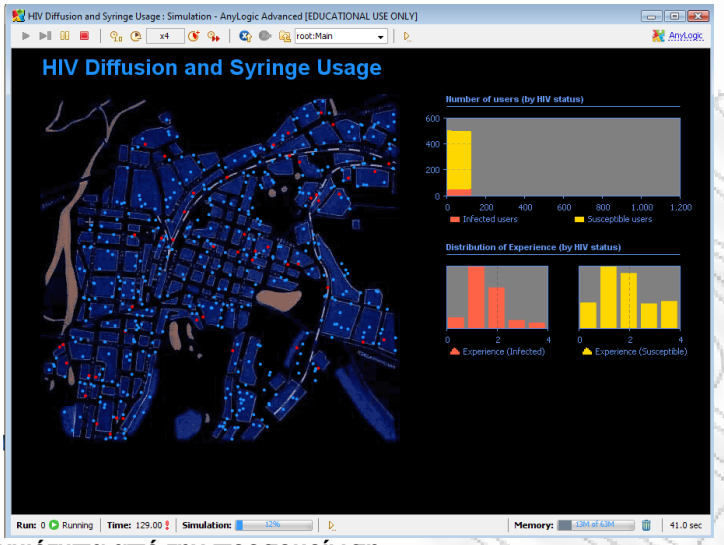
Οι εικόνες 7-15(α) και 7-15(β) παρουσιάζουν τις γενικές ιδιότητες του κομβίου έναρξης της προσομοίωσης και της ράβδου κύλισης, αντίστοιχα. Το δεύτερο παράθυρο της προσομοίωσης που εμφανίζεται με την εκτέλεση της προσομοίωσης, αποτελείται από τον

οριοθετημένο χάρτη της γεωγραφικής περιοχής που μας ενδιαφέρει και τα τρία (3) διαγράμματα παρουσίασης των αποτελεσμάτων, δύο (2) ραβδογράμματα και ένα (1) διάγραμμα συνεχούς χρόνου.

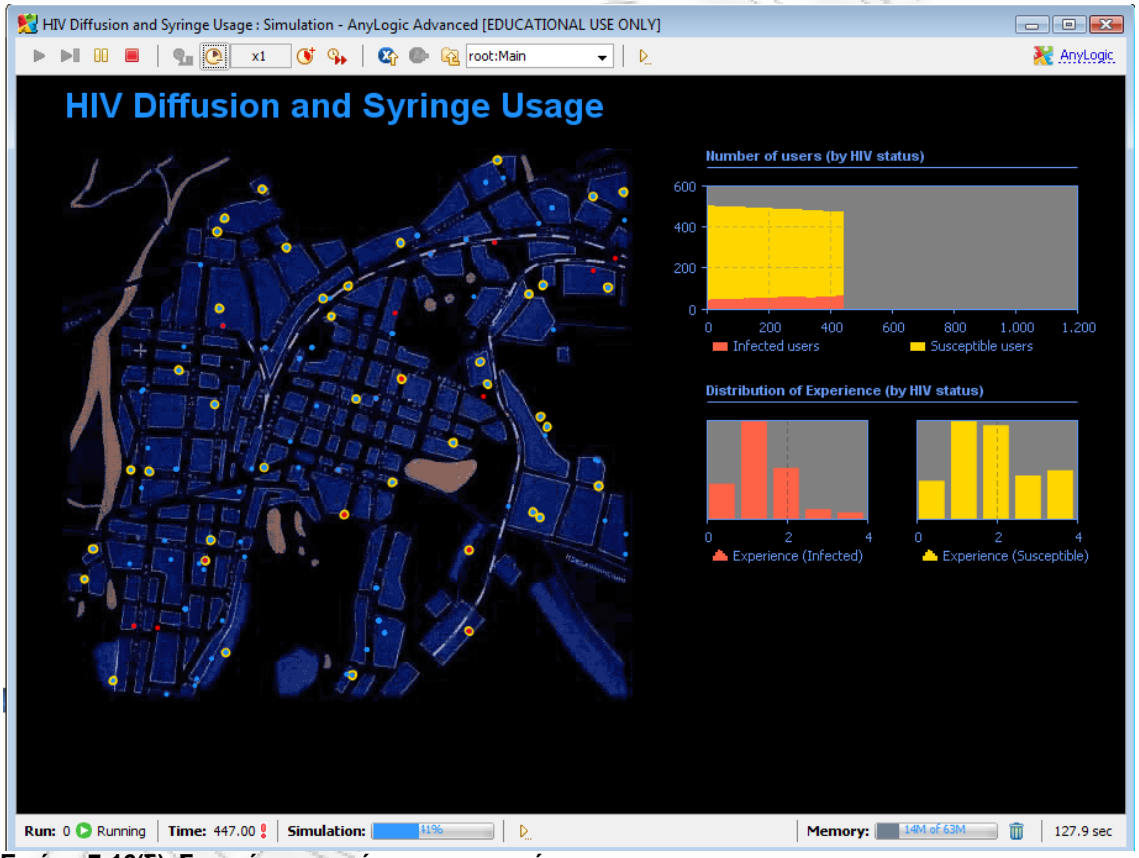
Εικόνα 7-16(α): Στιγμιότυπα από την προσομοίωση



Εικόνα 7-16(β): Στιγμιότυπα από την προσομοίωση



Εικόνα 7-16(γ): Στιγμιότυπα από την προσομοίωση



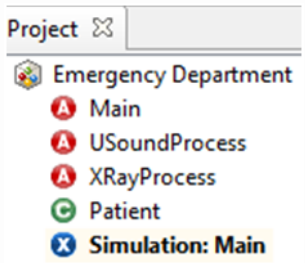
Εικόνα 7-16(δ): Στιγμιότυπα από την προσομοίωση

Τα διαγράμματα κλάσεων και ο πλήρης κώδικας του παραδείγματος βρίσκονται στο Παράρτημα 5.

ΚΕΦΑΛΑΙΟ 8

Μοντελοποίηση ενός Τμήματος Επειγόντων Περιστατικών με τη χρήση της Βιβλιοθήκης Enterprise του Λογισμικού

Για την υλοποίηση του παραδείγματος, στο οποίο θα αναφερθούμε στη συνέχεια, χρησιμοποιείται η ενσωματωμένη βιβλιοθήκη Enterprise του Λογισμικού, η οποία είναι η βιβλιοθήκη υλοποίησης μοντέλων διακριτών γεγονότων και συστημάτων πραγματικού χρόνου με αντικείμενα διαδικασιών όπως π.χ. η ουρά αναμονής, η καθυστέρηση, η παροχή μιας υπηρεσίας κλπ. Το θεωρητικό υπόβαθρο της βιβλιοθήκης Enterprise έχει τις ρίζες του στην οργάνωση και διοίκηση επιχειρήσεων και κυρίως στην επιχειρησιακή έρευνα. Με τη χρήση της συγκεκριμένης βιβλιοθήκης είναι εύκολο να δημιουργήσουμε μοντέλα, να συλλέξουμε στοιχεία δεδομένων, να επεξεργαστούμε στατιστικά στοιχεία δεδομένων και να οπτικοποιήσουμε τα αποτελέσματα προσομοιώσεων βιομηχανικών και επιχειρηματικών διαδικασιών, απλών αλλά και σύνθετων συστημάτων και διαδικασιών υπηρεσιών όπως π.χ. μοντέλα τραπεζών, αεροδρομίων, νοσοκομείων κλπ.

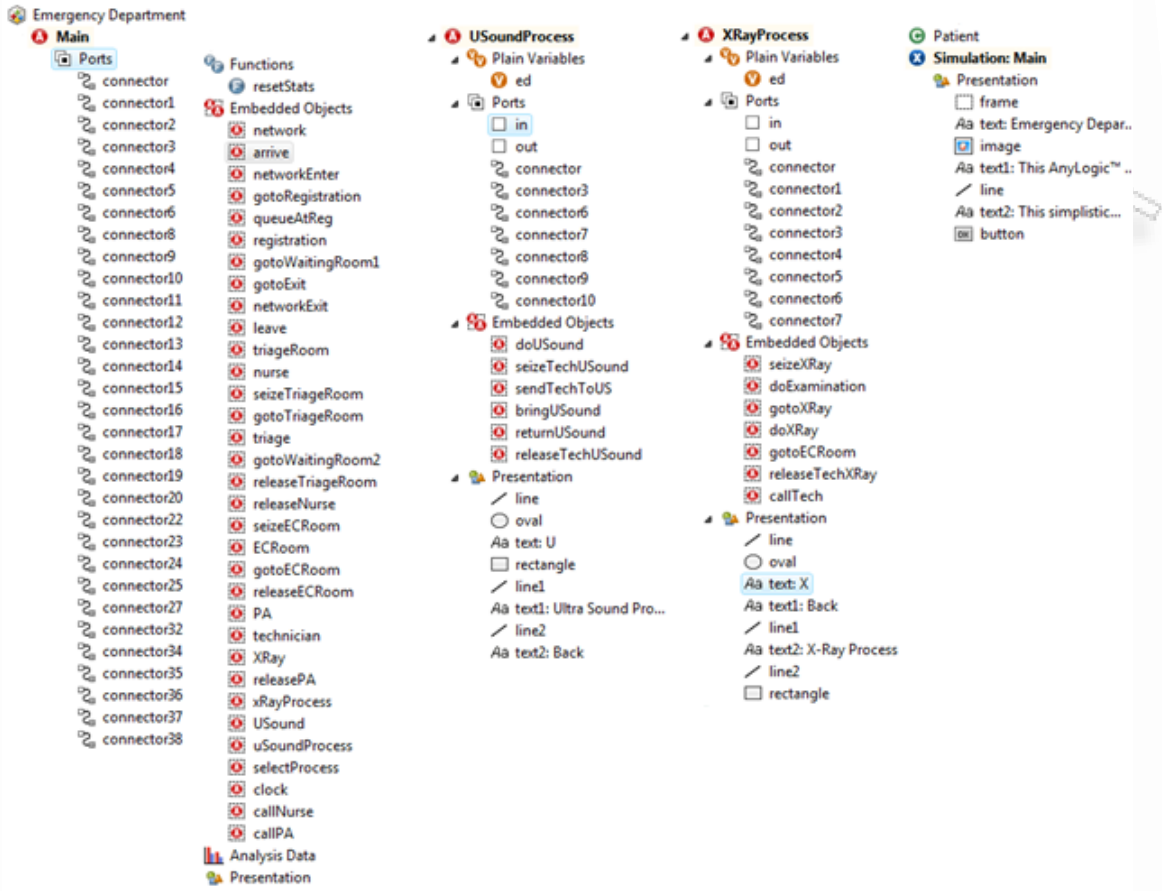


Εικόνα 8-1: Η ιεραρχία του έργου Emergency Department

Ορισμένα στοιχεία, αντικείμενα, της βιβλιοθήκης Enterprise είναι ειδικά σχεδιασμένα (α) για τη μοντελοποίηση διαδικασιών που λαμβάνουν χώρα σε συγκεκριμένους χώρους και (β) για την δυνατότητα μοντελοποίησης της κίνησης οντοτήτων και πόρων. Η μοντελοποίηση συστημάτων που υλοποιούνται με τη χρήση τέτοιων αντικειμένων, τα οποία έχουν το διακριτικό πρόθεμα network, ονομάζεται μοντελοποίηση που βασίζεται στα Δίκτυα, (Network Based Modeling). Ως Δίκτυο, (network), ορίζεται ένα σύνολο από κόμβους, (nodes), που είναι διασυνδεδεμένοι μέσω διαδρομών, (segments, paths). Η κίνηση γίνεται πάντα κατά μήκος της μικρότερης διαδρομής, μεταξύ του κόμβου αναχώρησης και του κόμβου άφιξης. Πάνω σε κάθε τέτοια διαδρομή μπορεί οποιαδήποτε οντότητα που έχει μοντελοποιηθεί και είναι μέλος του Δικτύου να κινείται στις διαδρομές του Δικτύου, να έχει ταχύτητα ανεξάρτητη από τις ταχύτητες των άλλων οντοτήτων. Η ταχύτητα των οντοτήτων μπορεί να αλλάζει δυναμικά.

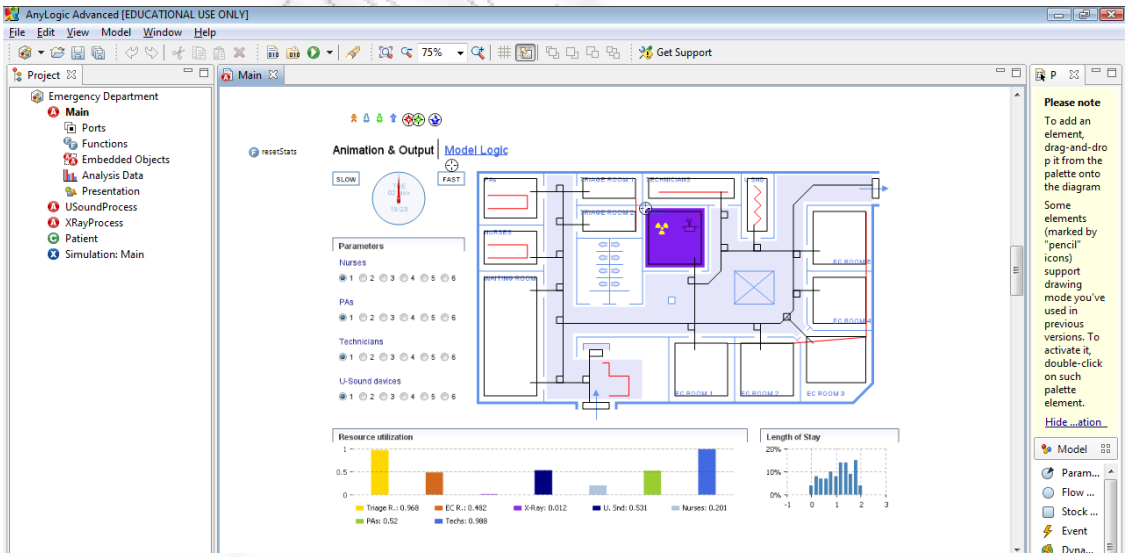
Το παράδειγμα που θα εξετάσουμε μοντελοποιεί ένα νοσοκομειακό τμήμα επειγόντων περιστατικών. Η εικόνα 8-1 παρουσιάζει την ιεραρχία του έργου Emergency Department. Το παράδειγμα μας αποτελείται από τρεις (3) κλάσεις ενεργών αντικειμένων, μια (1) κλάση η οποία υλοποιεί, μέσω κώδικα Java, το αντικείμενο Ασθενής και μία (1) κλάση που υλοποιεί το τμήμα της προσομοίωσης. Το κυρίως τμήμα του μοντέλου διαμορφώνεται με τη βοήθεια της κάτοψης του τμήματος επειγόντων περιστατικών πάνω στην οποία ορίζεται το Δίκτυο, αποτελούμενο από κόμβους και από διαδρομές όπως ειπώθηκε ήδη προηγούμενα. Στο μοντέλο, διακρίνονται τρεις (3) τύπους πόρων: (α) το προσωπικό του τμήματος των επειγόντων περιστατικών (ιατρικό, νοσηλευτικό και τεχνικό) και τα πάγια του τμήματος που διακρίνονται σε (β) σταθερά και (γ) κινητά.

Στην εικόνα 8-2 παρουσιάζεται σε πλήρη ανάπτυξη η δομή του έργου Emergency Department. Εμφανίζονται οι κλάσεις που αποτελούν το έργο, η δενδρική δομή κάθε κλάσης και τα αντίστοιχα δομικά στοιχεία που αποτελούν κάθε κλάση.

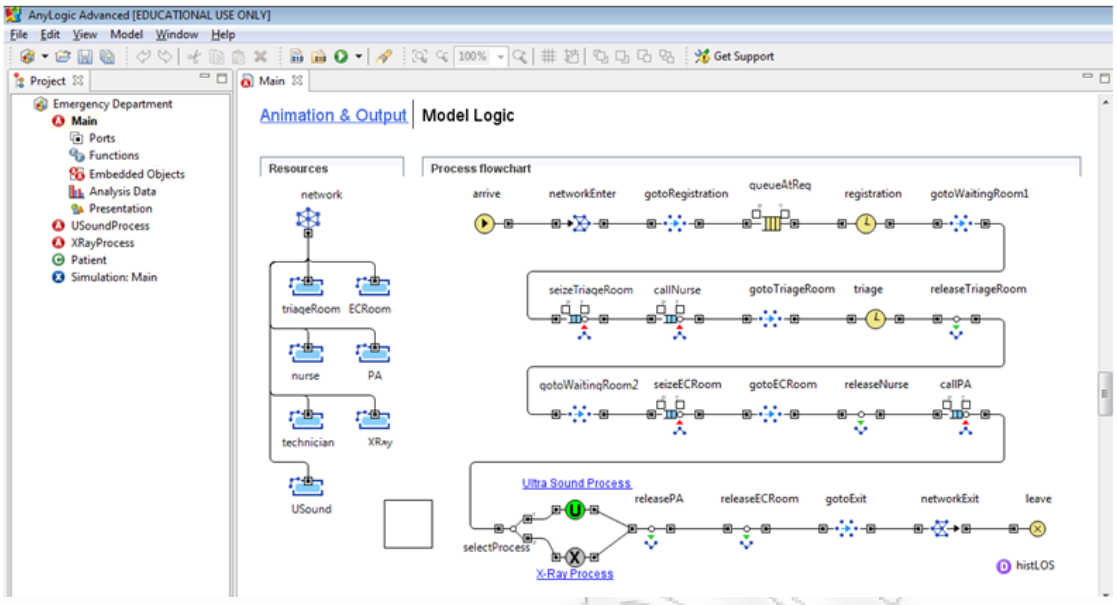


Εικόνα 8-2: Η ιεραρχία του έργου Emergency Department, σε πλήρη ανάπτυξη των κλάσεων που το αποτελούν.

8.1 Η κλάση Main



Εικόνα 8-3(α): Η γραφική υλοποίηση της κλάσης Main. Εμφανίζονται ξεκάθαρα οι διαδρομές και οι κόμβοι του Δικτύου πάνω στην κάτοψη.



Εικόνα 8-3(β): Η γραφική μοντελοποίηση της κλάσης Main.

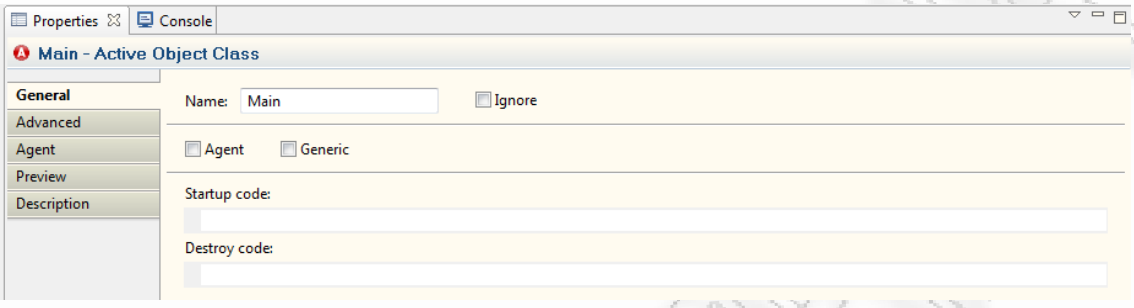
Η κλάση Main αποτελείται από ένα μεγάλο αριθμό θυρών, (ports), τριανταμία (31) τον αριθμό, με τις οποίες γίνεται η ανταλλαγή πληροφοριών μέσω του εσωτερικού μηχανισμού ανταλλαγής μηνυμάτων του Λογισμικού κατά την προσομοίωση. Οι ίδιες πόρτες, (και οι διαδρομές που τις συνδέουν), που στην εικόνα 8-3(α) αποτυπώνονται στην κάτοψη του τμήματος των επειγόντων περιστατικών σε συγκεκριμένες θέσεις δημιουργώντας το Δίκτυο, εμφανίζονται στην εικόνα 8-3(β) σε σειριακή μορφή. Το Δίκτυο, (network), στην εικόνα 8-3(β) το τμήμα των πόρων (Resources), αποτελείται από τις οντότητες triageRoom, ECRoom (Express Care Room), nurse, PA (Physicians Assistant), technician, XRay και USound.

Ένα ενδιαφέρον χαρακτηριστικό του Λογισμικού που εμφανίζεται στο συγκεκριμένο μοντέλο είναι το ότι μέσα από τον γραφικό συντάκτη του περιβάλλοντος ανάπτυξης του Λογισμικού μπορούμε να κάνουμε χρήση υπερσυνδέσμων. Στο μοντέλο μας, η κλάση Main αποτελείται από δύο (2) διαφορετικές καρτέλες. Επιλέγοντας το πεδίο Animation & Output βρισκόμαστε στο τμήμα της κλάσης που εμφανίζεται η κάτοψη του μοντέλου και τα ενδεικτικά αντικείμενα της. Επιλέγοντας το πεδίο Model Logic μεταφερόμαστε σε ένα δεύτερο επίπεδο αναπαράστασης της κλάσης όπου γίνεται χρήση των αντικειμένων της βιβλιοθήκης Enterprise για τη μοντελοποίηση των διακριτών γεγονότων.

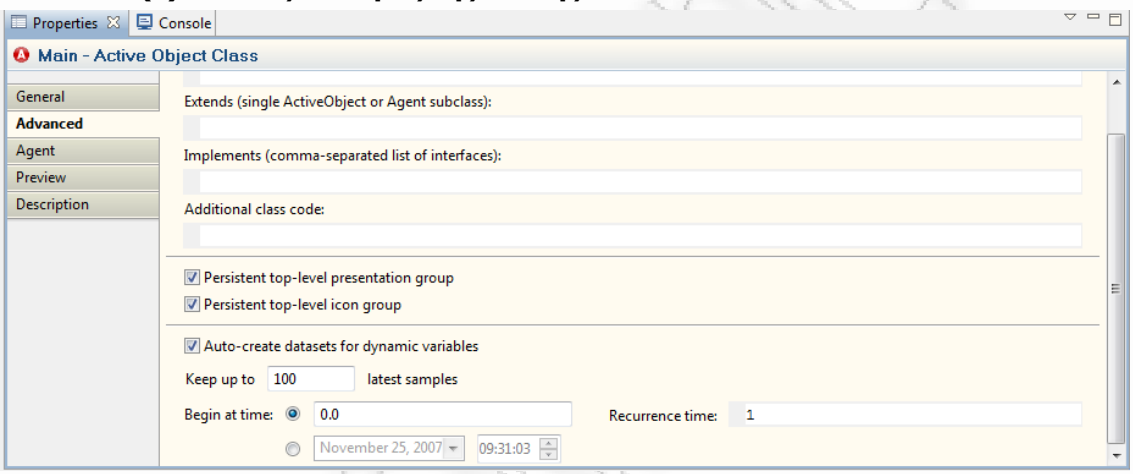
Το διάγραμμα ροής των διαδικασιών του μοντέλου, εικόνα 8-3(β), ξεκινά με την άφιξη του ασθενή στο τμήμα επειγόντων περιστατικών, την εγγραφή του ασθενούς μέσα από μια διαδικασία ουράς αναμονής, την μεταφορά του στην αίθουσα αναμονής, την αναζήτηση αίθουσας για την αξιολόγηση του περιστατικού (εξεταστήριο) και την αναζήτηση νοσοκόμας. Η διαδικασία συνεχίζεται με μεταφορά του ασθενή στην αίθουσα αξιολόγησης, γίνεται η εξέταση και αποδεσμεύεται η αίθουσα. Ο ασθενής μεταφέρεται σε νέα αίθουσα αναμονής, αναζητείται αίθουσα επείγουσας φροντίδας, οδηγείται στην αίθουσα των επειγόντων περιστατικών, αποδεσμεύεται η νοσοκόμα και καλείται το ειδικευμένο προσωπικό που θα αναλάβει το περιστατικό. Ανάλογα με το είδος του περιστατικού, ο ασθενής μεταφέρεται είτε στην αίθουσα του υπέρηχου είτε στην αίθουσα των ακτινογραφιών. Αποδεσμεύεται το προσωπικό και η αίθουσα των επειγόντων και ο ασθενής φεύγει από το τμήμα των επειγόντων περιστατικών. Τα παραπάνω υλοποιούνται μέσω των ενσωματωμένων αντικειμένων, των οντοτήτων, που παρουσιάζονται στην εικόνα 8-2.

Στην εικόνα 8-4(α) και στην εικόνα 8-4(β) παρουσιάζονται οι γενικές και οι προχωρημένες ιδιότητες της κλάσης Main. Στην εικόνα 8-5(α) δίνονται οι γενικές ιδιότητες της συνάρτησης resetStats, η οποία έχει ως αντικείμενο της αρχικοποίηση των στατιστικών. Η εικόνα 8-5(β) δείχνει τον κώδικα του σώματος της συνάρτησης.

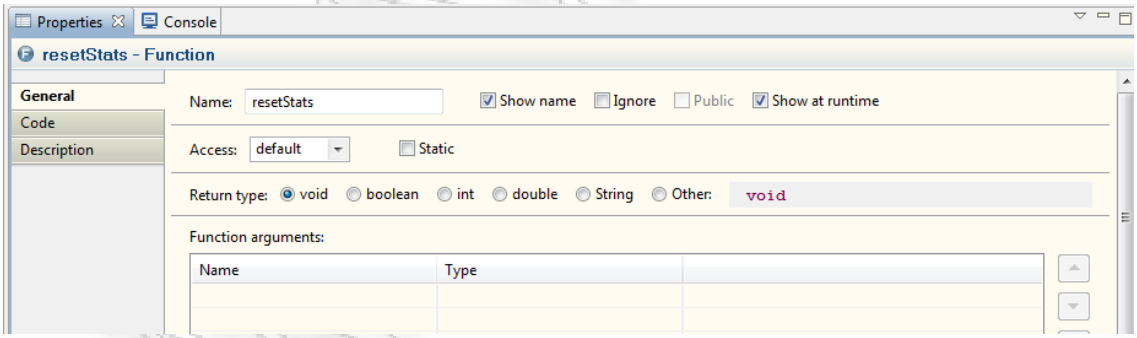
Οι εικόνες από την 8-6 έως και την 8-38 εμφανίζουν τις γενικές ιδιότητες των οντοτήτων. Όπου τα ενσωματωμένα αντικείμενα έχουν δηλώσεις κώδικα, στις καρτέλες των παραμέτρων τους, παρουσιάζονται επίσης. Σε τέτοιες περιπτώσεις οι εικόνες χωρίζονται στις κατηγορίες (α) και (β) αντίστοιχα.



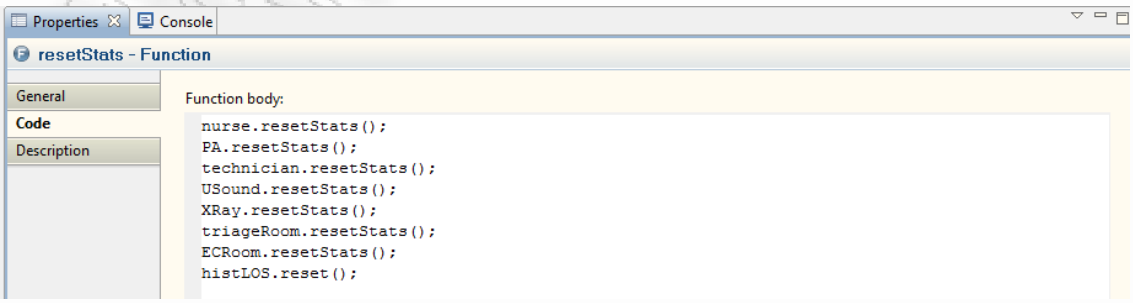
Εικόνα 8-4(α): Γενικές ιδιότητες της κλάσης Main.



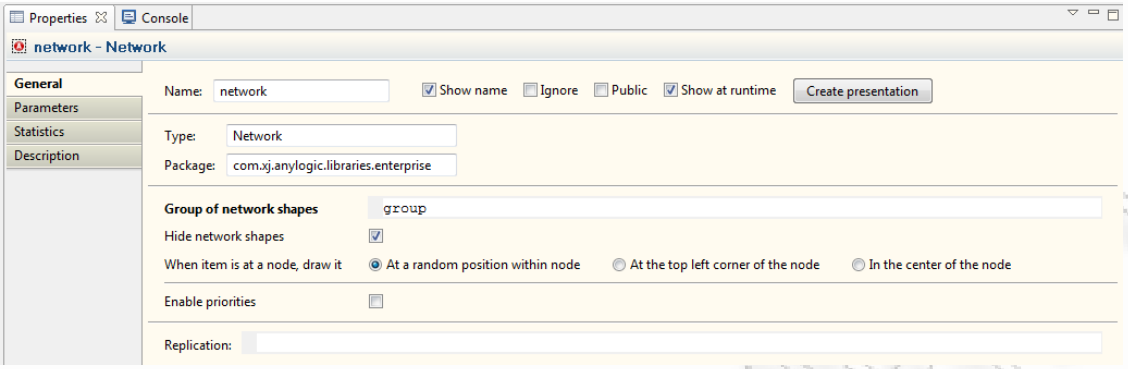
Εικόνα 8-4(β): Προχωρημένες ιδιότητες της κλάσης Main.



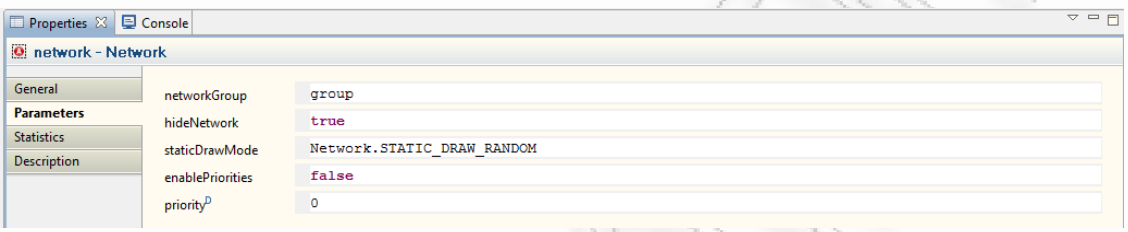
Εικόνα 8-5(α): Γενικές ιδιότητες της συνάρτησης resetStats.



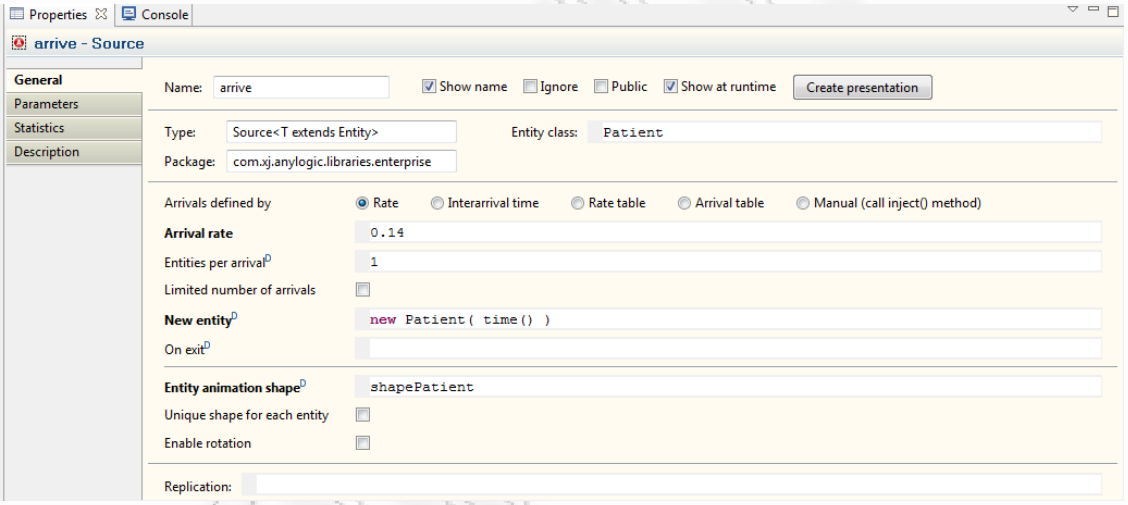
Εικόνα 8-5(β): Κώδικας της συνάρτησης resetStats.



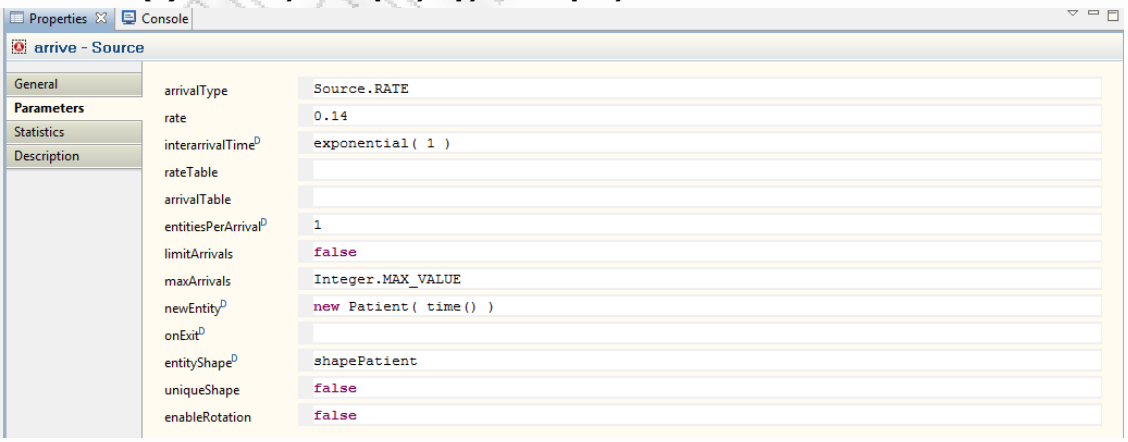
Εικόνα 8-6(α): Γενικές ιδιότητες της οντότητας network τύπου Network.



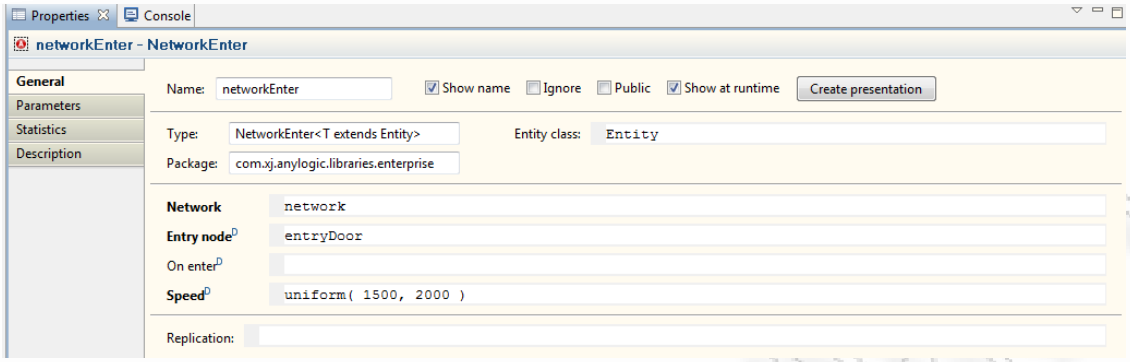
Εικόνα 8-6(β): Παράμετροι της οντότητας network τύπου Network.



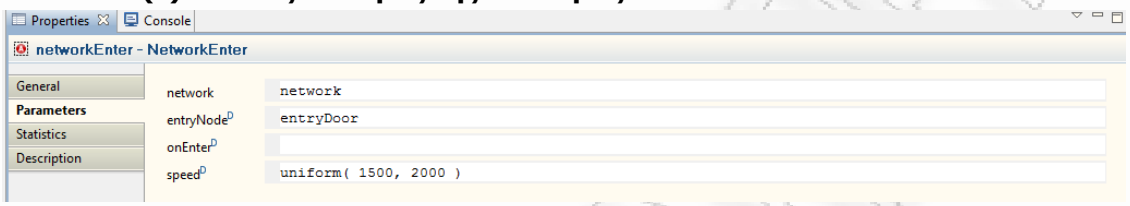
Εικόνα 8-7(α): Γενικές ιδιότητες της οντότητας arrive τύπου Source.



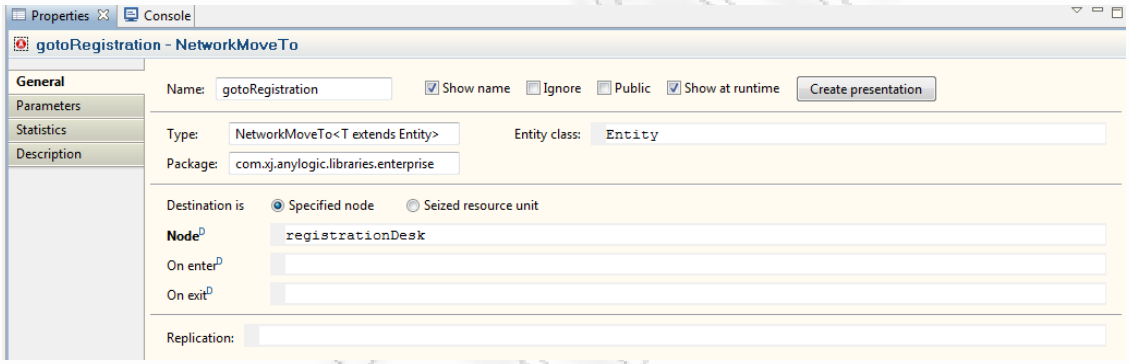
Εικόνα 8-7(β): Παράμετροι της οντότητας arrive τύπου Source.



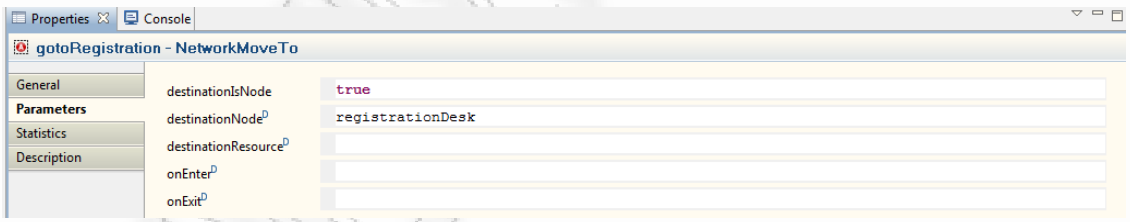
Εικόνα 8-8(α): Γενικές ιδιότητες της οντότητας networkEnter τύπου NetworkEnter.



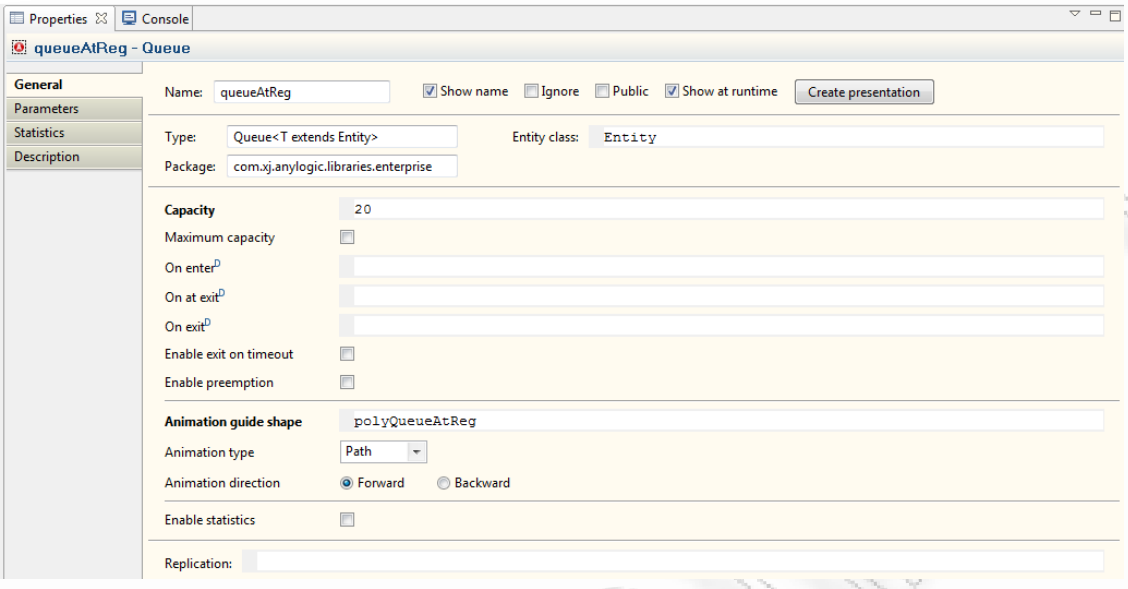
Εικόνα 8-8(β): Παράμετροι της οντότητας networkEnter τύπου NetworkEnter.



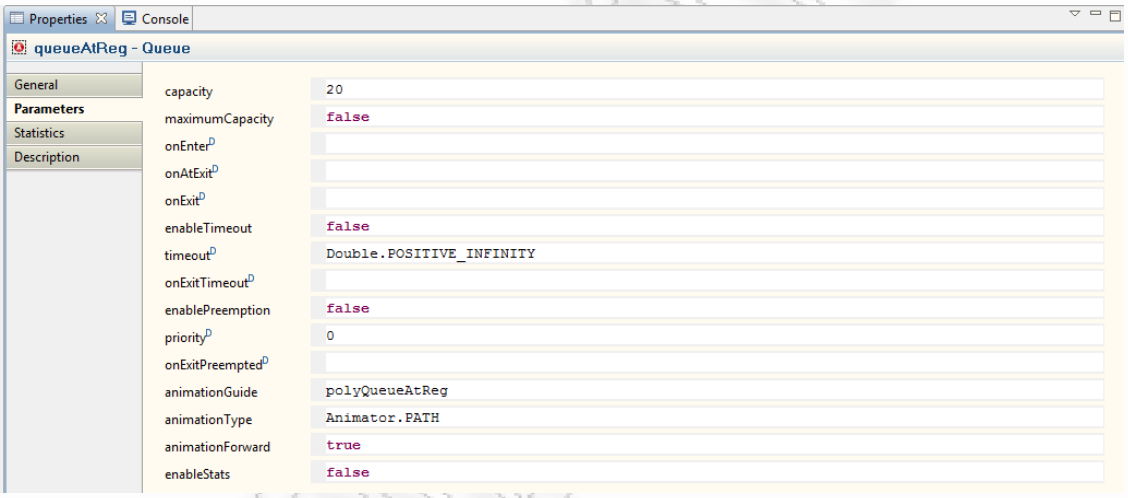
Εικόνα 8-9(α): Γενικές ιδιότητες της οντότητας gotoRegistration τύπου NetworkMoveTo.



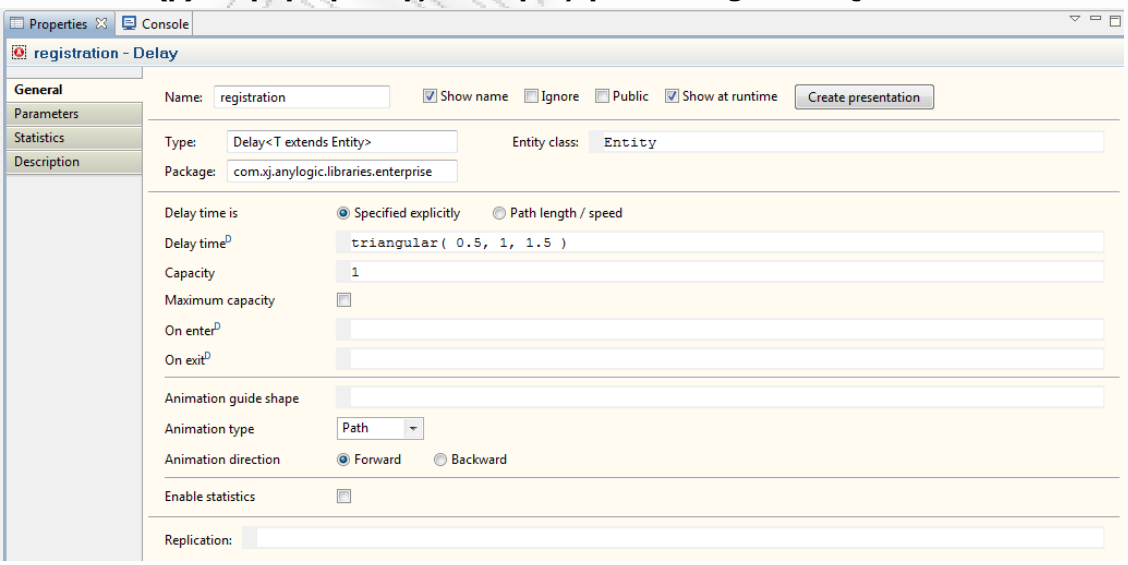
Εικόνα 8-9(β): Παράμετροι της οντότητας gotoRegistration τύπου NetworkMoveTo.



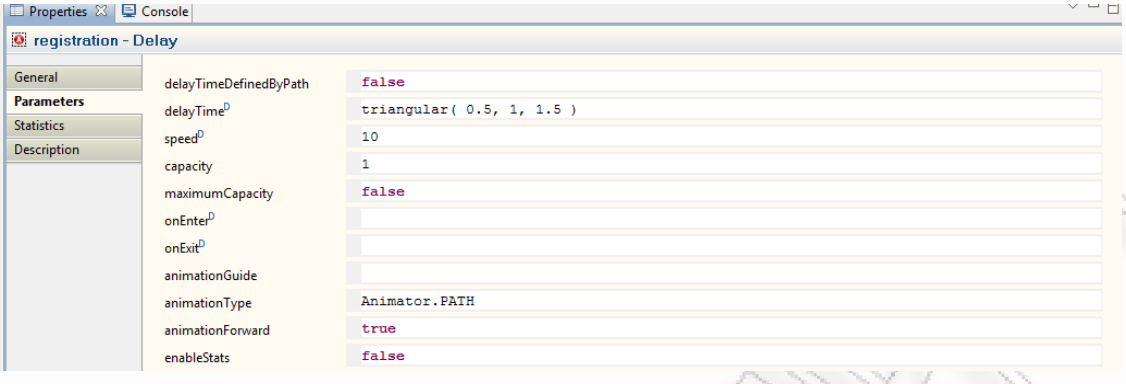
Εικόνα 8-10(α): Γενικές ιδιότητες της οντότητας queueAtReg τύπου Queue.



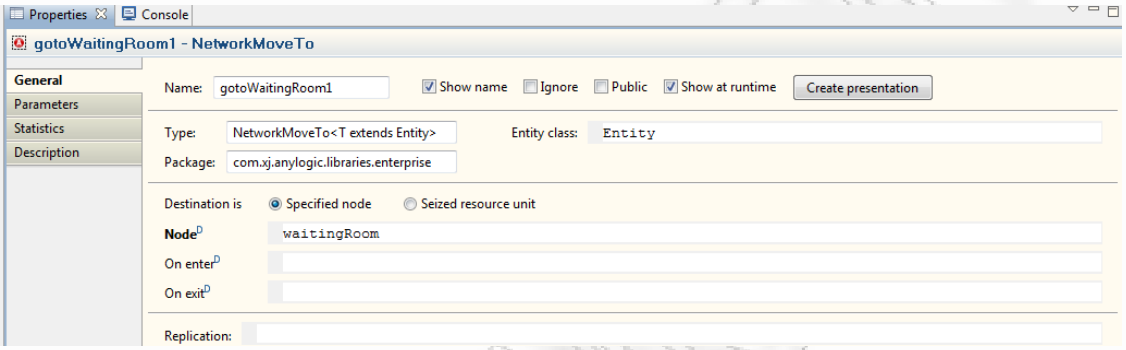
Εικόνα 8-10(β): Παράμετροι της οντότητας queueAtReg τύπου Queue.



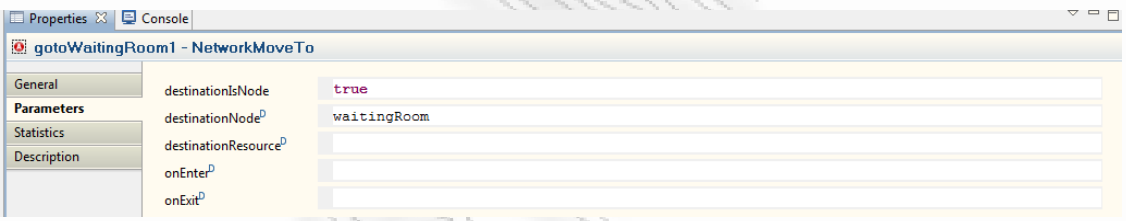
Εικόνα 8-11(α): Γενικές ιδιότητες της οντότητας registration τύπου Delay.



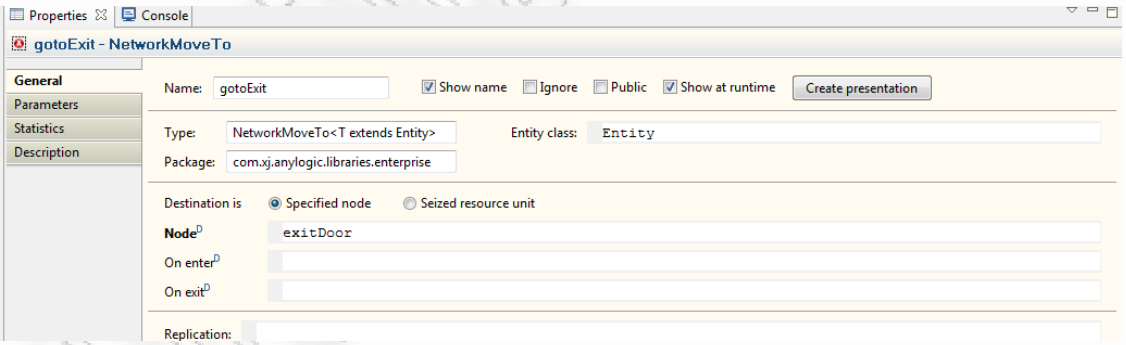
Εικόνα 8-11(β): Παράμετροι της οντότητας registration τύπου Delay.



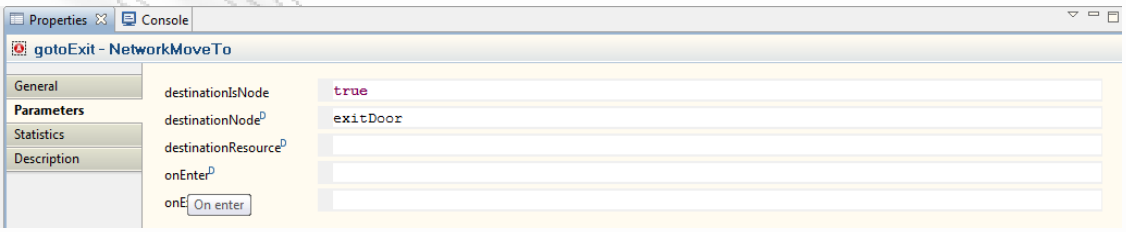
Εικόνα 8-12(α): Γενικές ιδιότητες της οντότητας gotoWaitingRoom1 τύπου NetworkMoveTo.



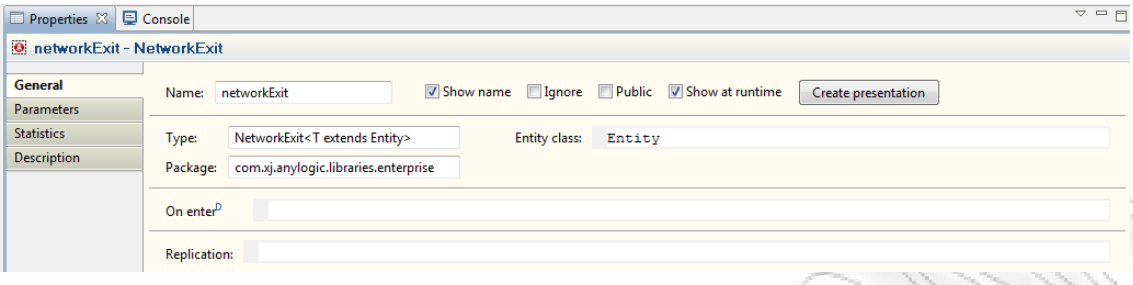
Εικόνα 8-12(β): Παράμετροι της οντότητας gotoWaitingRoom1 τύπου NetworkMoveTo.



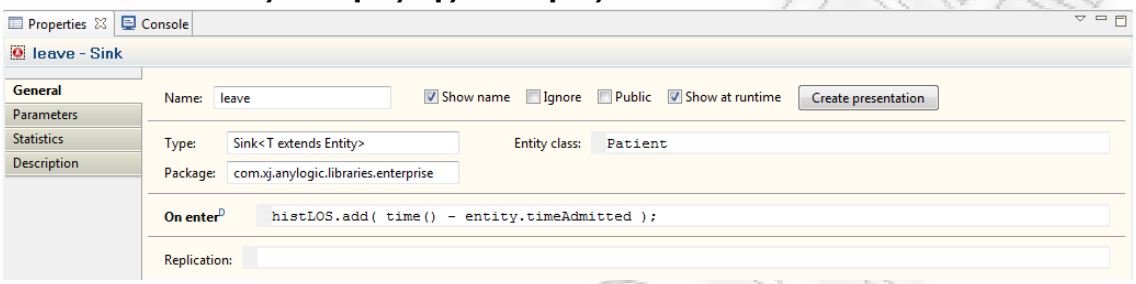
Εικόνα 8-13(α): Γενικές ιδιότητες της οντότητας gotoExit τύπου NetworkMoveTo.



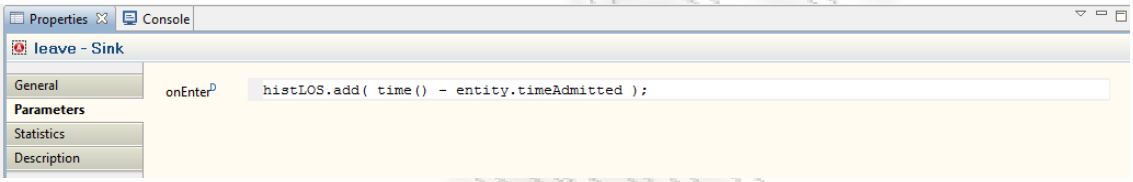
Εικόνα 8-13(β): Παράμετροι της οντότητας gotoExit τύπου NetworkMoveTo.



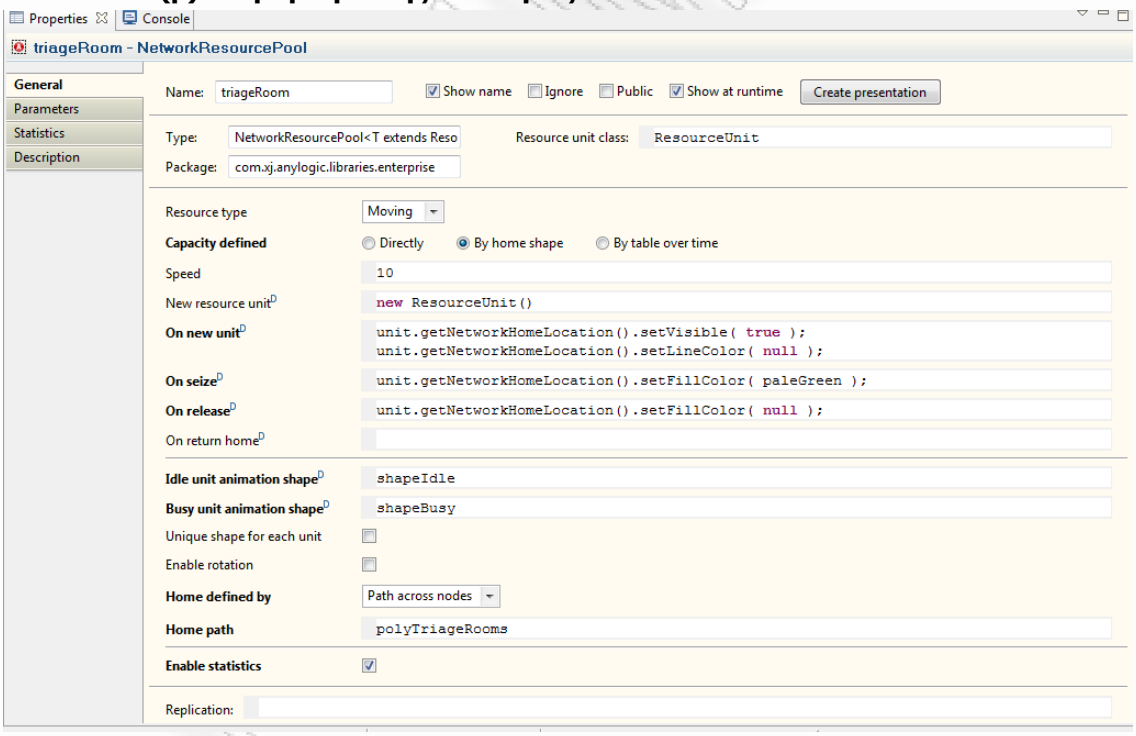
Εικόνα 8-14: Γενικές ιδιότητες της οντότητας networkExit τύπου NetworkExit.



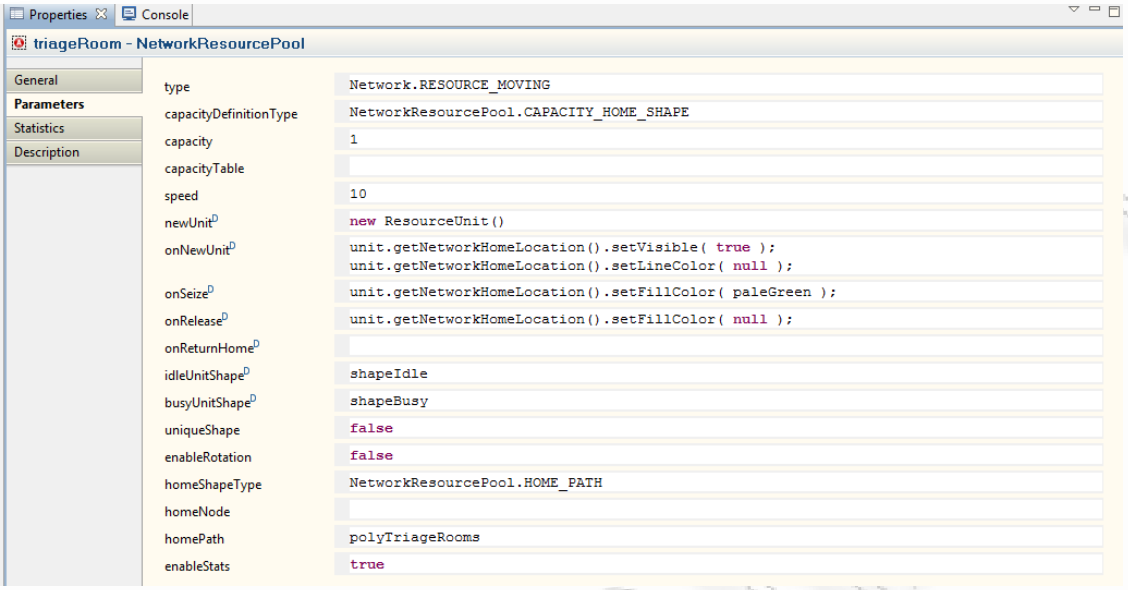
Εικόνα 8-15(α): Γενικές ιδιότητες της οντότητας leave τύπου Sink.



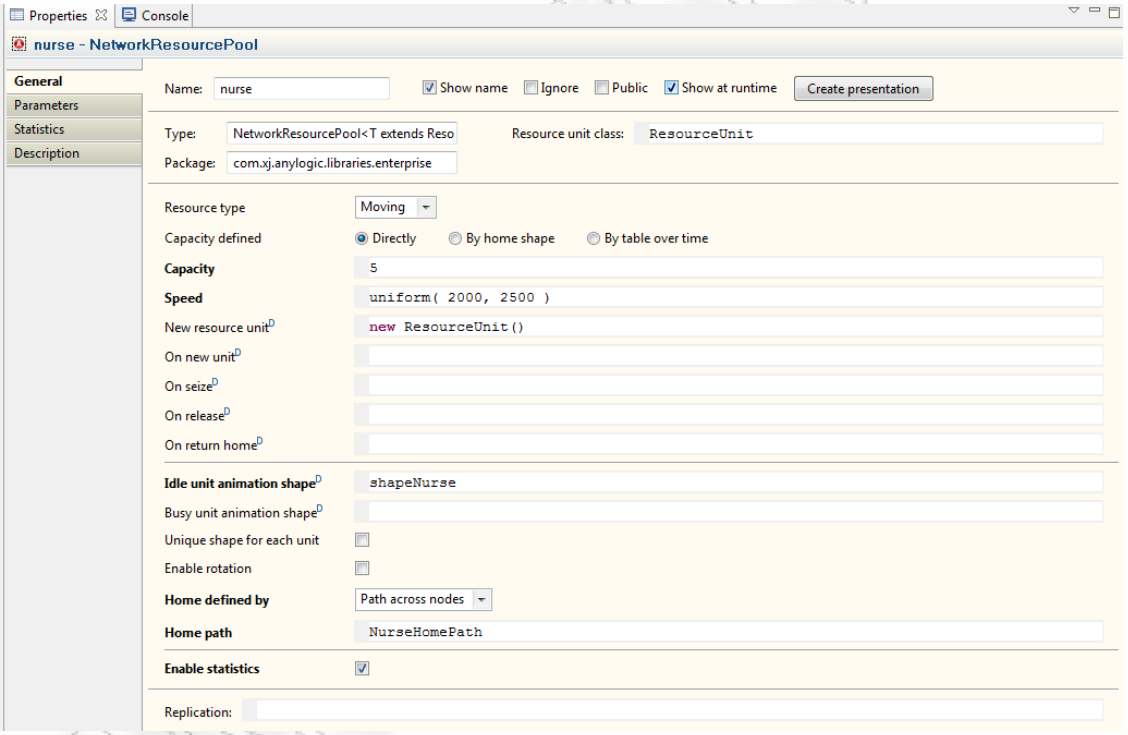
Εικόνα 8-15(β): Παράμετροι της οντότητας leave τύπου Sink.



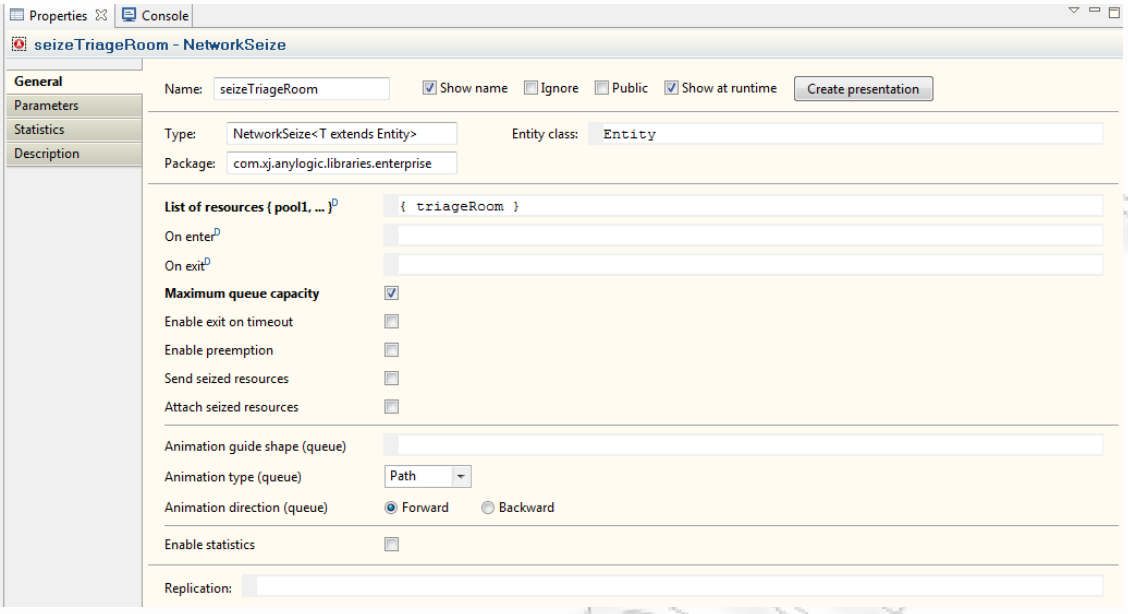
Εικόνα 8-16(α): Γενικές ιδιότητες της οντότητας triageRoom τύπου NetworkResourcePool.



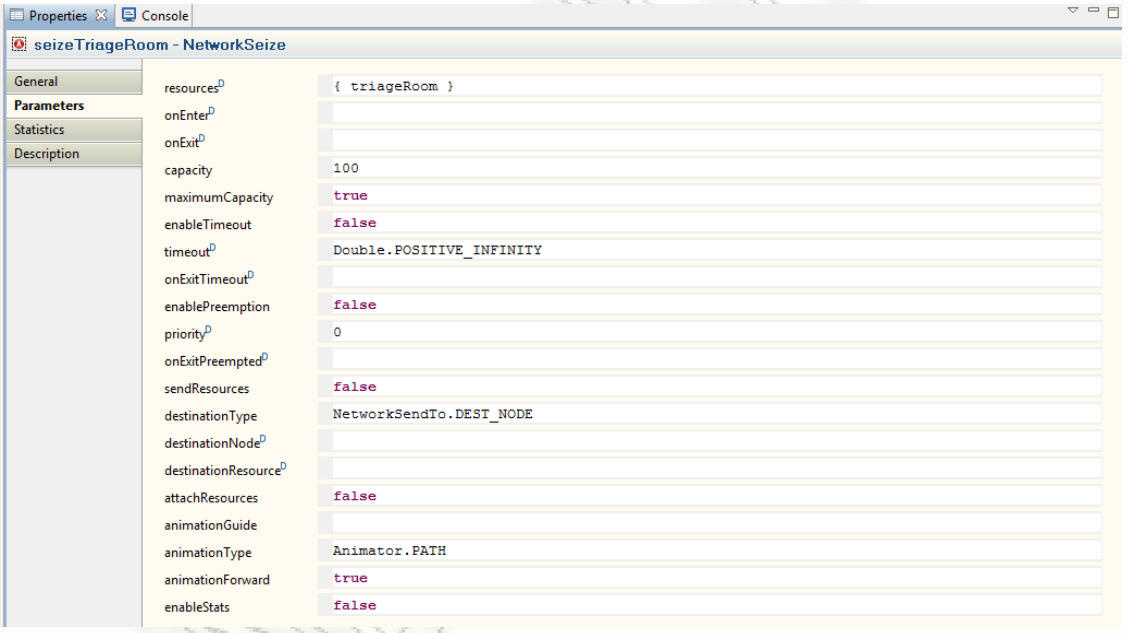
Εικόνα 8-16(β): Παράμετροι της οντότητας triageRoom τύπου NetworkResourcePool.



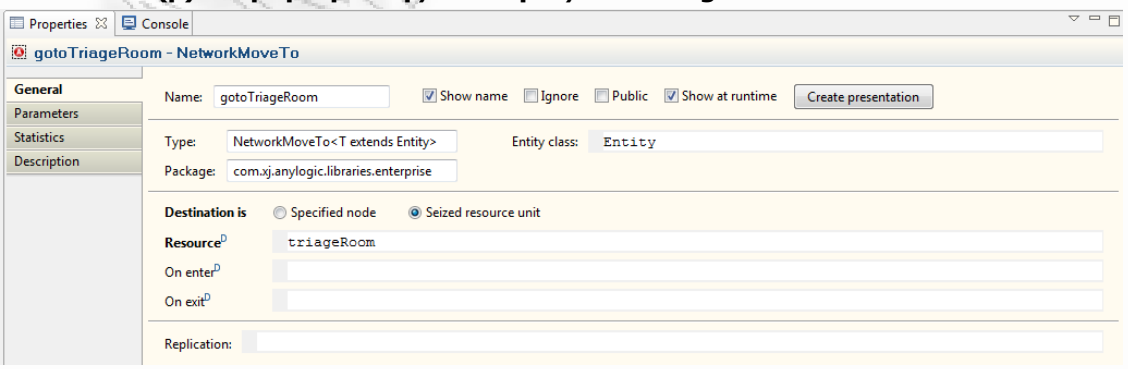
Εικόνα 8-17: Γενικές ιδιότητες της οντότητας nurse τύπου NetworkResourcePool.



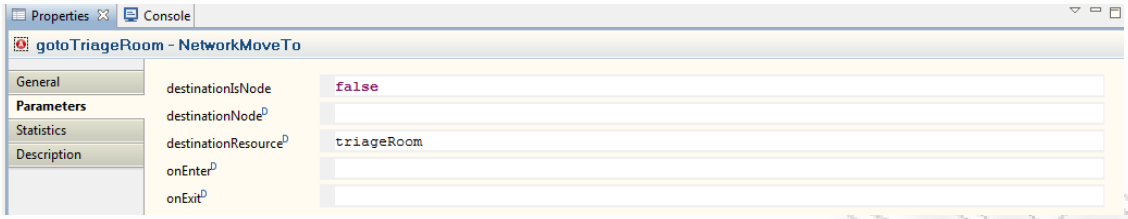
Εικόνα 8-18(α): Γενικές ιδιότητες της οντότητας seizeTriageRoom τύπου NetworkSeize.



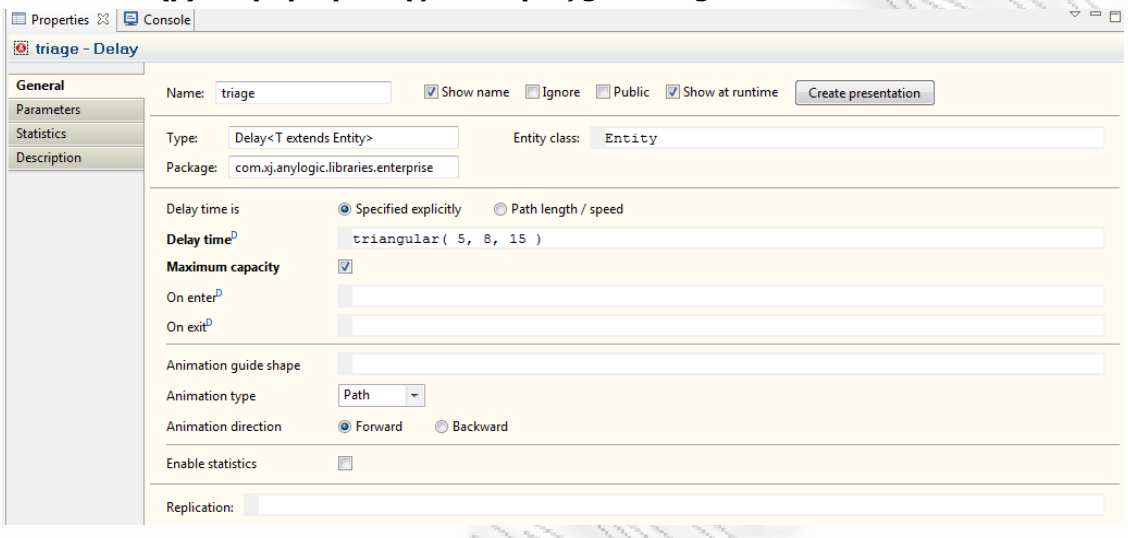
Εικόνα 8-18(β): Παράμετροι της οντότητας seizeTriageRoom τύπου NetworkSeize.



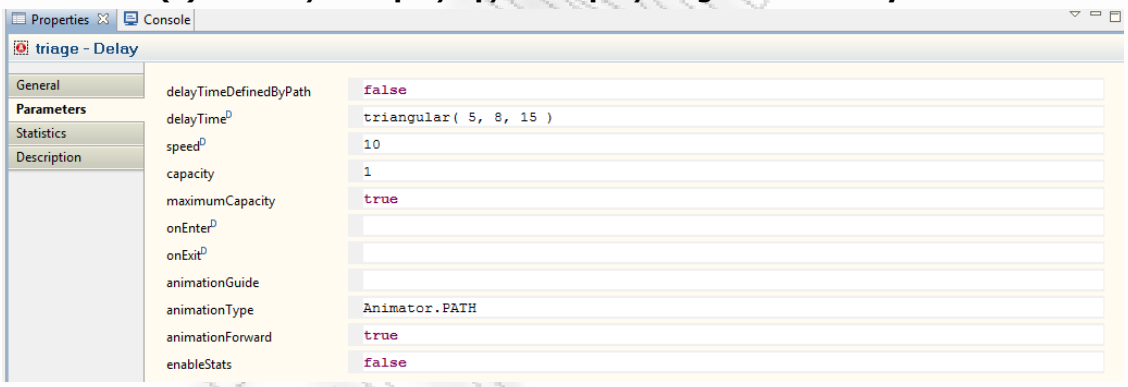
Εικόνα 8-19(α): Γενικές ιδιότητες της οντότητας gotoTriageRoom τύπου NetworkMoveTo.



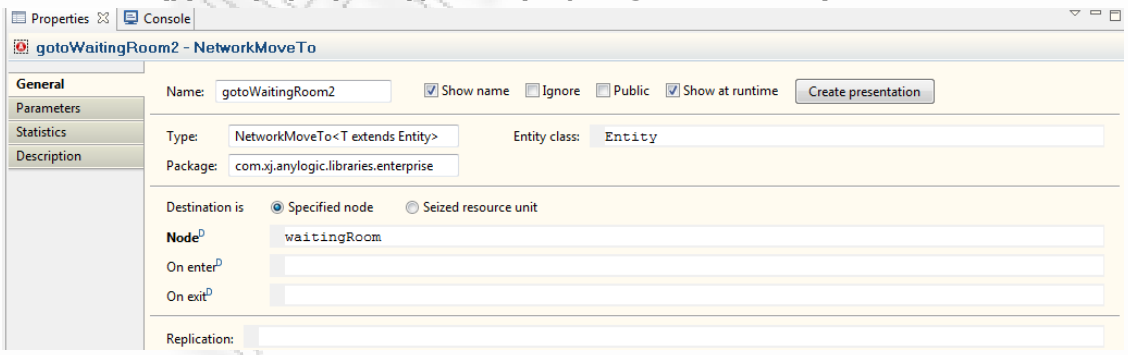
Εικόνα 8-19(β): Παράμετροι της οντότητας gotoTriageRoom τύπου NetworkMoveTo.



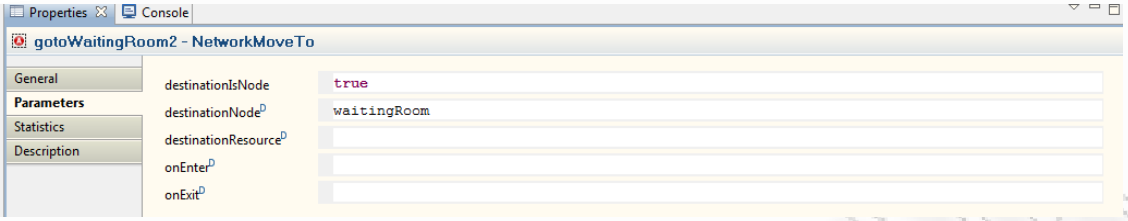
Εικόνα 8-20(α): Γενικές ιδιότητες της οντότητας triage τύπου Delay.



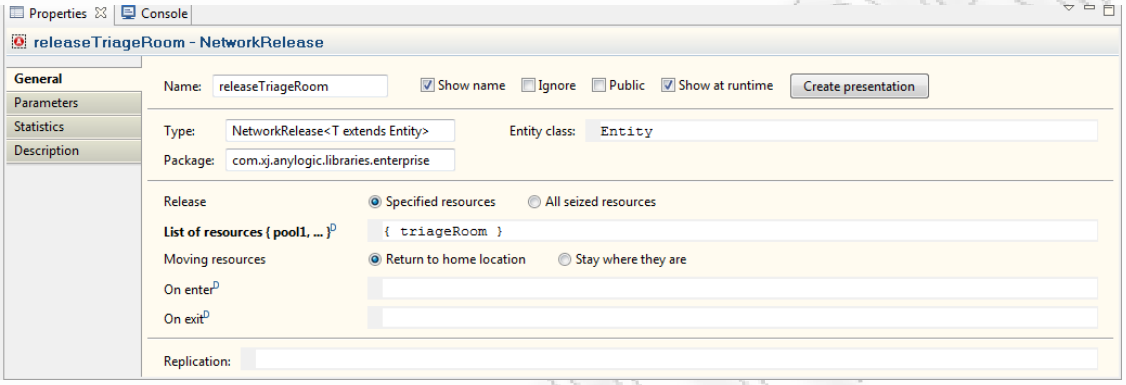
Εικόνα 8-20(β): Παράμετροι της οντότητας triage τύπου Delay.



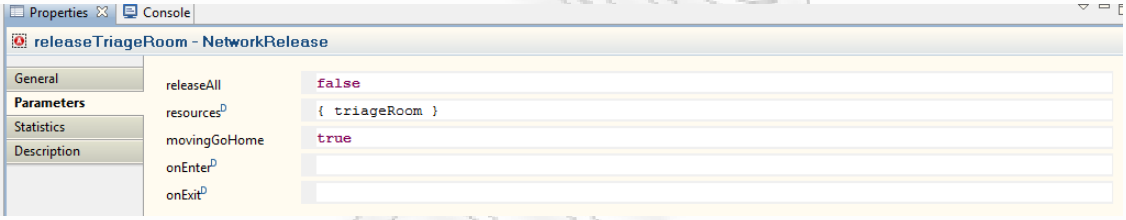
Εικόνα 8-21(α): Γενικές ιδιότητες της οντότητας gotoWaitingRoom2 τύπου NetworkMoveTo.



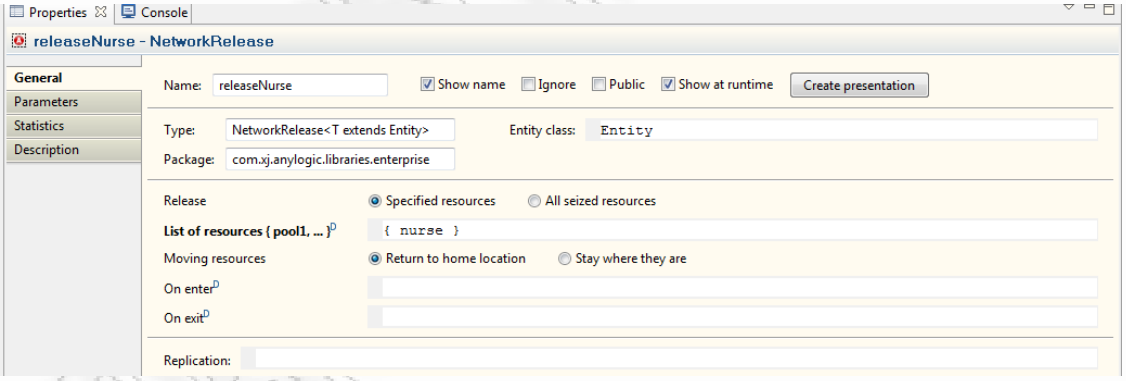
Εικόνα 8-21(β): Παράμετροι της οντότητας gotoWaitingRoom2 τύπου NetworkMoveTo.



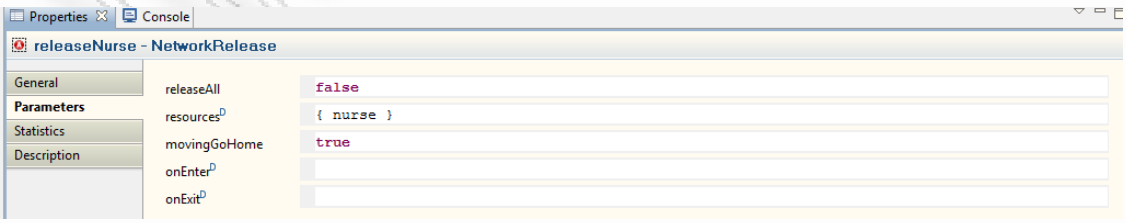
Εικόνα 8-22(α): Γενικές ιδιότητες της οντότητας releaseTriageRoom τύπου NetworkRelease.



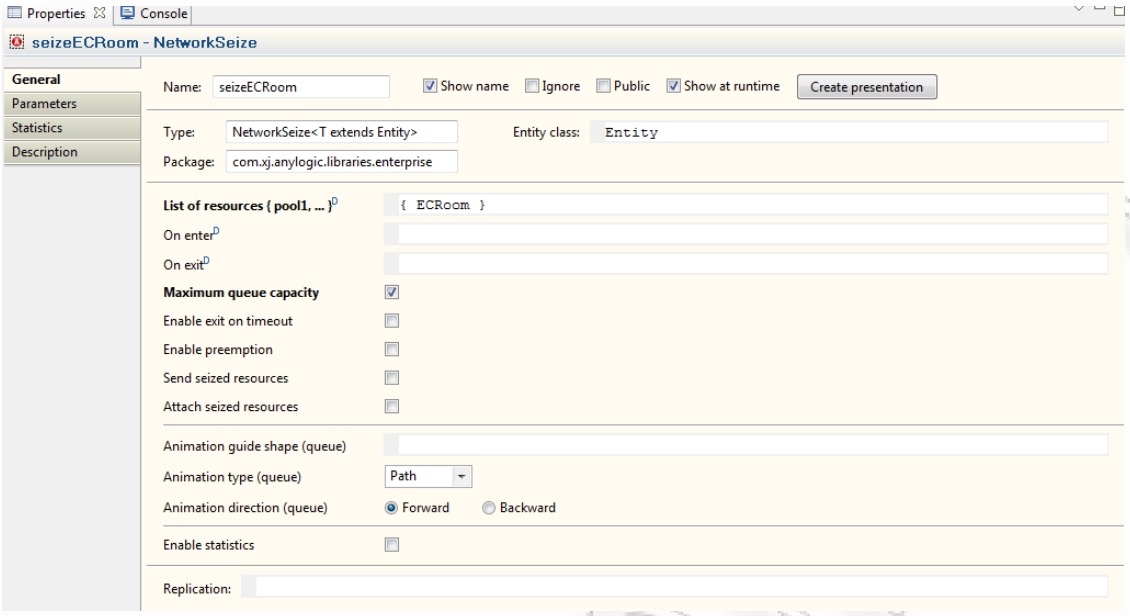
Εικόνα 8-22(β): Παράμετροι της οντότητας releaseTriageRoom τύπου NetworkRelease.



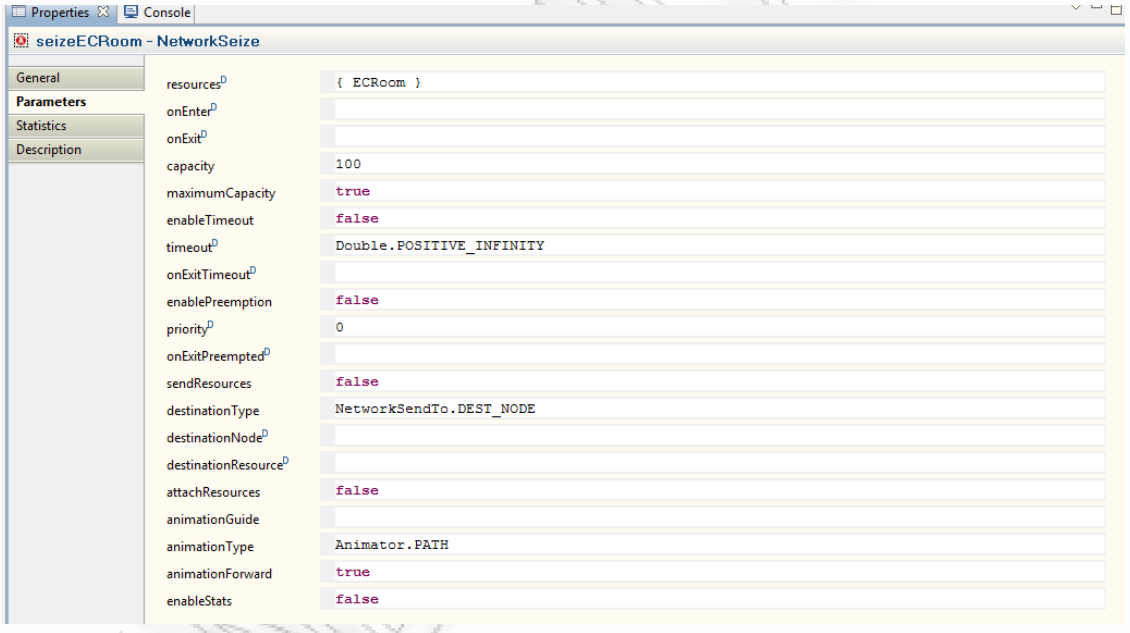
Εικόνα 8-23(α): Γενικές ιδιότητες της οντότητας releaseNurse τύπου NetworkRelease.



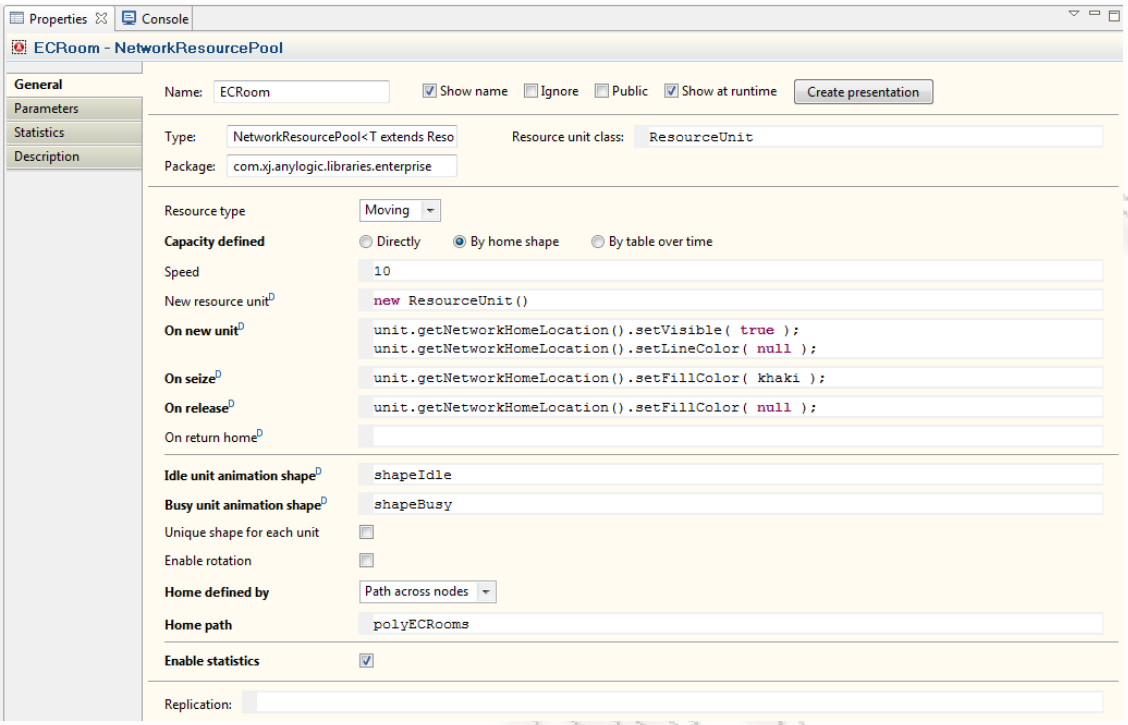
Εικόνα 8-23(β): Παράμετροι της οντότητας releaseNurse τύπου NetworkRelease.



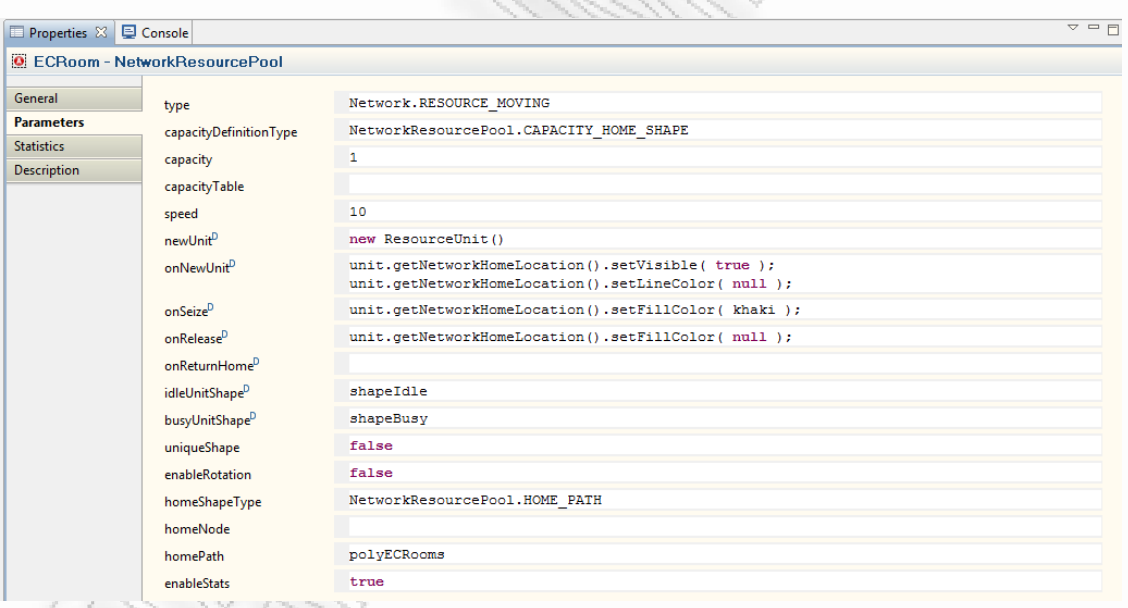
Εικόνα 8-24(α): Γενικές ιδιότητες της οντότητας seizeECRoom τύπου NetworkSeize.



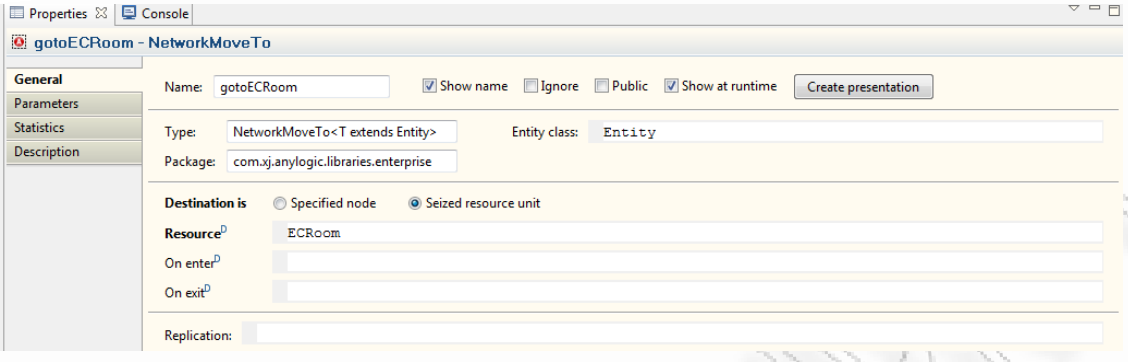
Εικόνα 8-24(β): Παράμετροι της οντότητας seizeECRoom τύπου NetworkSeize.



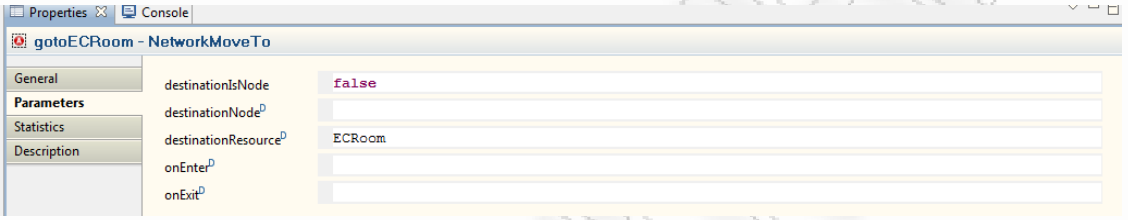
Εικόνα 8-25(α): Γενικές ιδιότητες της οντότητας ECRoom τύπου NetworkResourcePool.



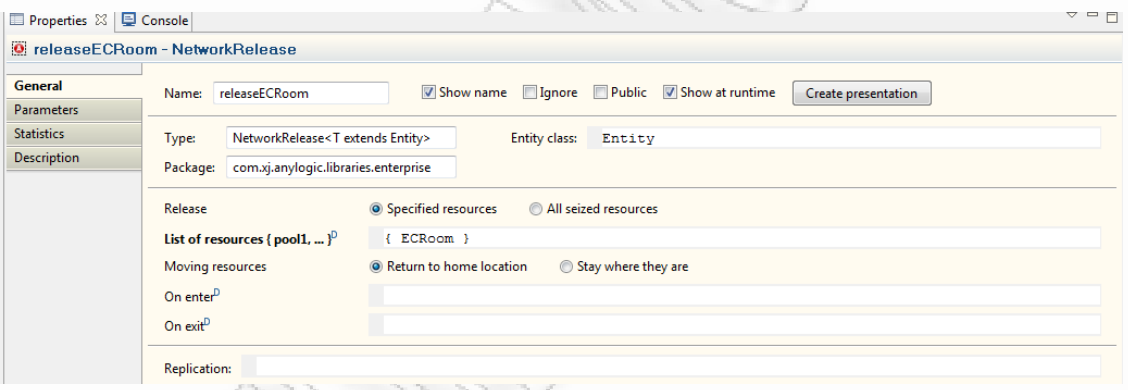
Εικόνα 8-25(β): Παράμετροι της οντότητας ECRoom τύπου NetworkResourcePool.



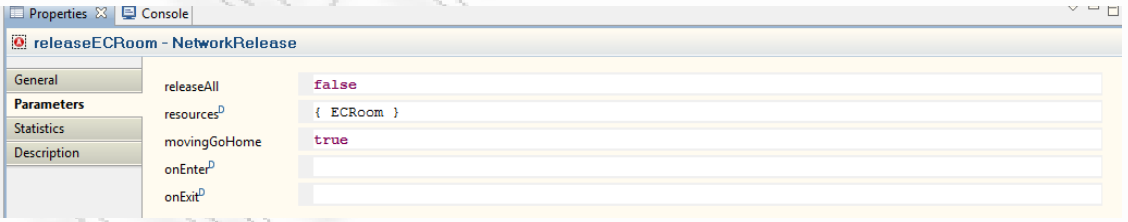
Εικόνα 8-26(α): Γενικές ιδιότητες της οντότητας gotoECRoom τύπου NetworkMoveTo.



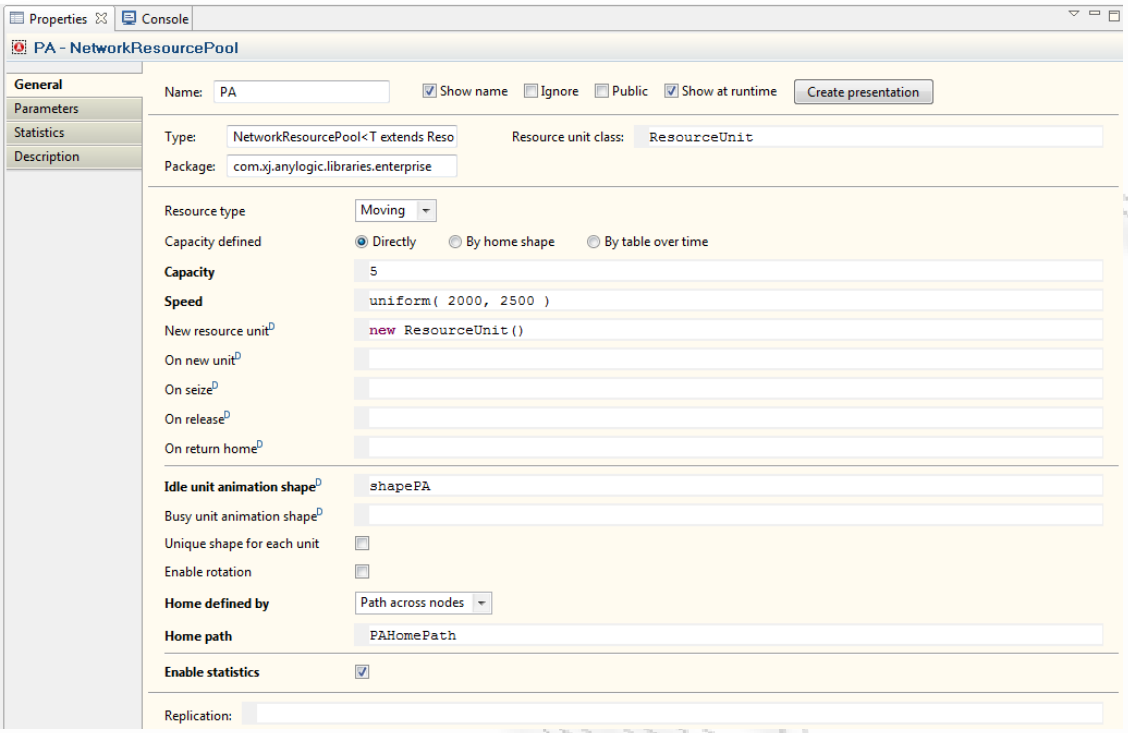
Εικόνα 8-26(β): Παράμετροι της οντότητας gotoECRoom τύπου NetworkMoveTo.



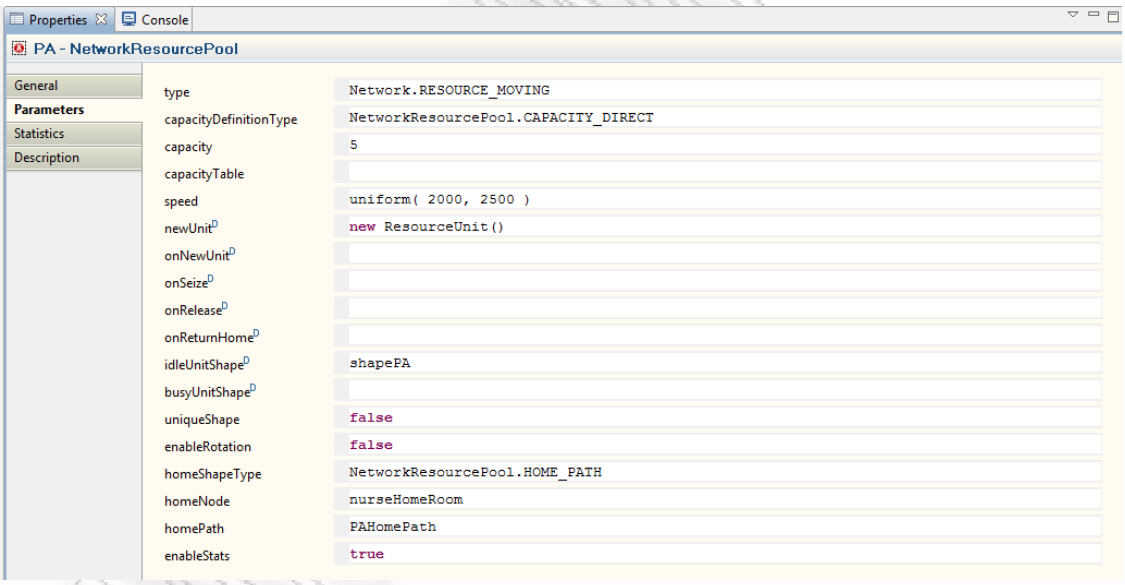
Εικόνα 8-27(α): Γενικές ιδιότητες της οντότητας releaseECRoom τύπου NetworkRelease.



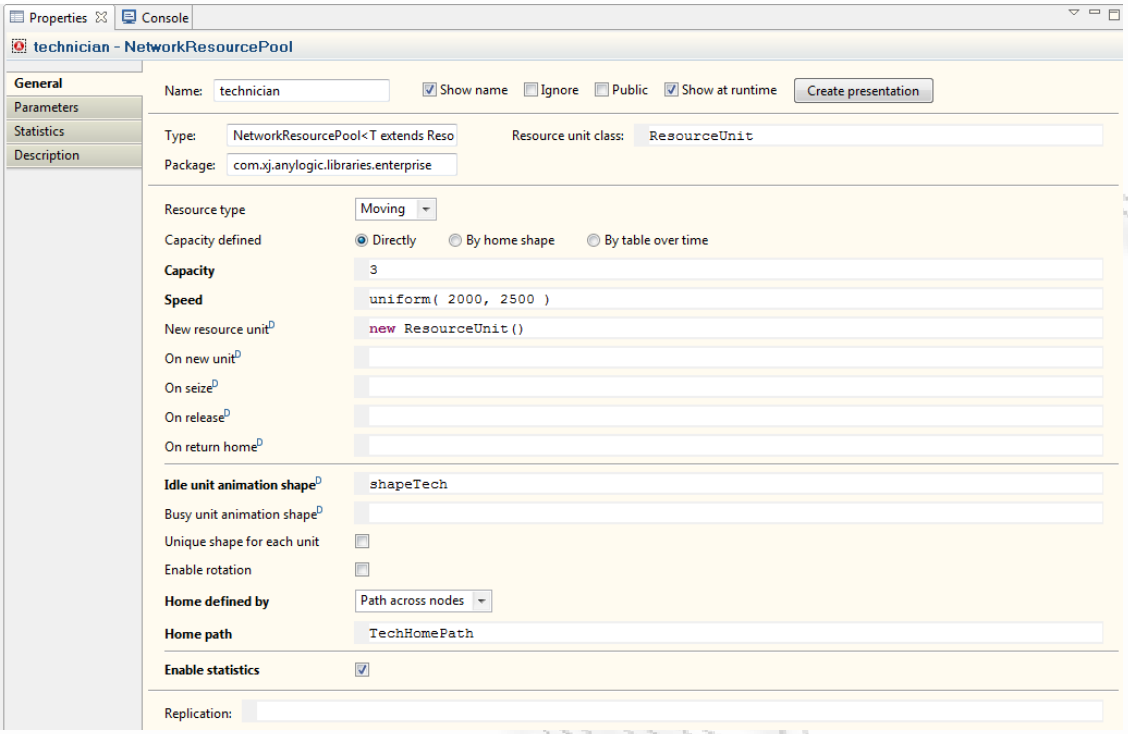
Εικόνα 8-27(β): Παράμετροι της οντότητας releaseECRoom τύπου NetworkRelease.



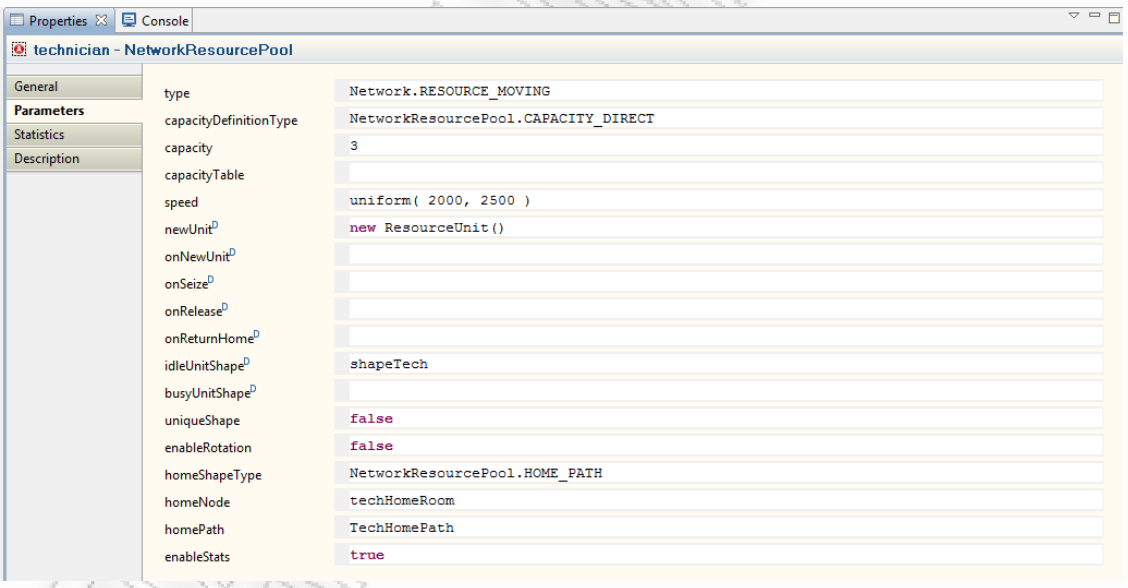
Εικόνα 8-28(α): Γενικές ιδιότητες της οντότητας PA τύπου NetworkResourcePool.



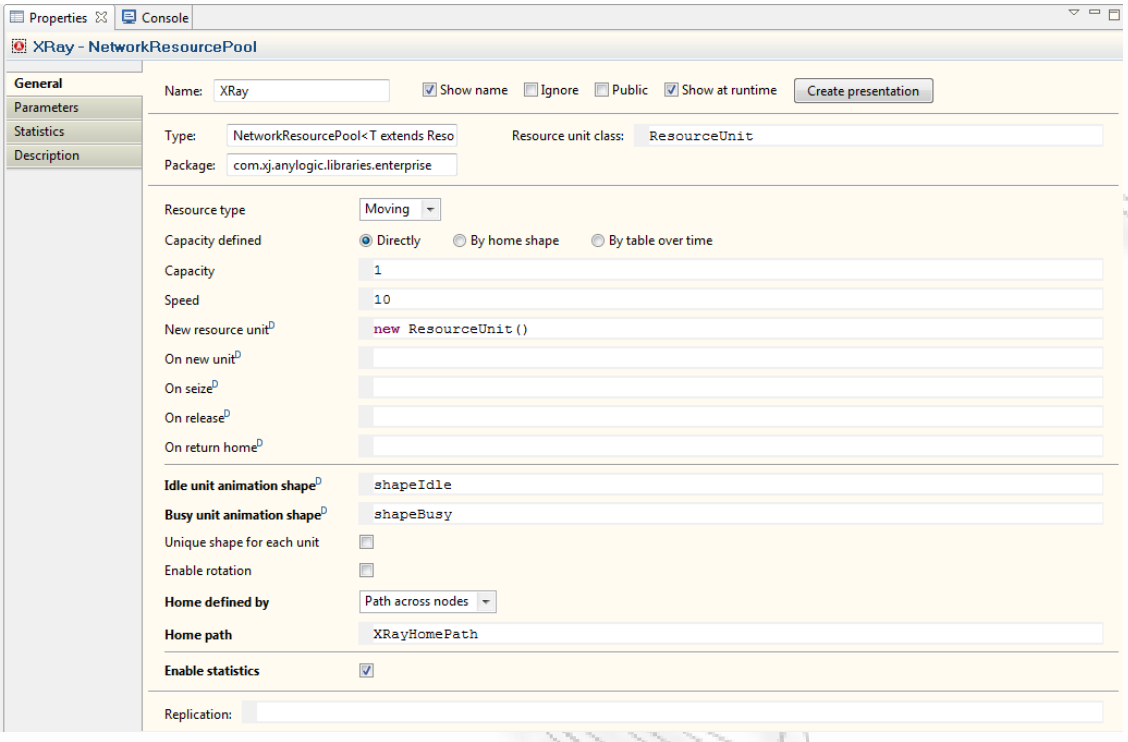
Εικόνα 8-28(β): Παράμετροι της οντότητας PA τύπου NetworkResourcePool.



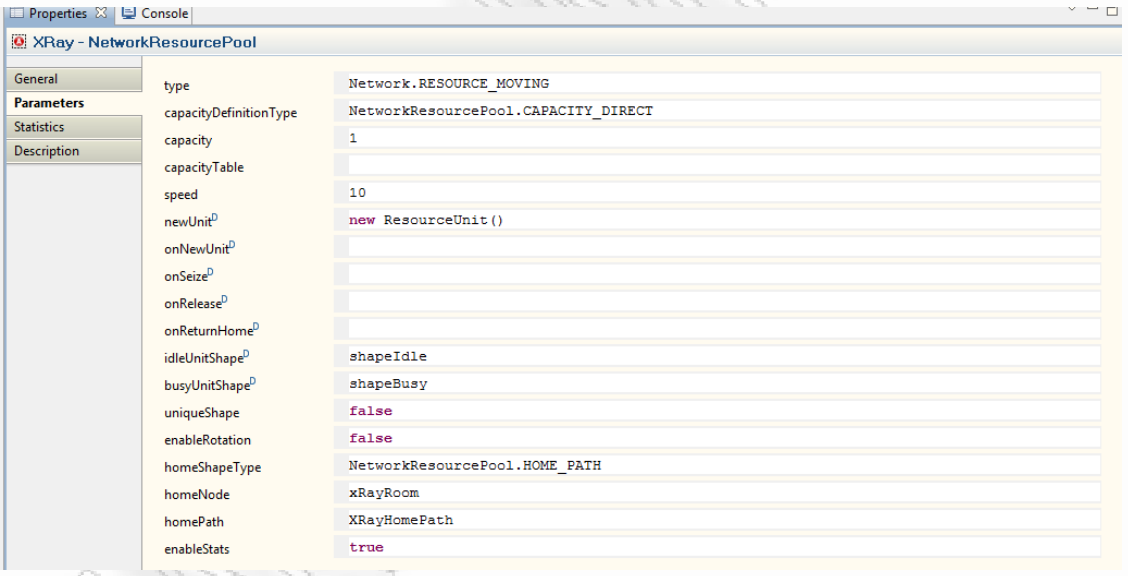
Εικόνα 8-29(α): Γενικές ιδιότητες της οντότητας technician τύπου NetworkResourcePool.



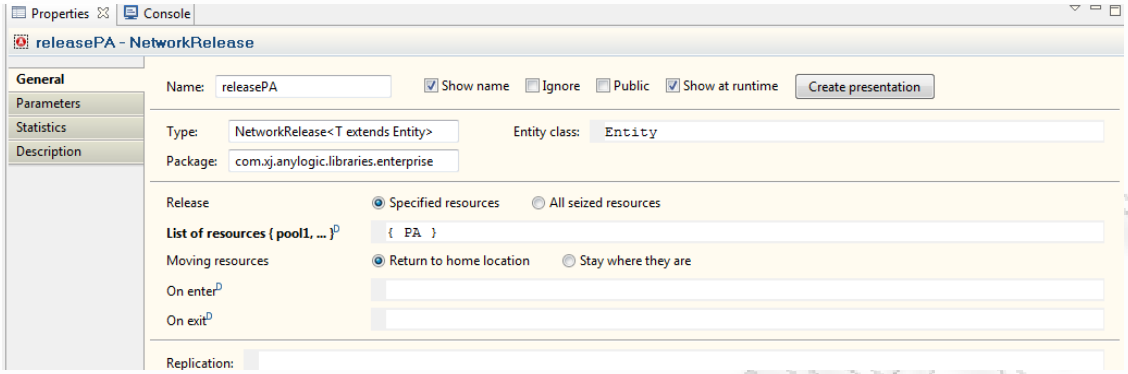
Εικόνα 8-29(β): Παράμετροι της οντότητας technician τύπου NetworkResourcePool.



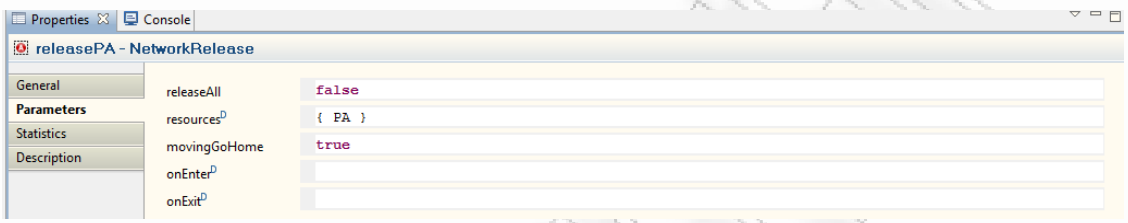
Εικόνα 8-30(α): Γενικές ιδιότητες της οντότητας XRay τύπου NetworkResourcePool.



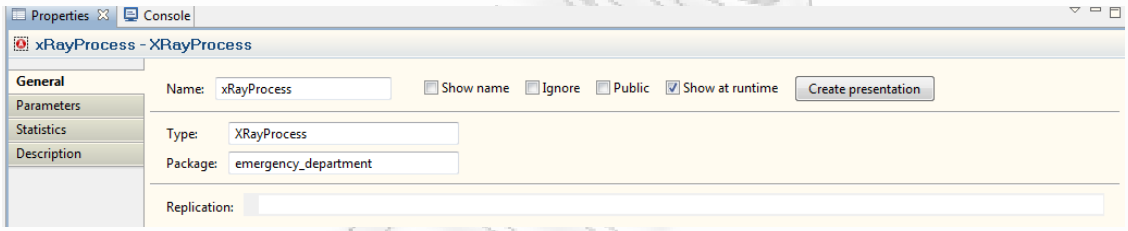
Εικόνα 8-30(β): Παράμετροι της οντότητας XRay τύπου NetworkResourcePool.



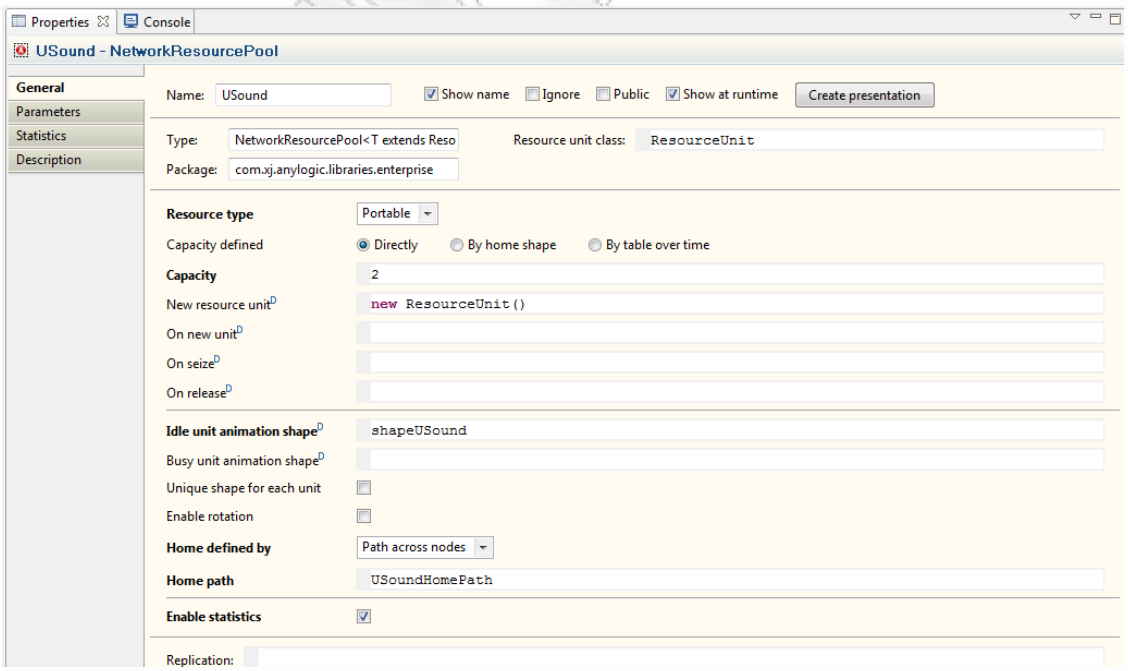
Εικόνα 8-31(α): Γενικές ιδιότητες της οντότητας releasePA τύπου NetworkRelease.



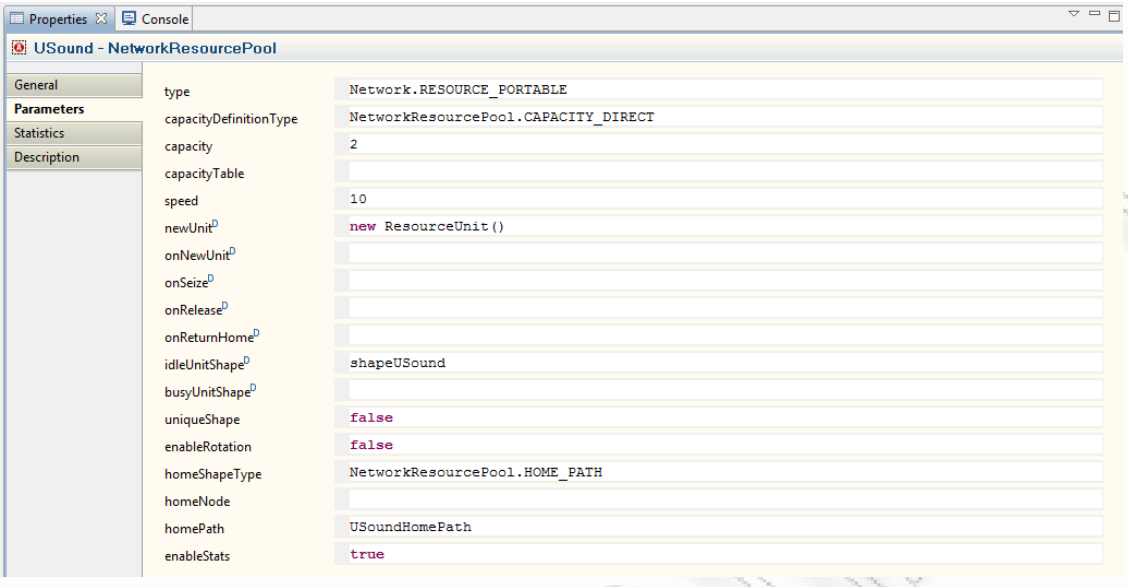
Εικόνα 8-31(β): Παράμετροι της οντότητας releasePA τύπου NetworkRelease.



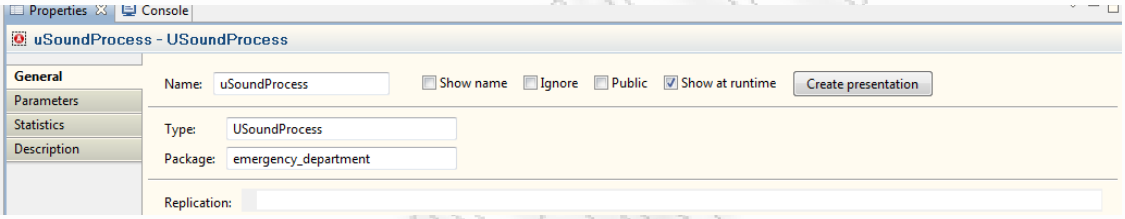
Εικόνα 8-32: Γενικές ιδιότητες της οντότητας xRayProcess τύπου XRayProcess.



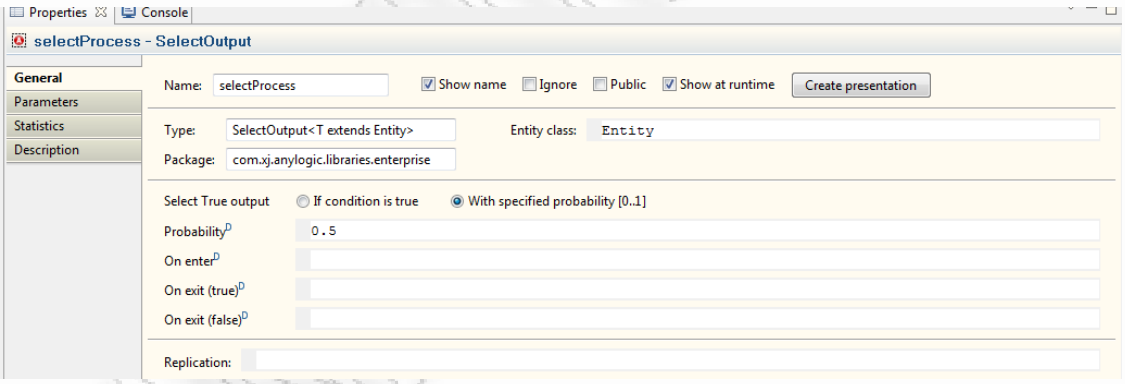
Εικόνα 8-33(α): Γενικές ιδιότητες της οντότητας USound τύπου NetworkResourcePool.



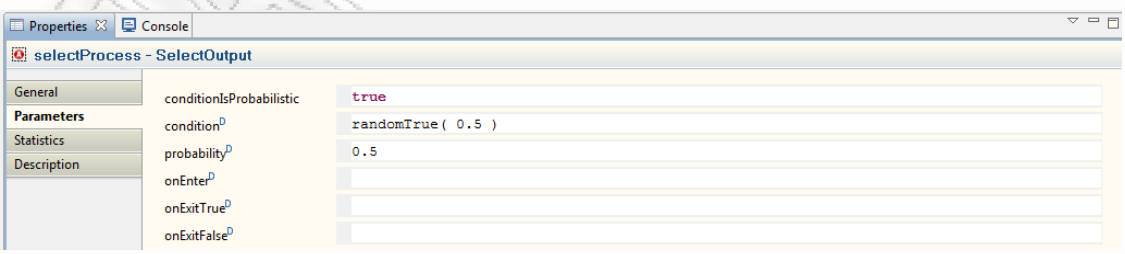
Εικόνα 8-33(β): Παράμετροι της οντότητας USound τύπου NetworkResourcePool.



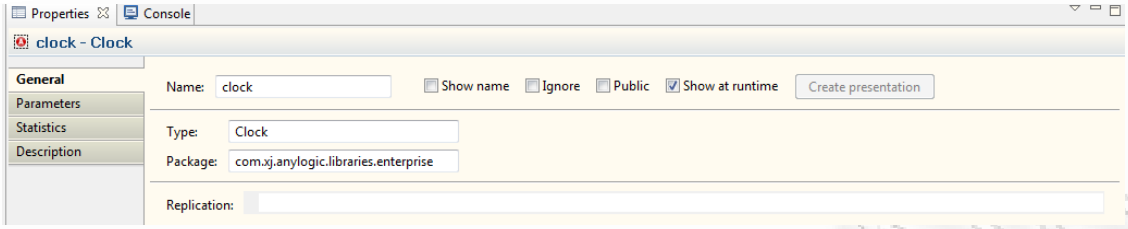
Εικόνα 8-34: Γενικές ιδιότητες της οντότητας uSoundProcess τύπου USoundProcess.



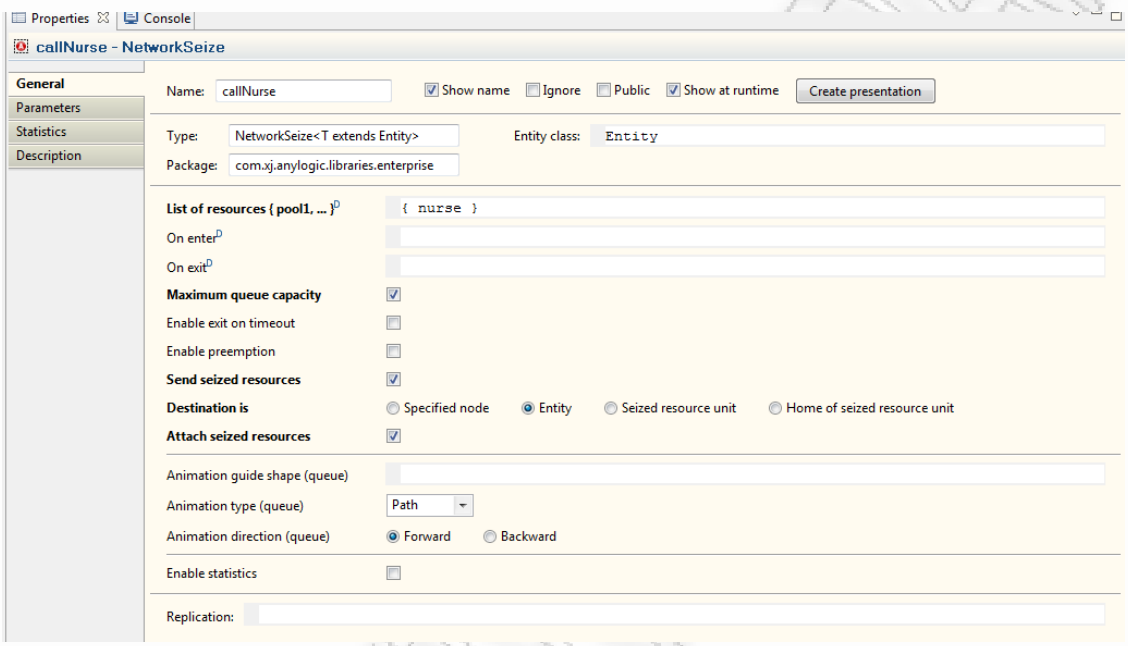
Εικόνα 8-35(α): Γενικές ιδιότητες της οντότητας selectProcess τύπου SelectOutput.



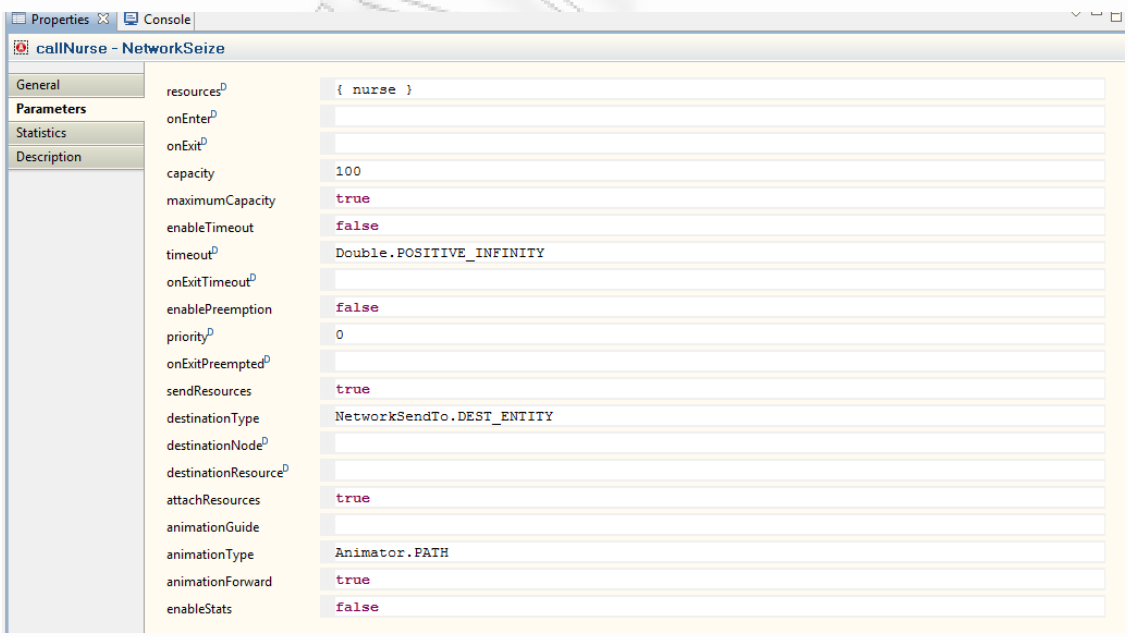
Εικόνα 8-35(β): Παράμετροι της οντότητας selectProcess τύπου SelectOutput.



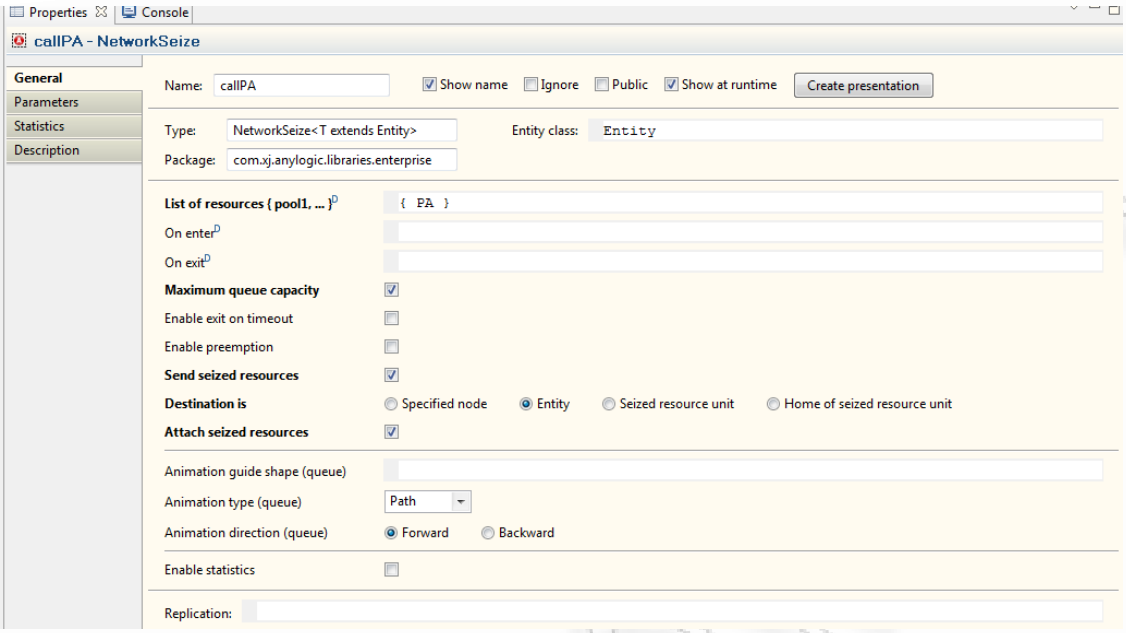
Εικόνα 8-36: Γενικές ιδιότητες της οντότητας clock τύπου Clock.



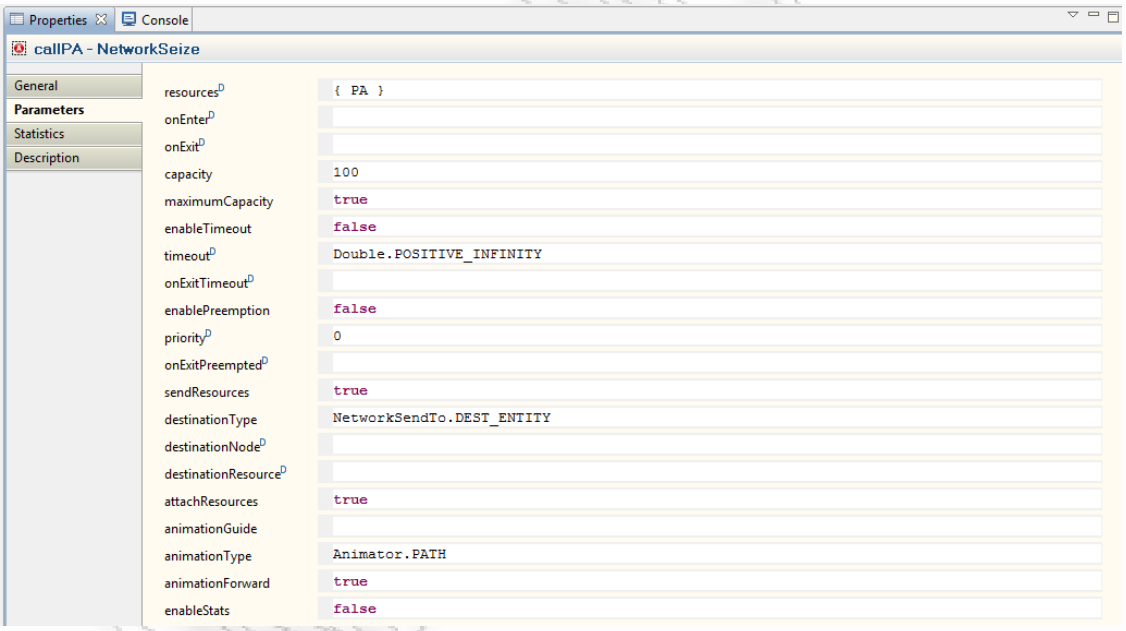
Εικόνα 8-37(α): Γενικές ιδιότητες της οντότητας callNurse τύπου NetworkSeize.



Εικόνα 8-37(β): Παράμετροι της οντότητας callNurse τύπου NetworkSeize.

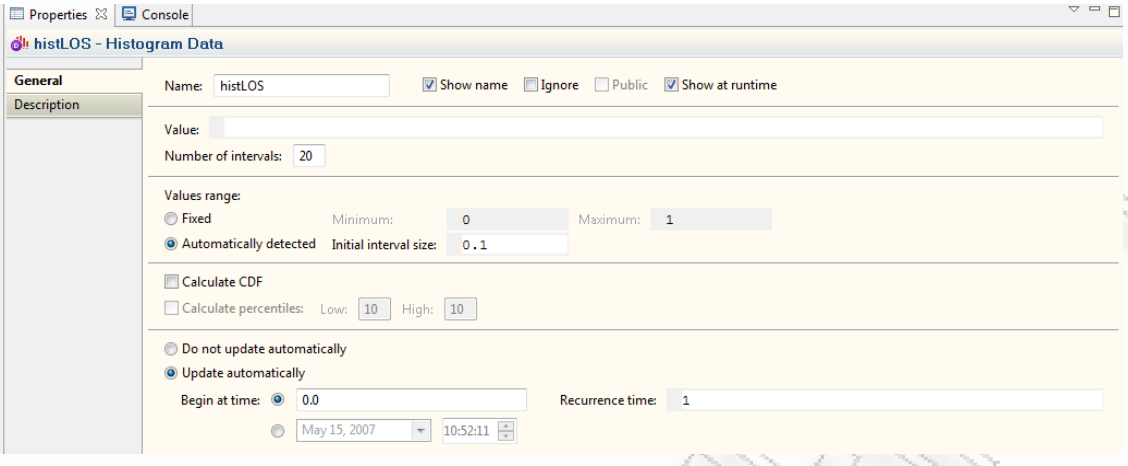


Εικόνα 8-38(α): Γενικές ιδιότητες της οντότητας callPA τύπου NetworkSeize.

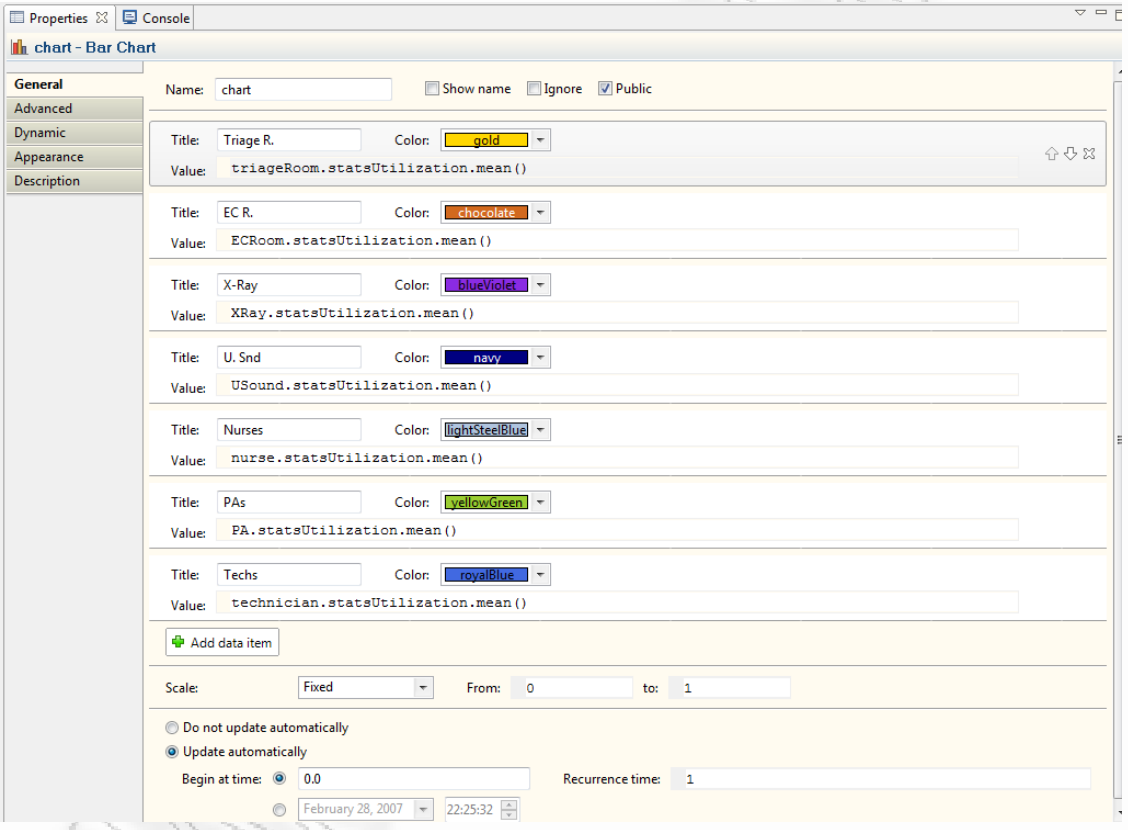


Εικόνα 8-38(β): Παράμετροι της οντότητας callPA τύπου NetworkSeize.

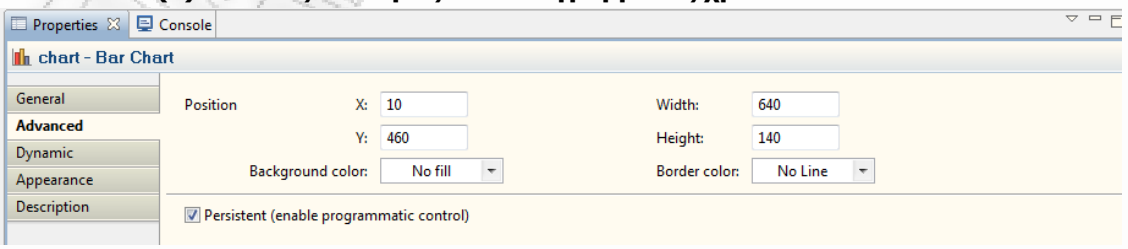
Στα επόμενα παρουσιάζονται τα στοιχεία που συμπληρώνουν την υλοποίηση της κλάσης Main. Στην εικόνα 8-39 δίνονται οι γενικές ιδιότητες του συνόλου δεδομένων, histLOS, για την στατιστική ανάλυση που παρουσιάζει τα αποτελέσματα με τη μορφή ιστογράμματος ενώ στις εικόνες από την 8-40(α) έως και την εικόνα 8-43(β) παρατίθενται τα στοιχεία παρουσίασης της κλάσης Main.



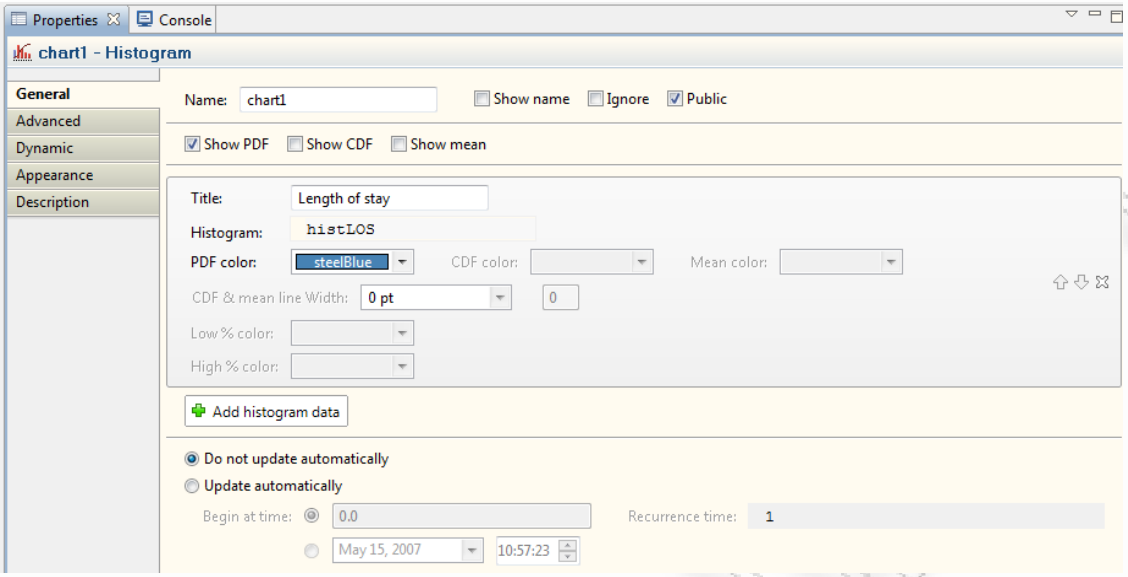
Εικόνα 8-39: Γενικές ιδιότητες του συνόλου στοιχείων δεδομένων ιστογράμματος histLOS.



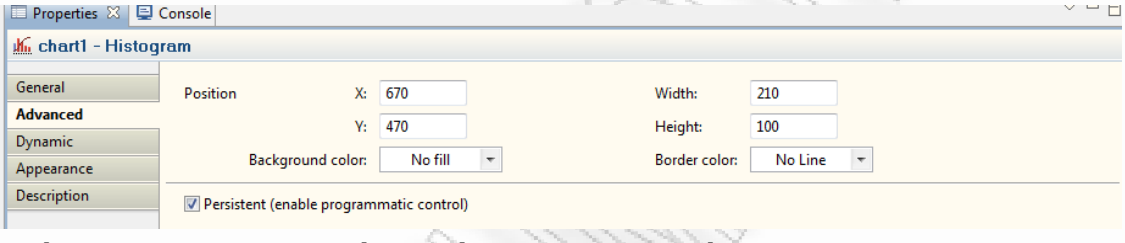
Εικόνα 8-40(α): Γενικές ιδιότητες του διαγράμματος χρόνου chart.



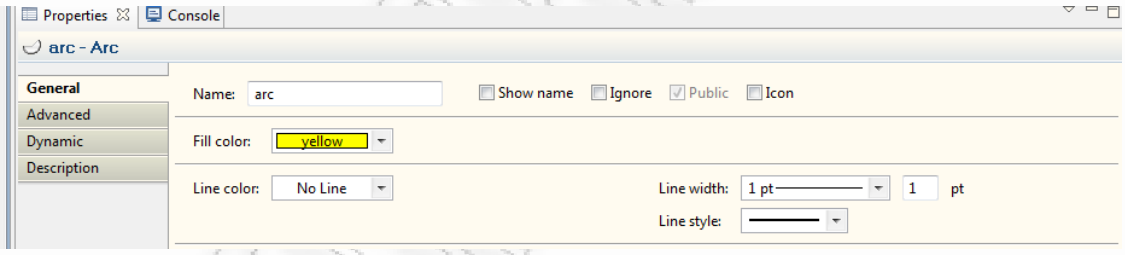
Εικόνα 8-40(β): Προχωρημένες ιδιότητες του διαγράμματος χρόνου chart.



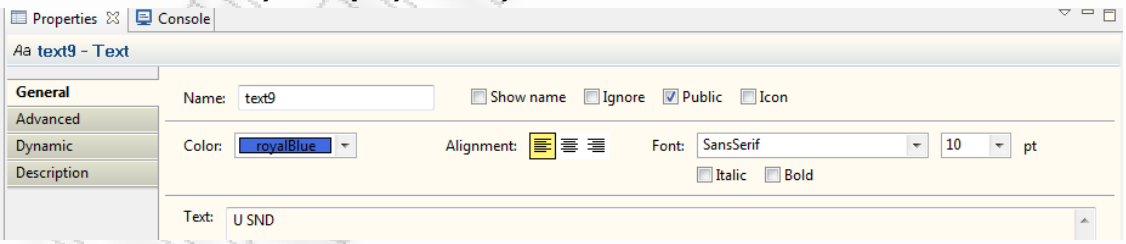
Εικόνα 8-41(α): Γενικές ιδιότητες του ιστογράμματος chart1.



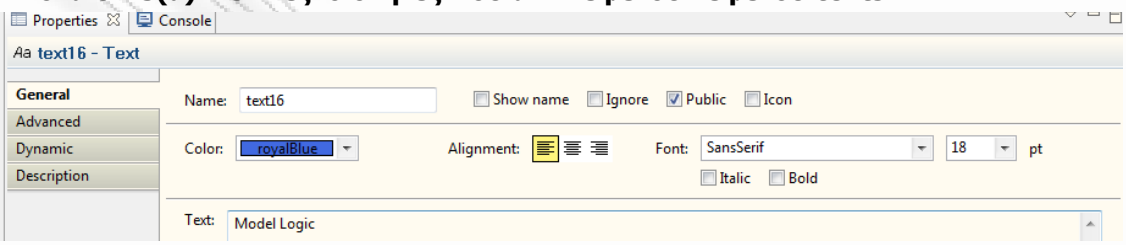
Εικόνα 8-41(β): Προχωρημένες ιδιότητες του ιστογράμματος chart1.



Εικόνα 8-42: Γενικές ιδιότητες του τόξου arc.



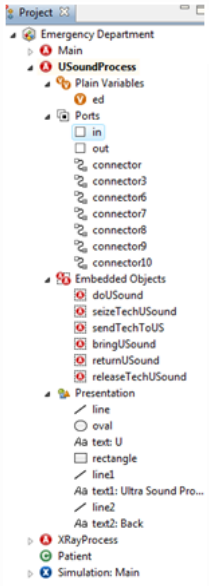
Εικόνα 8-43(α): Γενικές ιδιότητες του αντικειμένου κειμένου text9.



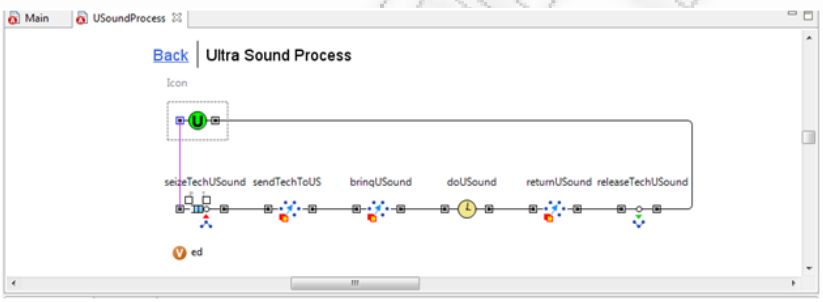
Εικόνα 8-43(β): Γενικές ιδιότητες του αντικειμένου κειμένου text16.

8.2 Η κλάση USoundProcess

Η κλάση USoundProcess μοντελοποιεί την διαδικασία της εξέτασης του ασθενούς με υπέρηχου. Στην εικόνα 8-44(α) παρουσιάζεται η ιεραρχική δομή της κλάσης και τα στοιχεία που την αποτελούν.



Εικόνα 8-44(α): Η ιεραρχική δομή της κλάσης USoundProcess.

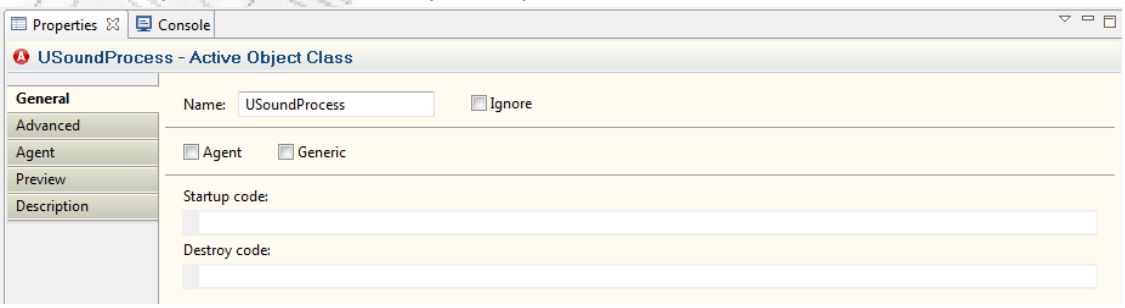


Εικόνα 8-44(β): Το εικονίδιο και το διάγραμμα διαδικασιών της κλάσης USoundProcess.

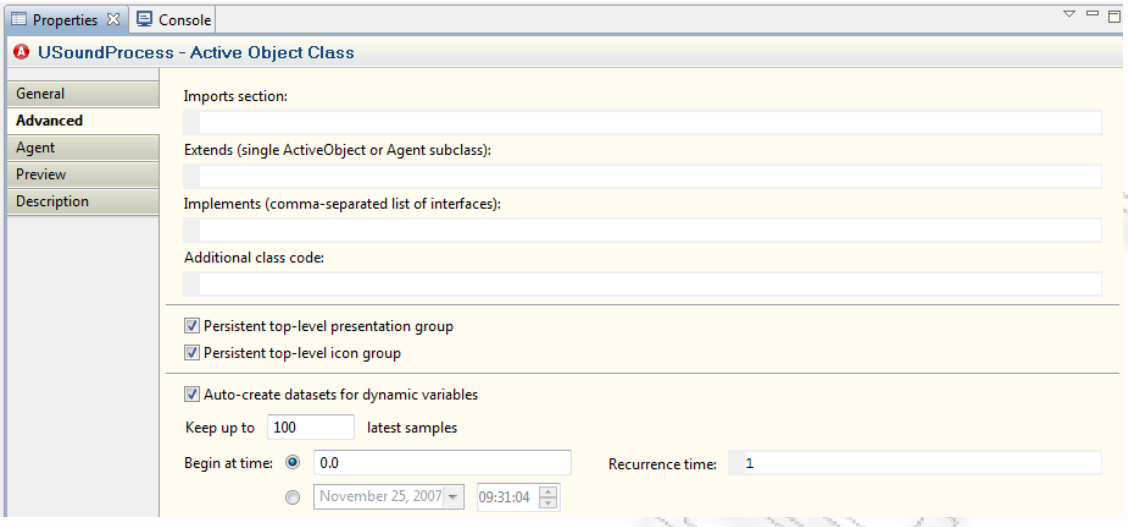
Στην εικόνα 8-44(β) εμφανίζεται το εικονίδιο της κλάσης και το διάγραμμα των διαδικασιών της κλάσης. Αρχικά αναζητείται ο πρώτος τεχνικός που είναι διαθέσιμος, seizeTechUSound, που θα χειριστεί το μηχάνημα του υπέρηχου. Αφού βρεθεί ο τεχνικός, sendTechToUS, ελέγχεται η διαθεσιμότητα των μηχανημάτων υπερήχου και το ποιό είναι ελεύθερο προς χρήση. Μόλις έρθει το μηχάνημα, bringUSound, γίνεται η εξέταση στον ασθενή, doUSound και αποδεσμεύονται πρώτα επιστρέφεται το μηχάνημα, returnUSound και ύστερα ο χειριστής, releaseTechUSound, οι πόροι της διαδικασίας.

Παρατηρείται στην εικόνα ότι η λέξη Back είναι υπογραμμισμένη και έχει μπλε χρώμα. Αυτό υποδεικνύει ότι η συγκεκριμένη λέξη είναι υπερσύνδεσμος. Το εικονίδιο της κλάσης είναι ο πράσινος συμπαγής κύκλος με το γράμμα U στο εσωτερικό του, το οποίο έχει μια πόρτα εισόδου και μια εξόδου. Η σειρά των διαδικασιών απεικονίζεται σε μορφή διαγράμματος και ορίζεται η αρχή και το τέλος της σειράς των διαδικασιών. Η αντιστοίχιση της αρχής της σειράς με την πόρτα εισόδου και η αντιστοίχιση του τέλους της σειράς με την πόρτα εξόδου είναι εσωτερική διαδικασία που εκτελεί το Λογισμικό.

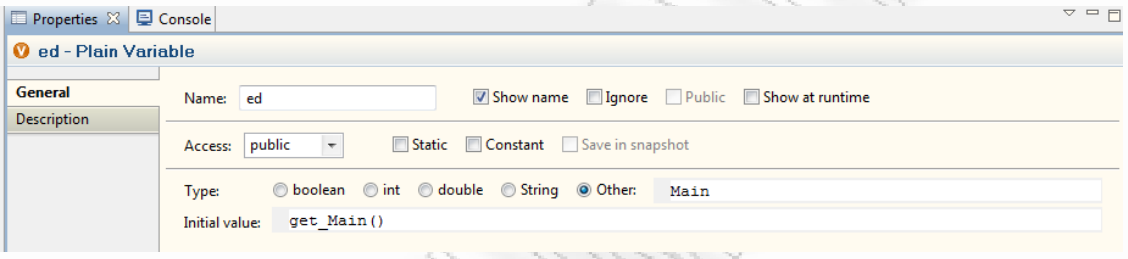
Στις εικόνες από 8-45(α) έως και την 8-55(γ) παρουσιάζονται οι ιδιότητες των στοιχείων και των οντοτήτων που αποτελούν την κλάση USoundProcess.



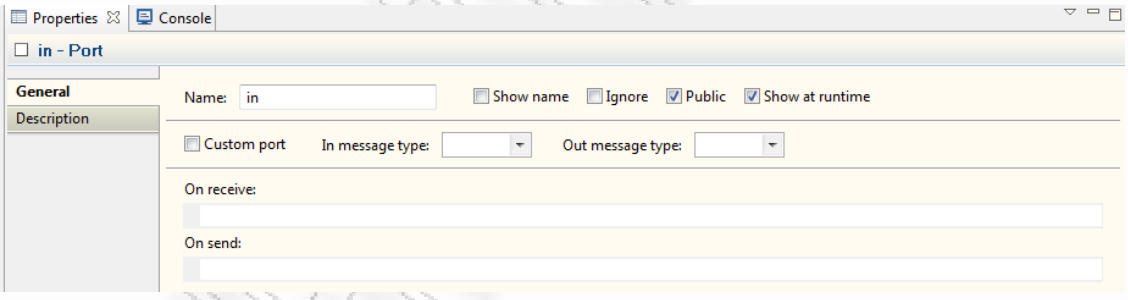
Εικόνα 8-45(α): Γενικές ιδιότητες της USoundProcess.



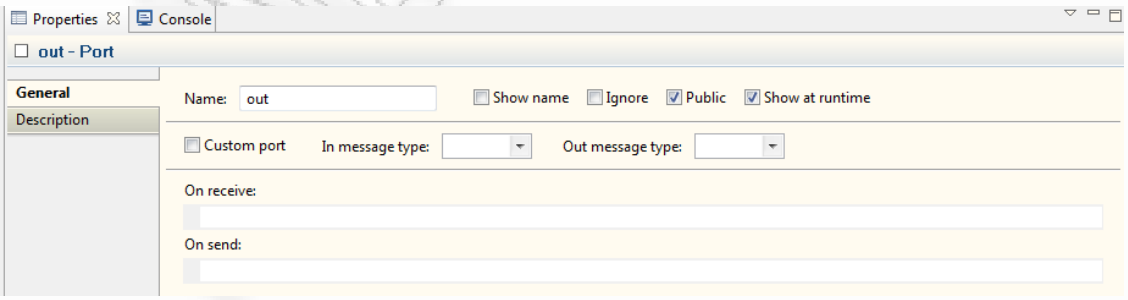
Εικόνα 8-45(β): Προχωρημένες ιδιότητες της USoundProcess.



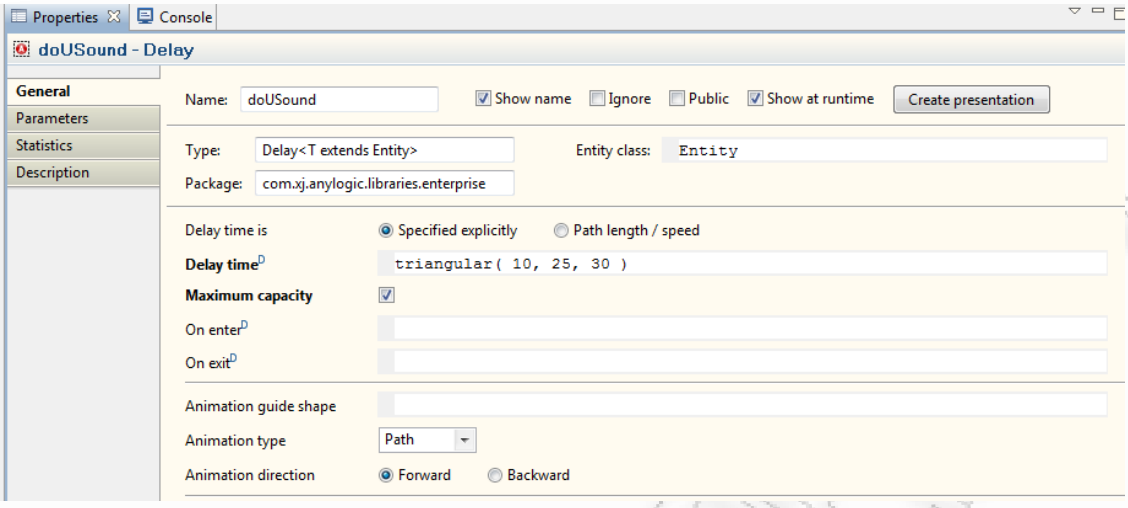
Εικόνα 8-46: Γενικές ιδιότητες της απλής μεταβλητής ed.



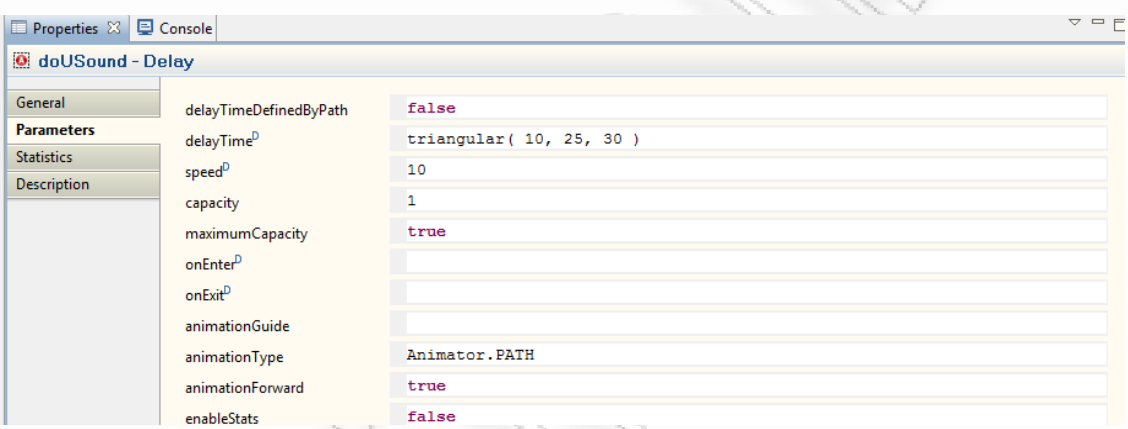
Εικόνα 8-47(α): Γενικές ιδιότητες της θύρας in.



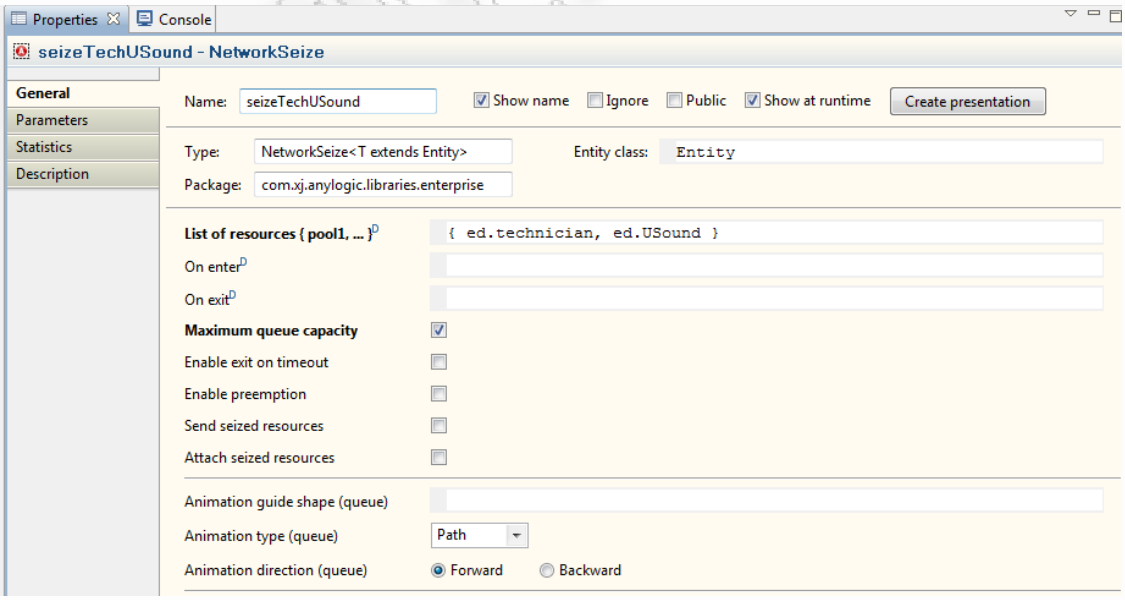
Εικόνα 8-47(β): Γενικές ιδιότητες της θύρας out.



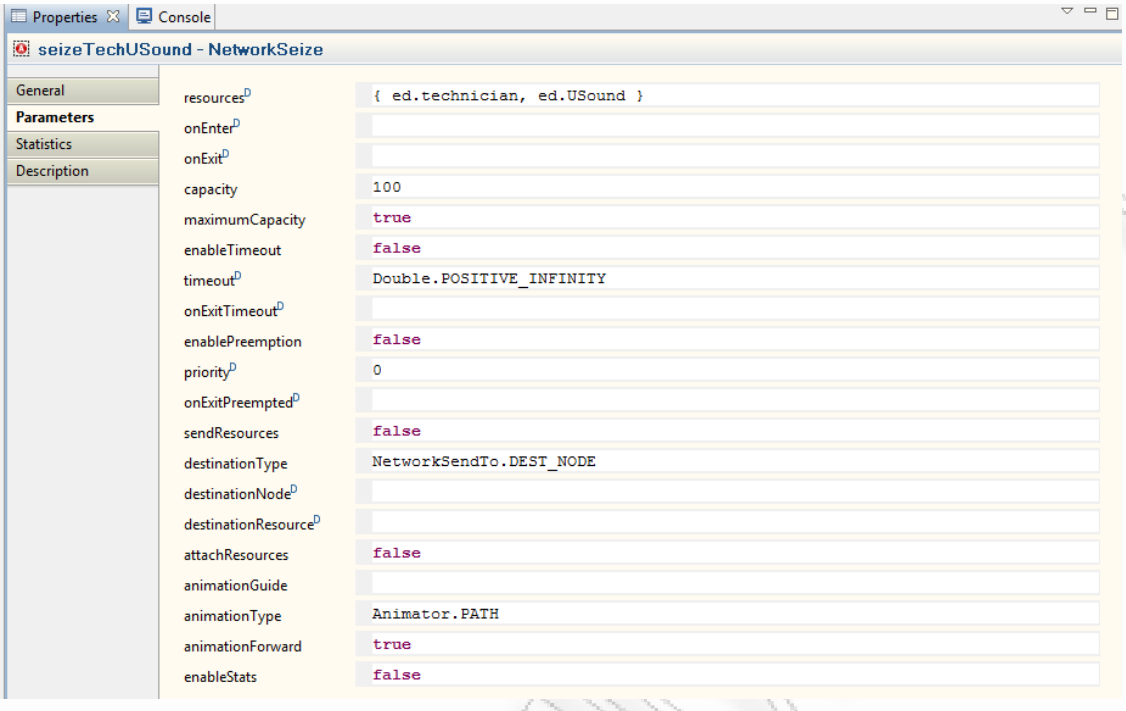
Εικόνα 8-48(α): Γενικές ιδιότητες της οντότητας doUSound τύπου Delay.



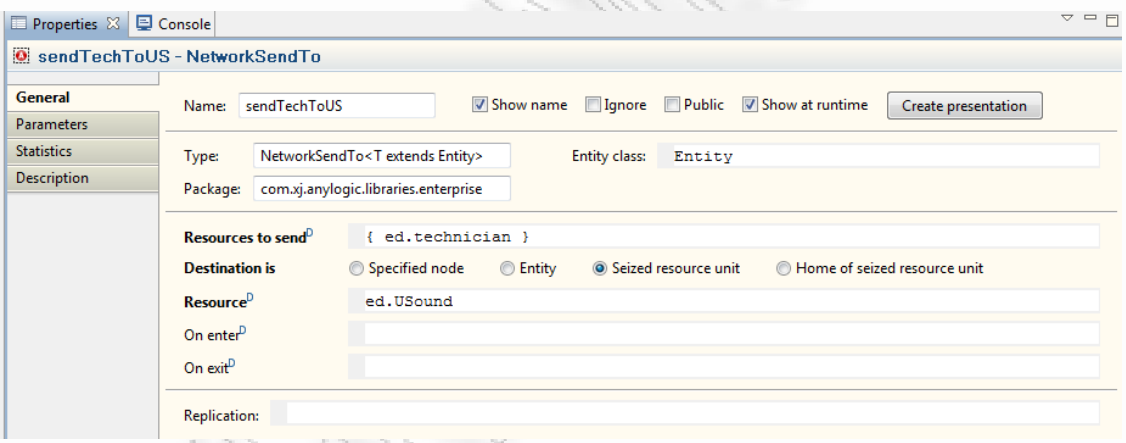
Εικόνα 8-48(β): Παράμετροι της οντότητας doUSound τύπου Delay.



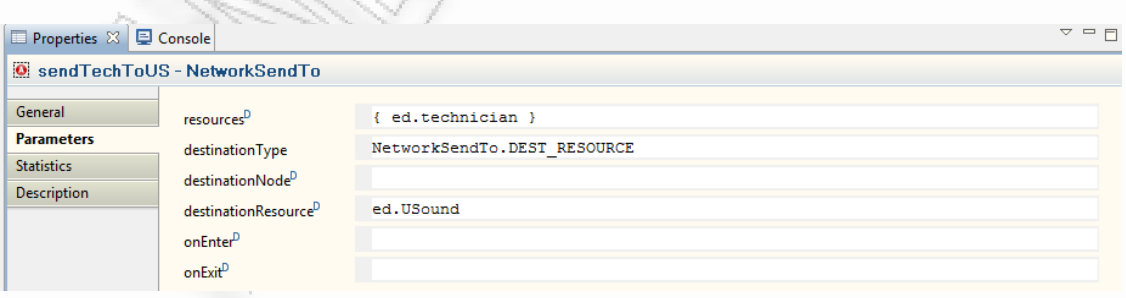
Εικόνα 8-49(α): Γενικές ιδιότητες της οντότητας seizeTechUSound τύπου NetworkSeize.



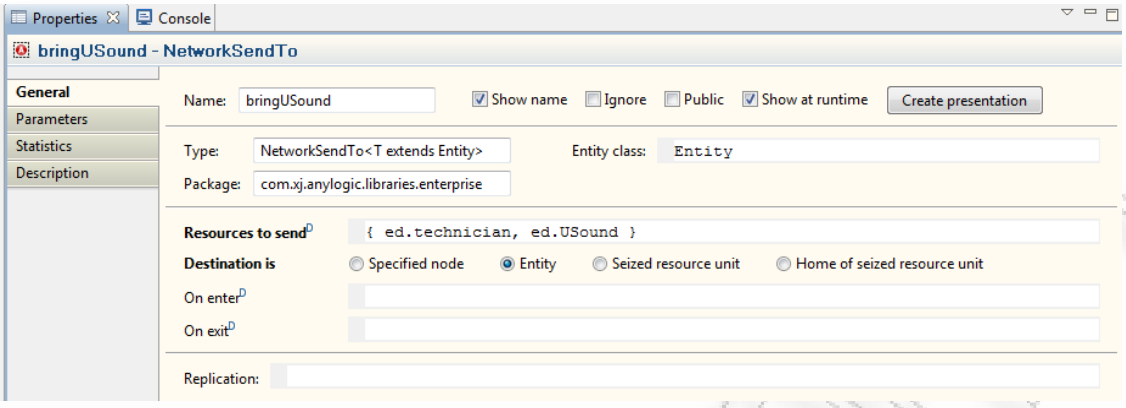
Εικόνα 8-49(β): Παράμετροι της οντότητας seizeTechUSound τύπου NetworkSeize.



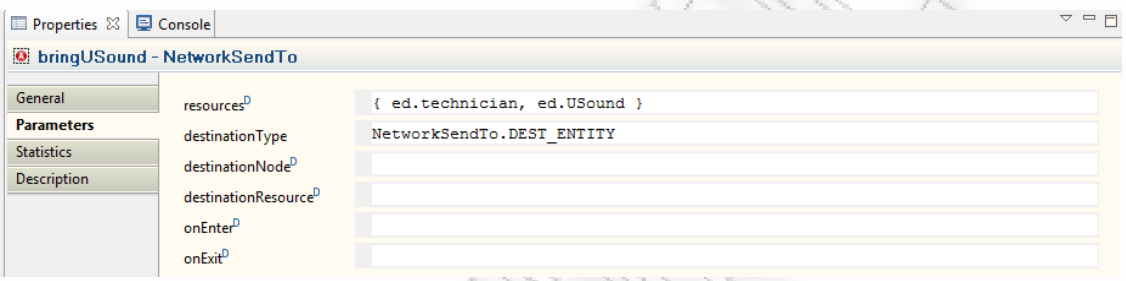
Εικόνα 8-50(α): Γενικές ιδιότητες της οντότητας sendTechToUS τύπου NetworkSendTo.



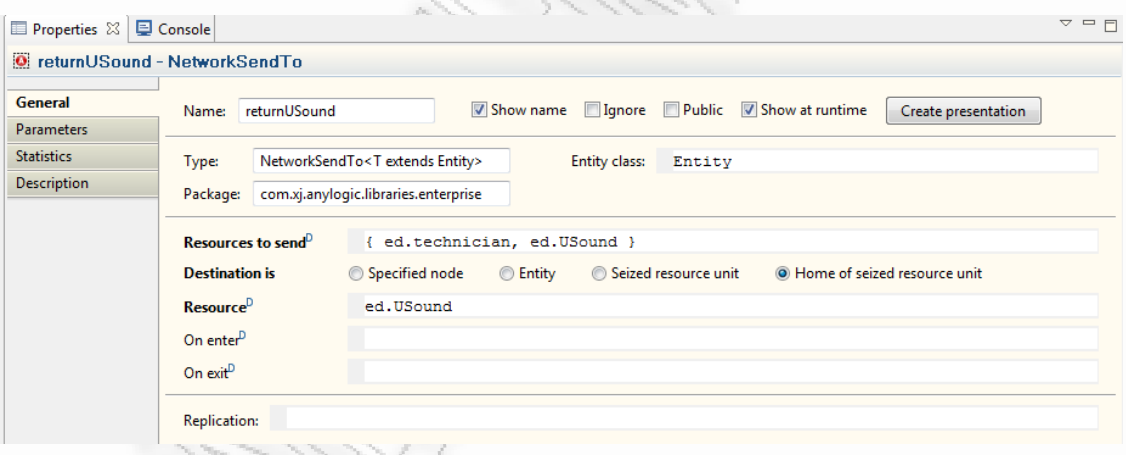
Εικόνα 8-50(β): Παράμετροι της οντότητας sendTechToUS τύπου NetworkSendTo.



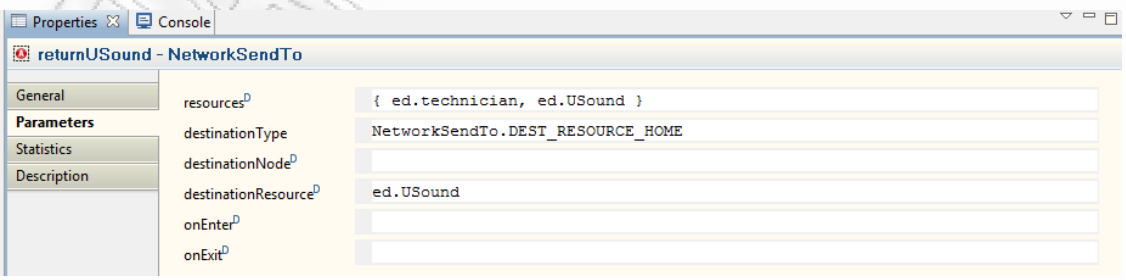
Εικόνα 8-51(α): Γενικές ιδιότητες της οντότητας bringUSound τύπου NetworkSendTo.



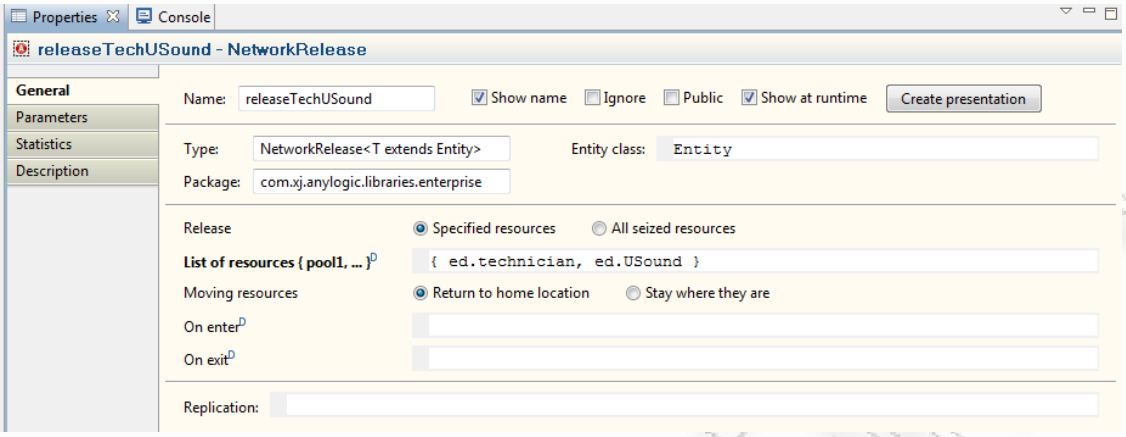
Εικόνα 8-51(β): Παράμετροι της οντότητας bringUSound τύπου NetworkSendTo.



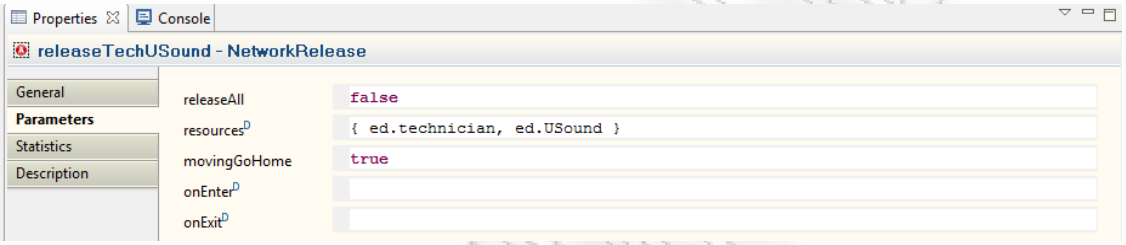
Εικόνα 8-52(α): Γενικές ιδιότητες της οντότητας returnUSound τύπου NetworkSendTo.



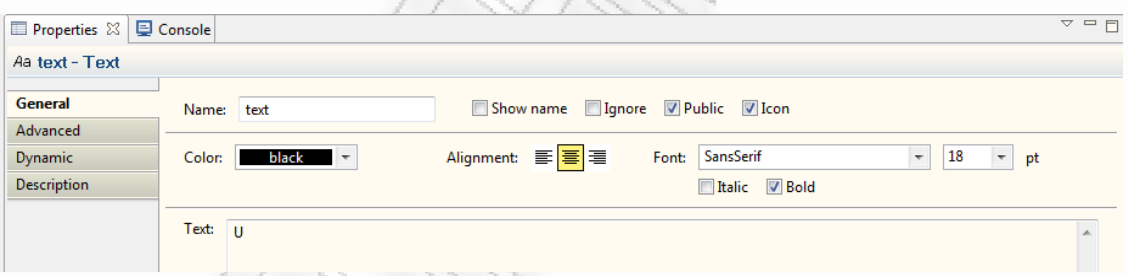
Εικόνα 8-52(β): Παράμετροι της οντότητας returnUSound τύπου NetworkSendTo.



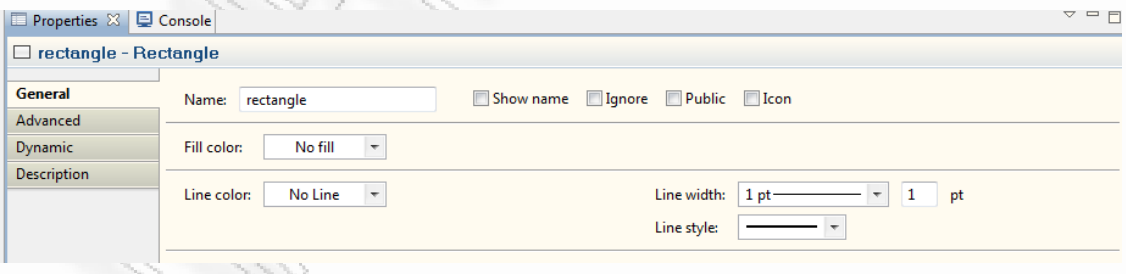
Εικόνα 8-53(α): Γενικές ιδιότητες της οντότητας releaseTechUSound τύπου NetworkRelease.



Εικόνα 8-53(β): Παράμετροι της οντότητας releaseTechUSound τύπου NetworkRelease.

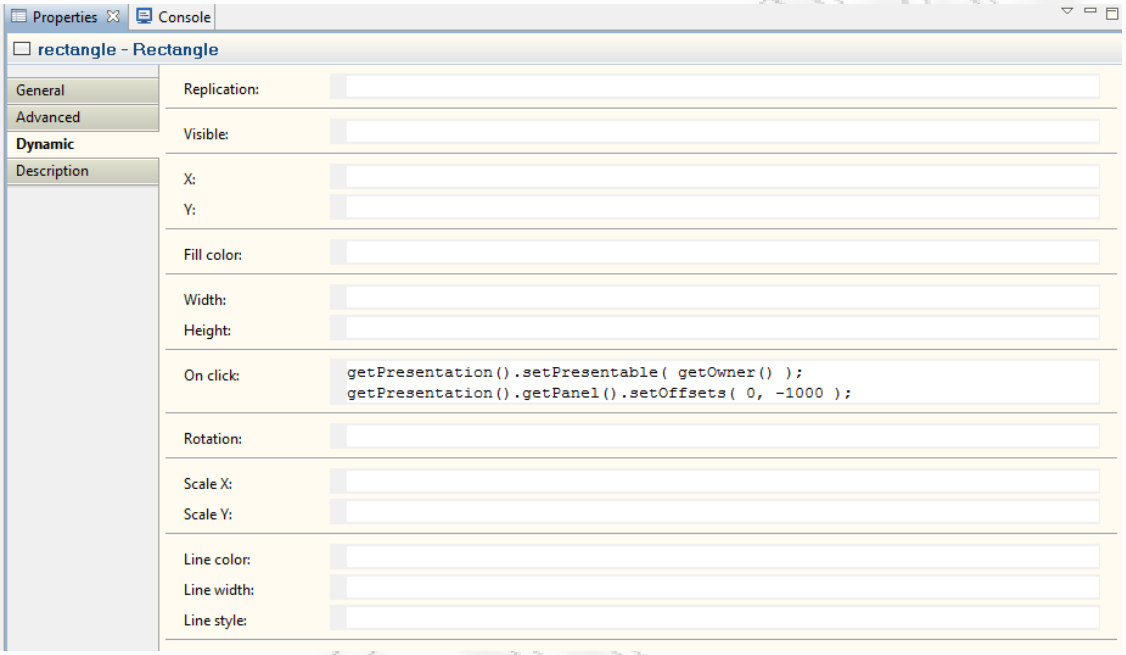


Εικόνα 8-54(α): Γενικές ιδιότητες του αντικειμένου κειμένου text.



Εικόνα 8-55(α): Γενικές ιδιότητες του αντικειμένου rectangle τύπου Rectangle.

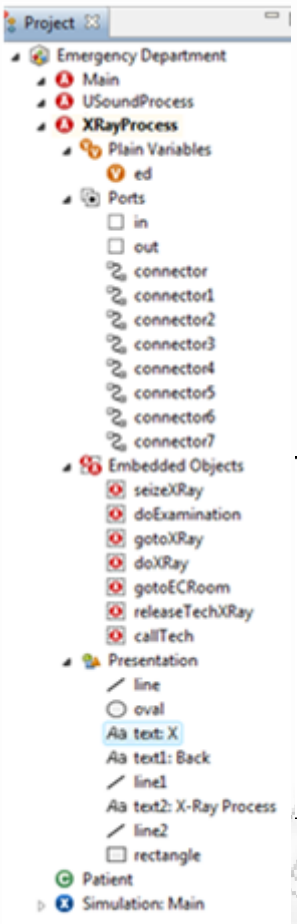
Εικόνα 8-55(β): Προχωρημένες ιδιότητες του αντικειμένου rectangle τύπου Rectangle.



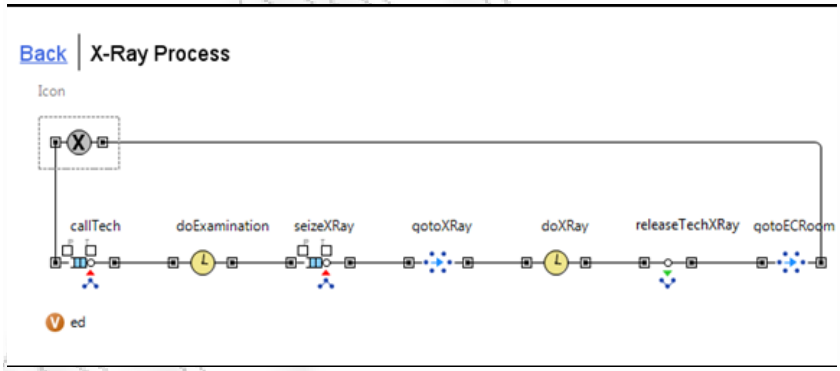
Εικόνα 8-55(γ): Δυναμικές ιδιότητες του αντικειμένου rectangle τύπου Rectangle (δεσμός Back).

8.3 Η κλάση XRayProcess

Η κλάση XRayProcess μοντελοποιεί την διαδικασία με την οποία γίνεται ακτινογραφία στον ασθενή. Στην εικόνα 8-56(α) παρουσιάζεται η ιεραρχική δομή της κλάσης και τα στοιχεία που την αποτελούν.



Εικόνα 8-56(α): Η ιεραρχική δομή της κλάσης XRayProcess.

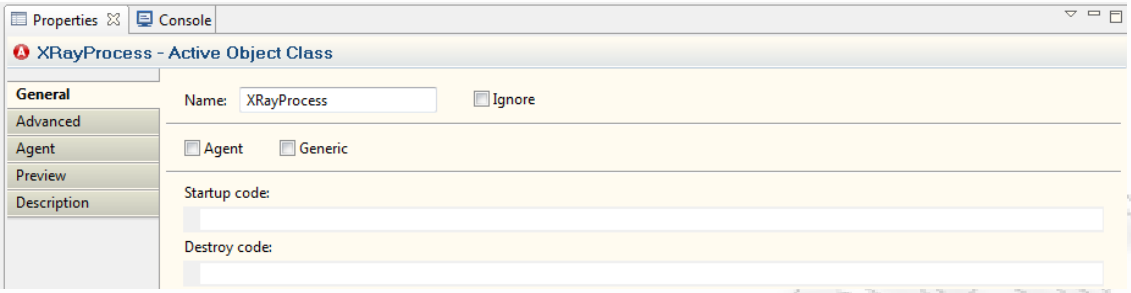


Εικόνα 8-56(β): Το εικονίδιο και το διάγραμμα διαδικασιών της κλάσης XRayProcess.

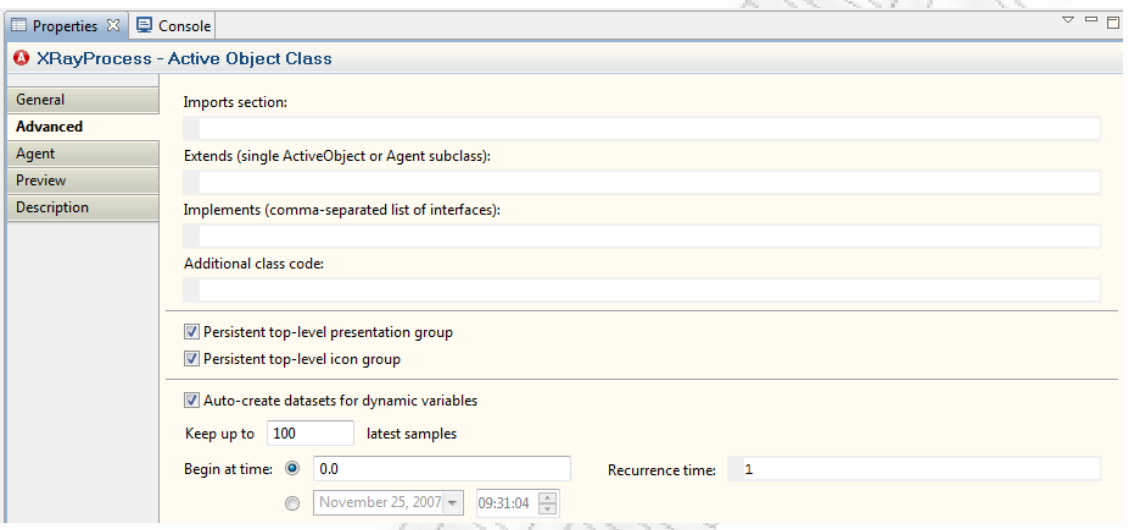
Στην εικόνα 8-56(β) εμφανίζεται το εικονίδιο της κλάσης και το διάγραμμα των διαδικασιών της κλάσης. Αρχικά καλείται ο πρώτος τεχνικός που είναι διαθέσιμος, callTech, γίνεται η εξέταση του ασθενούς, doExamination, ελέγχεται η διαθεσιμότητα των ακτινολογικών μηχανημάτων και το ποιό είναι ελεύθερο προς χρήση, seizeXRay. Γίνεται η ακτινογραφία στον ασθενή, doXRay, αποδεσμεύεται ο τεχνικός, releaseTechXRay και ο ασθενής επιστρέφει στο δωμάτιο της άμεσης φροντίδας, gotoECRoom.

Παρατηρείται και εδώ ότι η λέξη Back είναι υπογραμμισμένη και έχει μπλε χρώμα. Αυτό υποδεικνύει ότι η συγκεκριμένη λέξη είναι υπερσύνδεσμος. Το εικονίδιο της κλάσης είναι ο γκρι συμπαγής κύκλος με το γράμμα X στο εσωτερικό του, το οποίο έχει μια πόρτα εισόδου και μια εξόδου. Η σειρά των διαδικασιών απεικονίζεται σε μορφή διαγράμματος και ορίζεται η αρχή και το τέλος της σειράς των διαδικασιών. Η αντιστοίχιση της αρχής της σειράς με την πόρτα εισόδου και η αντιστοίχιση του τέλους της σειράς με την πόρτα εξόδου είναι εσωτερική διαδικασία που εκτελεί το Λογισμικό.

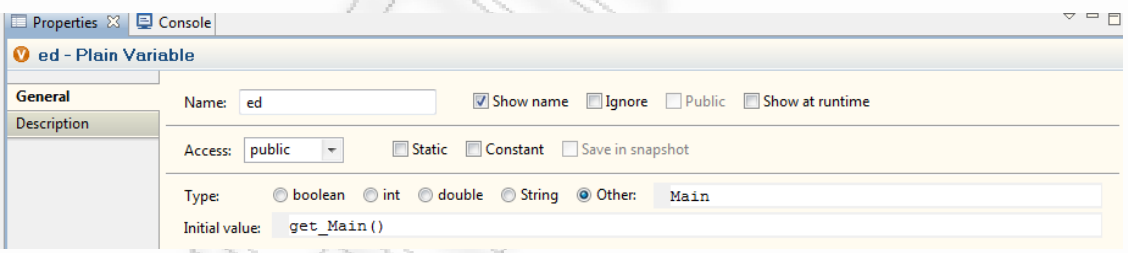
Στις εικόνες από την 8-57(α) έως και την 8-68 παρουσιάζονται οι ιδιότητες των στοιχείων και των οντοτήτων της κλάσης XRayProcess.



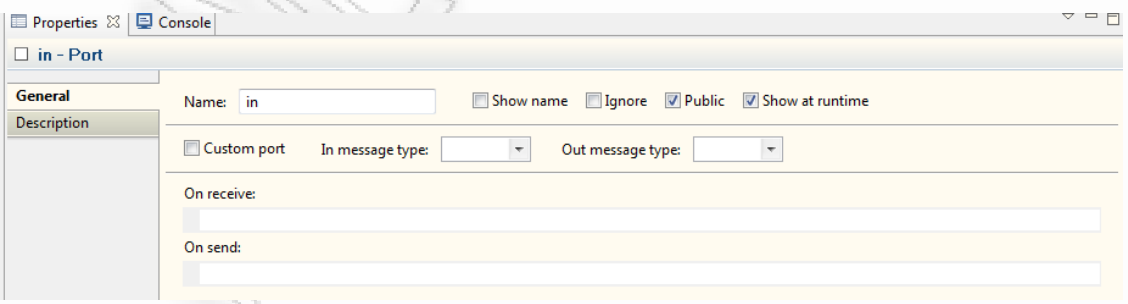
Εικόνα 8-57(α): Γενικές ιδιότητες της κλάσης XRayProcess.



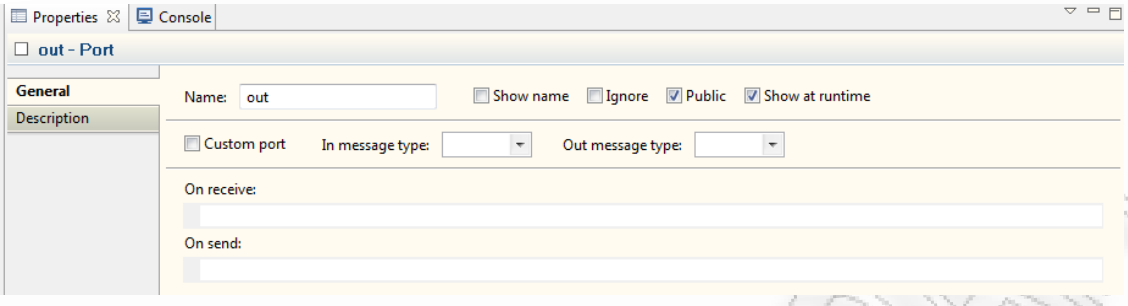
Εικόνα 8-57(β): Προχωρημένες ιδιότητες της κλάσης XRayProcess.



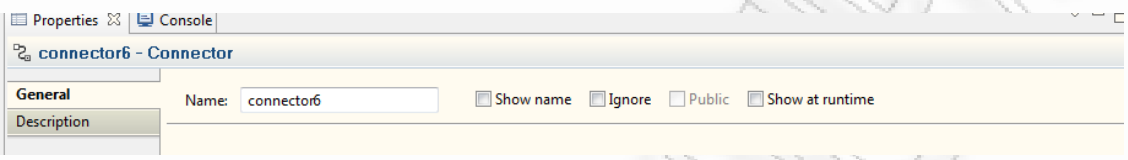
Εικόνα 8-58: Γενικές ιδιότητες της απλής μεταβλητής ed.



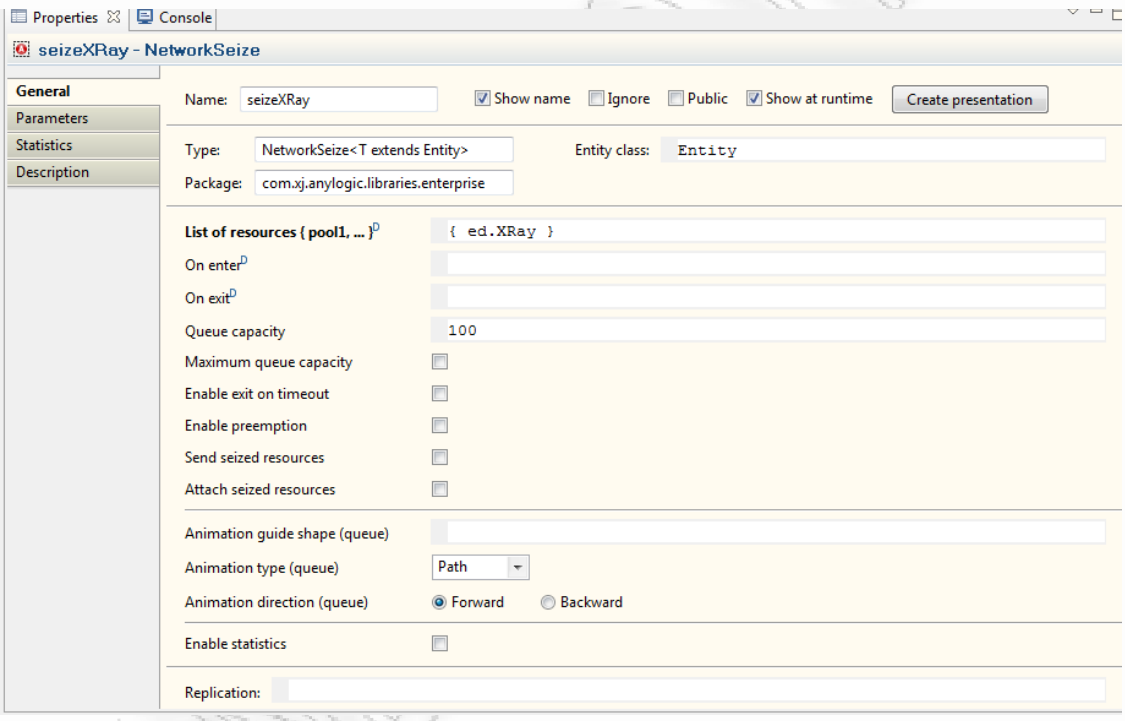
Εικόνα 8-59(α): Γενικές ιδιότητες της θύρας in.



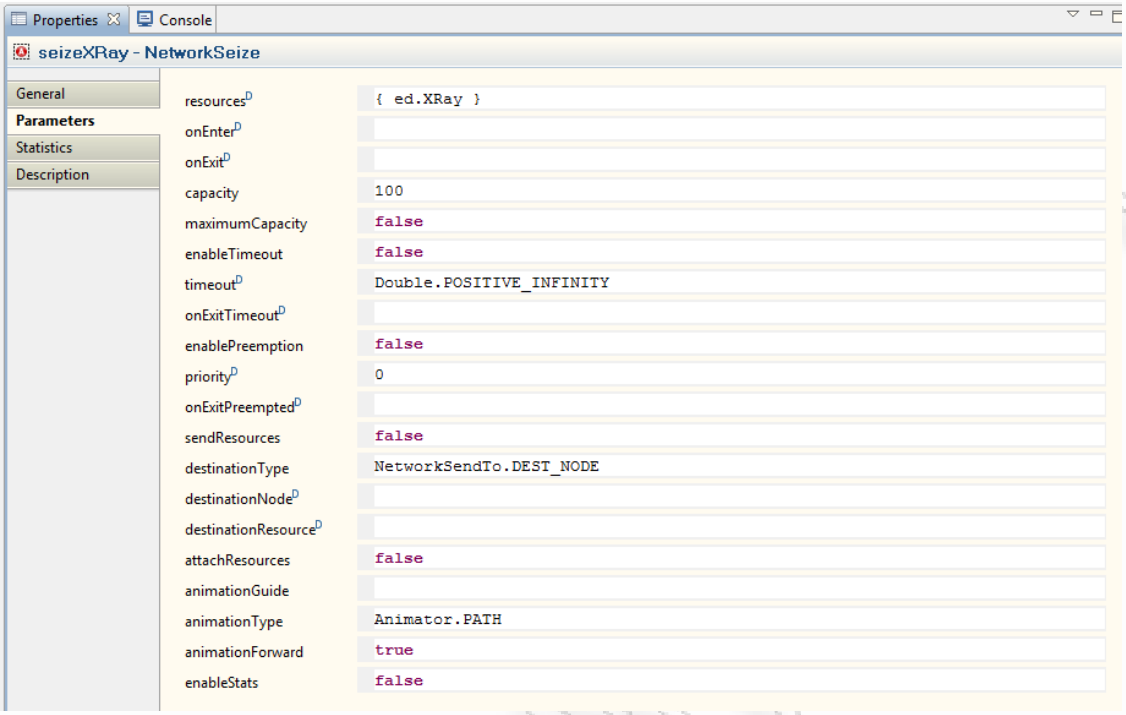
Εικόνα 8-59(β): Γενικές ιδιότητες της θύρας out.



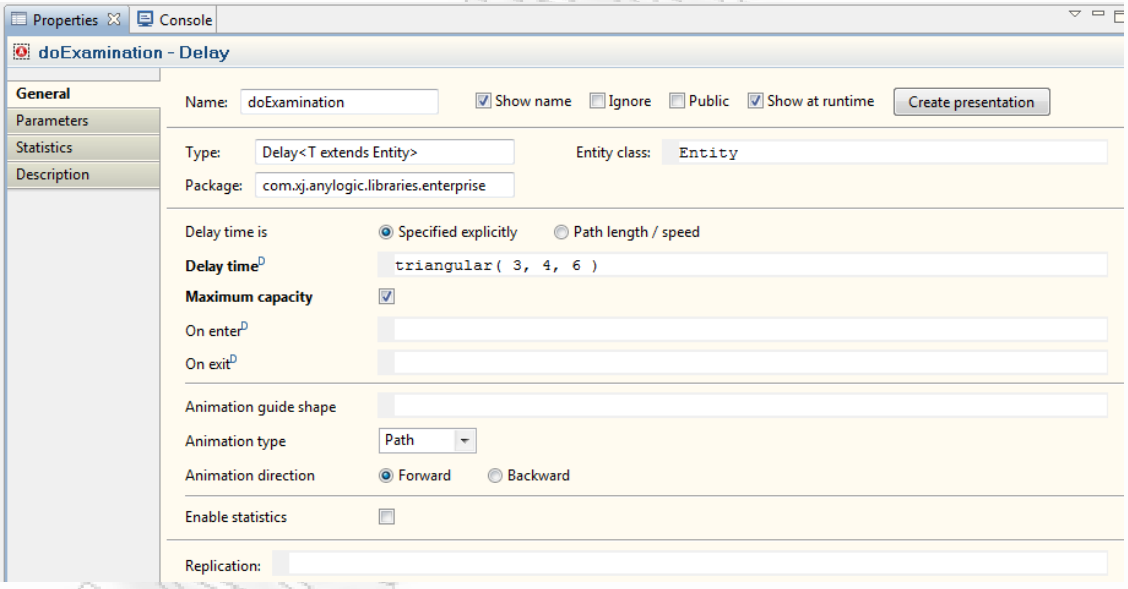
Εικόνα 8-59(γ): Γενικές ιδιότητες του συνδέσμου connector6.



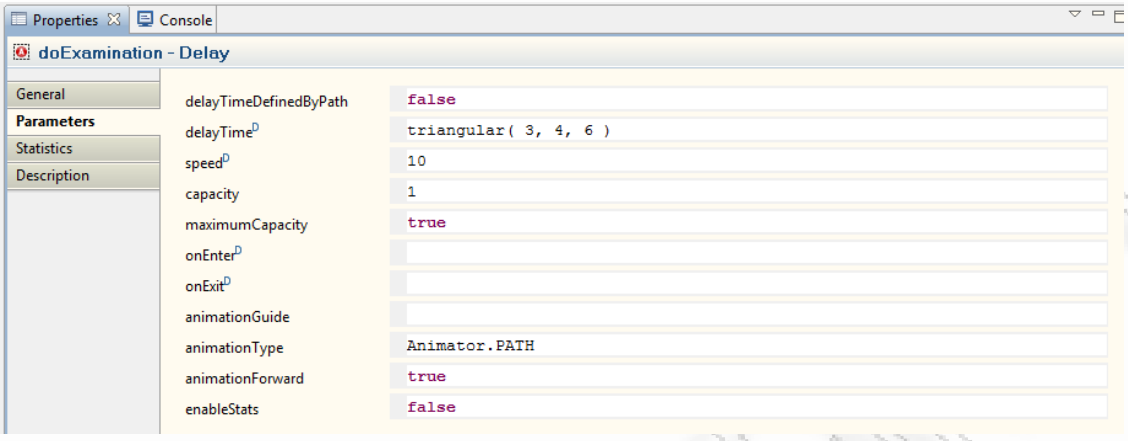
Εικόνα 8-60(α): Γενικές ιδιότητες της οντότητας seizeXRay τύπου NetworkSeize.



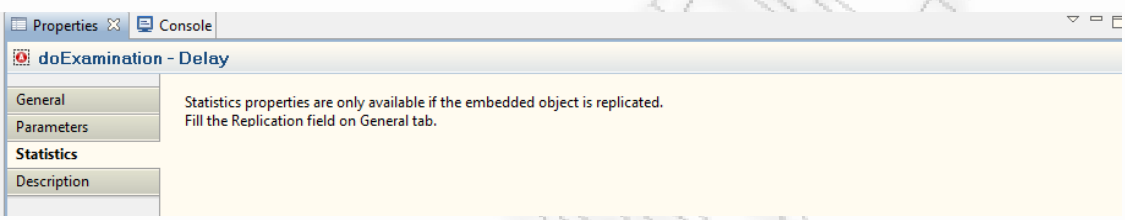
Εικόνα 8-60(β): Παράμετροι της οντότητας seizeXRay τύπου NetworkSeize.



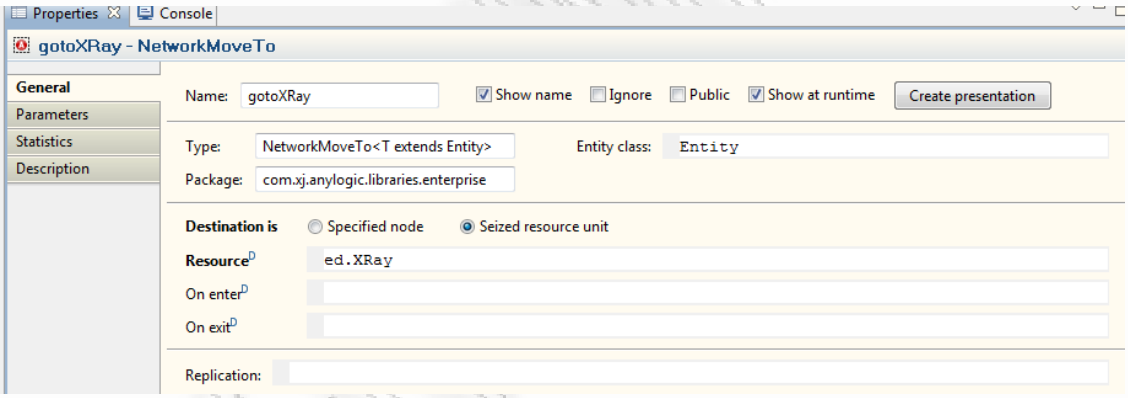
Εικόνα 8-61(α): Γενικές ιδιότητες της οντότητας doExamination τύπου Delay.



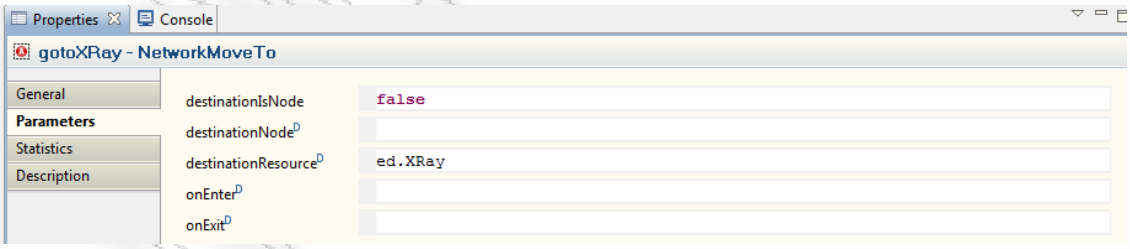
Εικόνα 8-61(β): Παράμετροι της οντότητας doExamination τύπου Delay.



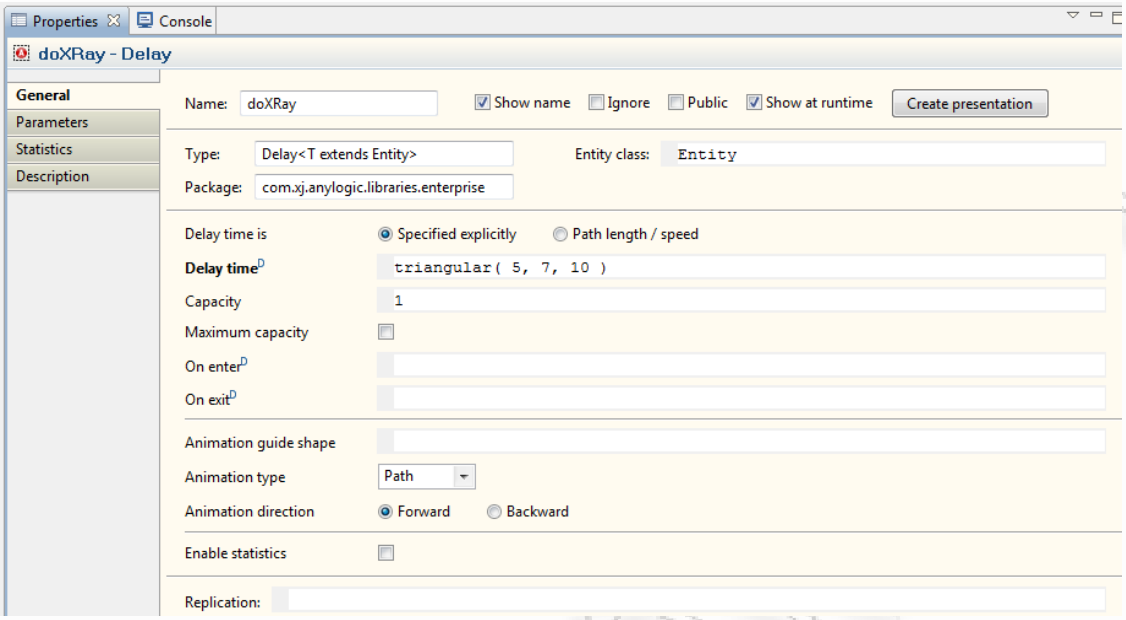
Εικόνα 8-61(γ): Στατιστικές ιδιότητες της οντότητας doExamination τύπου Delay.



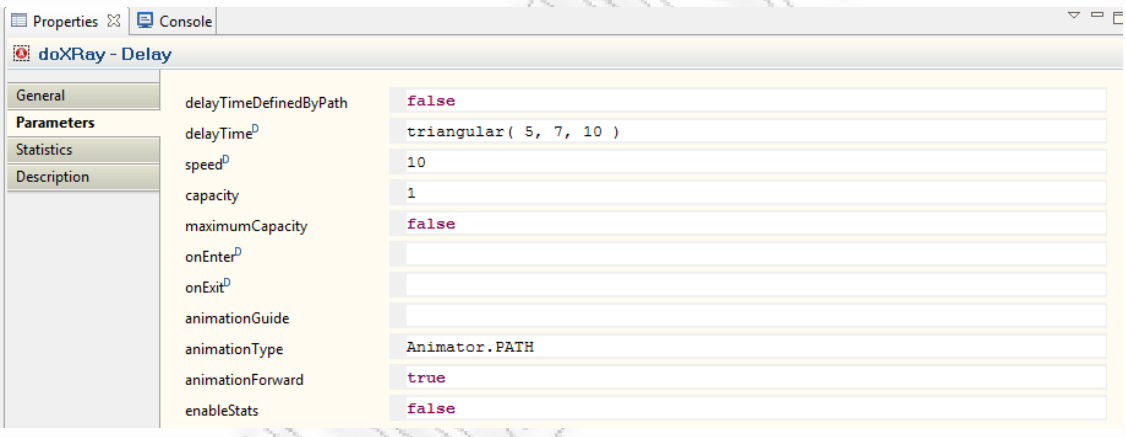
Εικόνα 8-62(α): Γενικές ιδιότητες της οντότητας gotoXRay τύπου NetworkMoveTo.



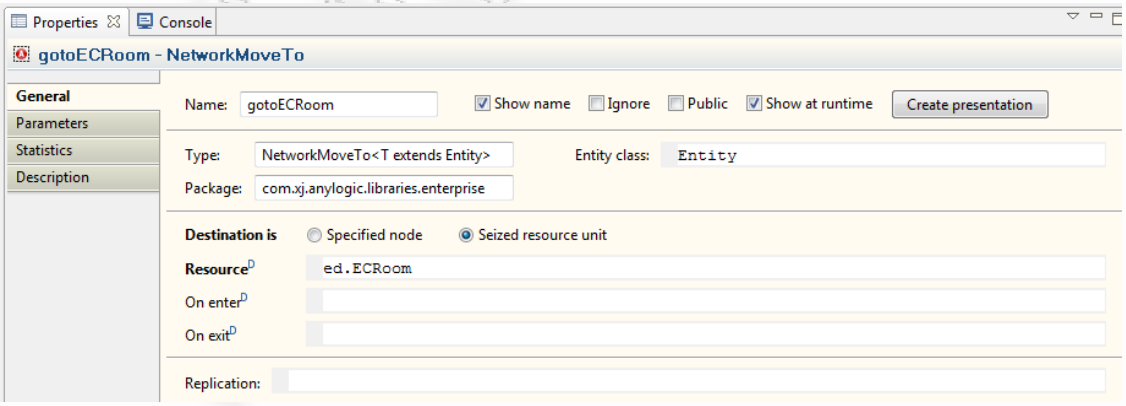
Εικόνα 8-62(β): Παράμετροι της οντότητας gotoXRay τύπου NetworkMoveTo.



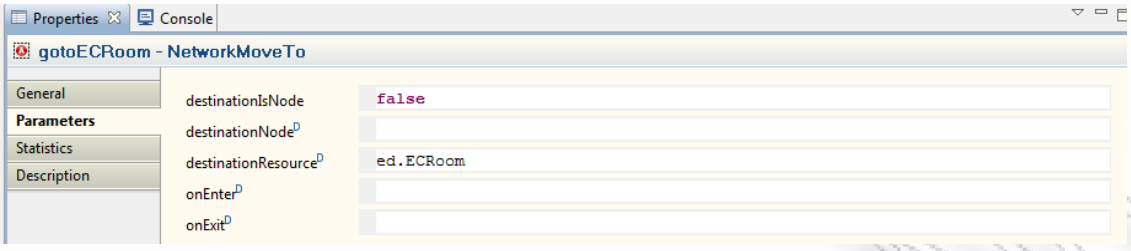
Εικόνα 8-63(α): Γενικές ιδιότητες της οντότητας doXRay τύπου Delay.



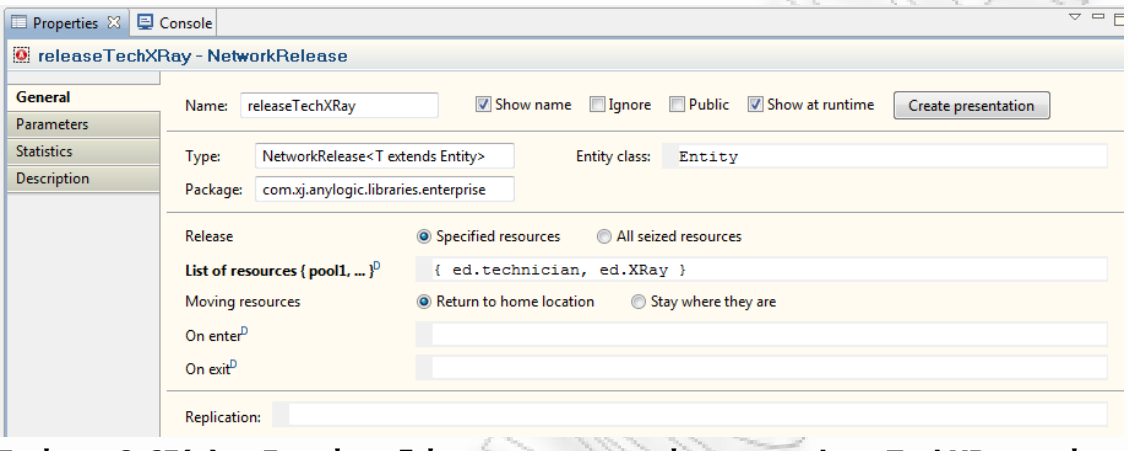
Εικόνα 8-63(β): Παράμετροι της οντότητας doXRay τύπου Delay.



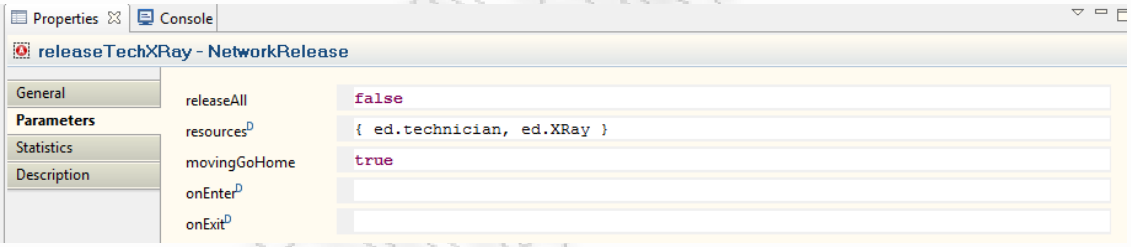
Εικόνα 8-64(α): Γενικές ιδιότητες της οντότητας gotoECRoom τύπου NetworkMoveTo.



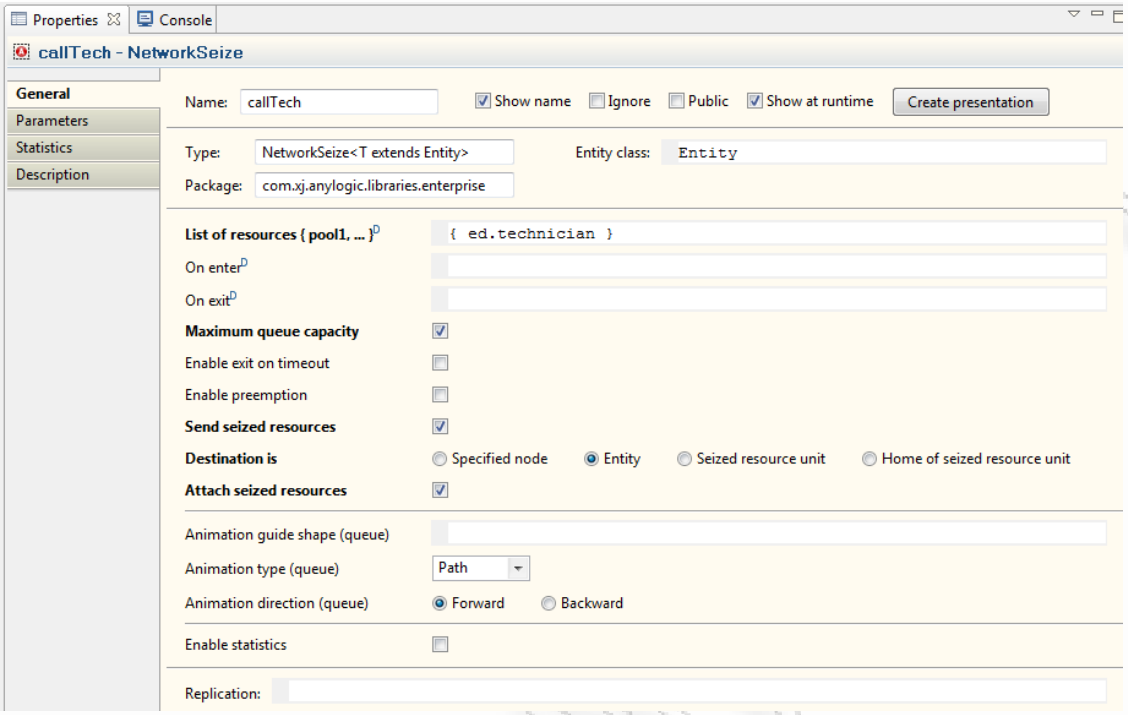
Εικόνα 8-64(β): Παράμετροι της οντότητας gotoECRoom τύπου NetworkMoveTo.



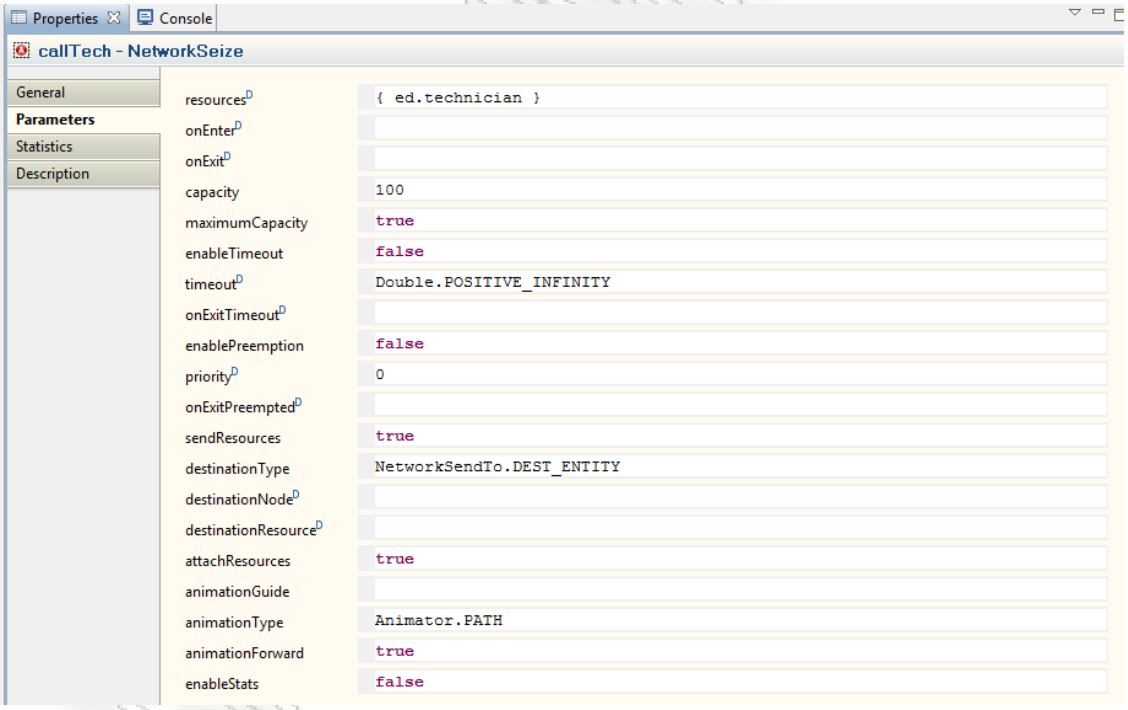
Εικόνα 8-65(α): Γενικές ιδιότητες της οντότητας releaseTechXRay τύπου NetworkRelease.



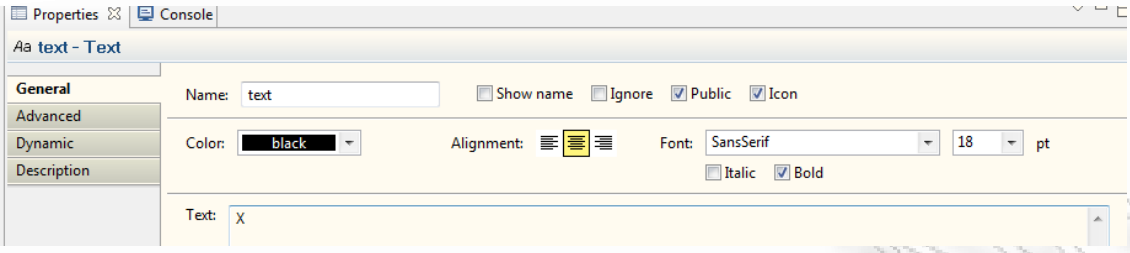
Εικόνα 8-65(β): Παράμετροι της οντότητας releaseTechXRay τύπου NetworkRelease.



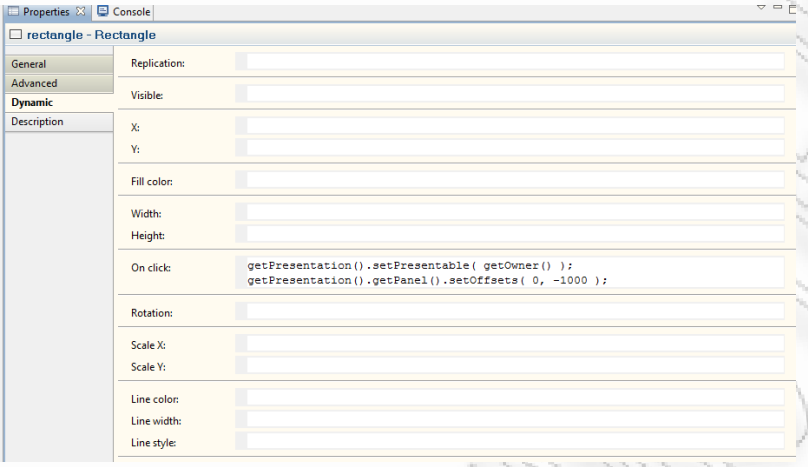
Εικόνα 8-66(α): Γενικές ιδιότητες της οντότητας callTech τύπου NetworkSeize.



Εικόνα 8-66(β): Παράμετροι της οντότητας callTech τύπου NetworkSeize.



Εικόνα 8-67: Γενικές ιδιότητες του αντικειμένου κειμένου text.

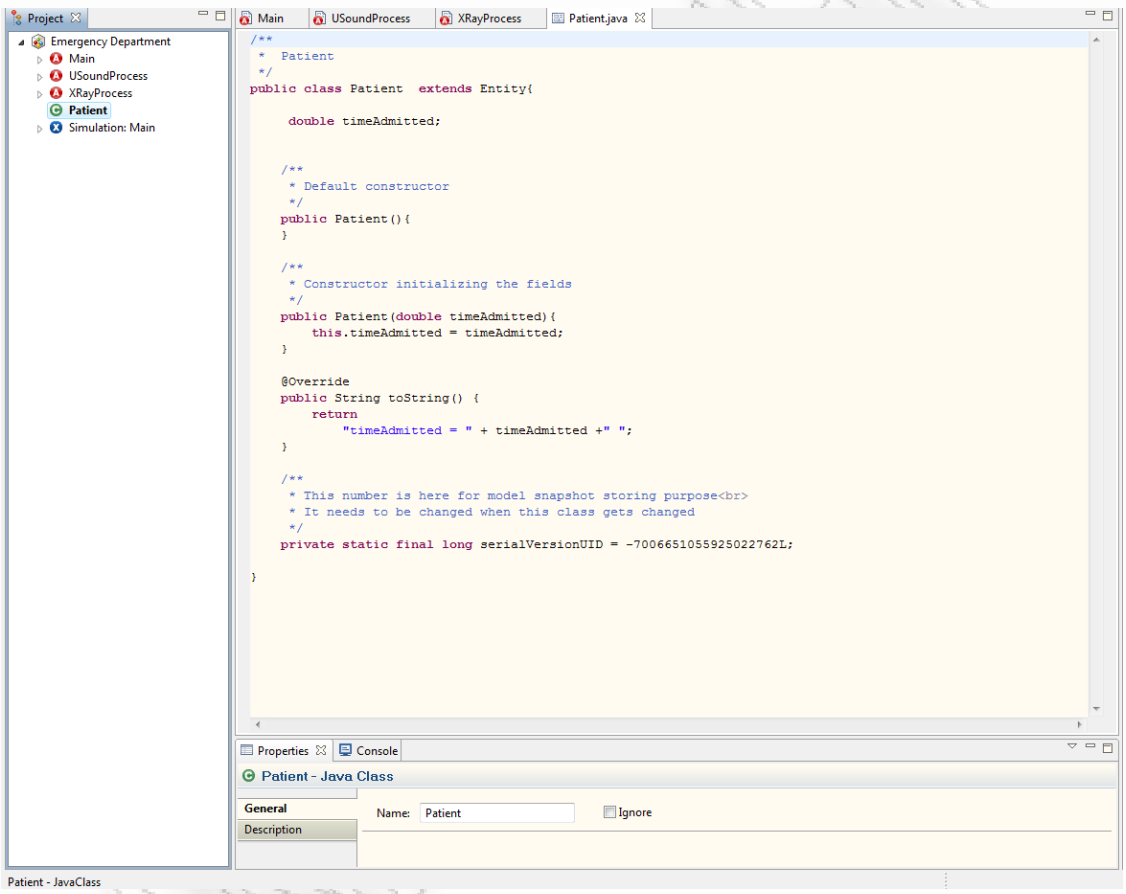


Εικόνα 8-68: Δυναμικές ιδιότητες του αντικειμένου rectangle τύπου Rectangle (δεσμός Back).

8.4 Η κλάση Patient.java

Η κλάση Patient υλοποιείται μέσω του κώδικα Patient.java που είναι γραμμένος στη Γλώσσα Προγραμματισμού Java, (για το λόγο αυτό το εικονίδιο της κλάσης, όπως εμφανίζεται στην ιεραρχία του έργου Emergency Department, είναι διαφορετικό). Με αυτό τον τρόπο δημιουργείται ο πράκτορας λογισμικού, Patient, που αναπαριστά τον ασθενή που χρήζει ιατρικής φροντίδας.

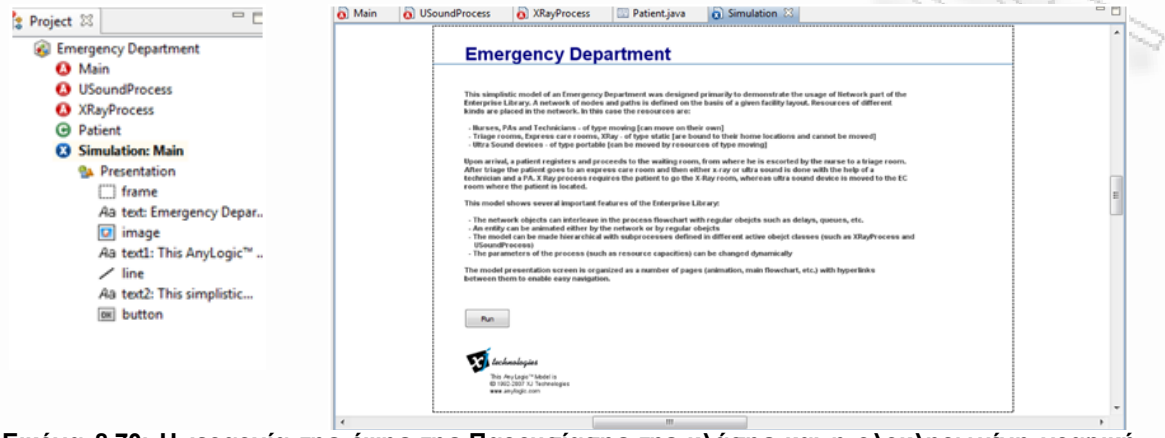
Στην εικόνα 8-69 δίνεται ο κώδικας που δημιουργεί τον πράκτορα λογισμικού. Μας ενδιαφέρει, για την προσομοίωσης του μοντέλου, ο χρόνος εισαγωγής του ασθενή, timeAdmitted.



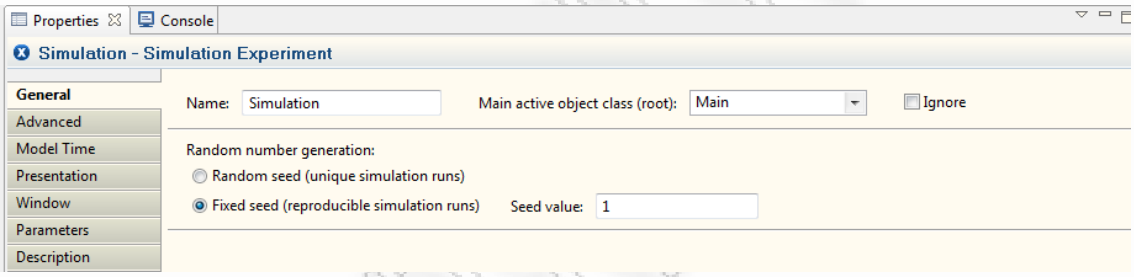
Εικόνα 8-69: Ο κώδικας, Patient.java, υλοποίησης του ασθενή ως πράκτορα λογισμικού, σε γλώσσα προγραμματισμού Java.

Η δήλωση: private static final long serialVersionUID = -7006651055925022762L;
στο τέλος του κώδικα υλοποίησης της κλάσης αφορά σε εσωτερική λειτουργία του Λογισμικού κατά τη διαδικασία της προσομοίωσης και δεν αφορά, δεν αλλάζει την λογική του μοντέλου.

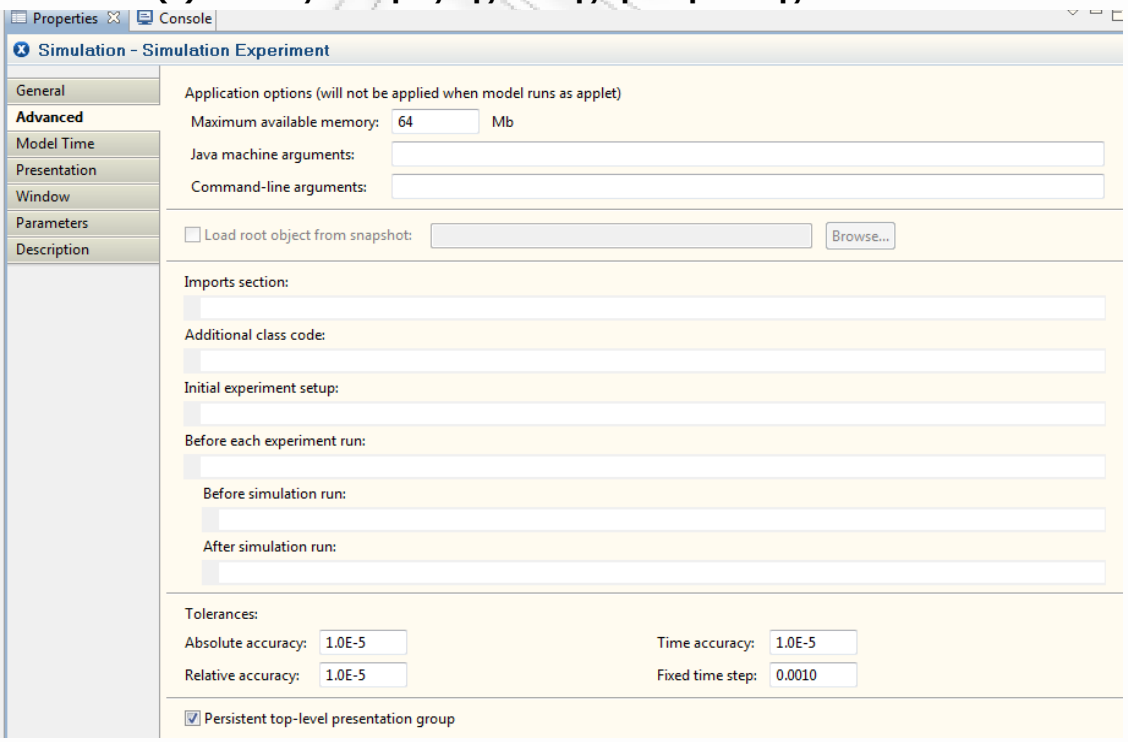
8.5 Η κλάση προσομοίωσης Simulation:Main του έργου Emergency Department



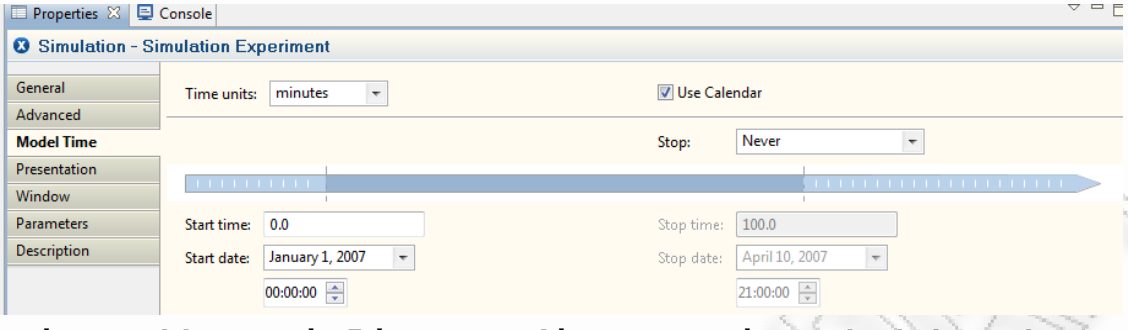
Εικόνα 8-70: Η ιεραρχία της όψης της Παρουσίασης της κλάσης και η ολοκληρωμένη γραφική εικόνα της κλάσης όπως αυτή παρουσιάζεται στο περιβάλλον ανάπτυξης του Λογισμικού.



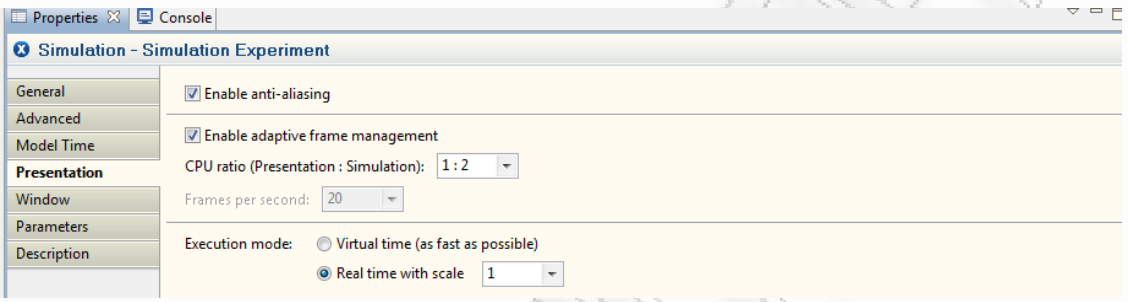
Εικόνα 8-71(α): Γενικές ιδιότητες της κλάσης προσομοίωσης Simulation:Main.



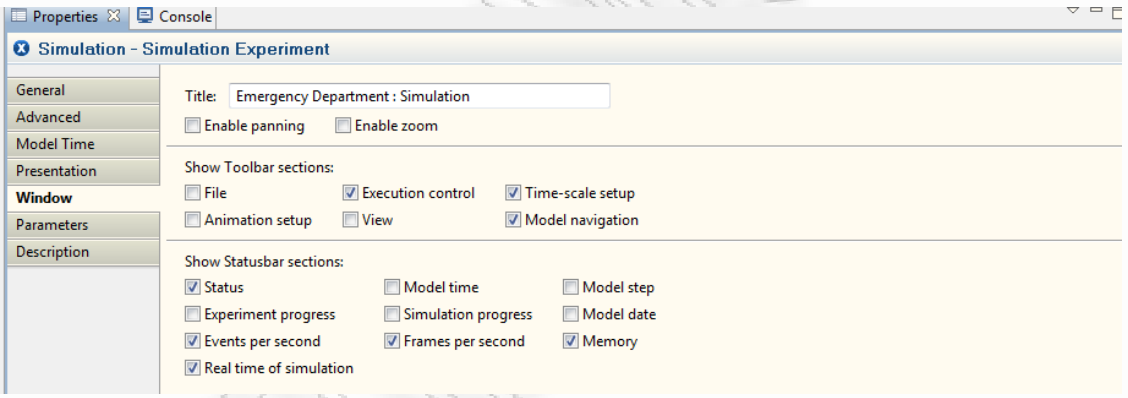
Εικόνα 8-71(β): Προχωρημένες ιδιότητες της κλάσης προσομοίωσης Simulation:Main.



Εικόνα 8-71(γ): Χρονικές ιδιότητες της κλάσης προσομοίωσης Simulation:Main.



Εικόνα 8-71(δ): Ιδιότητες παρουσίασης της κλάσης προσομοίωσης Simulation:Main.

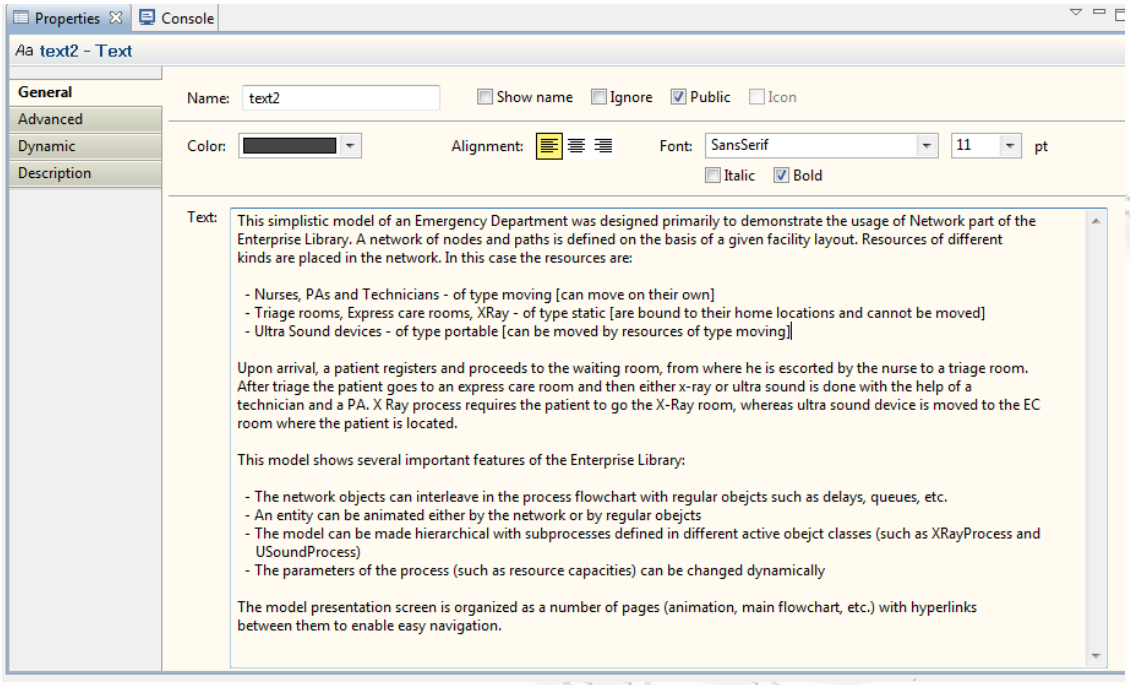


Εικόνα 8-71(ε): Ιδιότητες του παραθύρου της κλάσης προσομοίωσης Simulation:Main.

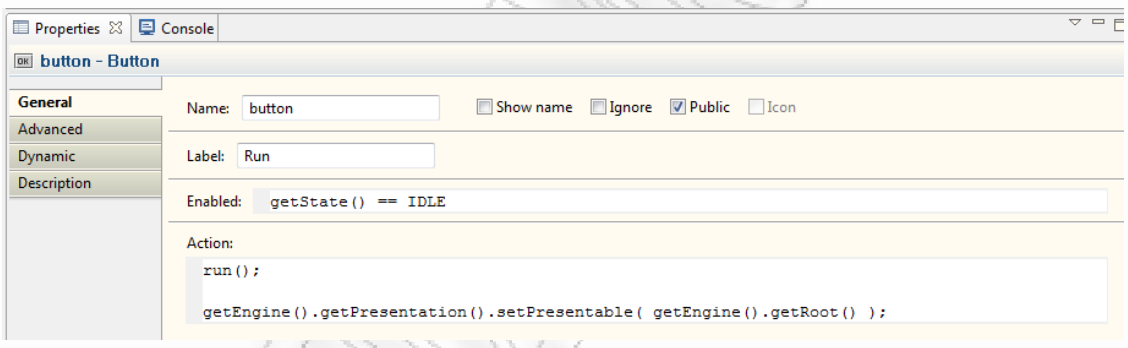
Η προσομοίωση του έργου Emergency Department ξεκινά μέσα από την κλάση παρουσίασης Simulation:Main πατώντας το κομβίο με την ένδειξη Run. Στην εικόνα 8-70, παρουσιάζεται το παράθυρο όπως αυτό εμφανίζεται στον γραφικό συντάκτη.

Στις εικόνες 8-71, από το (α) έως και το (ε), δίνονται όλες οι ιδιότητες και οι παράμετροι που χρησιμοποιούνται από το Λογισμικό για την προσομοίωση. Στην εικόνα 8-72 δίνονται οι γενικές ιδιότητες του παράθυρου κειμένου ενώ στις εικόνες 8-73(α) και 8-73(β) παρατίθενται οι ιδιότητες, γενικές και προχωρημένες αντίστοιχα, του κομβίου button.

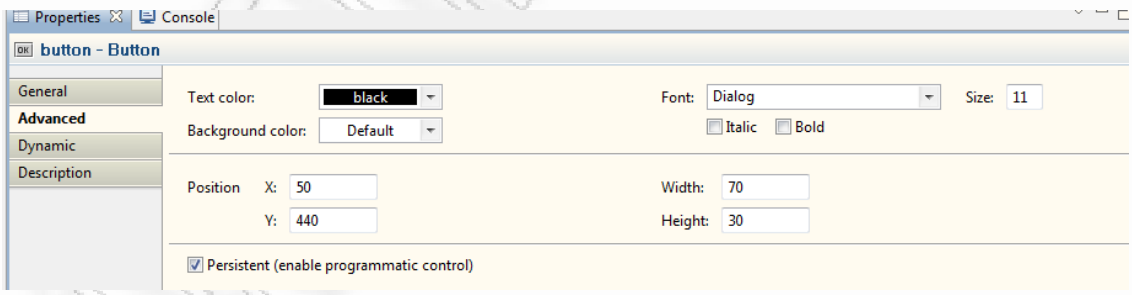
Στις εικόνες 8-74(α) έως και 8-74(ε) εμφανίζονται το αρχικό παράθυρο της προσομοίωσης και μερικά στιγμιότυπα από την προσομοίωση σε διαφορετικές χρονικές στιγμές. Παρουσιάζεται γραφικά η κλάση Main με όλα τα στοιχεία παρουσίασης που την αποτελούν. Οι πόροι του μοντέλου μπορούν να αλλάζουν δυναμικά κατά τη διάρκεια της προσομοίωσης, ενώ είναι δυνατό η προσομοίωση να γίνεται σε αργό ή σε γρήγορο ρυθμό πατώντας αντίστοιχα τα κομβία Slow ή Fast. Στην κάτοψη, κατά τη διάρκεια της προσομοίωσης, παρατηρούνται οι πράκτορες λογισμικού να «τρέχουν» πάνω στις προκαθορισμένες διαδρομές του Δικτύου. Τέλος, στην εικόνα 8-74(στ), εμφανίζεται ένα στιγμιότυπο από την προσομοίωση όταν ο χρήστης έχει επιλέξει να παρακολουθεί το Λογικό Μοντέλο του έργου.



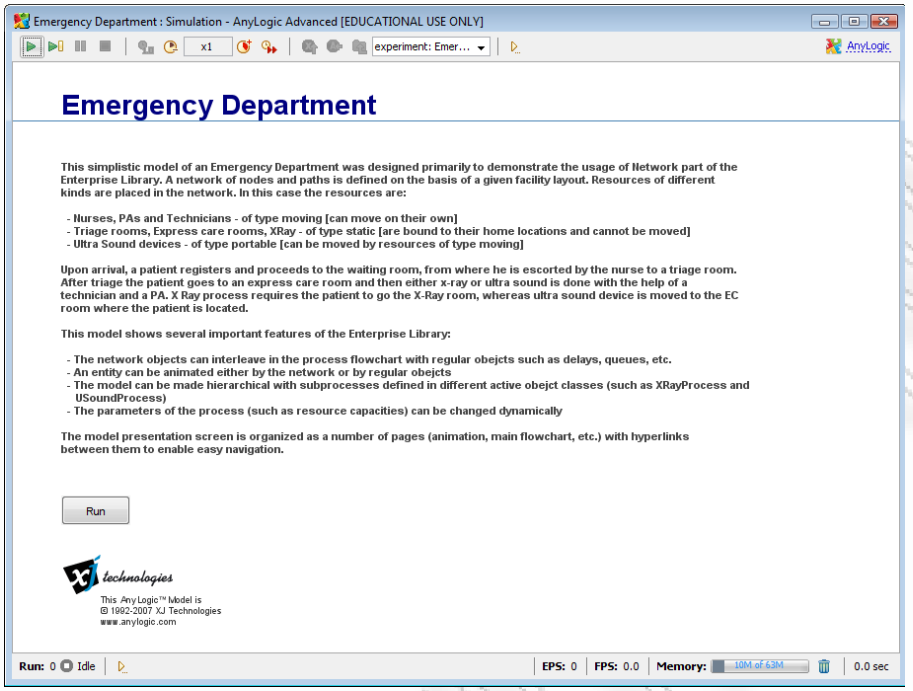
Εικόνα 8-72: Γενικές ιδιότητες του παραθύρου κειμένου text.



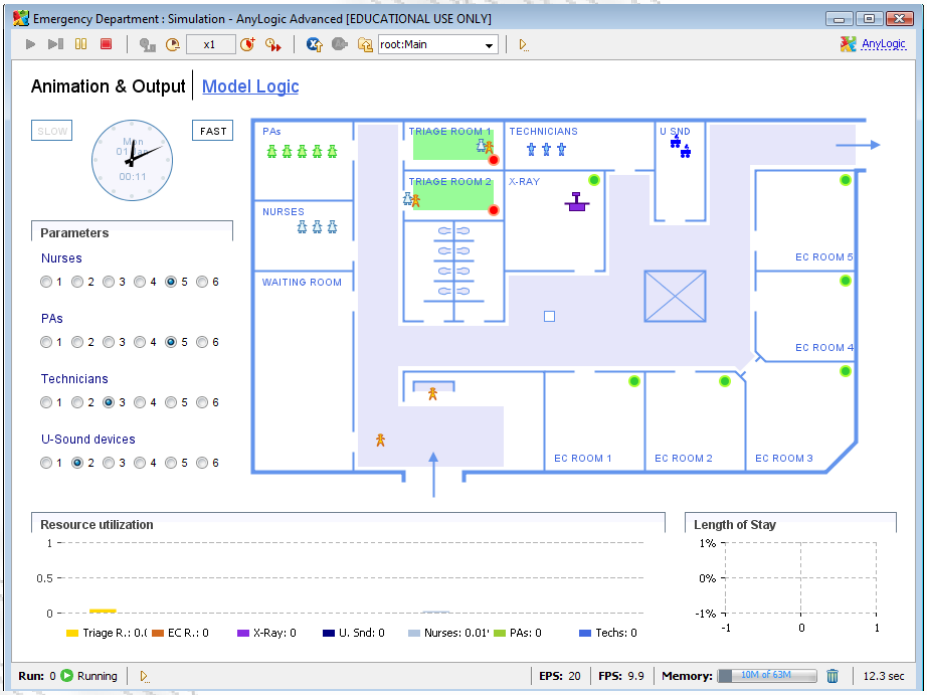
Εικόνα 8-73(α): Γενικές ιδιότητες του κομβίου button.



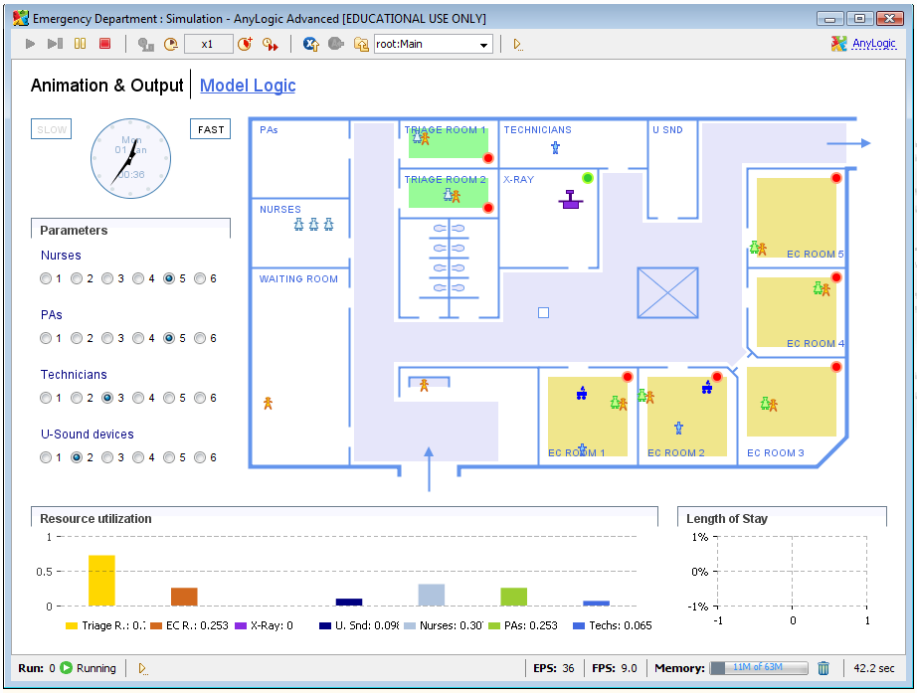
Εικόνα 8-73(β): Προχωρημένες ιδιότητες του κομβίου button.



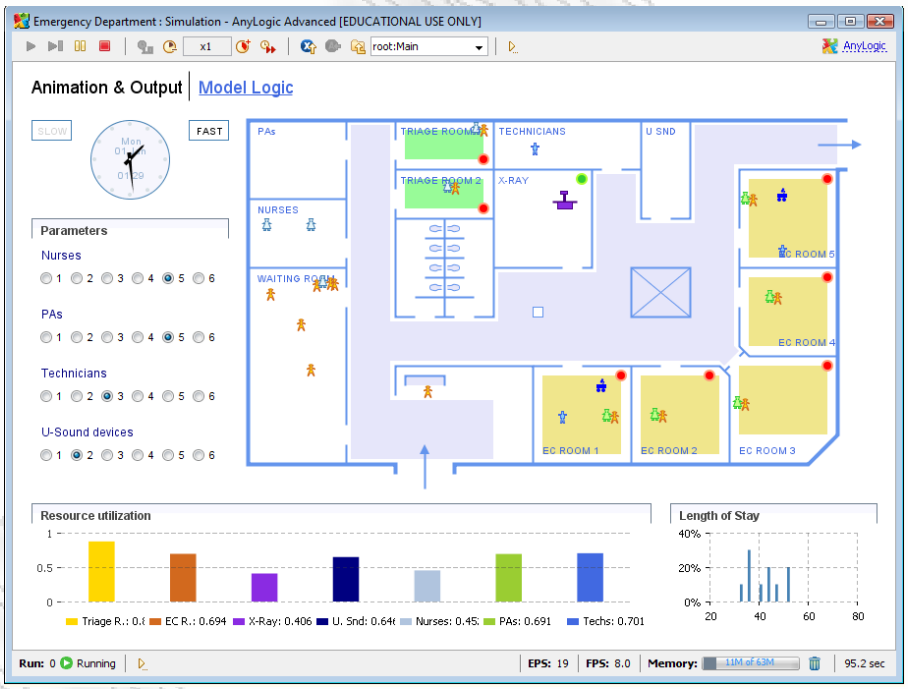
Εικόνα 8-74(α): Αρχικό παράθυρο της προσομοίωσης του έργου.



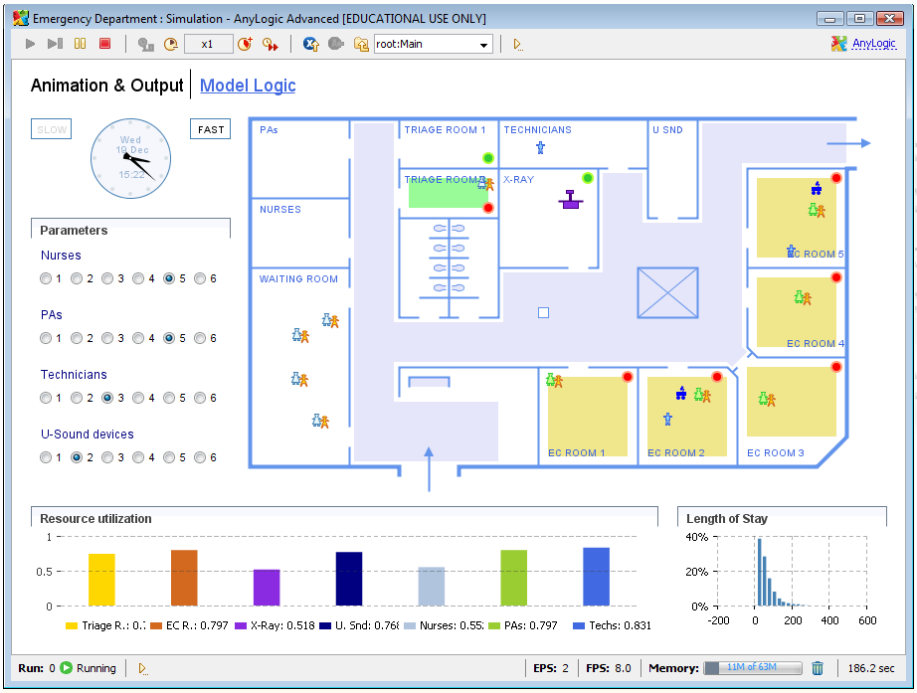
Εικόνα 8-74(β): Στιγμιότυπο από την προσομοίωση του έργου.



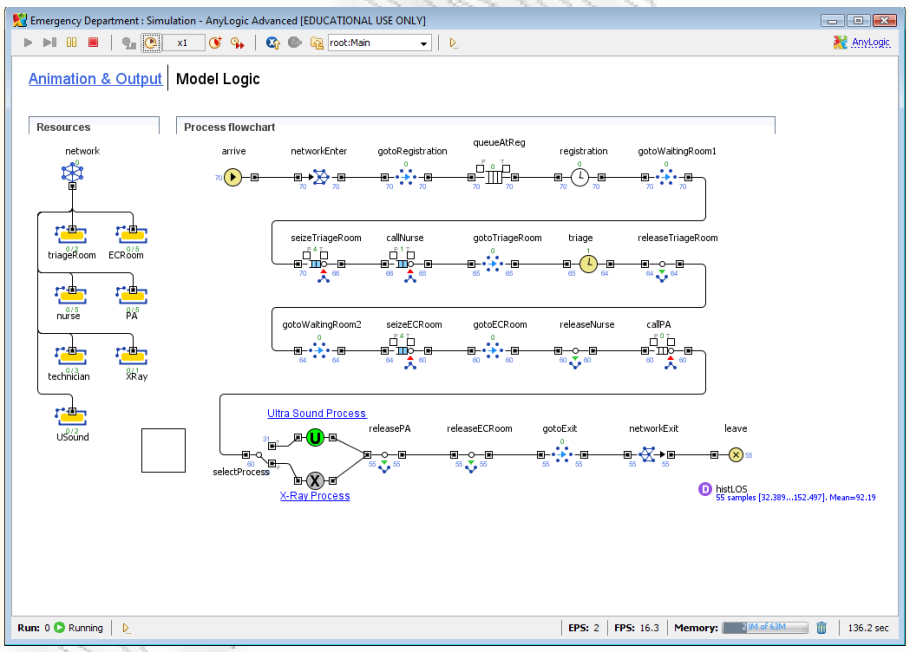
Εικόνα 8-74(γ): Στιγμιότυπο από την προσομοίωση του έργου.



Εικόνα 8-74(δ): Στιγμιότυπο από την προσομοίωση του έργου.



Εικόνα 8-74(ε): Στιγμιότυπο από την προσομοίωση του έργου.



Εικόνα 8-74(στ): Στιγμιότυπο από την προσομοίωση του έργου – Λογικό μοντέλο.

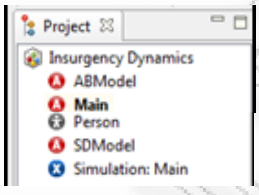
Τα διαγράμματα κλάσεων και ο πλήρης κώδικας του παραδείγματος βρίσκονται στο Παράρτημα 6.

ΚΕΦΑΛΑΙΟ 9

Μοντελοποίηση και προσομοίωση μιας κοινωνικής εξέγερσης

Το παράδειγμα που θα μας απασχολήσει στη συνέχεια, είναι αντικείμενο έρευνας και μελέτης της κοινωνικής δυναμικής, (Social Dynamics), ενός τομέα της κοινωνιολογίας ο οποίος ασχολείται με τη μελέτη της συμπεριφοράς ομάδων ατόμων. Αυτή η συμπεριφορά των ομάδων προκύπτει από τις εσωτερικές αλληλεπιδράσεις των μεμονωμένων μελών της κάθε ομάδας εντός και εκτός της ομάδας αλλά και από τις σχέσεις, τις εξαρτήσεις και τις επιδράσεις ξεχωριστών ομάδων μεταξύ των. Βασική παραδοχή της κοινωνικής δυναμικής είναι το ότι κάθε άτομο επηρεάζεται από τη συμπεριφορά του άλλου ή των άλλων και ενεργεί ανάλογα.

Το μοντέλο που θα εξετάσουμε, επιδεικνύει τις δυνατότητες του Λογισμικού στην κατασκευή μοντέλων για την κατανόηση της δυναμικής συμπεριφοράς των κοινωνικών συμπεριφορών, την εξερεύνηση των καταστάσεων κοινωνικής αναστάτωσης και συγκεκριμένα προσομοίωση μιας κοινωνικής εξέγερσης. Ως εξέγερση ή στάση (insurgency) γενικότερα ορίζεται η ένοπλη ανταρσία που προέρχεται από ένοπλη μη αναγνωρισμένη δύναμη ή ομάδα η οποία εγείρει αξιώσεις εξουσίας πάνω στην καθιερωμένη αρχή, κυβέρνηση ή διοίκηση. Στις μέρες μας, πρακτικά η έννοια της εξέγερσης έχει αποκτήσει διαφορετικό νόημα διότι συχνά εμπλέκεται η χρήση βίας με τις ασύμμετρες απειλές που εκτός από την ανταρσία ενδεικτικά περιλαμβάνουν τρομοκρατία, εκφοβισμό, δολιοφθορά και παρενόχληση. Αυτοί που συμμετέχουν στην εξέγερση είναι οι αντάρτες ή στασιαστές, (insurgents), οι οποίοι συνήθως είναι αντίθετοι με την αστική αρχή ή την κυβέρνηση τους και κατά κύριο λόγο δρουν με στόχο την αλλαγή μιας πολιτικής, οικονομικής, θρησκευτικής ή ιδεολογικής κατάστασης. Σε αντίθεση με τους στασιαστές που χρησιμοποιούν βία για την επίτευξη των σκοπών τους, μια άλλη κοινωνική ομάδα, οι διαφωνούντες, (dissidents), για την επίτευξη των στόχων τους δεν χρησιμοποιούν οποιασδήποτε μορφής βία.



Εικόνα 9-1: Η ιεραρχία του έργου Insurgency Dynamics.

Στο παράδειγμα μας, ο πληθυσμός μιας κοινωνίας χωρίζεται σε τέσσερις (4) ομάδες, στους υποστηριχτές της κυβέρνησης, (Government Supporters), στους διαφωνούντες, (Dissidents), στους αντάρτες, (Insurgents) και στους «απομακρυσμένους» αντάρτες (Removed Insurgents), αυτούς που η κυβέρνηση κατάφερε να εξαλείψει κατά την προσπάθεια της να μειώσει τον αριθμό των ανταρτών.

Οι διαφωνούντες και οι αντάρτες επικοινωνούν με τους υποστηρικτές της κυβέρνησης με σκοπό τον προσυλιτισμό τους. Η διαδικασία μετατροπής των υποστηρικτών της κυβέρνησης σε αντιφρονούντες δεν είναι απόλυτη. Κάποιοι, ένα ποσοστό, μετατρέπονται σε διαφωνούντες. Βέβαια, δεν είναι όλοι οι διαφωνούντες κατ΄ ανάγκη αντάρτες. Ένα κλάσμα των αντιφρονούντων επιστρέφει στην υποστήριξη της κυβέρνησης μετά από την άσκηση κυβερνητικής πολιτικής κατευνασμού του πληθυσμού. Το υπόλοιπο μέρος των αντιφρονούντων γίνονται τελικά αντάρτες. Η κυβέρνηση προσπαθεί να μειώσει το μέγεθος της ομάδας των ανταρτών, η αντοχή των οποίων εξαρτάται από τους πόρους που διατίθενται από την κυβέρνηση και είναι ανάλογη με το μέγεθος της ομάδας ανταρτών.

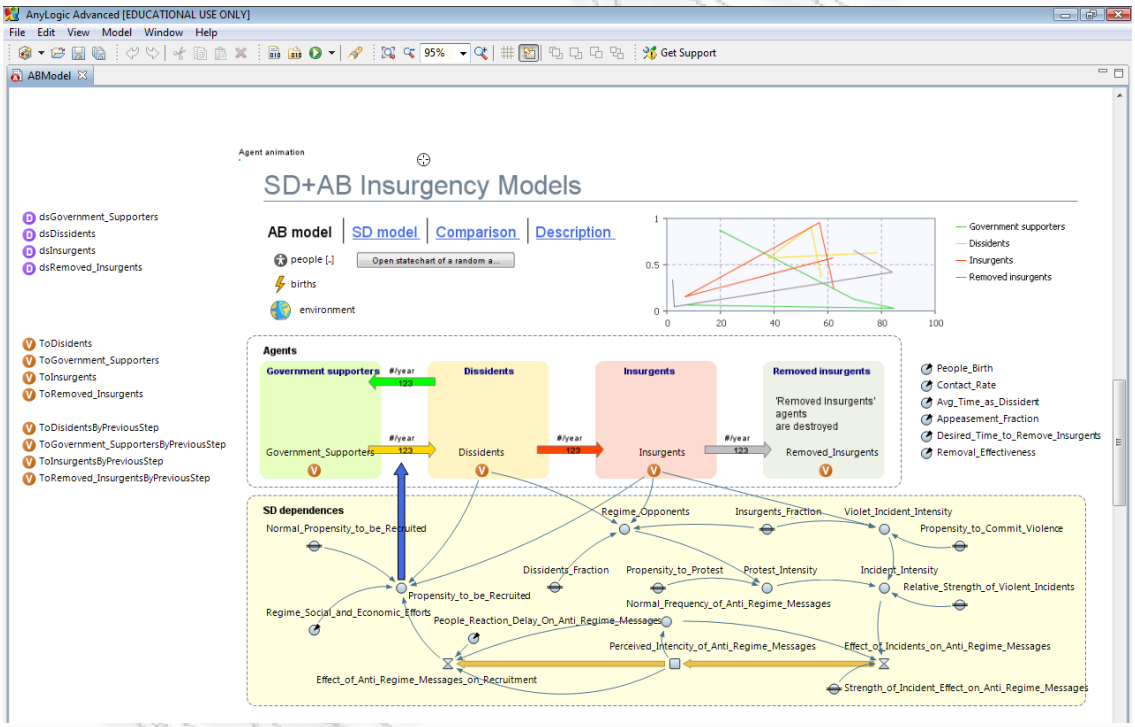
Το παράδειγμα έχει ενδιαφέρον διότι, όπως φαίνεται και από την εικόνα 9-1 που παρουσιάζεται η δομή του έργου Insurgency Dynamics, εφαρμόζονται δύο (2) διαφορετικά μοντέλα που τρέχουν παράλληλα και συγκρίνονται σε πραγματικό χρόνο. Έχουμε, (α) το υβριδικό μοντέλο, υλοποιείται μέσω της κλάσης ABModel, όπου η μοντελοποίηση γίνεται με τη χρήση πρακτόρων

λογισμικού (AB) σε συνδυασμό με την εφαρμογή συστημικής δυναμικής (SD) και (β) το μοντέλο που υλοποιείται αποκλειστικά μέσω της συστημικής δυναμικής, κλάση SDModel.

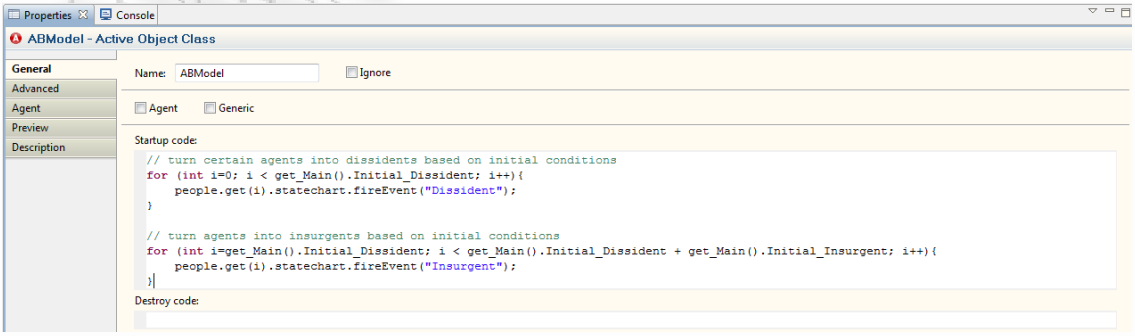
9.1 Η κλάση ABModel

Η δομή της κλάσης ABModel, εικόνα 9-2(α), αποτελείται από δύο βασικά τμήματα, το τμήμα των πρακτόρων (Agents) και το τμήμα των εξαρτήσεων (SD dependencies) που είναι το τμήμα του μοντέλου που υλοποιείται μέσω της συστημικής δυναμικής. Το τμήμα των εξαρτήσεων σε συνδυασμό με τις τιμές των παραμέτρων της κλάσης, μεταβάλουν τη συμπεριφορά των πρακτόρων κατά την προσομοίωση. Κάθε πράκτορας λογισμικού αντιστοιχεί σε ένα άτομο και η επικοινωνία μεταξύ των πρακτόρων επιτυγχάνεται με την αποστολή και τη λήψη μηνυμάτων, μέσω του εσωτερικού μηχανισμού του Λογισμικού. Στις εικόνες 9-2(β) και 9-2(γ) παρουσιάζονται οι γενικές και οι προχωρημένες ιδιότητες της κλάσης αντίστοιχα.

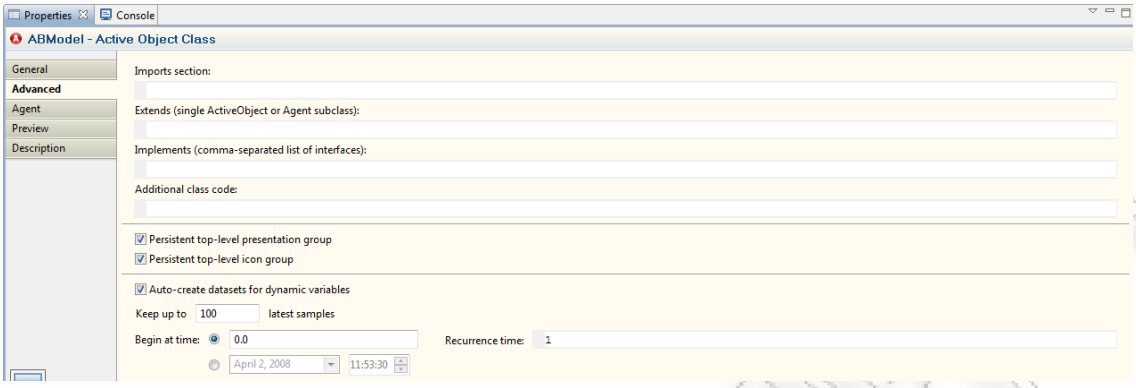
Το τμήμα των εξαρτήσεων συγκροτείται από δύο (2) ροές, ένα (1) σημείο συσσώρευσης, δύο (2) παραμέτρους, επτά (7) βοηθητικές παραμέτρους και πέντε (5) βοηθητικές μεταβλητές. Το τμήμα των πρακτόρων λογισμικού, αποτελείται από έξι (6) παραμέτρους, δώδεκα (12) μεταβλητές, τέσσερα (4) σύνολα δεδομένων, ένα (1) γράφημα, ένα (1) κόμβιο, ένα (1) απλό γεγονός, ένα (1) σύνολο πρακτόρων, ένα (1) περιβάλλον, ένα (1) στοιχείο πράκτορα και ένα (1) στοιχείο ομαδοποίησης μαζί με τα αντικείμενα που το αποτελούν.



Εικόνα 9-2(α): Η κλάση ABModel.

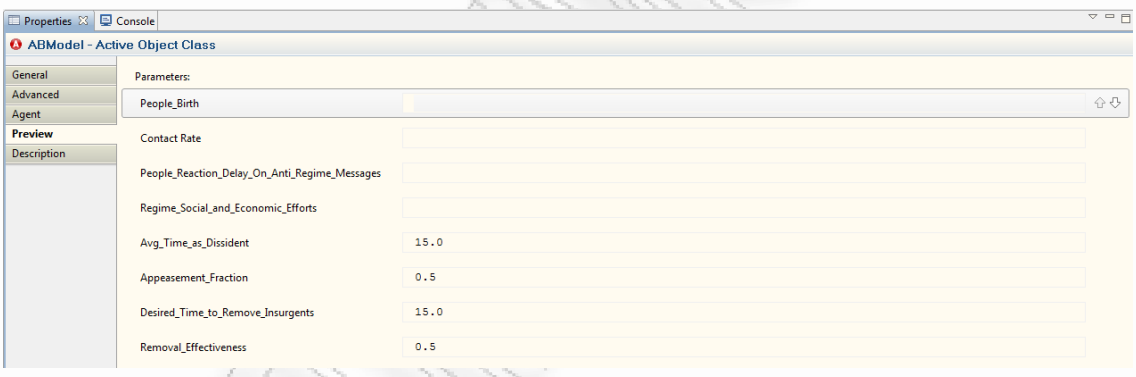


Εικόνα 9-2(β): Γενικές ιδιότητες της κλάσης ABModel.

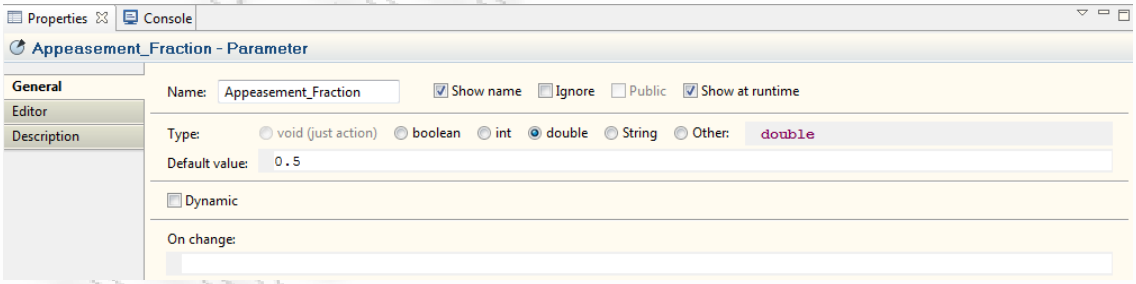


Εικόνα 9-2(γ): Προχωρημένες ιδιότητες της κλάσης ABModel.

Στην εικόνα 9-2(δ) γίνεται προεπισκόπηση των παραμέτρων της κλάσης και στις εικόνες 9-3(α) έως και 9-3(στ) παρουσιάζονται οι ιδιότητες των παραμέτρων. Η παράμετρος People_Birth καθορίζει τις γεννήσεις με βάση τον πληθυσμό των κυβερνητικών υποστηρικτών. Η παράμετρος Contact_Rate δίνει το ρυθμό επαφής των στασιαστών με τους κυβερνητικούς υποστηρικτές. Η παράμετρος Appeasement_Fraction ορίζει το κλάσμα κατευνασμού, το ποσοστό χειραγώγησης των πολιτών από την κεντρική εξουσία. Η παράμετρος Avg_Time_as_Dissident ορίζει το μέσο χρόνο που ένας πολίτης βρίσκεται στην κατάσταση του διαφωνούντα. Η παράμετρος Desired_Time_to_Remove_Insurgents καθορίζει τον επιθυμητό χρόνο μέσα στον οποίο πρέπει να γίνεται η εξάλειψη των στασιαστών ενώ η παράμετρος Removal_Effectiveness καθορίζει την αποτελεσματικότητα απομάκρυνσης τους.

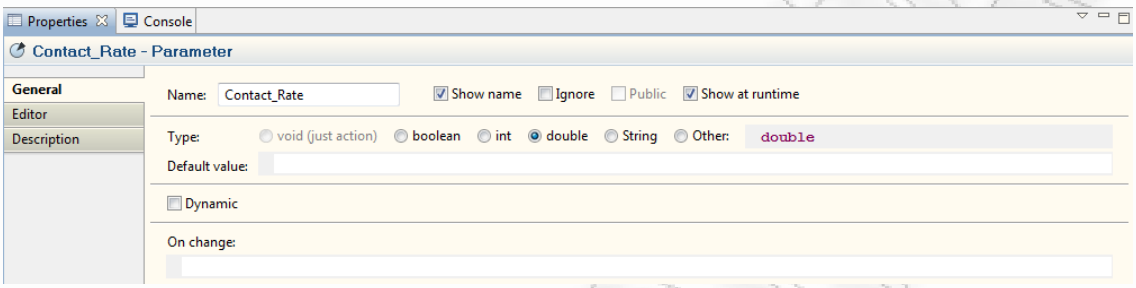


Εικόνα 9-2(δ): Προεπισκόπηση των παραμέτρων της κλάσης ABModel.

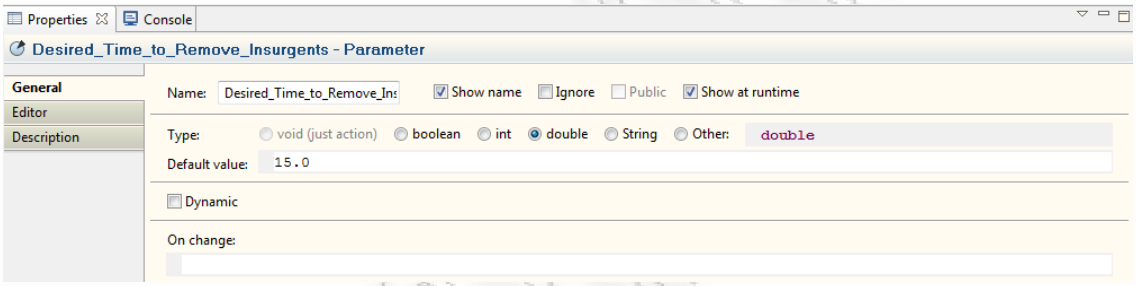


Εικόνα 9-3(α): Γενικές ιδιότητες της παραμέτρου Appeasement_Fraction.

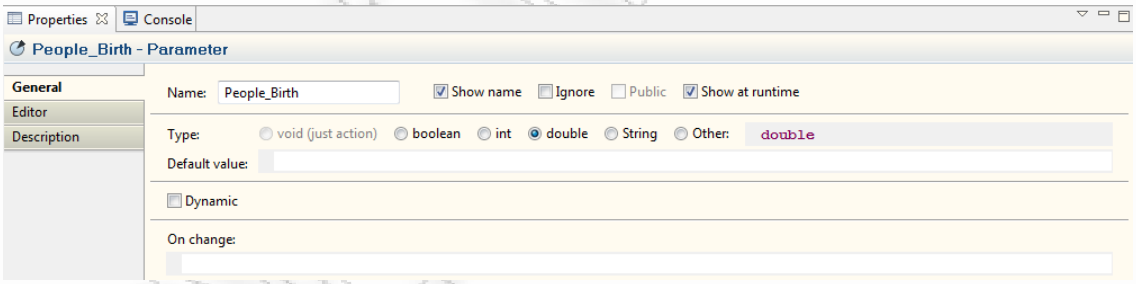
Εικόνα 9-3(β): Γενικές ιδιότητες της παραμέτρου Avg_Time_as_Dissident.



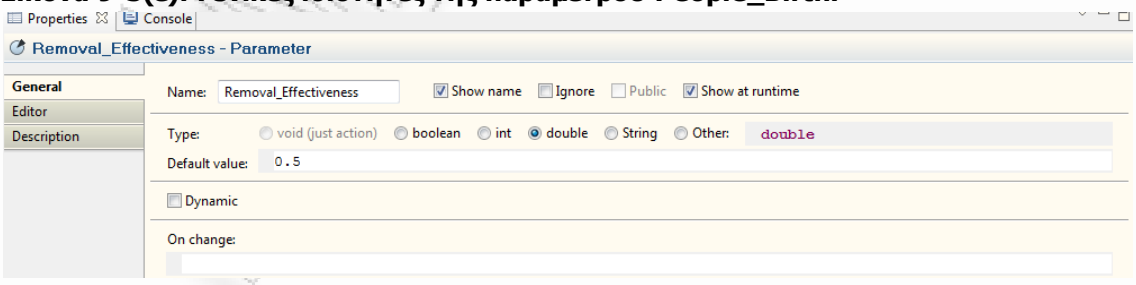
Εικόνα 9-3(γ): Γενικές ιδιότητες της παραμέτρου Contact_Rate.



Εικόνα 9-3(δ): Γενικές ιδιότητες της παραμέτρου Desired_Time_to_Remove_Insurgents.



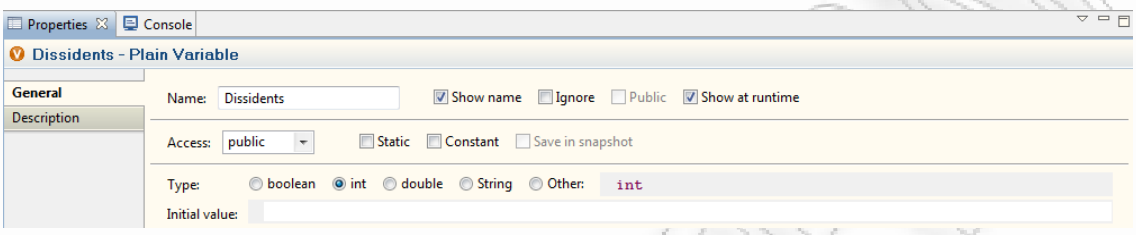
Εικόνα 9-3(ε): Γενικές ιδιότητες της παραμέτρου People_Birth.



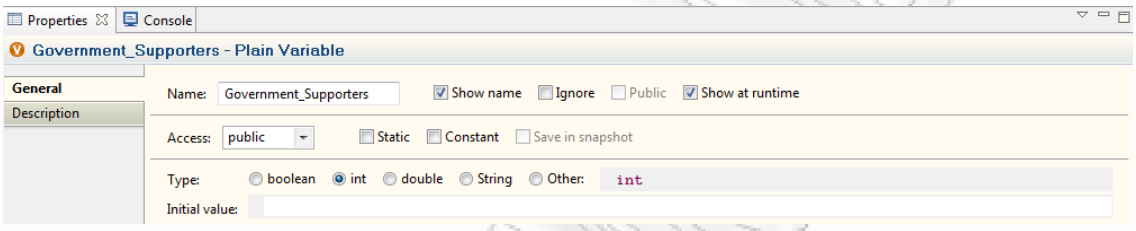
Εικόνα 9-3(στ): Γενικές ιδιότητες της παραμέτρου Removal_Effectiveness.

Στις εικόνες 9-4(α) έως και 9-4(ια) παρουσιάζονται οι γενικές ιδιότητες των απλών μεταβλητών που ορίζονται στην κλάση. Καθώς ορίζονται στο μοντέλο τέσσερις (4) ομάδες του πληθυσμού, οι μεταβλητές χωρίζονται σε τρεις ομάδες: (α) ορίζονται οι απλές μεταβλητές στις οποίες καταχωρούνται ο αριθμός των τεσσάρων ομάδων πληθυσμού Government_Supporters,

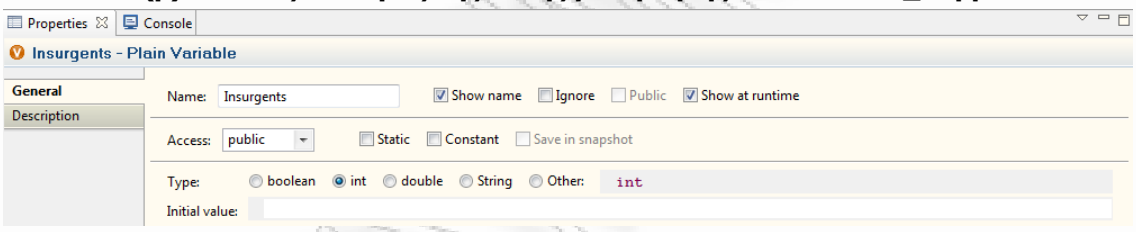
Dissidents, Insurgents και Removed_Insurgents, (β) ορίζονται οι απλές μεταβλητές ToGovernment_Supporters, ToDissidents, ToInsurgents, ToRemoved_Insurgents στις οποίες καταχωρούνται οι τιμές του πλήθους των ατόμων που αλλάζουν ομάδα υποστήριξης αντίστοιχα και (γ) ορίζονται οι απλές μεταβλητές ToGovernment_SupportersByPreviousStep, ToDissidentsByPreviousStep, ToInsurgentsByPreviousStep και ToRemovedInsurgentsByPreviousStep που εκφράζουν αντίστοιχα τον αριθμό των υποστηρικτών των ομάδων στο προηγούμενο στάδιο της προσομοίωσης.



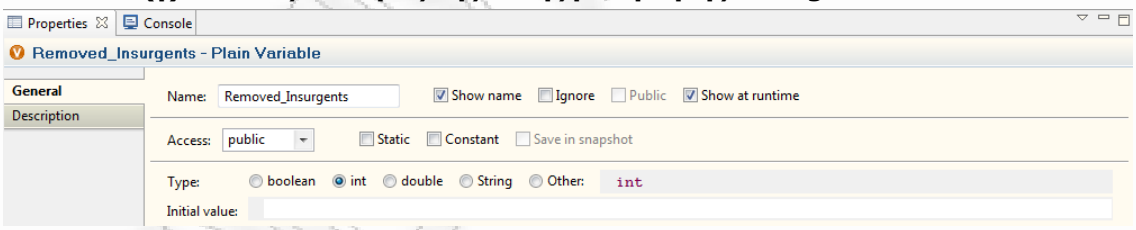
Εικόνα 9-4(α): Γενικές ιδιότητες της απλής μεταβλητής Dissidents.



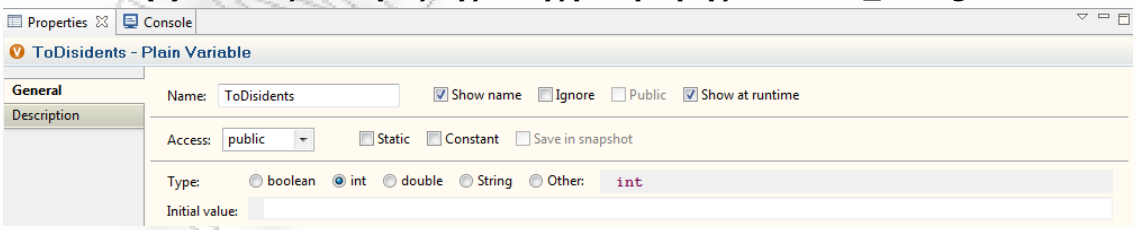
Εικόνα 9-4(β): Γενικές ιδιότητες της απλής μεταβλητής Government_Supporters.



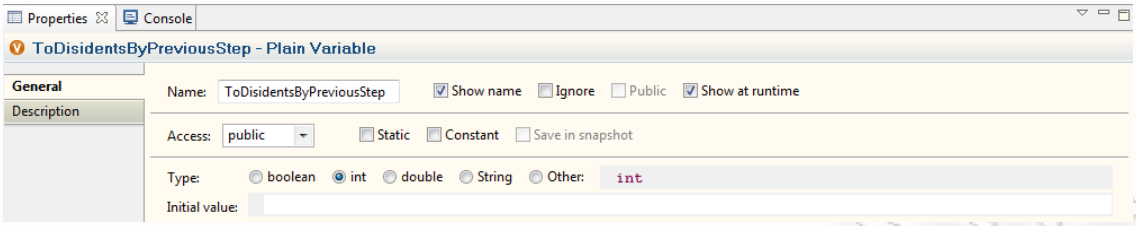
Εικόνα 9-4(γ): Γενικές ιδιότητες της απλής μεταβλητής Insurgents.



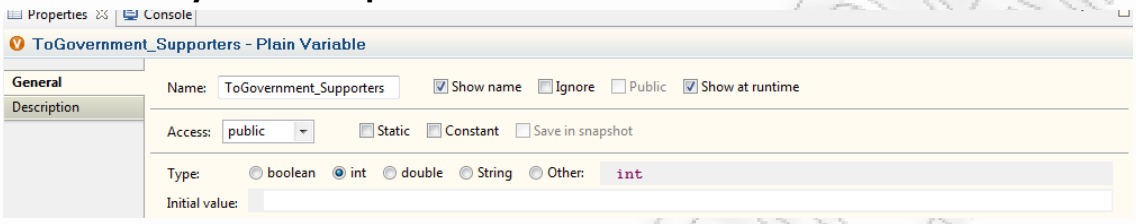
Εικόνα 9-4(δ): Γενικές ιδιότητες της απλής μεταβλητής Removed_Insurgents.



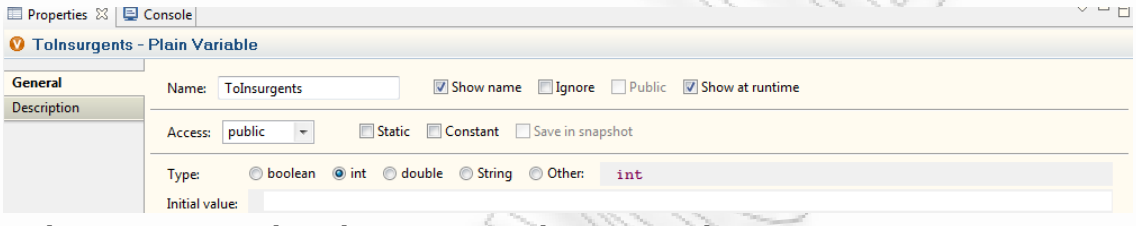
Εικόνα 9-4(ε): Γενικές ιδιότητες της απλής μεταβλητής ToDissidents.



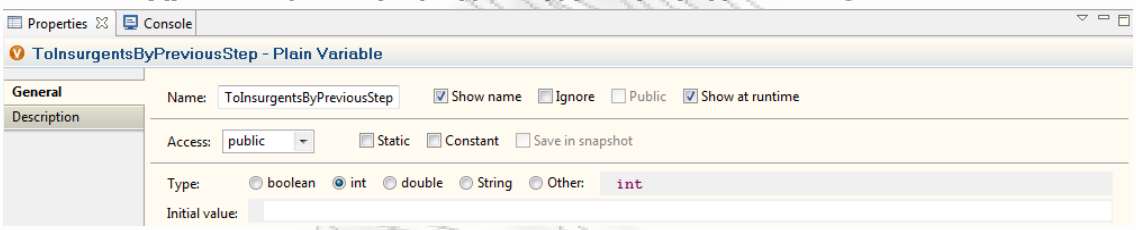
Εικόνα 9-4(στ): Γενικές ιδιότητες της απλής μεταβλητής ToDissidentsByPreviousStep.



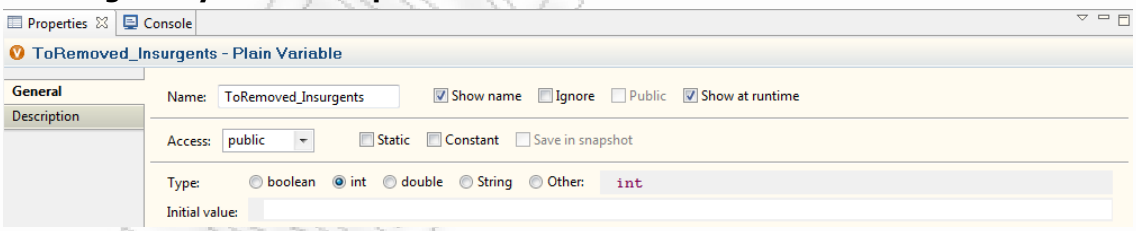
Εικόνα 9-4(ζ): Γενικές ιδιότητες της απλής μεταβλητής ToGovernment_Supporters.



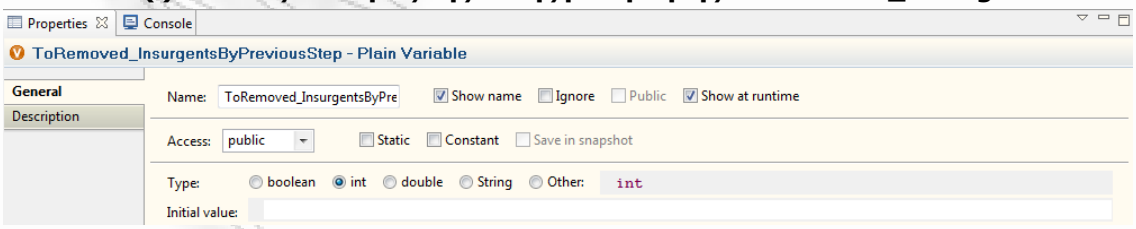
Εικόνα 9-4(η): Γενικές ιδιότητες της απλής μεταβλητής ToInsurgents.



Εικόνα 9-4(θ): Γενικές ιδιότητες της απλής μεταβλητής ToInsurgentsByPreviousStep.



Εικόνα 9-4(ι): Γενικές ιδιότητες της απλής μεταβλητής ToRemoved_Insurgents.

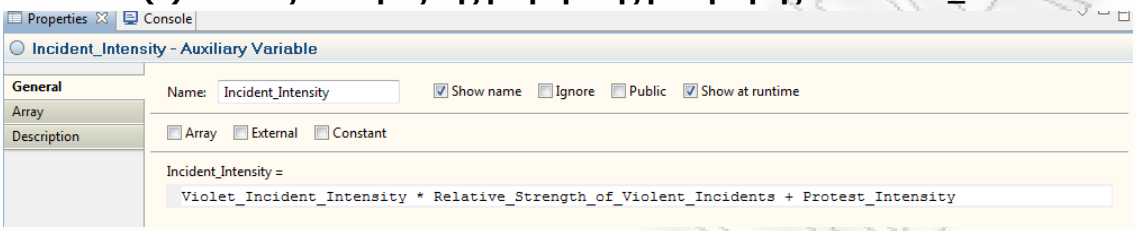


Εικόνα 9-4(ια): Γενικές ιδιότητες της απλής μεταβλητής ToRemoved_InsurgentsByPreviousStep.

Στις εικόνες 9-5(α) έως 9-5(ιε) παρουσιάζονται τις βοηθητικές μεταβλητές που χρησιμοποιούνται στο τμήμα του μοντέλου της συστημικής δυναμικής και παρατίθενται οι

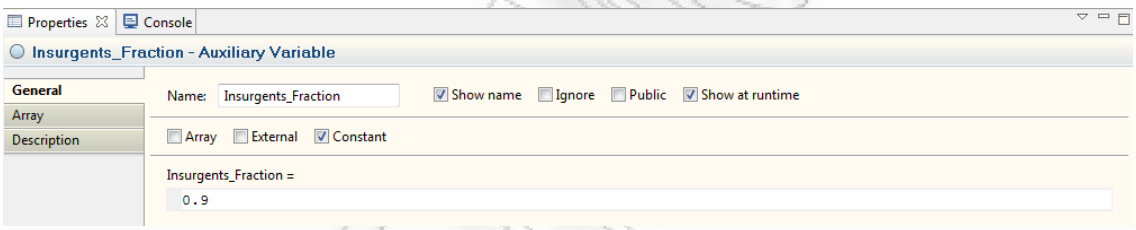
γενικές ιδιότητες τους. Σε ορισμένες περιπτώσεις οι τιμές των βοηθητικών μεταβλητών εξαρτώνται από μαθηματικές παραστάσεις ή μαθηματικές εξισώσεις τις οποίες και παραθέτουμε.

Εικόνα 9-5(α): Γενικές ιδιότητες της βοηθητικής μεταβλητής Dissidents_Fraction.

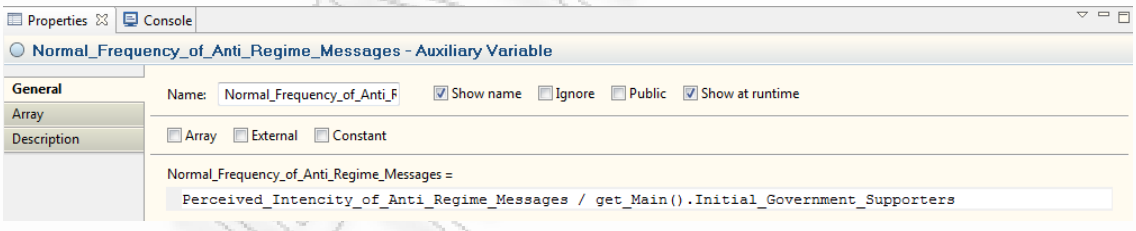


Εικόνα 9-5(β): Γενικές ιδιότητες της βοηθητικής μεταβλητής Incident_Intensity, όπου

$$Incident\_Intensity = Violet\_Incident\_Intensity * Relative\_Strength\_of\_Violent\_Incidents + Protest\_Intensity$$



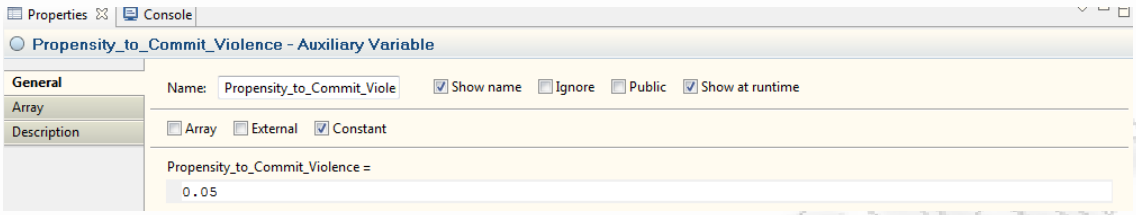
Εικόνα 9-5(γ): Γενικές ιδιότητες της βοηθητικής μεταβλητής Insurgents_Fraction.



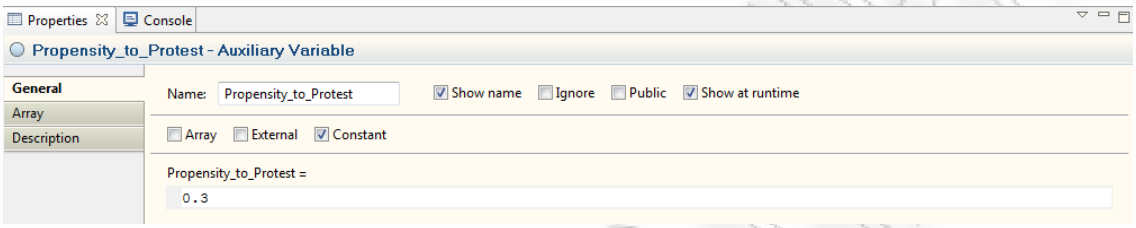
Εικόνα 9-5(δ): Γενικές ιδιότητες της βοηθητικής μεταβλητής Normal_Frequency_of_Anti_Regime_Messages, όπου

$$Normal\_Frequency\_of\_Anti\_Regime\_Messages = \frac{Perceived\_Intencity\_of\_Anti\_Regime\_Messages}{get\_Main().Initial\_Government\_Supporters}$$

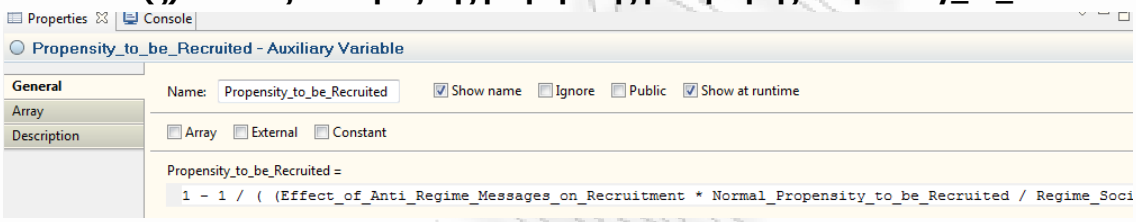
Εικόνα 9-5(ε): Γενικές ιδιότητες της βοηθητικής μεταβλητής Normal_Propensity_to_be_Recruited.



Εικόνα 9-5(στ): Γενικές ιδιότητες της βοηθητικής μεταβλητής Propensity_to_Commit_Violence.

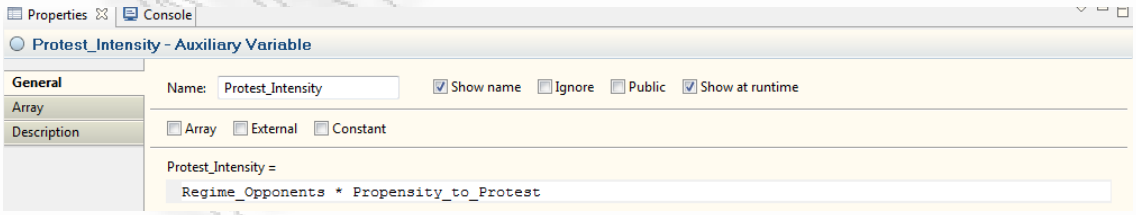


Εικόνα 9-5(ζ): Γενικές ιδιότητες της βοηθητικής μεταβλητής Propensity_to_Protest.



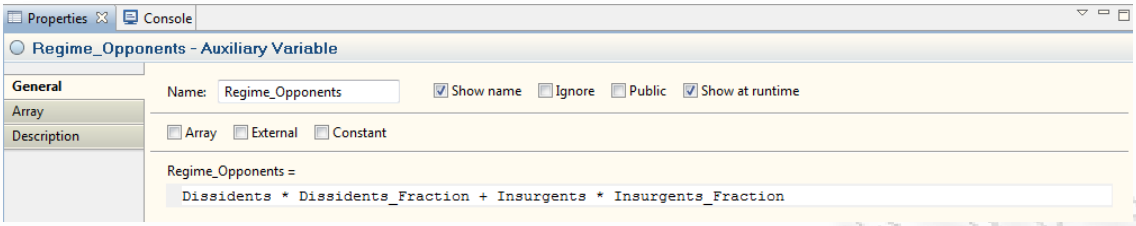
Εικόνα 9-5(η): Γενικές ιδιότητες της βοηθητικής μεταβλητής Propensity_to_be_Recruited, όπου

$$\alpha_1 = Effect\_of\_Anti\_Regime\_Messages\_on\_Recruitment$$
$$\alpha_2 = Normal\_Propensity\_to\_be\_Recruited$$
$$\alpha_3 = Regime\_Social\_and\_Economic\_Efforts$$
$$\alpha_4 = Dissidents + Insurgents$$
$$A_1 = \frac{\frac{\alpha_1 * \alpha_2}{\alpha_3}}{\alpha_4}$$
$$Propensity\_to\_be\_Recruited = 1 - \frac{1}{(A_1 * A_1 + 1)}$$



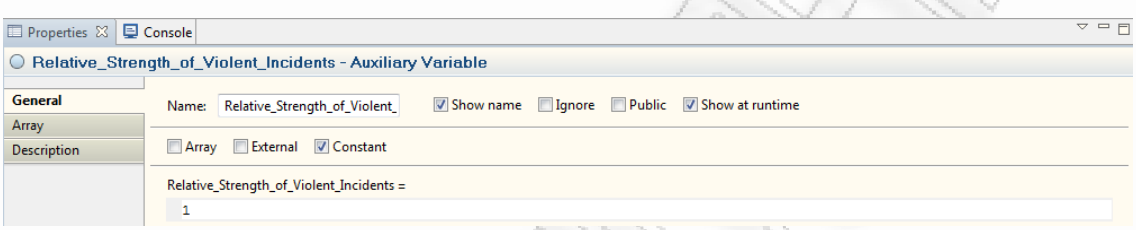
Εικόνα 9-5(θ): Γενικές ιδιότητες της βοηθητικής μεταβλητής Protest_Intensity, όπου

$$Protest\_Intensity = Regime\_Opponents * Propensity\_to\_Protest$$

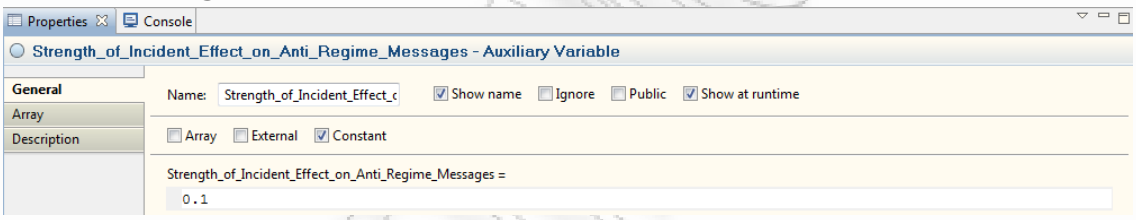


Εικόνα 9-5(ι): Γενικές ιδιότητες της βοηθητικής μεταβλητής Regime_Opponents, όπου

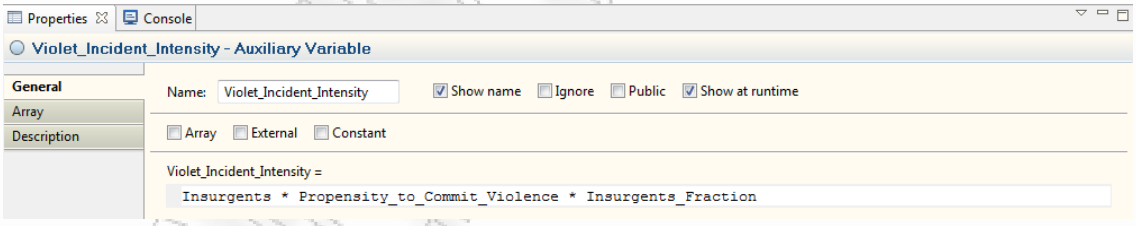
$$Regime\_Opponents = (Dissidents * Dissidents\_Fraction) + (Insurgents * Insurgents\_Fraction)$$



Εικόνα 9-5(ια): Γενικές ιδιότητες της βοηθητικής μεταβλητής Relative_Strength_of_Violent_Incidents.

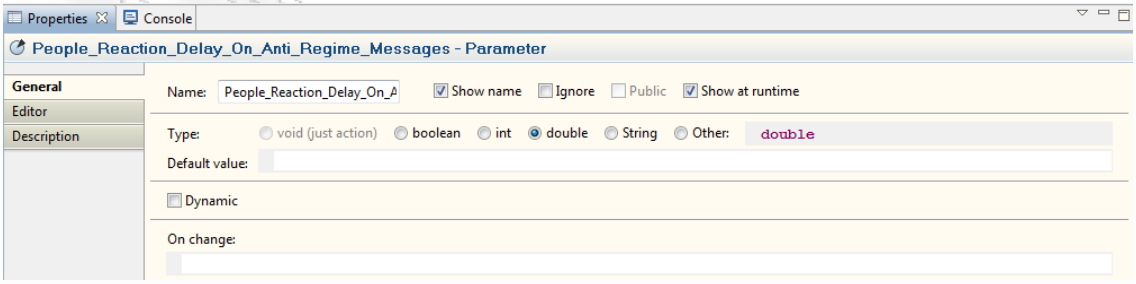


Εικόνα 9-5(ιβ): Γενικές ιδιότητες της βοηθητικής μεταβλητής Strength_of_Incident_Effect_on_Anti_Regime_Messages.

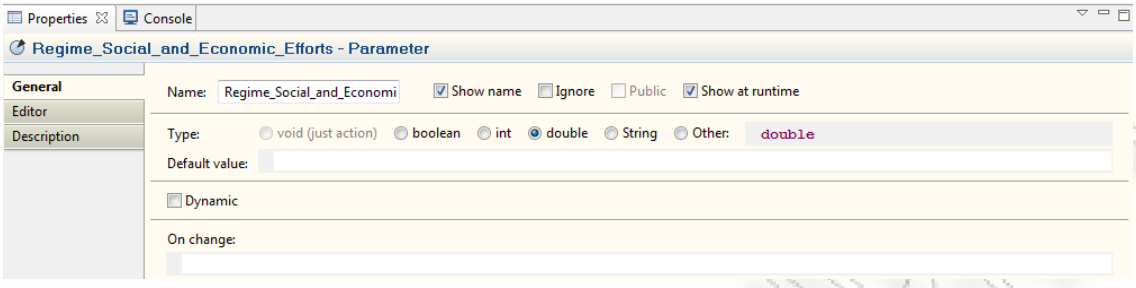


Εικόνα 9-5(ιγ): Γενικές ιδιότητες της βοηθητικής μεταβλητής Violet_Incident_Intensity, όπου

$$Violet\_Incident\_Intensity = Insurgents * Propensity\_to\_Commit\_Violence * Insurgents\_Fraction$$

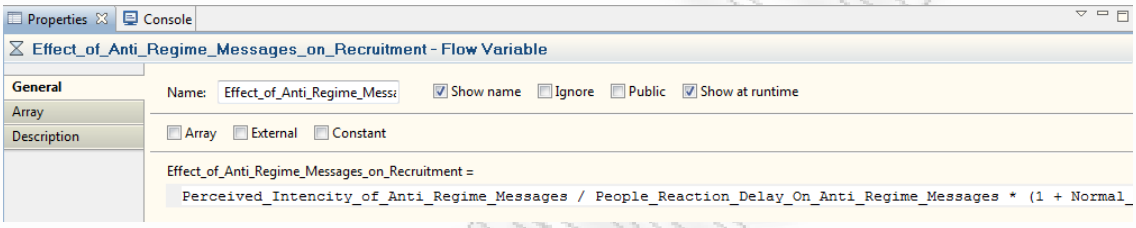


Εικόνα 9-5(ιδ): Γενικές ιδιότητες της παραμέτρου People_Reaction_Delay_On_Anti_Regime_Messages.



Εικόνα 9-5(ιε): Γενικές ιδιότητες της παραμέτρου Regime_Social_and_Economic_Efforts.

Στις εικόνες 9-6(α) και 9-6(β) παρουσιάζονται οι γενικές ιδιότητες των δύο (2) ροών και στην εικόνα 9-7 οι γενικές ιδιότητες του σημείου συσσώρευσης. Δίνονται σε κάθε περίπτωση οι μαθηματικές εξισώσεις που τις καθορίζουν.

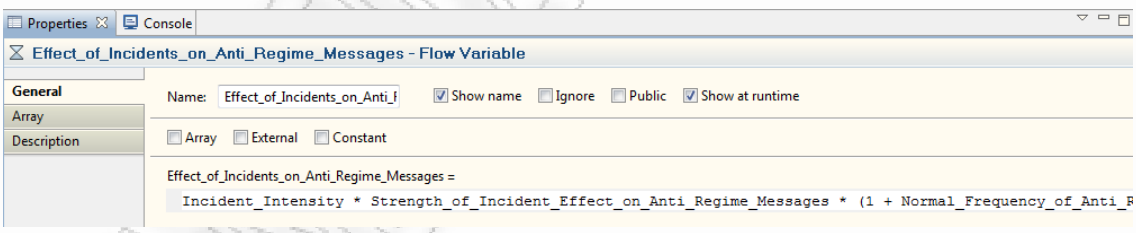


Εικόνα 9-6(α): Γενικές ιδιότητες της ροής Effect_of_Anti_Regime_Messages_on_Recruitment, όπου

$$\beta_1 = 1 + Normal\_Frequency\_of\_Anti\_Regime\_Messages$$

$$\beta_2 = \frac{Perceived\_Intencity\_of\_Anti\_Regime\_Messages}{People\_Reaction\_Delay\_On\_Anti\_Regime\_Messages}$$

$$Effect\_of\_Anti\_Regime\_Messages\_on\_Recruitment = \beta_1 + \beta_2$$

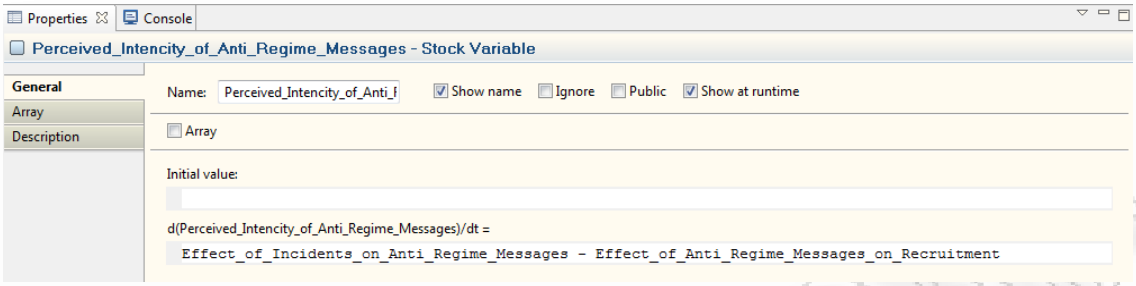


Εικόνα 9-6(β): Γενικές ιδιότητες της ροής Effect_of_Incidents_on_Anti_Regime_Messages, όπου

$$\gamma_1 = Incident\_Intensity * Strength\_of\_Incident\_Effect\_on\_Anti\_Regime\_Messages$$

$$\gamma_2 = (1 + Normal\_Frequency\_of\_Anti\_Regime\_Messages)$$

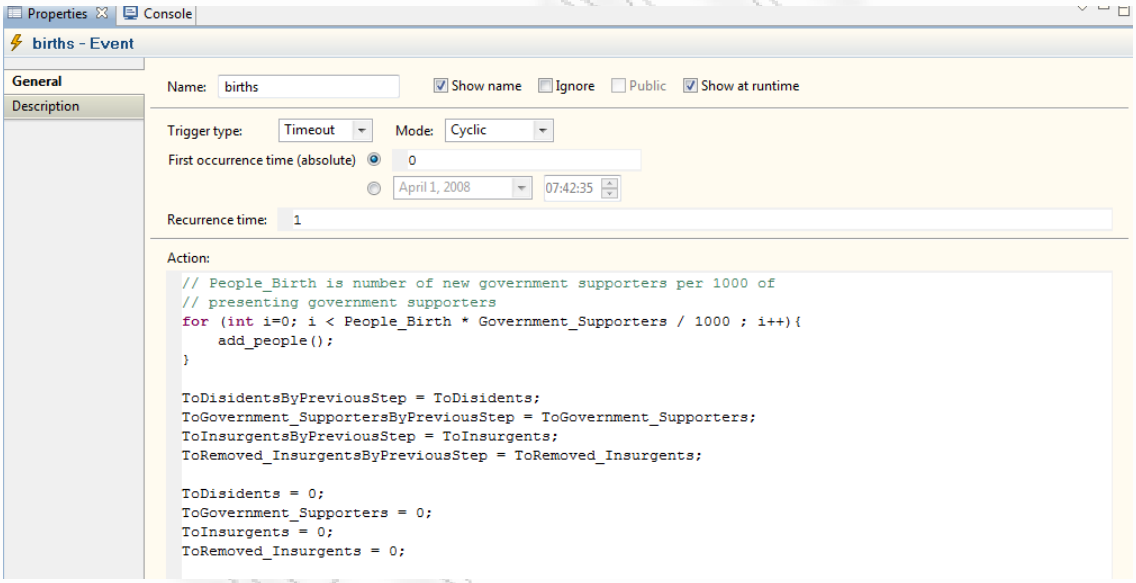
$$Effect\_of\_Incidents\_on\_Anti\_Regime\_Messages = \gamma_1 + \gamma_2$$



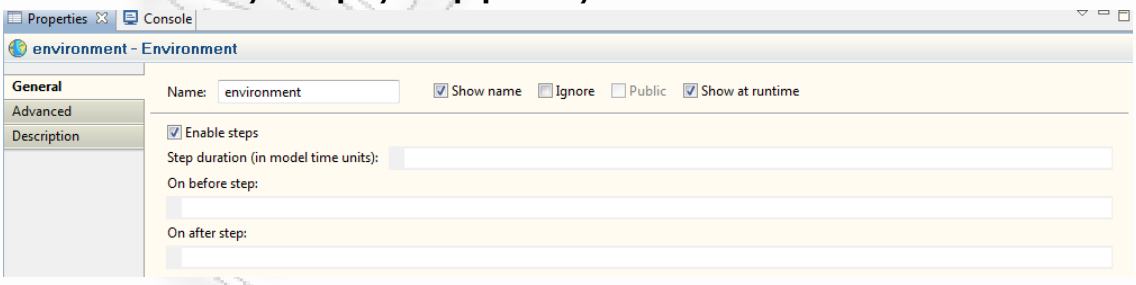
Εικόνα 9-7: Γενικές ιδιότητες του σημείου συσσώρευσης Perceived_Intencity_of_Anti_Regime_Messages, όπου

$$\frac{\mathbf{d(Perceived\_Intencity\_of\_Anti\_Regime\_Messages)}}{\mathbf{dt}} = \mathbf{Effect\_of\_Incidents\_on\_Anti\_Regime\_Messages} - -\mathbf{Effect\_of\_Anti\_Regime\_Messages\_on\_Recruitment}$$

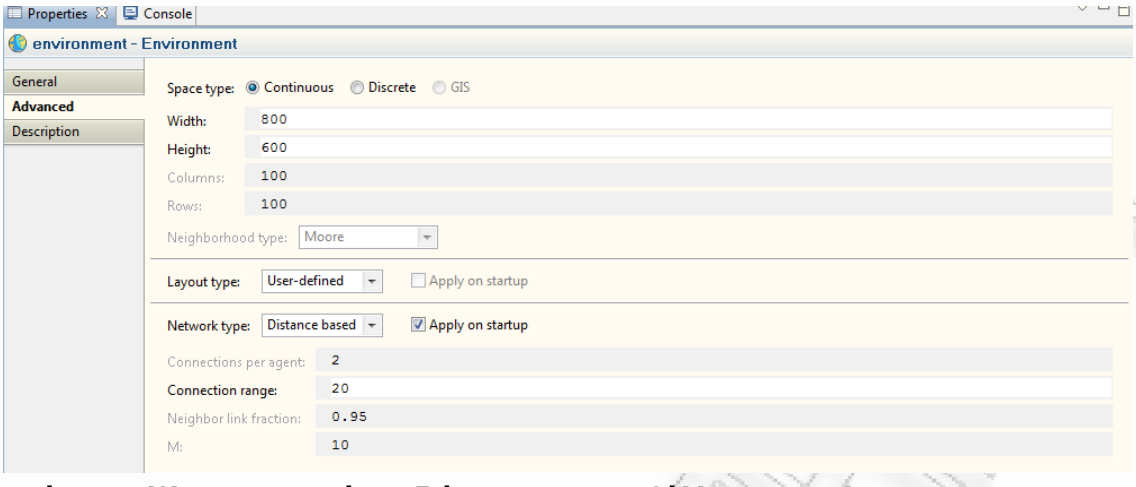
Οι εικόνες, από την 9-8 έως και την 9-13 παρουσιάζουν τις γενικές ή και τις προχωρημένες ιδιότητες των υπολοίπων στοιχείων που αποτελούν την κλάση.



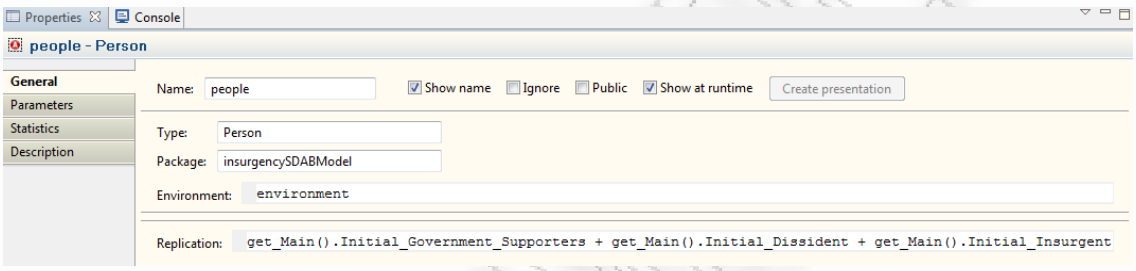
Εικόνα 9-8: Γενικές ιδιότητες του γεγονότος births.



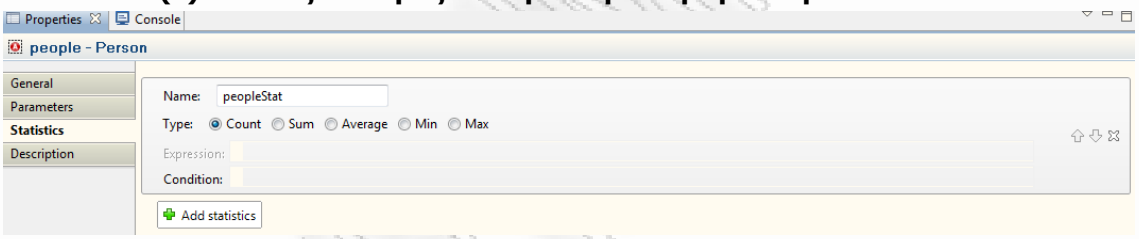
Εικόνα 9-9(α): Γενικές ιδιότητες του περιβάλλοντος environment.



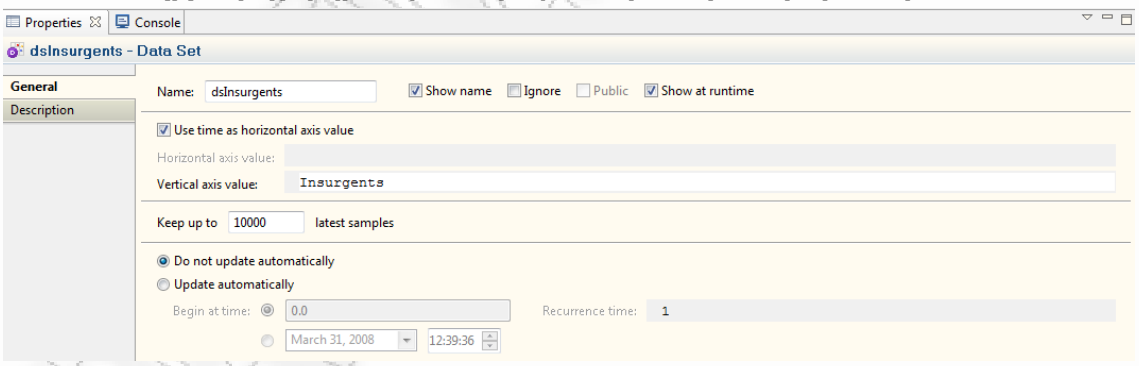
Εικόνα 9-9(β): Προχωρημένες ιδιότητες του περιβάλλοντος environment.



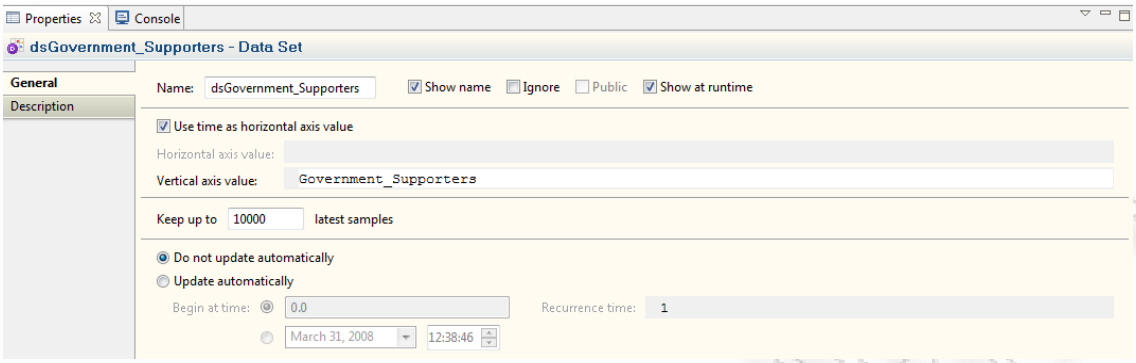
Εικόνα 9-10(α): Γενικές ιδιότητες του πράκτορα λογισμικού person.



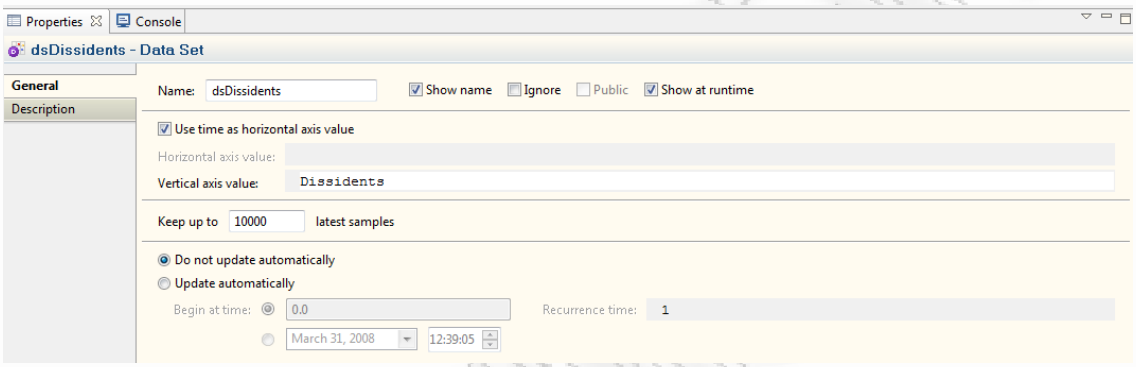
Εικόνα 9-10(β): Προχωρημένες ιδιότητες του πράκτορα λογισμικού person.



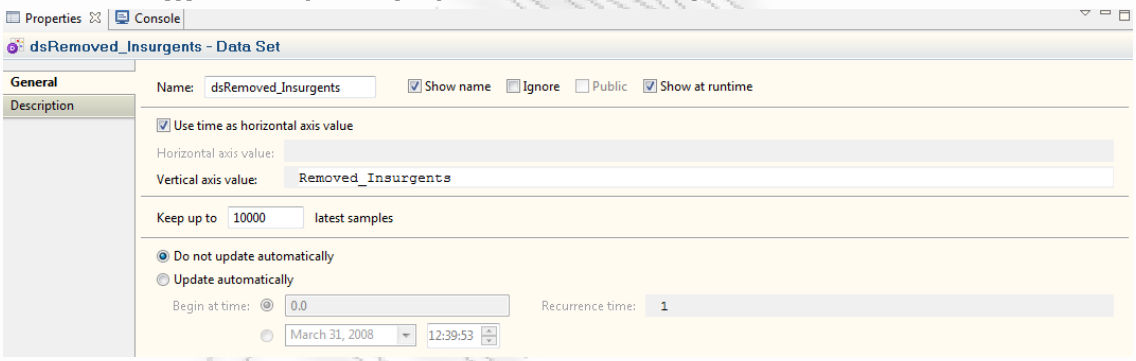
Εικόνα 9-11(α): Γενικές ιδιότητες του συνόλου δεδομένων dsInsurgents.



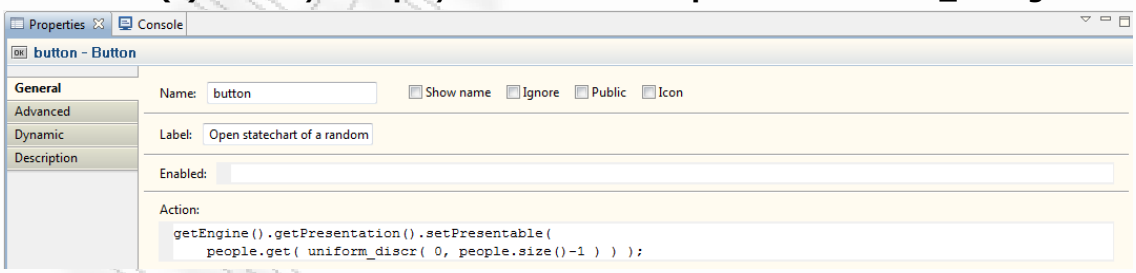
Εικόνα 9-11(β): Γενικές ιδιότητες του συνόλου δεδομένων dsGovernment_Supporters.



Εικόνα 9-11(γ): Γενικές ιδιότητες του συνόλου δεδομένων dsDissidents.

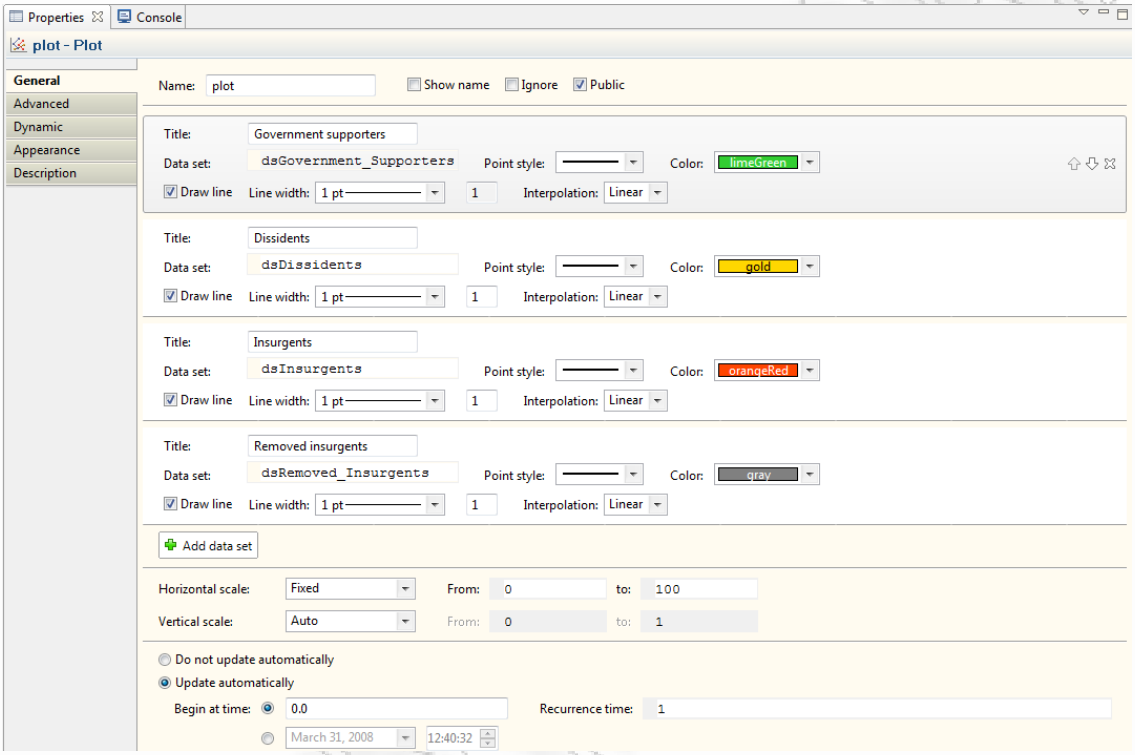


Εικόνα 9-11(δ): Γενικές ιδιότητες του συνόλου δεδομένων dsRemoved_Insurgents.



Εικόνα 9-12(α): Γενικές ιδιότητες του κομβίου button.

Εικόνα 9-12(β): Προχωρημένες ιδιότητες του κομβίου button.



Εικόνα 9-13: Γενικές ιδιότητες του διαγράμματος χρόνου plot.

Εκτός των παραπάνω, η κλάση αποτελείται επίσης από ορισμένα στοιχεία κειμένου, και από τα στοιχεία γεωμετρικών σχημάτων που αναπαριστούν τα τόξα διαφόρων χρωματισμών, που φαίνονται στην εικόνα 9-2(α), τα οποία όμως δεν τα παρουσιάζουμε για λόγους οικονομίας χώρου. Τα τόξα συνεισφέρουν στο μοντέλο διότι χρησιμοποιώντας τις δυναμικές τους ιδιότητες τα καθιστούμε ενδιάμεσους μεταφορείς πληροφορίας (α) μεταξύ των ομάδων των πρακτόρων λογισμικού και (β) μεταξύ των δύο τμημάτων των μεθοδολογιών υλοποίησης του μοντέλου.

Ενδιαφέρον τέλος έχει το μενού, εικόνα 9-14, με το οποίο μπορούμε να αλλάζουμε καρτέλες και να μεταφερόμαστε στις διαφορετικές κλάσεις του μοντέλου, μέσω των υπερσυνδέσμων AB Model, SD Model, Comparison και Description έχουμε τη δυνατότητα να μετακινούμαστε στην αντίστοιχη κλάση, κατά τη διάρκεια της προσομοίωσης και να λαμβάνουμε πληροφορίες. Όπως ισχύει και για τις υπόλοιπες κλάσεις του έργου Insurgency Dynamics, (SDModel, Main και Simulation:Main τις οποίες θα παρουσιάσουμε στη συνέχεια), το μενού αποτελείται από την ομάδα των τεσσάρων (4) υπερσυνδέσμων.

Ο κώδικας που εκτελείται στο παρασκήνιο κατά την επιλογή των υπερσυνδέσμων είναι σε κάθε περίπτωση:

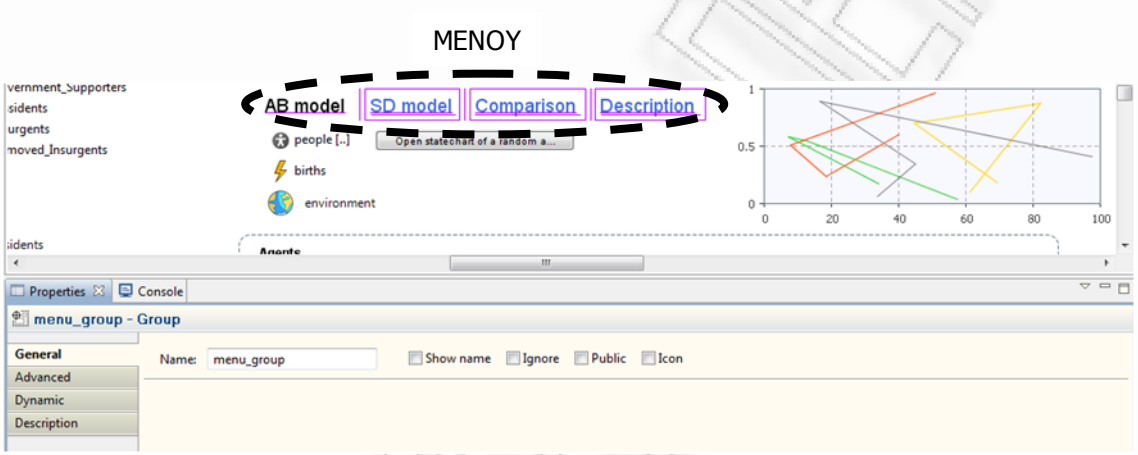
Για την κλάση ABModel:

```
getEngine().getPresentation().setPresentable(((Main)getEngine().getRoot()).aM);
getPresentation().getPanel().setOffsets(0,0);
```

Για την κλάση SDModel:
getEngine().getPresentation().setPresentable(((Main)getEngine().getRoot()).sD);
getPresentation().getPanel().setOffsets(0,0);

Για την κλάση Comparison (Main):
getEngine().getPresentation().setPresentable(((Main)getEngine().getRoot()));
getPresentation().getPanel().setOffsets(0,0);

Για την κλάση Description (Simulation:Main):
getEngine().getPresentation().setPresentable(((Main)getEngine().getRoot()));
getPresentation().getPanel().setOffsets(0,1500);

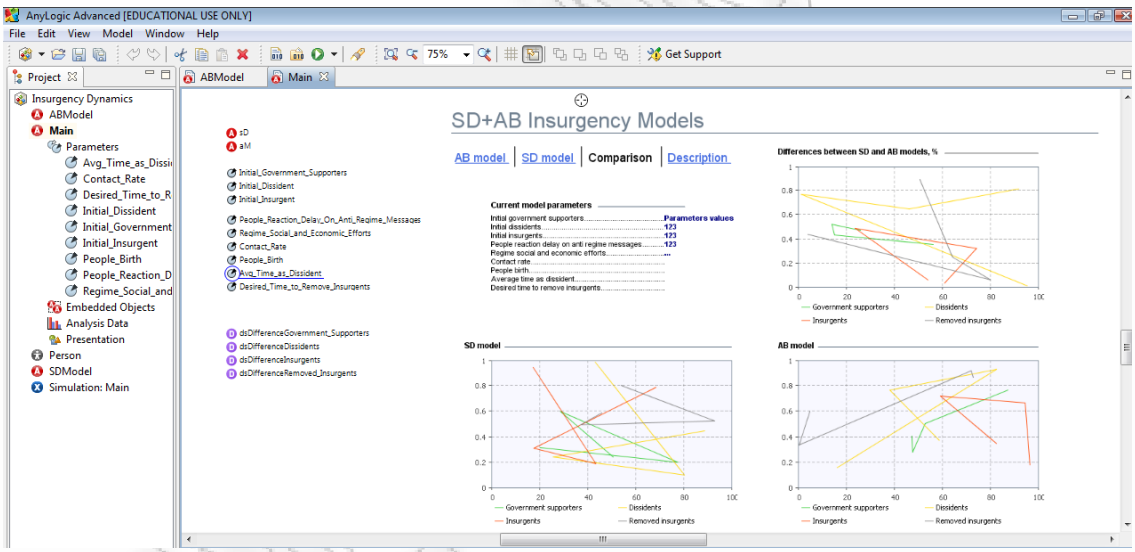


Εικόνα 9-14: Το μενού menu_group τύπου Group.

9.2 Η κλάση Main

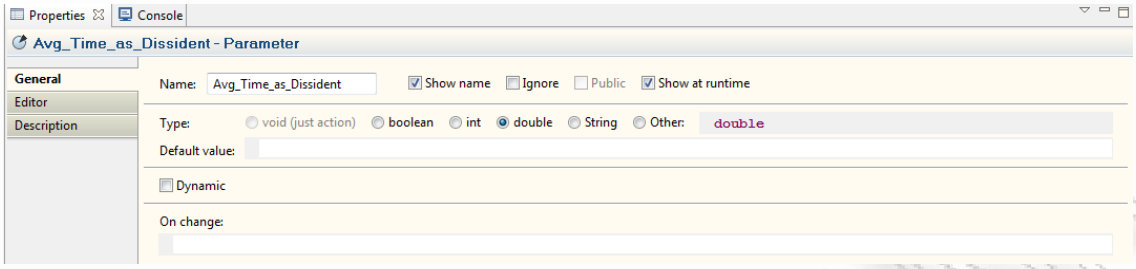
Η κλάση Main, εικόνα 9-15, είναι αυτή που παρουσιάζεται στο προσκήνιο κατά την προσομοίωση του μοντέλου στο Λογισμικό.

Αποτελείται από τα στιγμιότυπα aM και sD των κλάσεων ABModel και SDModel αντίστοιχα, από τις παραμέτρους Initial_Government_Supporters, Initial_Dissidents και Initial_Insurgents που αποδίδουν αρχικές τιμές στις αντίστοιχες ομάδες του πληθυσμού. Οι παράμετροι Contact_Rate, People_Birth, People_Reaction_Delay_On_Anti_Regime_Messages, Regime_Social_and_Economic_Efforts, Desired_Time_to_Remove_Insurgents και Avg_Time_as_Dissident ορίζονται όπως ακριβώς έχουν ήδη οριστεί στην κλάση ABModel. Η κλάση περιέχει επίσης τέσσερα (4) σύνολα δεδομένων, τα dsDifferenceGovernment_Supporters, dsDifferenceDissidents, dsDifferenceInsurgents και dsDifferenceRemoved_Insurgents στα οποία καταχωρούνται οι διαφορές των τιμών των αντίστοιχων ομάδων κατά την προσομοίωση. Η οπτικοποίηση των διαφορών των δύο (2) μοντέλων παρουσιάζεται στο διάγραμμα χρόνου «Differencies between AB and SD models» σε ποσοστό επί τοις εκατό. Επίσης, παρατίθενται μέσω δύο (2) ακόμη διαγραμμάτων χρόνου οι πληθυσμοί των κοινωνικών ομάδων όπως αυτοί λαμβάνονται από τις κλάσεις AB Model και SD Model σε ποσοστό επί τοις εκατό. Οι τιμές των παραμέτρων της κλάσης Main εμφανίζονται επίσης ως αριθμητικές τιμές κατά την προσομοίωση, σε «πραγματικό χρόνο».

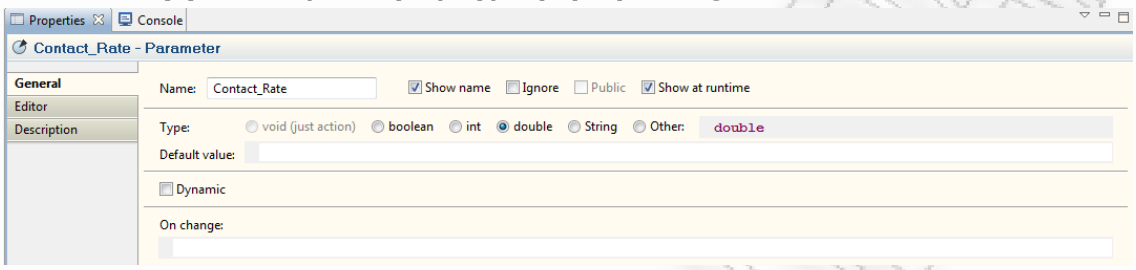


Εικόνα 9-15: Η κλάση Main.

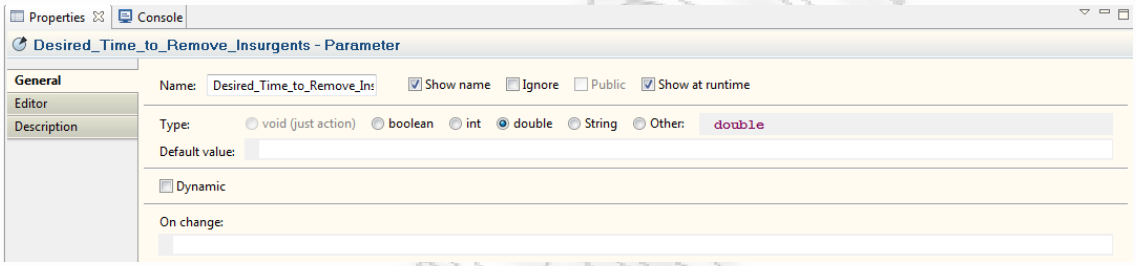
Στις εικόνες 9-16(α) έως και 9-16(θ) εμφανίζονται οι γενικές ιδιότητες των παραμέτρων που ορίζονται στην κλάση Main, ενώ στις εικόνες 9-17(α) και 9-17(β) παρουσιάζονται οι ιδιότητες και οι παράμετροι αντίστοιχα του στιγμιότυπου aM της κλάσης ABModel. Στις εικόνες 9-18(α) και 9-18(β) παρουσιάζονται οι ιδιότητες και οι παράμετροι αντίστοιχα του στιγμιότυπου sD της κλάσης SDModel.



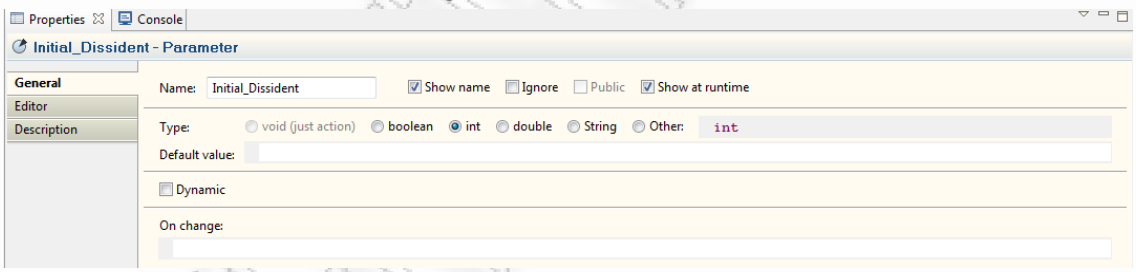
Εικόνα 9-16(α): Γενικές ιδιότητες της παραμέτρου Avg_Time_as_Dissident.



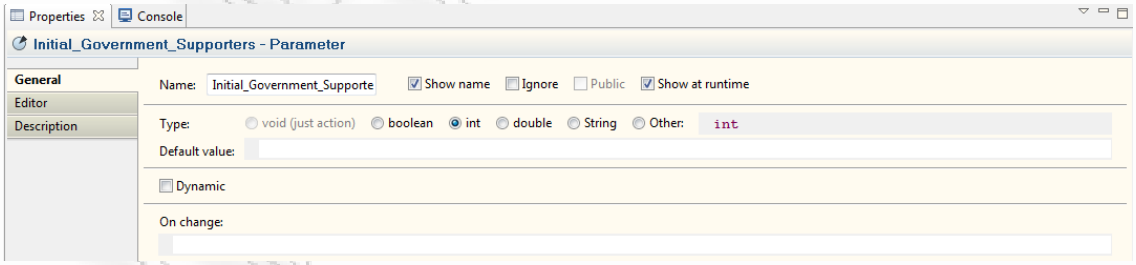
Εικόνα 9-16(β): Γενικές ιδιότητες της παραμέτρου Contact_Rate.



Εικόνα 9-16(γ): Γενικές ιδιότητες της παραμέτρου Desired_Time_to_Remove_Insurgents.

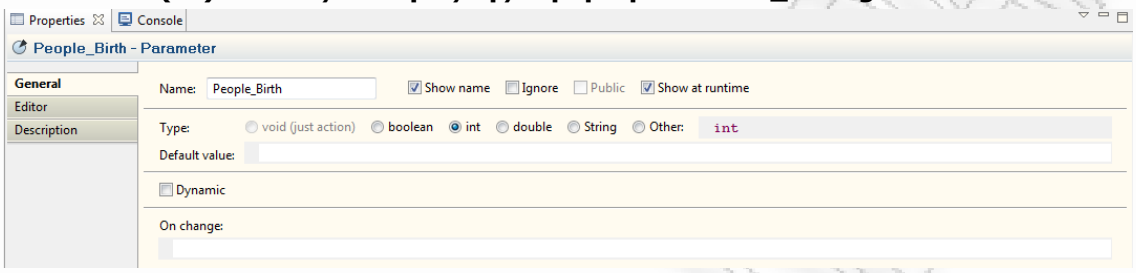


Εικόνα 9-16(δ): Γενικές ιδιότητες της παραμέτρου Initial_Dissident.

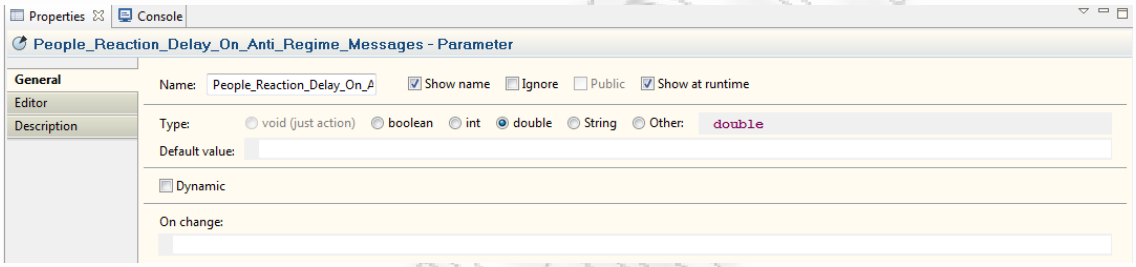


Εικόνα 9-16(ε): Γενικές ιδιότητες της παραμέτρου Initial_Government_Supporters.

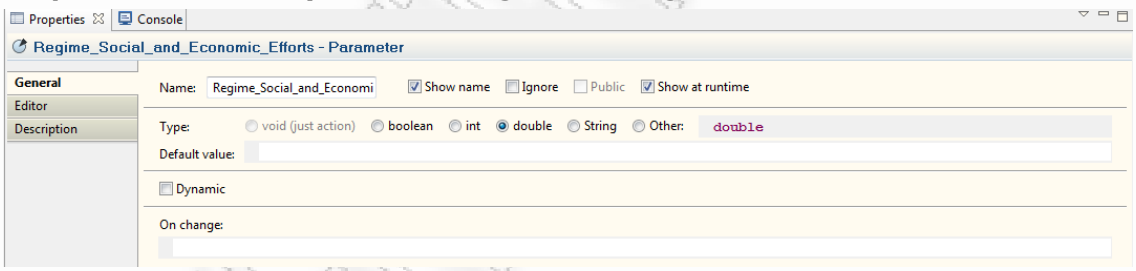
Εικόνα 9-16(στ): Γενικές ιδιότητες της παραμέτρου Initial_Insurgent.



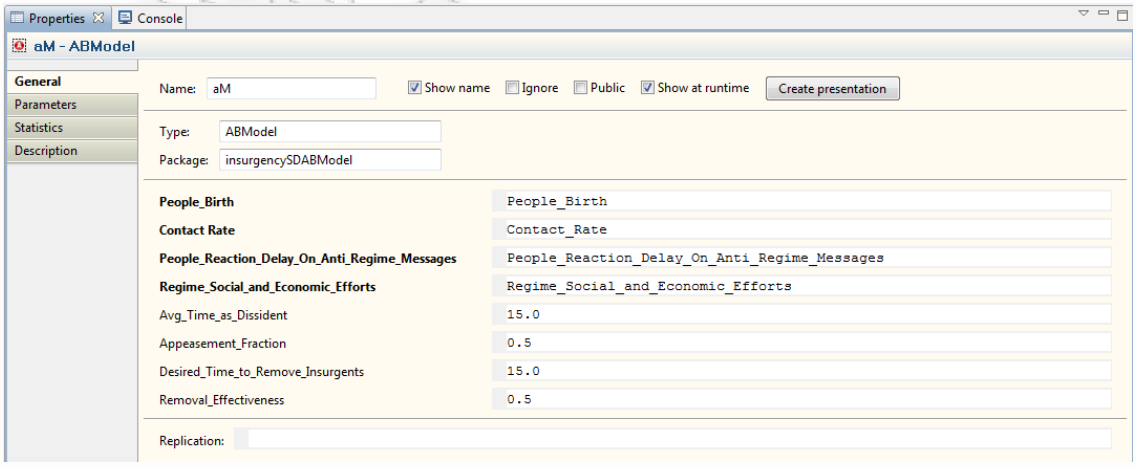
Εικόνα 9-16(ζ): Γενικές ιδιότητες της παραμέτρου People_Birth.



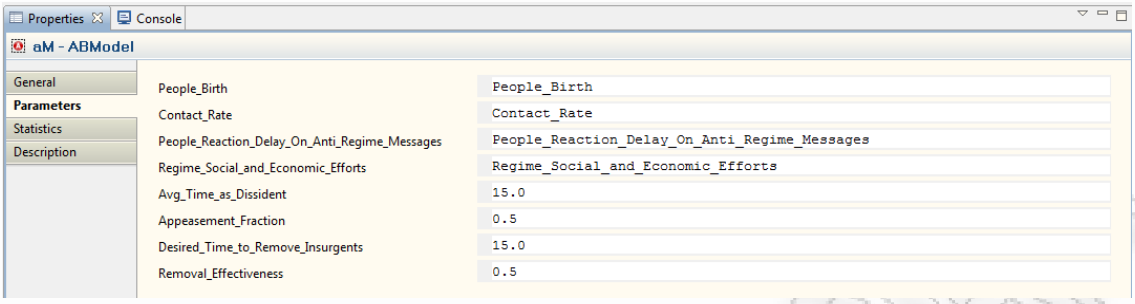
Εικόνα 9-16(η): Γενικές ιδιότητες της παραμέτρου People_Reaction_Delay_On_Anti_Regime_Messages.



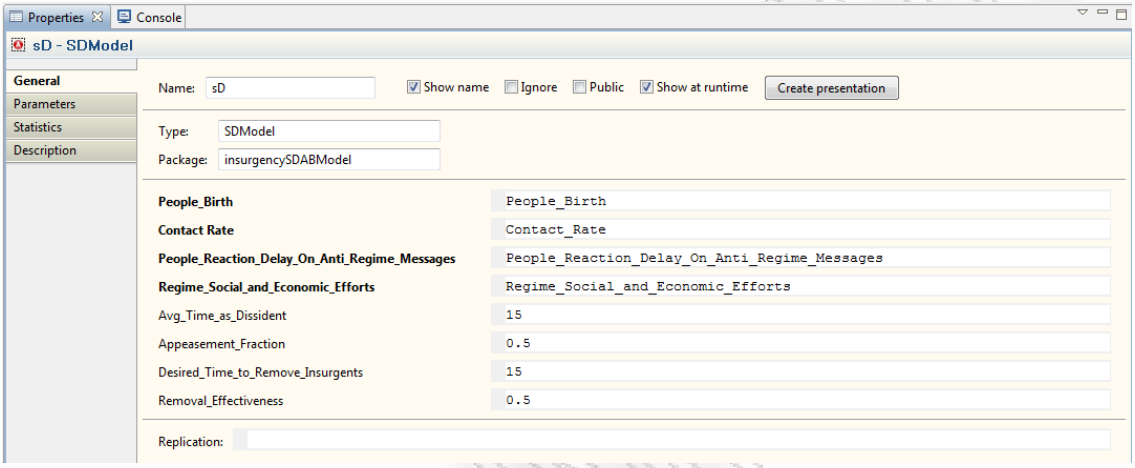
Εικόνα 9-16(θ): Γενικές ιδιότητες της παραμέτρου Regime_Social_and_Economic_Efforts.



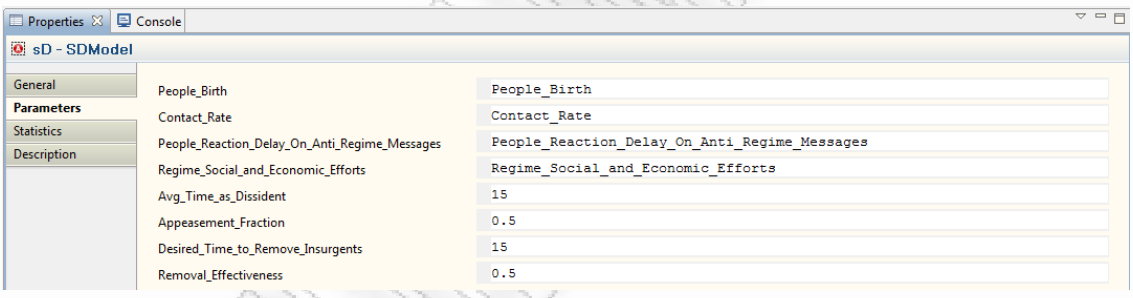
Εικόνα 9-17(α): Γενικές ιδιότητες του στιγμιότυπου aM της κλάσης ABModel.



Εικόνα 9-17(β): Παράμετροι του στιγμιότυπου aM της κλάσης ABModel.

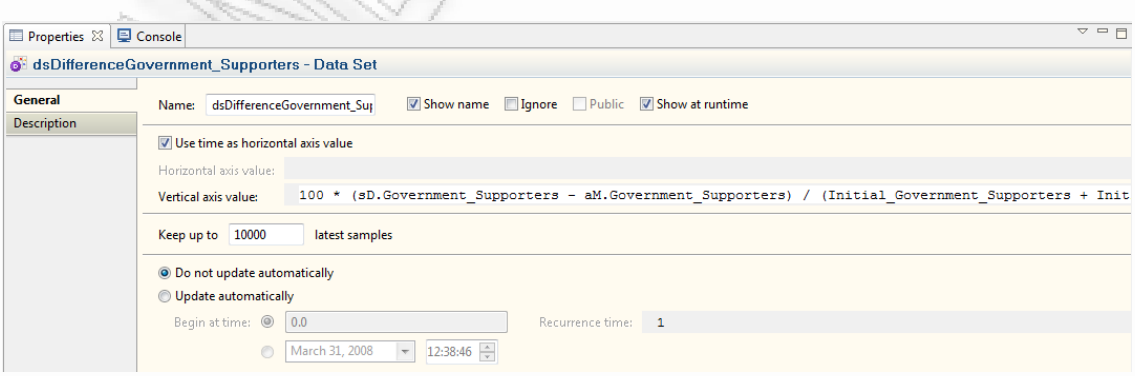


Εικόνα 9-18(α): Γενικές ιδιότητες του στιγμιότυπου sD της κλάσης SDModel.



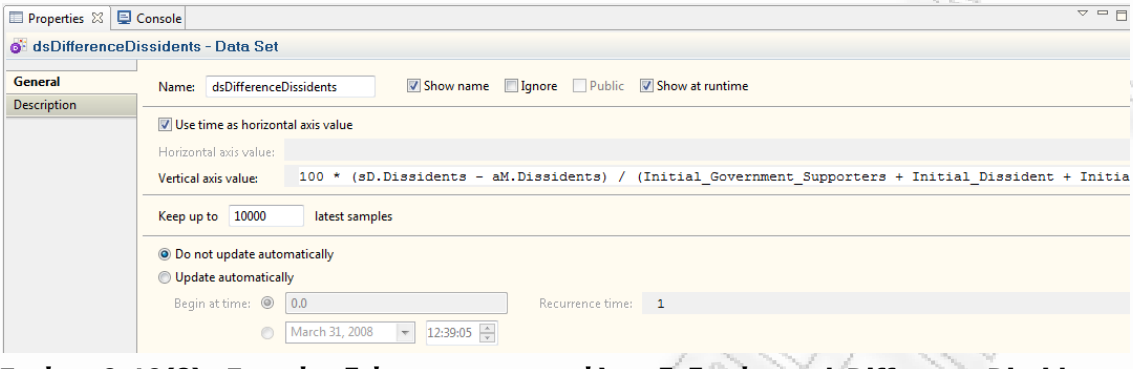
Εικόνα 9-18(β): Παράμετροι του στιγμιότυπου sD της κλάσης SDModel.

Στις εικόνες 9-19(α) έως και 9-19(δ) δίνονται αντίστοιχα οι γενικές ιδιότητες των συνόλων δεδομένων και στις εικόνες 9-20(α) έως και 9-20(γ) εμφανίζονται αντίστοιχα οι γενικές ιδιότητες των χρονικών διαγραμμάτων της κλάσης Main.



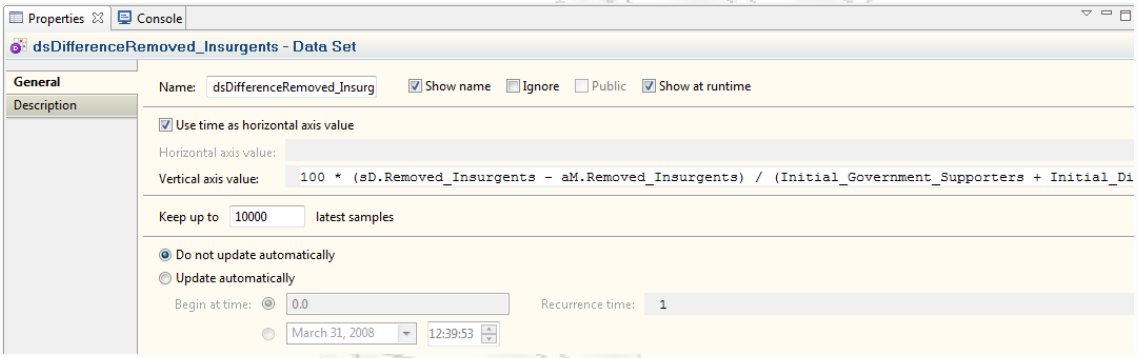
Εικόνα 9-19(α): Γενικές ιδιότητες του συνόλου δεδομένων dsDifferenceGovernment_Supporters, όπου ο κάθετος άξονας έχει τιμή

$$100 * \frac{(sD.Government\_Supporters - aM.Government\_Supporters)}{(Initial\_Government\_Supporters + Initial\_Dissident + Initial\_Insurgent)}$$



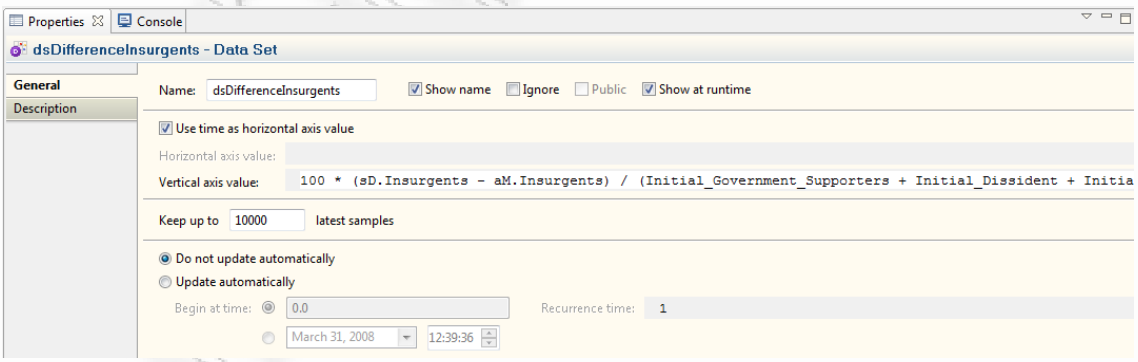
Εικόνα 9-19(β): Γενικές ιδιότητες του συνόλου δεδομένων dsDifferenceDissidents, όπου ο κάθετος άξονας έχει τιμή

$$100 * \frac{(sD.Dissidents - aM.Dissidents)}{(Initial\_Government\_Supporters + Initial\_Dissident + Initial\_Insurgent)}$$



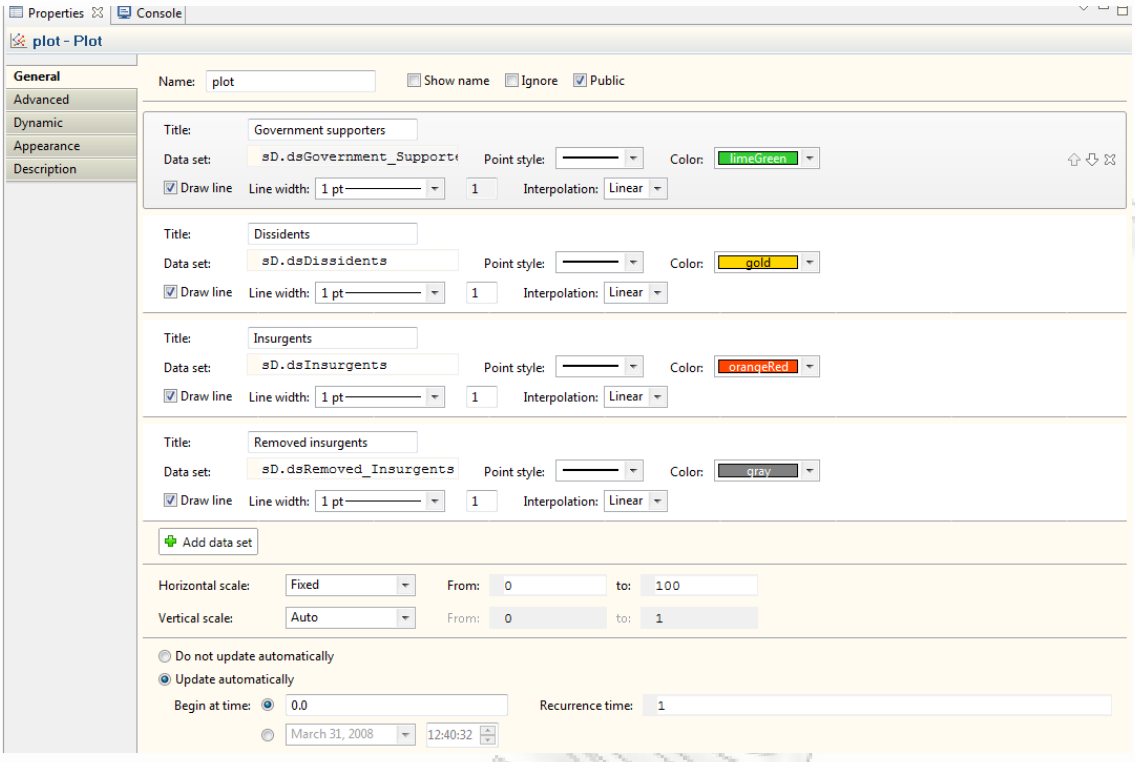
Εικόνα 9-19(γ): Γενικές ιδιότητες του συνόλου δεδομένων dsDifferenceRemoved_Insurgents, όπου ο κάθετος άξονας έχει τιμή

$$100 * \frac{(sD.Removed\_Insurgents - aM.Removed\_Insurgents)}{(Initial\_Government\_Supporters + Initial\_Dissident + Initial\_Insurgent)}$$

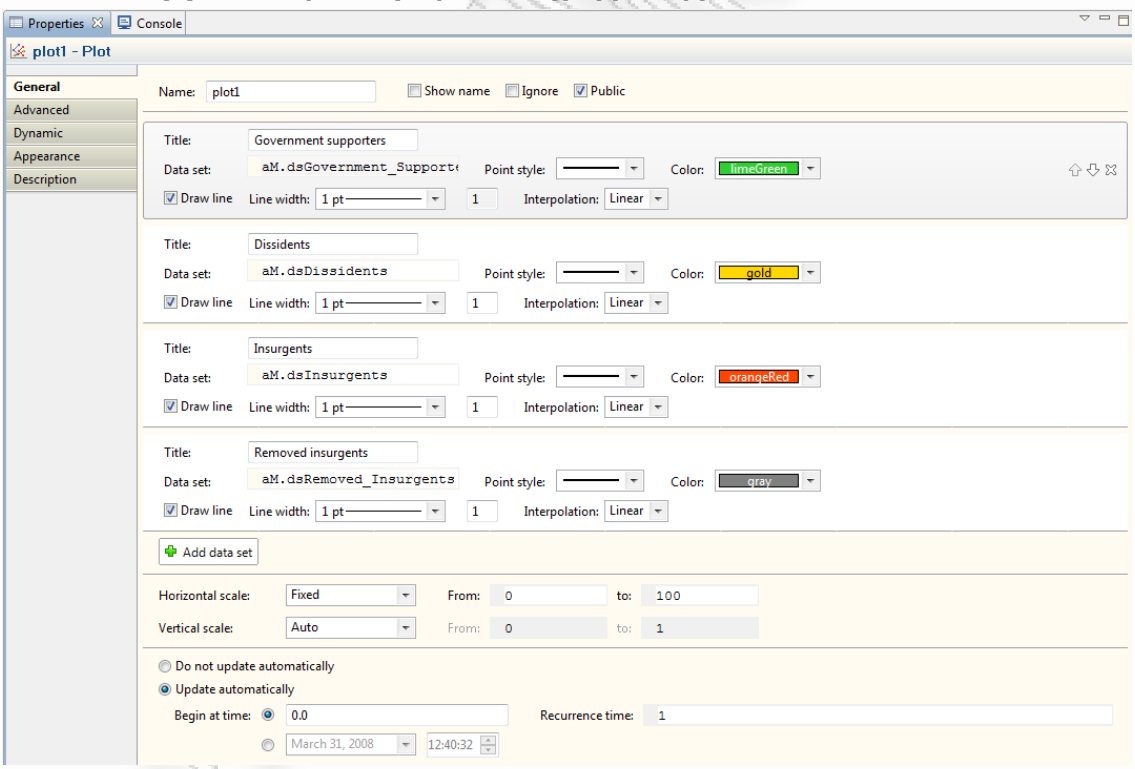


Εικόνα 9-19(δ): Γενικές ιδιότητες του συνόλου δεδομένων dsDifferenceInsurgents, όπου ο κάθετος άξονας έχει τιμή

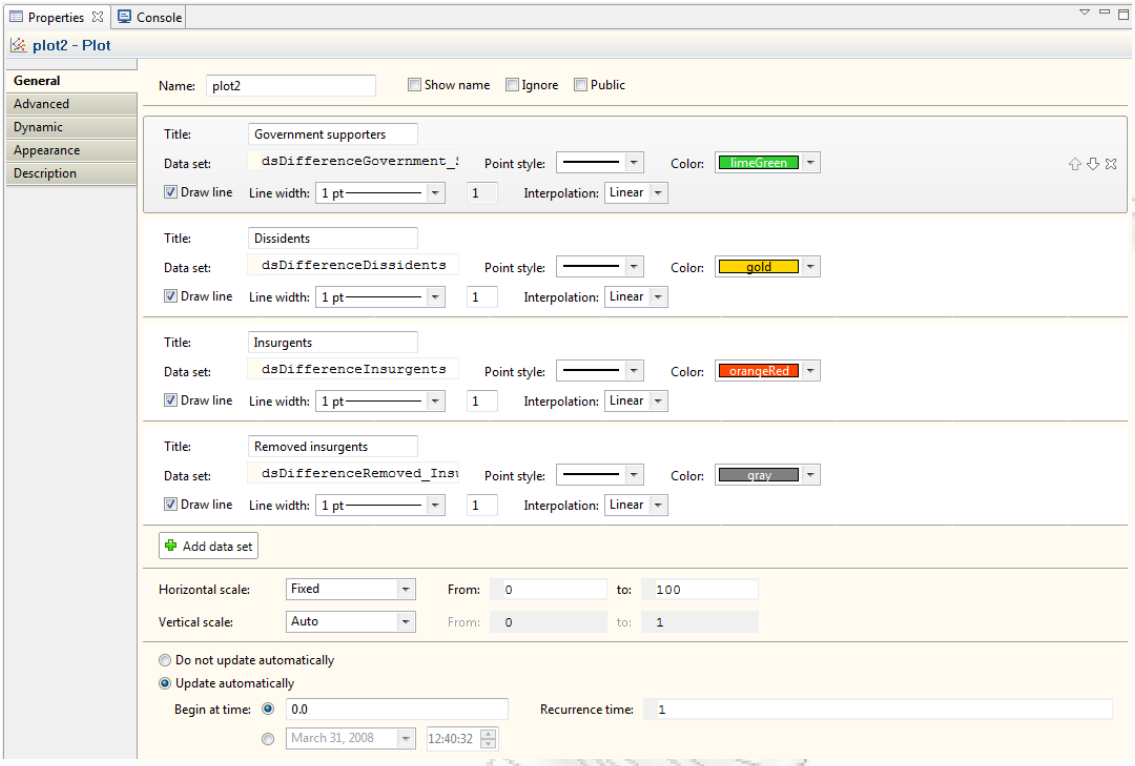
$$100 * \frac{(sD.Insurgents - aM.Insurgents)}{(Initial\_Government\_Supporters + Initial\_Dissident + Initial\_Insurgent)}$$



Εικόνα 9-20(α): Γενικές ιδιότητες του διαγράμματος plot.



Εικόνα 9-20(β): Γενικές ιδιότητες του διαγράμματος plot1.



Εικόνα 9-20(γ): Γενικές ιδιότητες του διαγράμματος plot2.

9.3 Η κλάση Person

Η κλάση ενεργού αντικειμένου Person, εικόνες 9-21, 9-22(α) έως και 9-22(γ), υλοποιείται μέσω ενός διαγράμματος καταστάσεων. Καθορίζει τον πράκτορα λογισμικού και μοντελοποιεί τις πιθανές καταστάσεις στις οποίες μπορεί να βρεθεί ο πράκτορας λογισμικού που μιμείται το άτομο ως μέλος μιας κοινωνικής ομάδας. Αποτελείται από το διάγραμμα καταστάσεων, εικόνες 9-23(α) έως και 9-23(ιθ), το στοιχείο που υλοποιεί τον πράκτορα λογισμικού, εικόνες 9-24(α) και 9-24(β), δύο (2) κομβία που έχουν ως λειτουργία τη μετάβαση από τον ένα πράκτορα στον άλλο, εικόνες 9-27(α) και 9-27(β), ένα (1) στοιχείο ομάδας, εικόνες 9-25(α) έως και 9-25(γ) που αποτελείται από τα δύο κομβία, μερικά στοιχεία κειμένου, εικόνες 9-26(α) έως και 9-26(δ) και το τμήμα των υπερσυνδέσμων, εικόνες 9-28(α) έως 9-28(δ), όπως αυτοί έχουν ήδη ορισθεί στην κλάση Main .

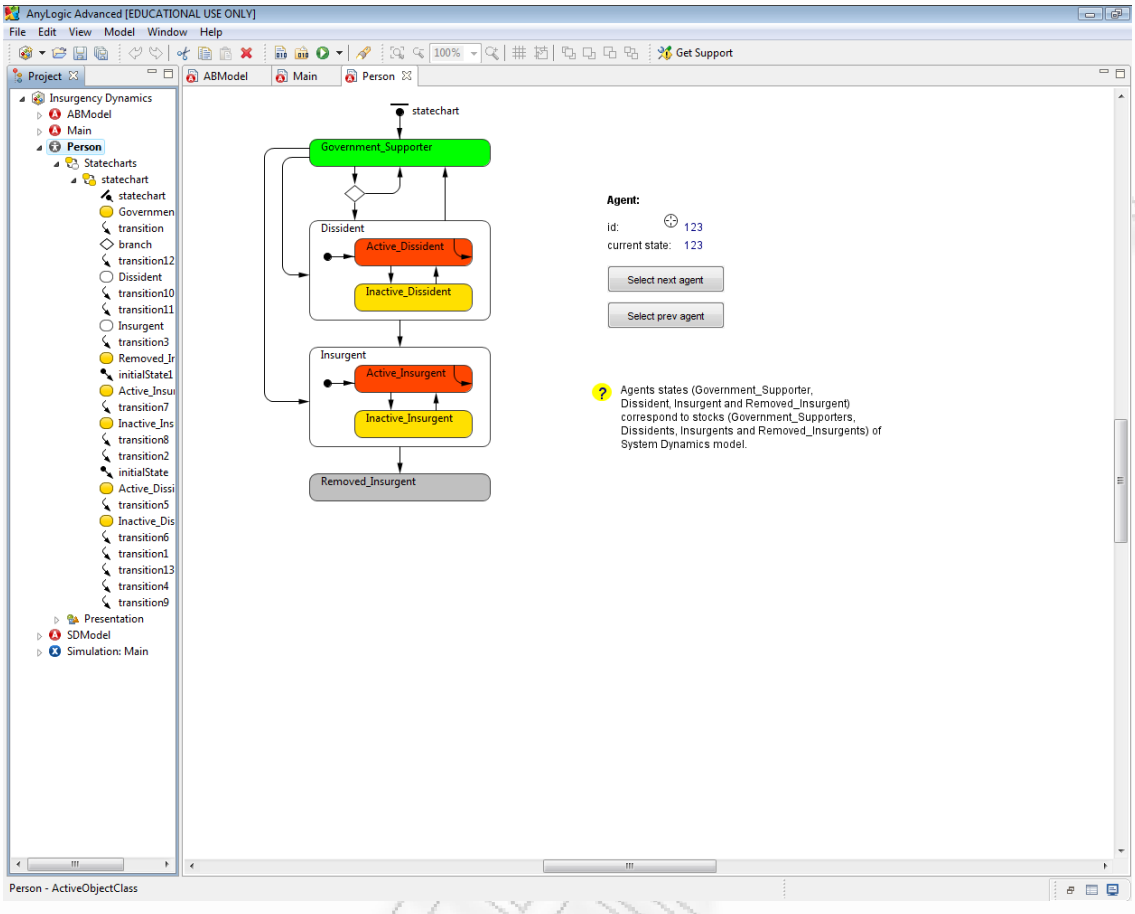
Η μοντελοποίηση του πράκτορα λογισμικού ξεκινά θεωρώντας τον υποστηρικτή της κυβέρνησης, κατάσταση Government_Supporter στο διάγραμμα. Από αυτή την κατάσταση, ο πράκτορας μπορεί: (α) να αλλάξει άμεσα στην κατάσταση του διαφωνούντα, (Dissident, αριστερό τόξο στην εικόνα 9-21), (β) να αλλάξει άμεσα στην κατάσταση του αντάρτη - στασιαστή, (Insurgent, ακραίο αριστερό τόξο στην εικόνα 9-21) και (γ) να αλλάζει κατάσταση από υποστηρικτή της κυβέρνησης, Government_Supporter, σε διαφωνούντα, Dissident. Μέσω της μετάβασης transition, η ροή του διαγράμματος φτάνει στο σημείο επιλογής απόφασης. Αν η συνθήκη απόφασης δεν ικανοποιείται τότε μέσω της μετάβασης transition10 ο πράκτορας παραμένει στην κατάσταση του κυβερνητικού υποστηρικτή. Αν όμως η συνθήκη απόφασης ικανοποιείται, τότε μέσω της μετάβασης transition12 ο πράκτορας αλλάζει κατάσταση σε διαφωνούντα, αλλάζει χρώμα εμφάνισης και μεταβάλλεται ο αριθμός των διαφωνούντων. Η εκπλήρωση της συνθήκης εξαρτάται από την τιμή της βοηθητικής μεταβλητής Propensity_to_be_Recruited και τη σχέση της με την τρέχουσα τιμή μιας ομοιόμορφης κατανομής που μεταβάλλεται μέσω του εσωτερικού μηχανισμού του Λογισμικού κατά τη διάρκεια της προσομοίωσης.

Όταν ο πράκτορας λογισμικού βρεθεί στην κατάσταση του διαφωνούντα, ο πράκτορας θα παραμείνει στην κατάσταση αυτή για όσο χρόνο ορίζει η μεταβλητή Avg_Time_as_Dissident και λαμβάνοντας υπόψη το κλάσμα κατευνασμού, η αλλαγή κατάστασης θα εξαρτηθεί από τη μεταβλητή Appeasement_Fraction, ο πράκτορας θα μεταβεί είτε στην κατάσταση του κυβερνητικού υποστηρικτή ή στην κατάσταση του αντάρτη - στασιαστή. Κατά την παραμονή στην κατάσταση του διαφωνούντα, ένα δεύτερο εσωτερικό επίπεδο καταστάσεων λαμβάνει χώρα. Ο πράκτορας λογισμικού αρχικά βρίσκεται στην ενεργή κατάσταση, (Active_Dissident) και μπορεί να αλλάζει κατάσταση από ενεργός σε ανενεργός , (Inactive_Dissident), σύμφωνα με τις παραμέτρους του μοντέλου, μετάβαση transition5 και transition6. Η διαφορά των δύο καταστάσεων είναι ότι μόνο ο ενεργός διαφωνών μπορεί να προσηλυτίσει άλλους πράκτορες που βρίσκονται στη κατάσταση του κυβερνητικού υποστηρικτή, κάτι που εξαρτάται από τη μεταβλητή Contact_Rate.

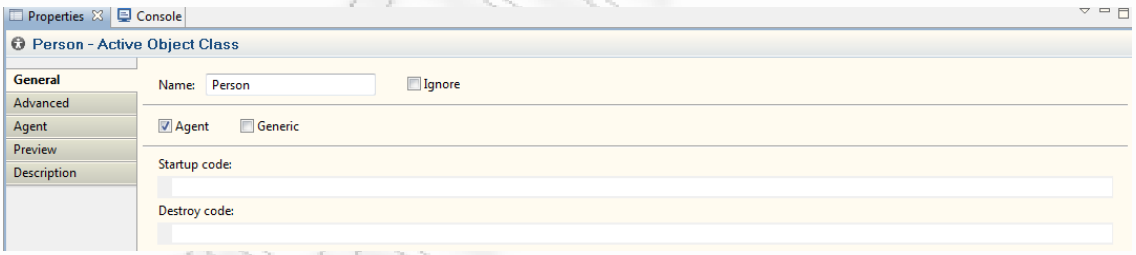
Όταν ο πράκτορας λογισμικού βρεθεί στην κατάσταση του αντάρτη - στασιαστή παραμένει σε αυτή έως ότου ικανοποιηθεί η συνθήκη transition3 για τη μετάβαση στην κατάσταση των «απομακρυσμένων» ανταρτών – στασιαστών, (Removed_Insurgent). Κατά την παραμονή στην κατάσταση του αντάρτη - στασιαστή, ένα δεύτερο εσωτερικό επίπεδο καταστάσεων λαμβάνει χώρα. Ο πράκτορας λογισμικού αρχικά βρίσκεται στην ενεργή κατάσταση, (Active_Insurgent) και μπορεί να αλλάζει κατάσταση από ενεργός σε ανενεργός , (Inactive_Insurgent), σύμφωνα με τις παραμέτρους του μοντέλου, μετάβαση transition7 και transition8. Ο πράκτορας που βρίσκεται στην κατάσταση Active_Insurgent μπορεί να προσηλυτίσει άλλους πράκτορες ενώ ο πράκτορας που βρίσκεται στην κατάσταση Inactive Insurgent δεν μπορεί.

Από την κατάσταση του κυβερνητικού υποστηρικτή ο πράκτορας μπορεί να αλλάξει άμεσα κατάσταση σε διαφωνούντα ή σε αντάρτη - στασιαστή, (οι περιπτώσεις α και β που αναφέρθηκαν προηγούμενα), αν οι πράκτορες λογισμικού που βρίσκονται ήδη στην ενεργή κατάσταση των διαφωνούντων, (Active_Dissident) και των ανταρτών - στασιαστών, (Active_Insurgent) στείλουν το κατάλληλο μήνυμα μέσω του εσωτερικού μηχανισμού επικοινωνίας πρακτόρων του λογισμικού και η ροπή για στρατολόγηση λάβει κατάλληλη τιμή, ανάλογα με τις τιμές των κατανομών που χρησιμοποιούνται στην προσομοίωση, αλλιώς ο πράκτορας θα παραμένει στην ίδια κατάσταση την Government_Supporter.

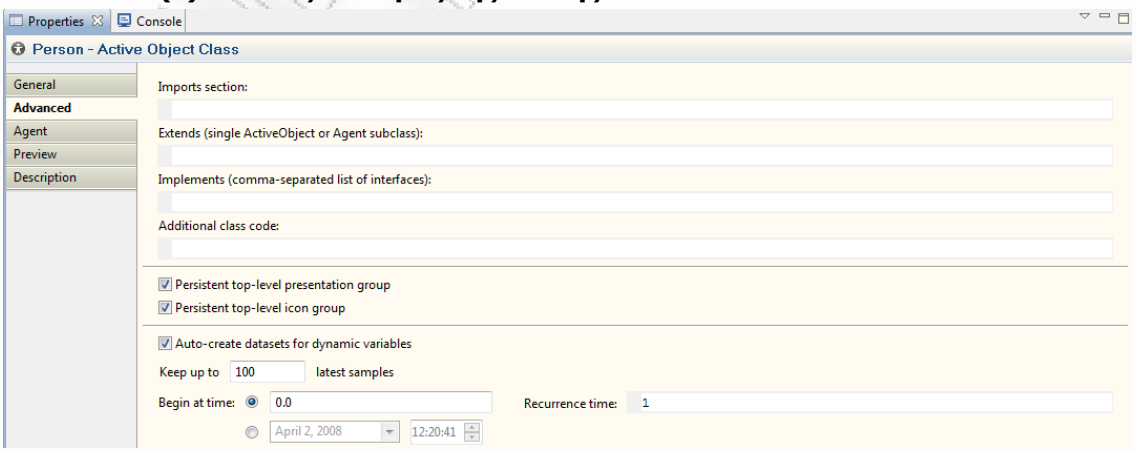
Τέλος, ο πράκτορας αλλάζει στην κατάσταση των «απομακρυσμένων» ανταρτών – στασιαστών όταν οι μεταβλητές Removal_Effectiveness Desired_Time_to_Remove_Insurgents λάβουν τις κατάλληλες τιμές.



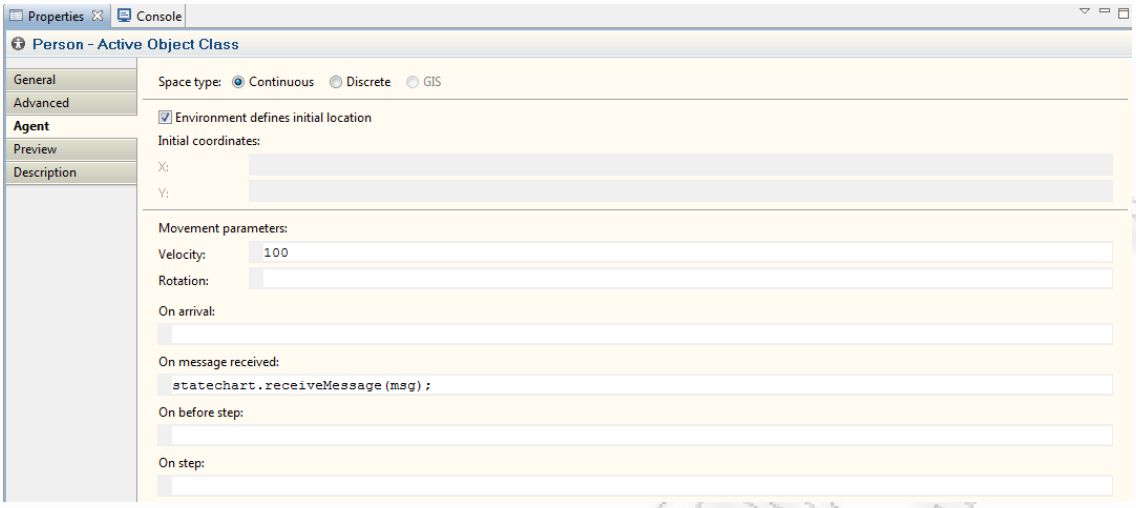
Εικόνα 9-21: Η κλάση ενεργού αντικειμένου Person.



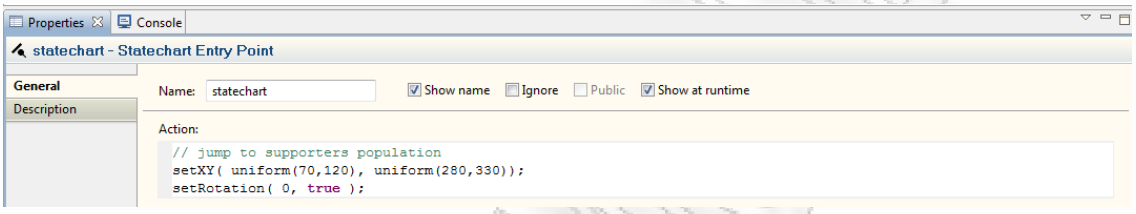
Εικόνα 9-22(α): Γενικές ιδιότητες της κλάσης Person.



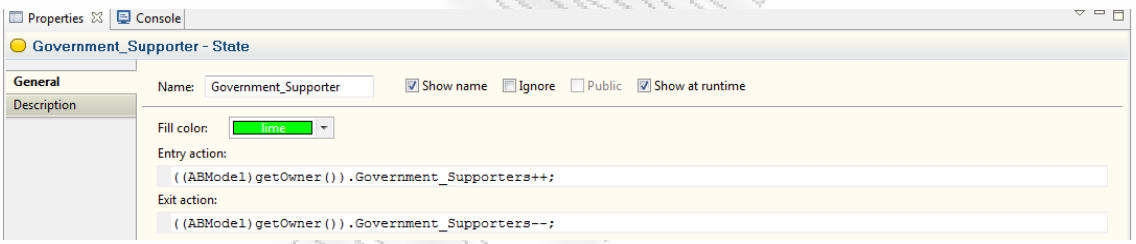
Εικόνα 9-22(β): Προχωρημένες ιδιότητες της κλάσης Person.



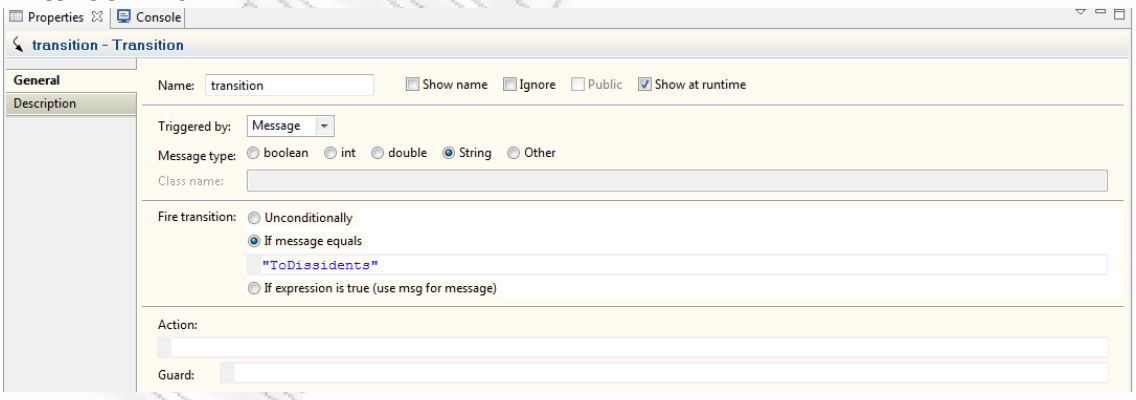
Εικόνα 9-22(γ): Ιδιότητες του πράκτορα της κλάσης Person.



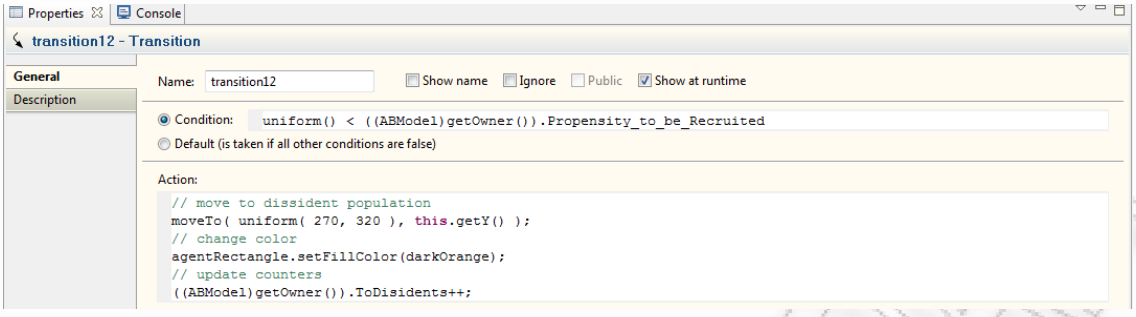
Εικόνα 9-23(α): Γενικές ιδιότητες του σημείου εκκίνησης του διαγράμματος καταστάσεων.



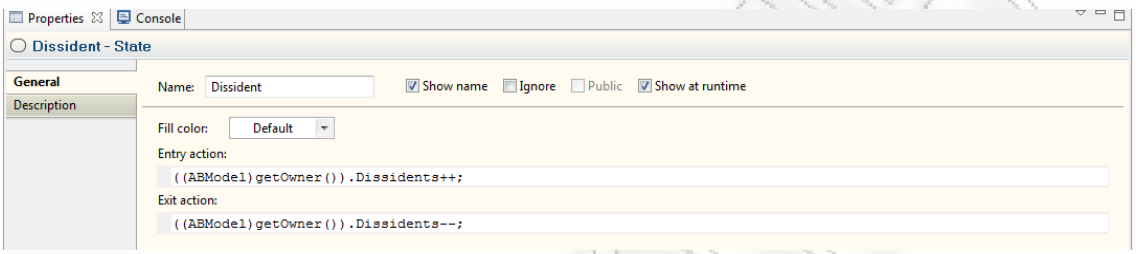
Εικόνα 9-23(β): Γενικές ιδιότητες της κατάστασης Government_Supporter του διαγράμματος καταστάσεων.



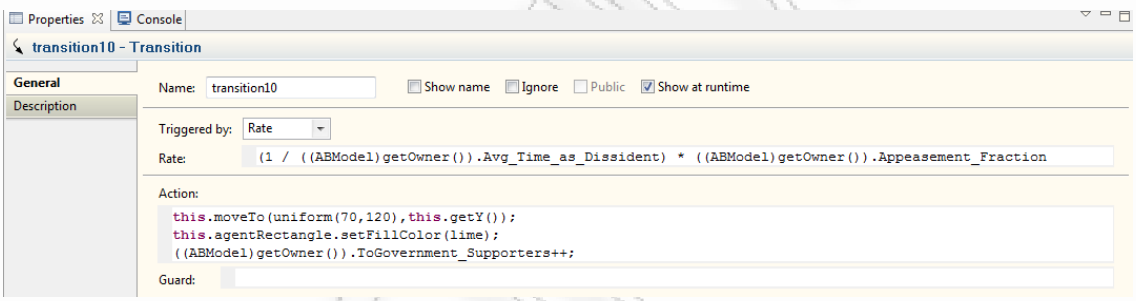
Εικόνα 9-23(γ): Γενικές ιδιότητες της μετάβασης transition του διαγράμματος καταστάσεων.



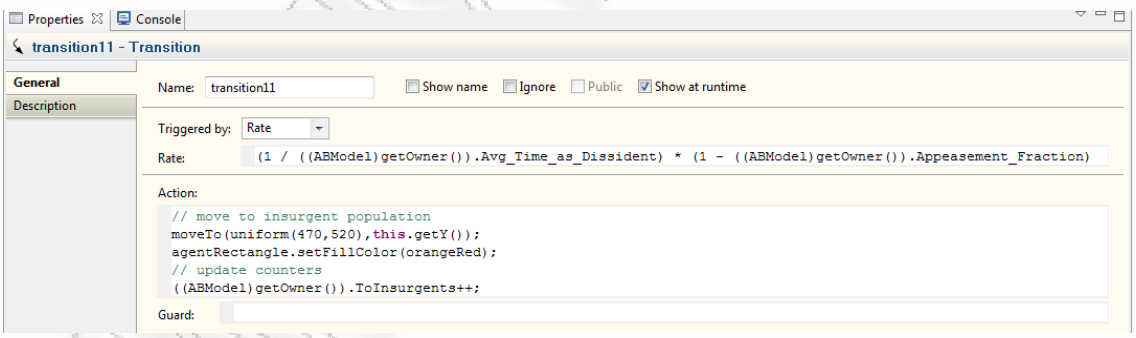
Εικόνα 9-23(δ): Γενικές ιδιότητες της μετάβασης transition12 του διαγράμματος καταστάσεων.



Εικόνα 9-23(ε): Γενικές ιδιότητες της κατάστασης Dissident του διαγράμματος καταστάσεων.

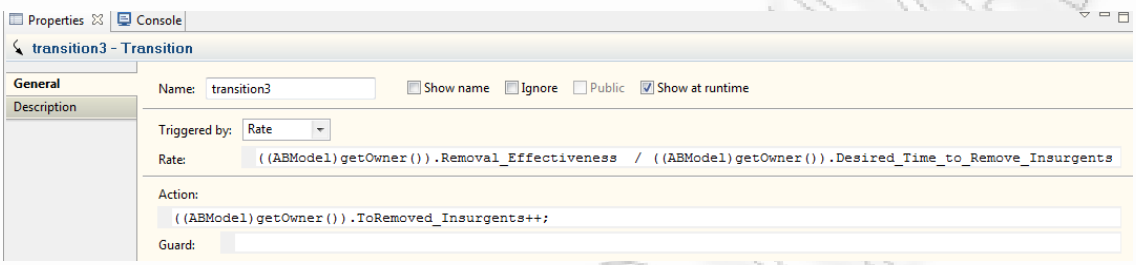


Εικόνα 9-23(στ): Γενικές ιδιότητες της μετάβασης transition10 του διαγράμματος καταστάσεων.

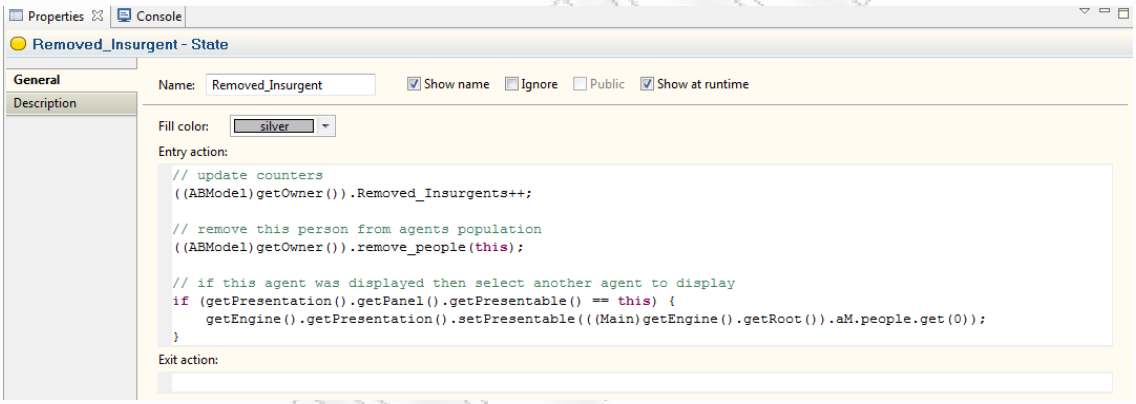


Εικόνα 9-23(ζ): Γενικές ιδιότητες της μετάβασης transition11 του διαγράμματος καταστάσεων.

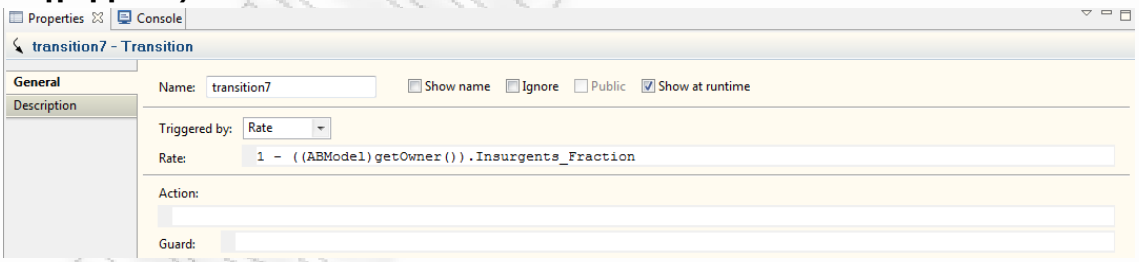
Εικόνα 9-23(η): Γενικές ιδιότητες της κατάστασης Insurgent του διαγράμματος καταστάσεων.



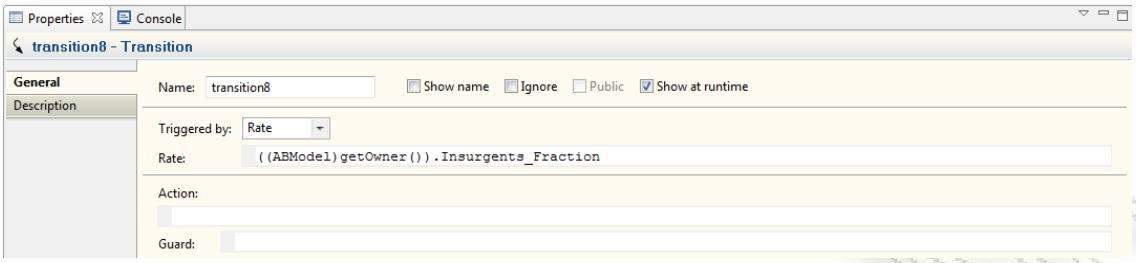
Εικόνα 9-23(θ): Γενικές ιδιότητες της μετάβασης transition3 του διαγράμματος καταστάσεων.



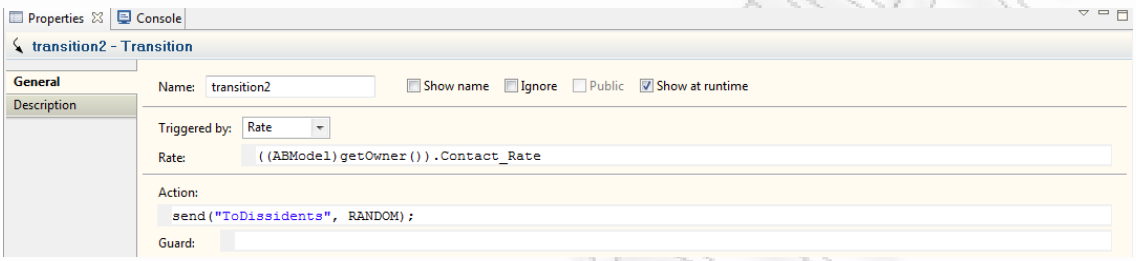
Εικόνα 9-23(ι): Γενικές ιδιότητες της κατάστασης Removed_Insurgent του διαγράμματος καταστάσεων.



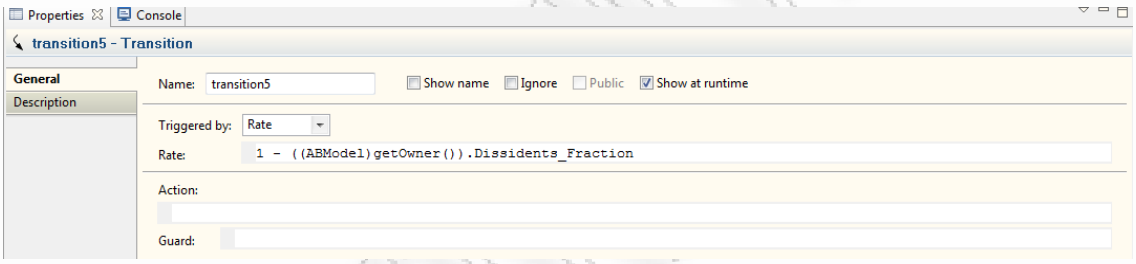
Εικόνα 9-23(ια): Γενικές ιδιότητες της μετάβασης transition7 του διαγράμματος καταστάσεων.



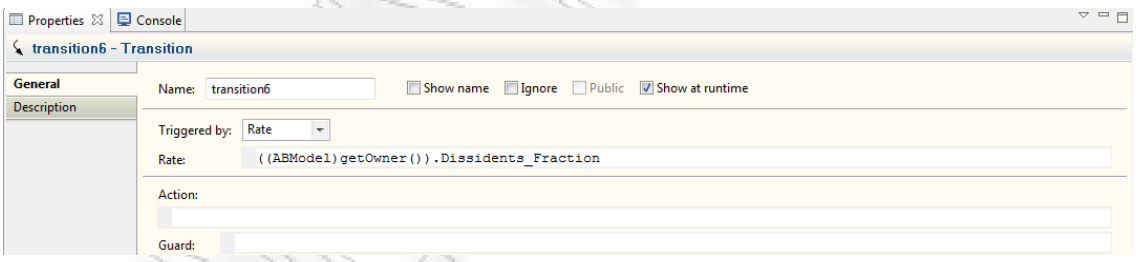
Εικόνα 9-23(ιβ): Γενικές ιδιότητες της μετάβασης transition8 του διαγράμματος καταστάσεων.



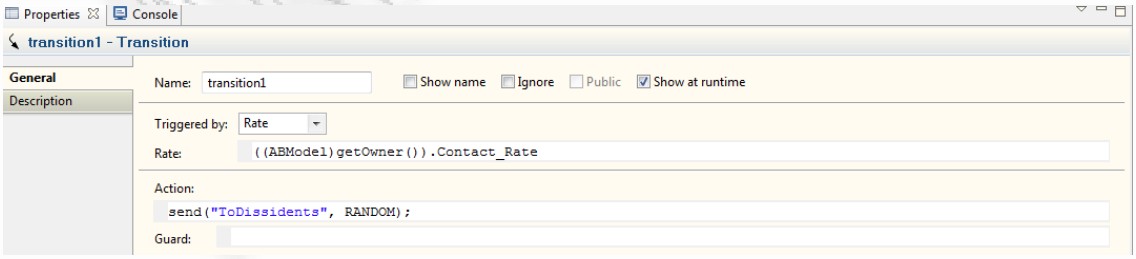
Εικόνα 9-23(ιγ): Γενικές ιδιότητες της μετάβασης transition2 του διαγράμματος καταστάσεων.



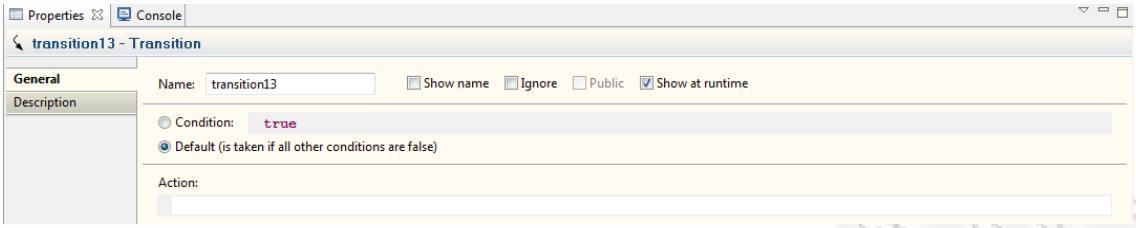
Εικόνα 9-23(ιδ): Γενικές ιδιότητες της μετάβασης transition5 του διαγράμματος καταστάσεων.



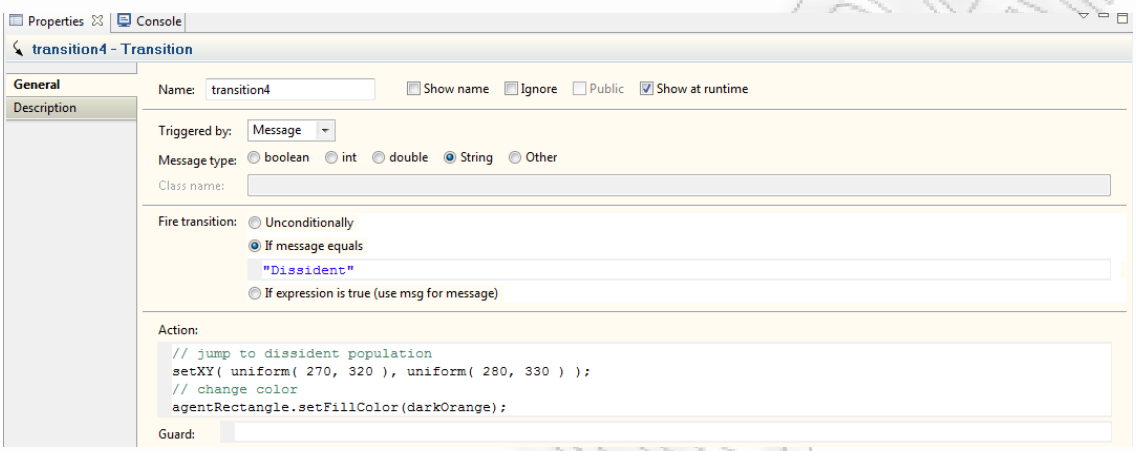
Εικόνα 9-23(ιε): Γενικές ιδιότητες της μετάβασης transition6 του διαγράμματος καταστάσεων.



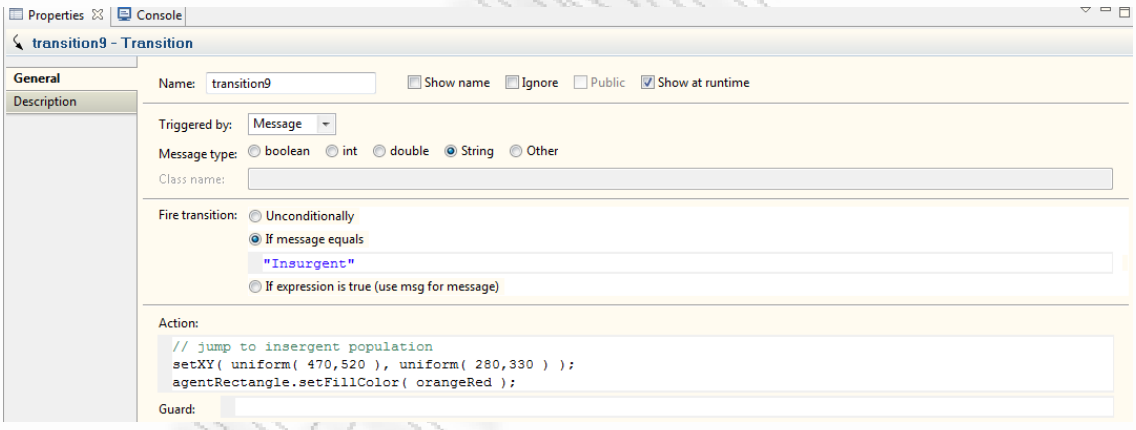
Εικόνα 9-23(ιστ): Γενικές ιδιότητες της μετάβασης transition1 του διαγράμματος καταστάσεων.



Εικόνα 9-23(ιζ): Γενικές ιδιότητες της μετάβασης transition13 του διαγράμματος καταστάσεων.

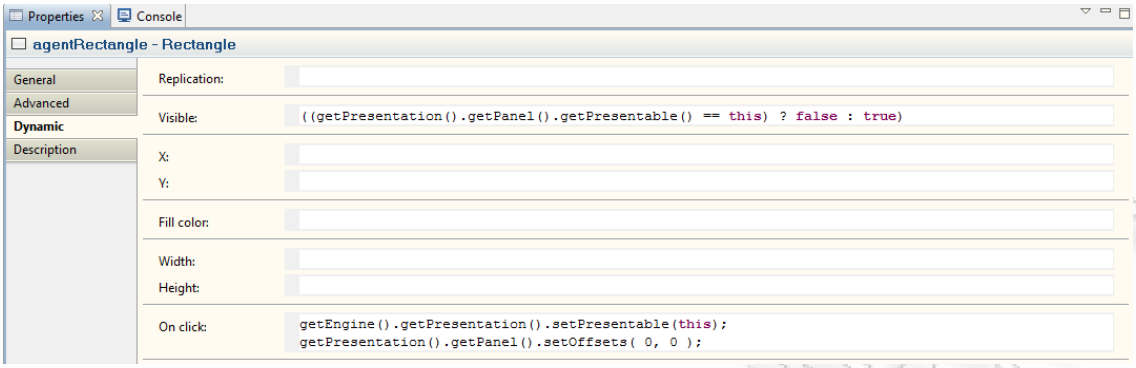


Εικόνα 9-23(ιη): Γενικές ιδιότητες της μετάβασης transition4 του διαγράμματος καταστάσεων.

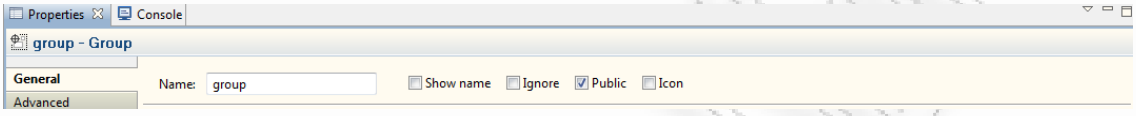


Εικόνα 9-23(ιθ): Γενικές ιδιότητες της μετάβασης transition9 του διαγράμματος καταστάσεων.

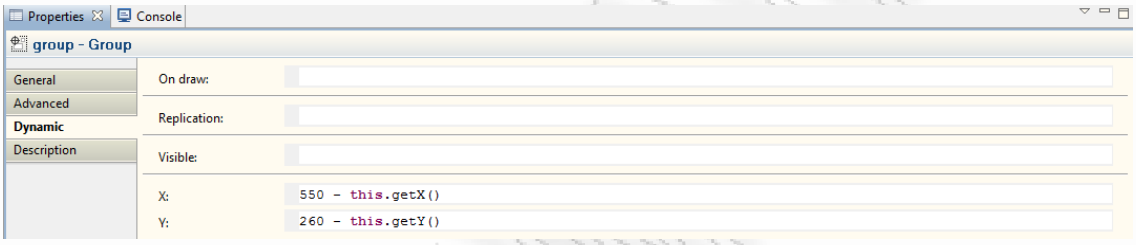
Εικόνα 9-24(α): Γενικές ιδιότητες του σχήματος του πράκτορα.



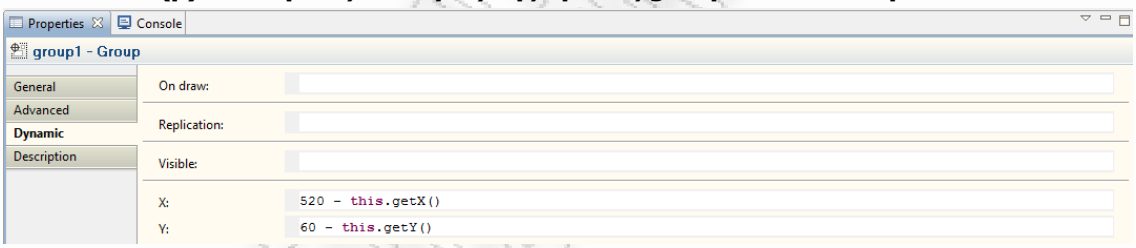
Εικόνα 9-24(β): Δυναμικές ιδιότητες του σχήματος του πράκτορα.



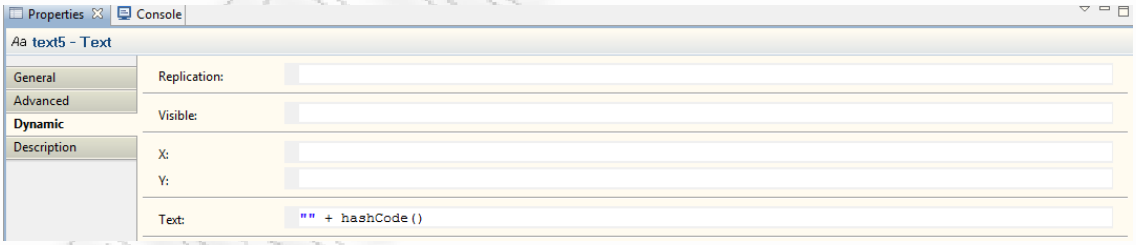
Εικόνα 9-25(α): Γενικές ιδιότητες της ομάδας group τύπου Group.



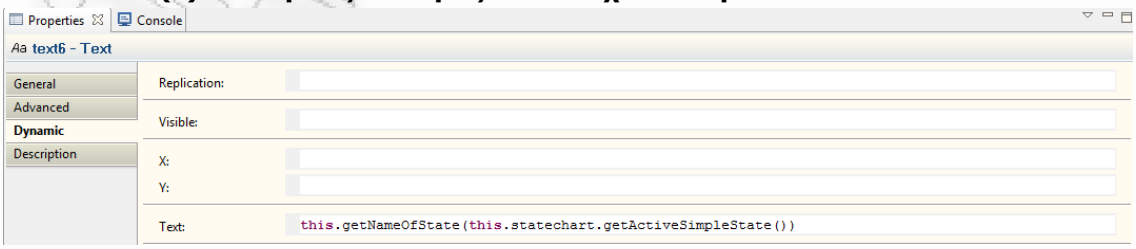
Εικόνα 9-25(β): Δυναμικές ιδιότητες της ομάδας group τύπου Group.



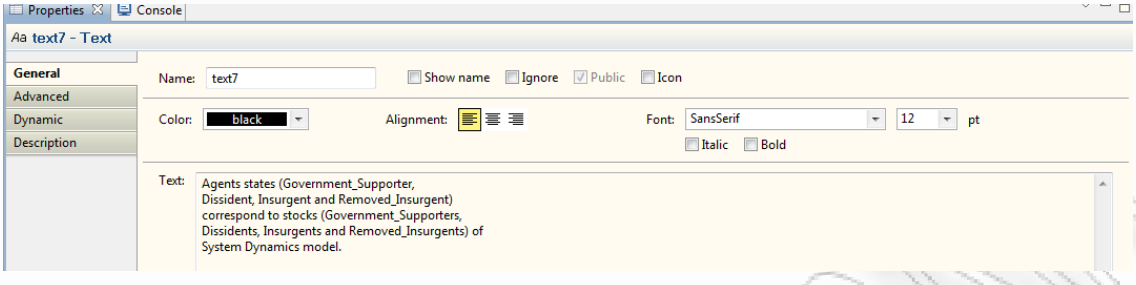
Εικόνα 9-25(γ): Δυναμικές ιδιότητες της ομάδας group1 τύπου Group.



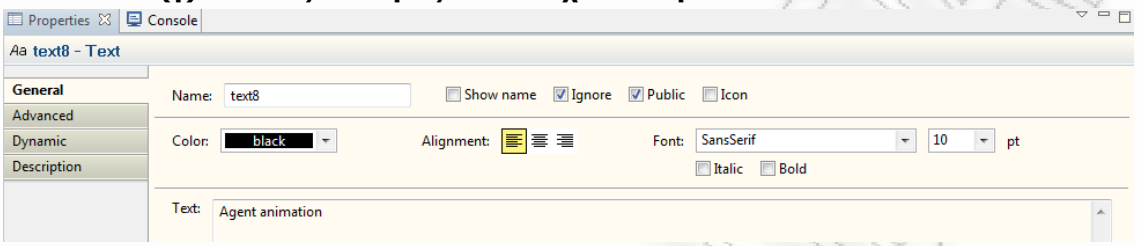
Εικόνα 9-26(α): Δυναμικές ιδιότητες του στοιχείου κειμένου text5.



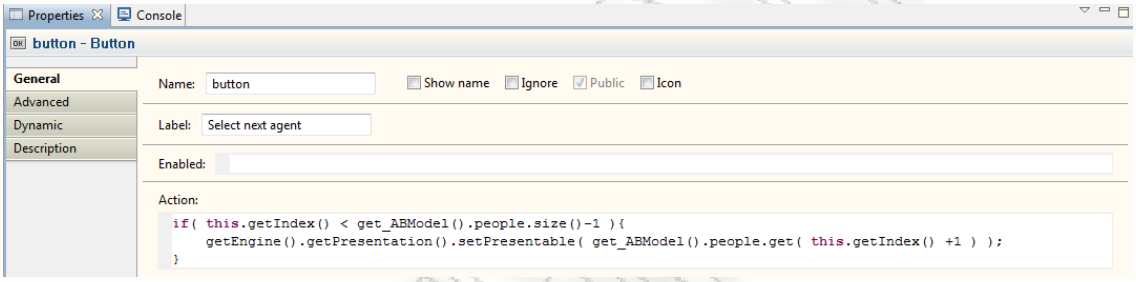
Εικόνα 9-26(β): Δυναμικές ιδιότητες του στοιχείου κειμένου text6.



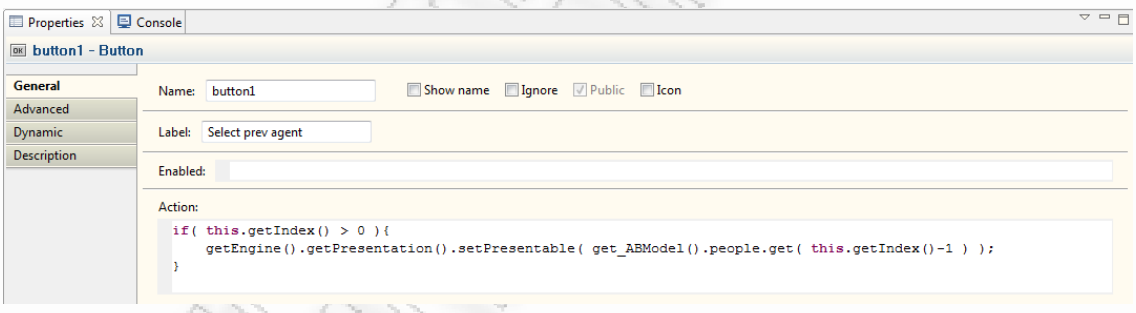
Εικόνα 9-26(γ): Γενικές ιδιότητες του στοιχείου κειμένου text7.



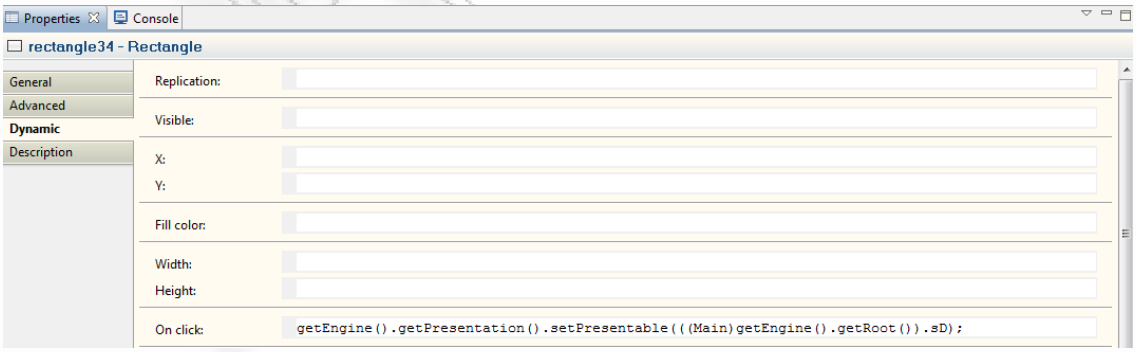
Εικόνα 9-26(δ): Γενικές ιδιότητες του στοιχείου κειμένου text8.



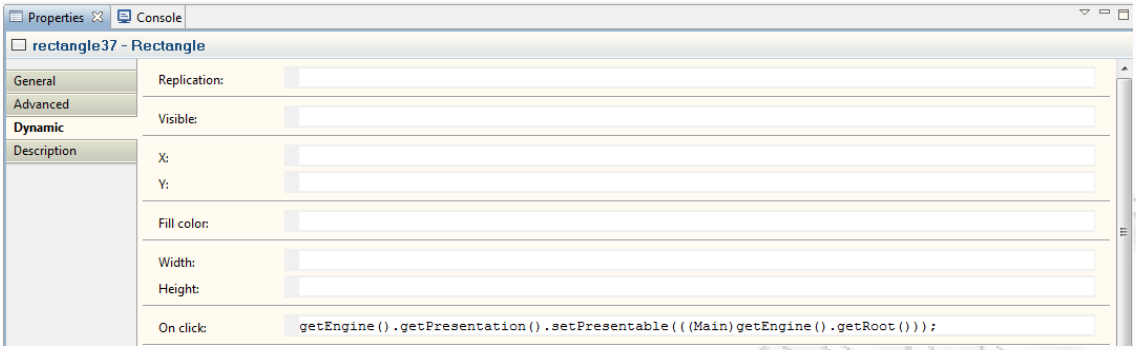
Εικόνα 9-27(α): Γενικές ιδιότητες του κομβίου button τύπου Button.



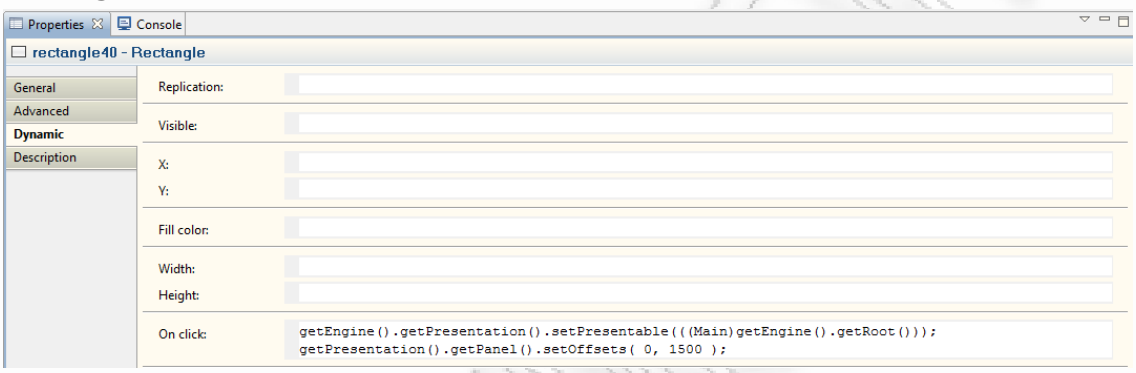
Εικόνα 9-27(β): Γενικές ιδιότητες του κομβίου button1 τύπου Button.



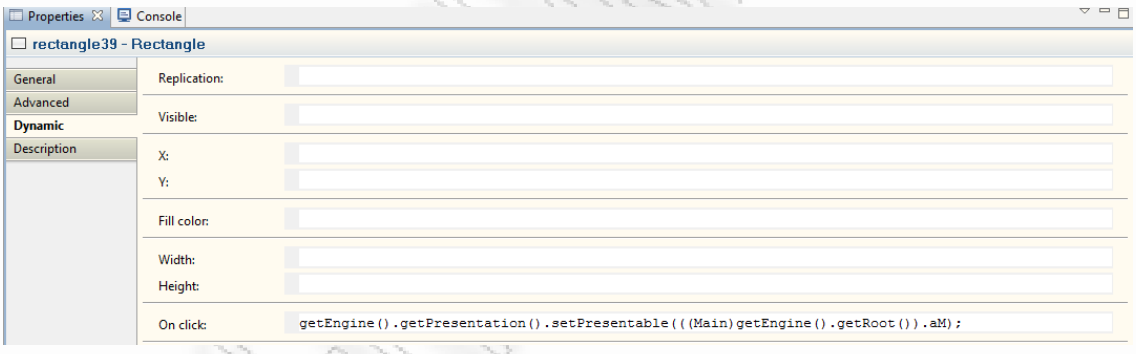
Εικόνα 9-28(α): Δυναμικές ιδιότητες του παραλληλογράμμου rectangle34 τύπου Rectangle.



Εικόνα 9-28(β): Δυναμικές ιδιότητες του παραλληλογράμμου rectangle37 τύπου Rectangle.



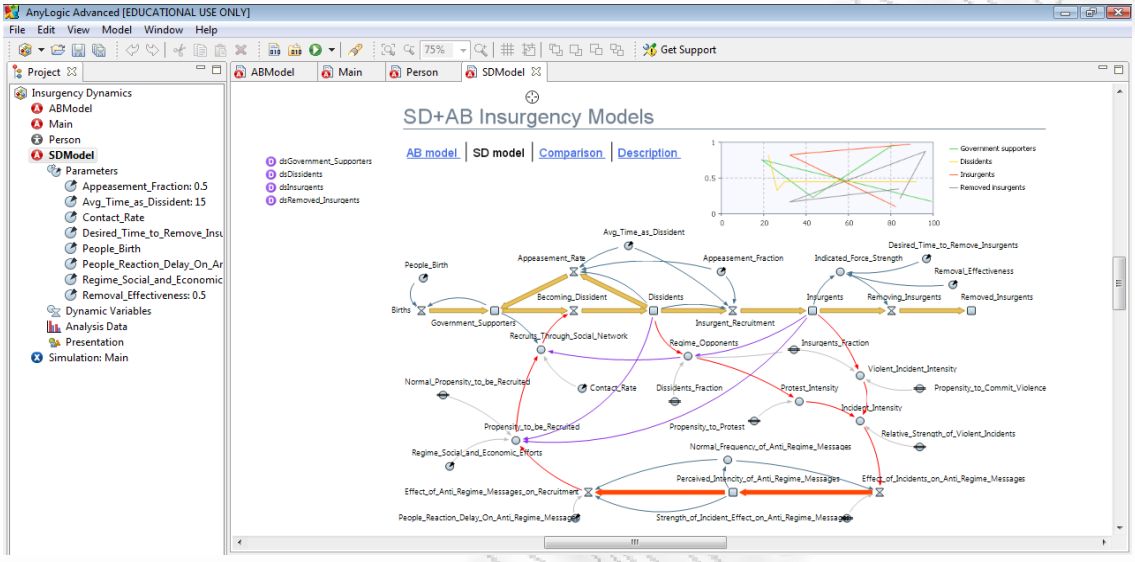
Εικόνα 9-28(γ): Δυναμικές ιδιότητες του παραλληλογράμμου rectangle40 τύπου Rectangle.



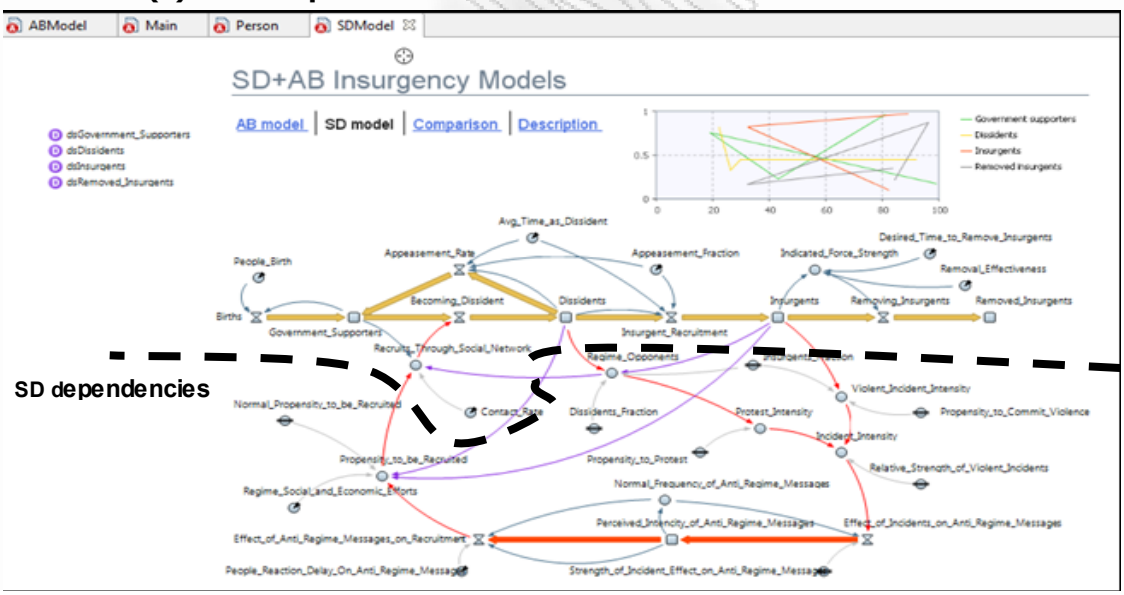
Εικόνα 9-28(δ): Δυναμικές ιδιότητες του παραλληλογράμμου rectangle39 τύπου Rectangle.

9.4 Η κλάση SDModel

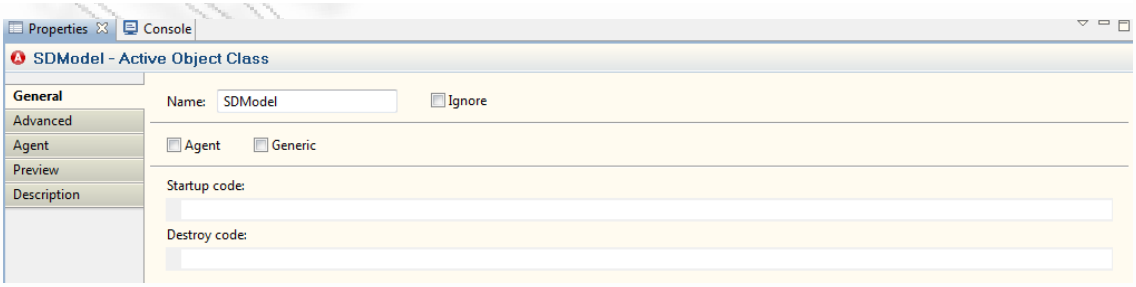
Η κλάση SDModel είναι η κλάση με την οποία υλοποιείται το μοντέλο μέσω της Συστημικής Δυναμικής.



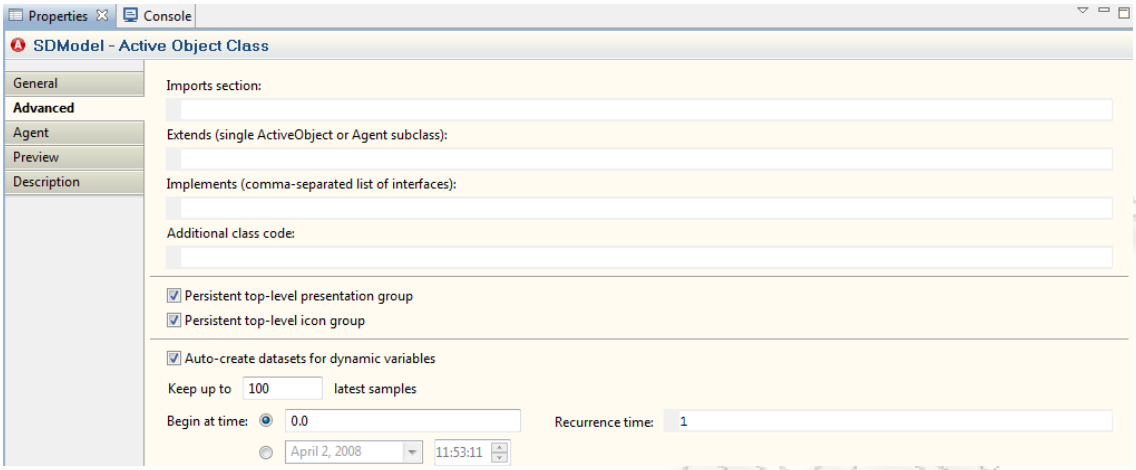
Εικόνα 9-29(α): Η κλάση SDModel.



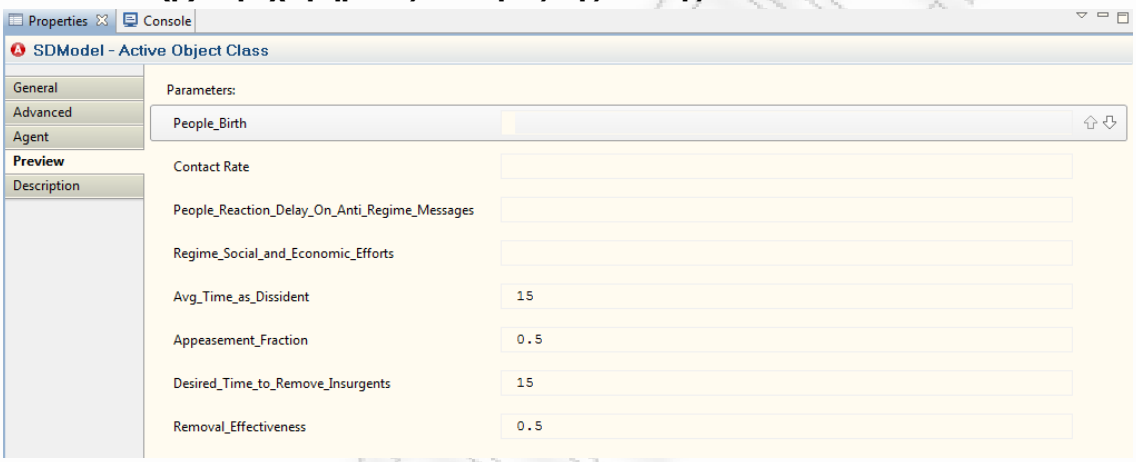
Εικόνα 9-29(β): Το τμήμα των εξαρτήσεων, (SD dependencies) της κλάσης SDModel, οριοθετημένο από την διακεκομμένη γραμμή, παραμένει το ίδιο όπως και στην κλάση ABModel.



Εικόνα 9-30(α): Γενικές ιδιότητες της κλάσης SDModel.



Εικόνα 9-30(β): Προχωρημένες ιδιότητες της κλάσης SDModel.



Εικόνα 9-30(γ): Προεπισκόπιση των παραμέτρων της κλάσης SDModel.

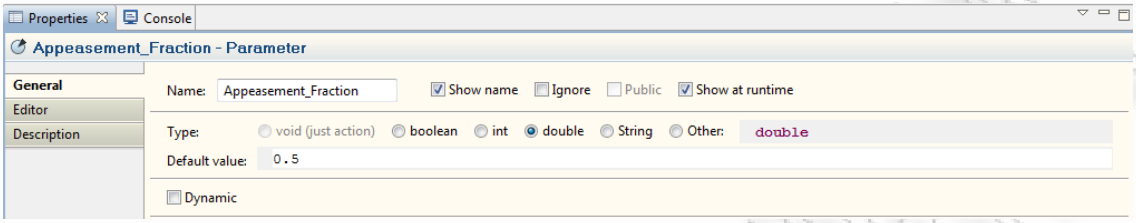
Στην εικόνα 9-29(α) παρουσιάζεται η κλάση όπως αυτή φαίνεται στον γραφικό συντάκτη του Λογισμικού μετά τη δημιουργία της ενώ στην εικόνα 9-29(β) καταδεικνύονται οι διαφορές του μοντέλου σε σχέση με αυτό της κλάσης ABModel. Στις εικόνες 9-30(α) και 9-30(β) δίνονται οι γενικές και οι προχωρημένες ιδιότητες της κλάσης.

Στην εικόνα 9-29(γ) δίνονται συγκεντρωμένες οι οκτώ (8) παράμετροι της κλάσης. Εκτός των παραμέτρων, η κλάση αποτελείται επίσης από το στοιχείο ομαδοποίησης συν τα στοιχεία που αποτελούν την ομάδα, ένα (1) διάγραμμα χρόνου, τέσσερα (4) σύνολα δεδομένων, τέσσερα (4) σημεία συσσώρευσης, πέντε (5) ροές και δύο (2) βοηθητικές μεταβλητές. Το τμήμα των εξαρτήσεων, (όπως έχει ήδη περιγραφεί προηγούμενα), συγκροτείται από δύο (2) ροές, ένα (1) σημείο συσσώρευσης, δύο (2) παραμέτρους, επτά (7) βοηθητικές παραμέτρους και πέντε (5) βοηθητικές μεταβλητές. Στις εικόνες 9-31(α) έως και 9-31(η) δίνονται οι γενικές ιδιότητες των παραμέτρων του μοντέλου.

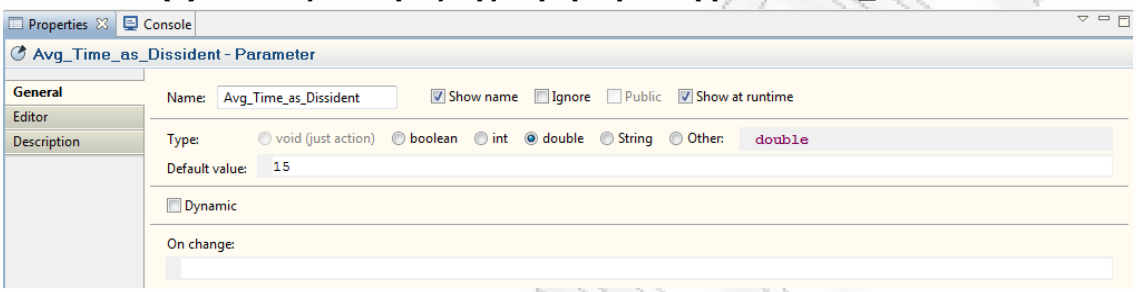
Στο μοντέλο, που υλοποιείται μέσω της κλάσης SDModel, η ροή Births που εξαρτάται από την παράμετρο People_Birth, εισαγάγει ένα δυναμικά μεταβαλλόμενο αριθμό από άτομα που γίνονται μέλη στην ομάδα των υποστηρικτών της κυβέρνησης, ανά χρονική μονάδα της προσομοίωσης. Οι πληθυσμοί των τεσσάρων (4) κοινωνικών ομάδων, (Government_Supporters, Dissidents, Insurgents και Removed_Insurgents), αποθηκεύονται στα τέσσερα σημεία συσσώρευσης. Το πέμπτο σημείο συσσώρευσης που βρίσκεται στο τμήμα των εξαρτήσεων, Perceived_Intencity_of_Anti_Regime_Messages, είναι αυτό που αποθηκεύει τα μηνύματα που δημιουργούνται από τους αντάρτες – στασιαστές και μοντελοποιεί ένα ανάχωμα, έναν απομονωτή μεταξύ των ανταρτών και της κοινωνίας.

Ανάμεσα στα τέσσερα σημεία συσσώρευσης των πληθυσμών των τεσσάρων κοινωνικών ομάδων βρίσκονται οι ροές Becoming_Dissident, Insurgent_Recruitment και

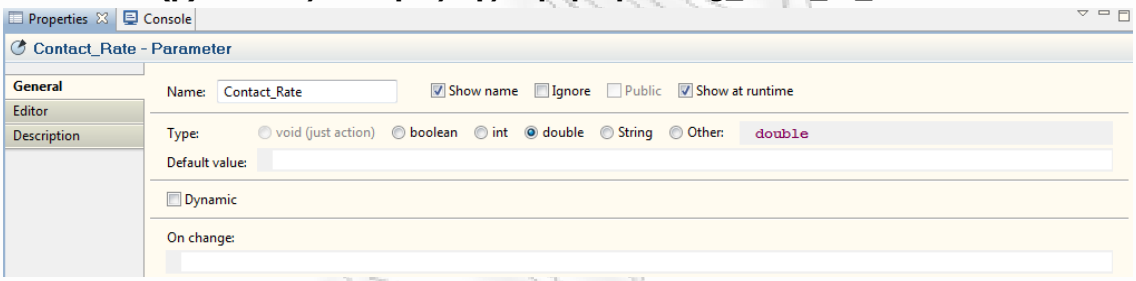
Removing_Insurgents. Η ροή Appeasement_Rate μοντελοποιεί την πολιτική κατευνασμού που προωθεί η κυβέρνηση για να προλάβει τα χειρότερα. Στην εικόνα 9-29, φαίνεται ο βρόχος θετικής ανάδρασης που δημιουργείται.



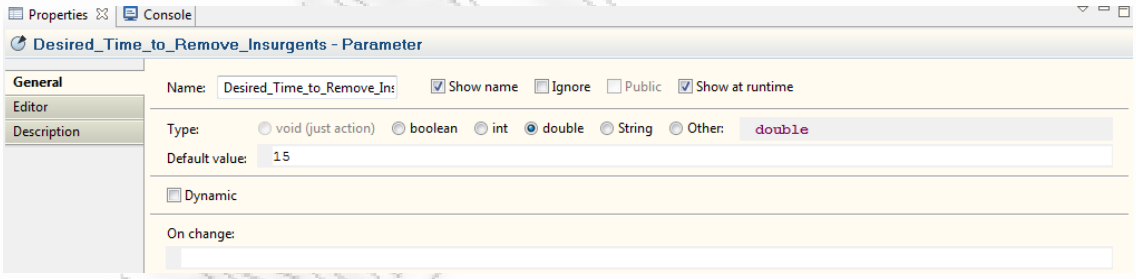
Εικόνα 9-31(α): Γενικές ιδιότητες της παραμέτρου Appeasement_Fraction.



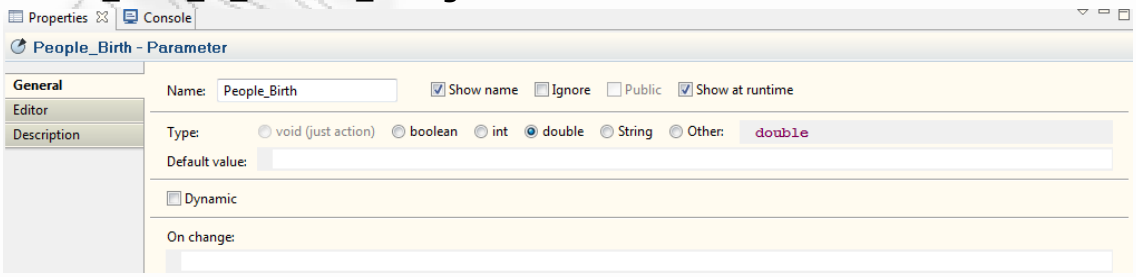
Εικόνα 9-31(β): Γενικές ιδιότητες της παραμέτρου Avg_Time_as_Dissident.



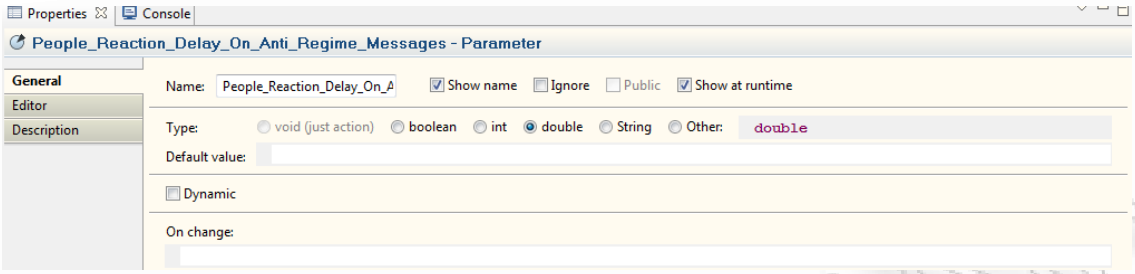
Εικόνα 9-31(γ): Γενικές ιδιότητες της παραμέτρου Contact_Rate.



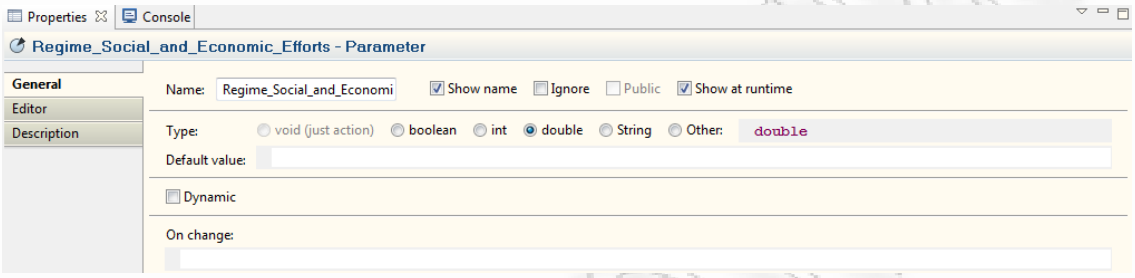
Εικόνα 9-31(δ): Γενικές ιδιότητες της παραμέτρου Desired_Time_to_Remove_Insurgents.



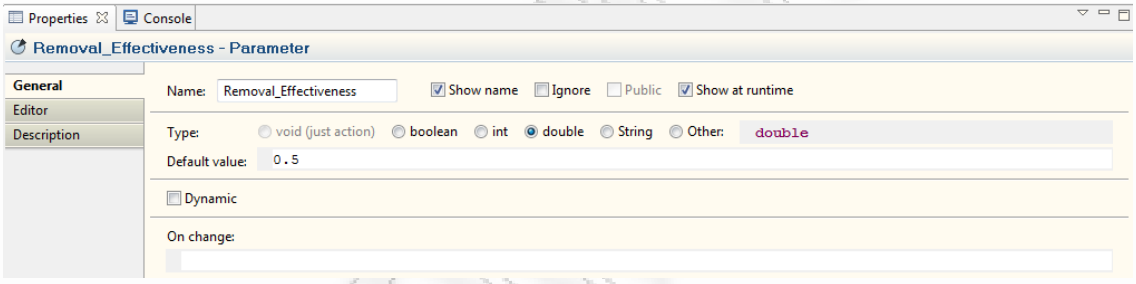
Εικόνα 9-31(ε): Γενικές ιδιότητες της παραμέτρου People_Birth.



Εικόνα 9-31(στ): Η παράμετρος People_Reaction_Delay_On_Anti_Regime_Messages και οι γενικές της ιδιότητες.

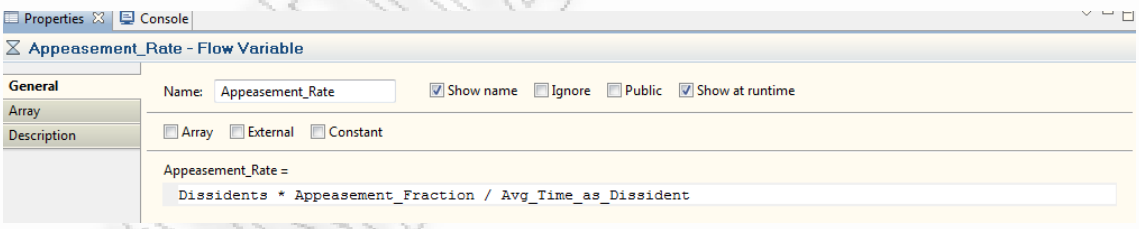


Εικόνα 9-31(ζ): Γενικές ιδιότητες της παραμέτρου Regime_Social_and_Economic_Efforts.



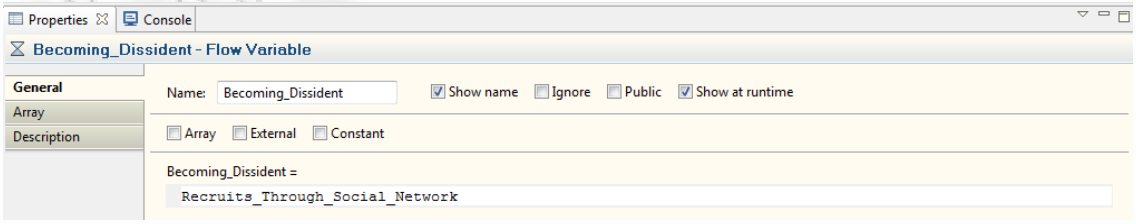
Εικόνα 9-31(η): Γενικές ιδιότητες της παραμέτρου Removal_Effectiveness.

Στις εικόνες 9-32(α) έως και 9-32(ζ) παρουσιάζονται οι γενικές ιδιότητες των ροών που απαρτίζουν την κλάση SDModel.



Εικόνα 9-32(α): Γενικές ιδιότητες της ροής Appeasement_Rate, όπου

$$Appeasement\_Rate = \frac{Dissidents * Appeasement\_Fraction}{Avg\_Time\_as\_Dissident}$$

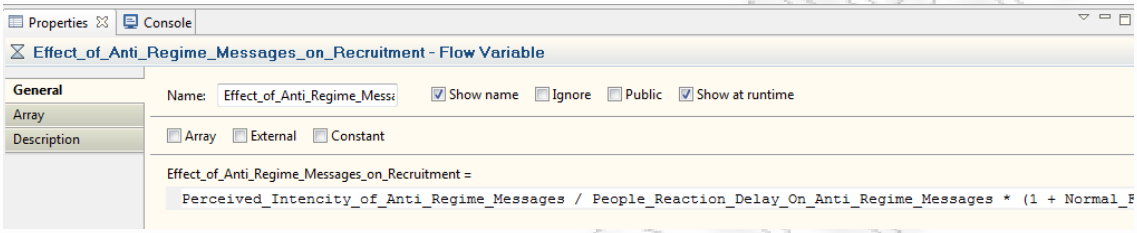


Εικόνα 9-32(β): Γενικές ιδιότητες της ροής Becoming_Dissident, όπου

$$Becoming\_Dissident = Recruits\_Through\_Social\_Network$$

Εικόνα 9-32(γ): Γενικές ιδιότητες της ροής Births, όπου

$$Births = \frac{People\_Birth}{1000 * Government\_Supporters}$$



Εικόνα 9-32(δ): Γενικές ιδιότητες της ροής Effect_of_Anti_Regime_Messages_on_Recruitment, όπου

$$\delta_1 = \frac{Perceived\_Intencity\_of\_Anti\_Regime\_Messages}{People\_Reaction\_Delay\_On\_Anti\_Regime\_Messages}$$

$$\delta_2 = (1 + Normal\_Frequency\_of\_Anti\_Regime\_Messages)$$

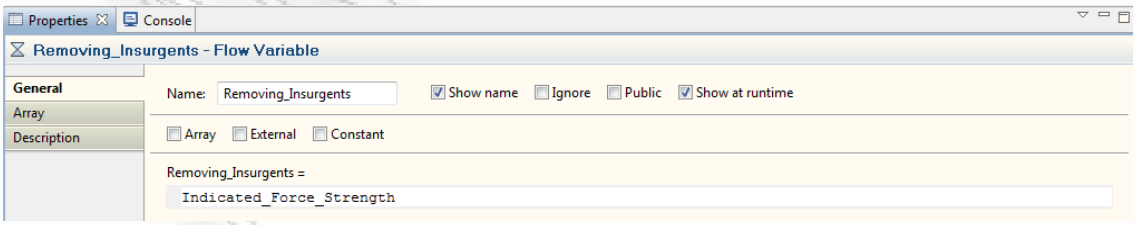
$$Effect\_of\_Anti\_Regime\_Messages\_on\_Recruitment = \delta_1 * \delta_2$$

Εικόνα 9-32(ε): Γενικές ιδιότητες της ροής Effect_of_Incidents_on_Anti_Regime_Messages, όπου

$$\varepsilon_1 = Incident\_Intensity * Strength\_of\_Incident\_Effect\_on\_Anti\_Regime\_Messages$$

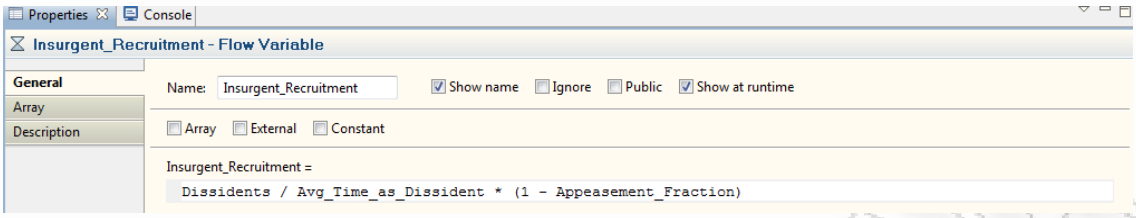
$$\varepsilon_2 = (1 + Normal\_Frequency\_of\_Anti\_Regime\_Messages)$$

$$Effect\_of\_Incidents\_on\_Anti\_Regime\_Messages = \varepsilon_1 * \varepsilon_2$$



Εικόνα 9-32(στ): Γενικές ιδιότητες της ροής Removing_Insurgents, όπου

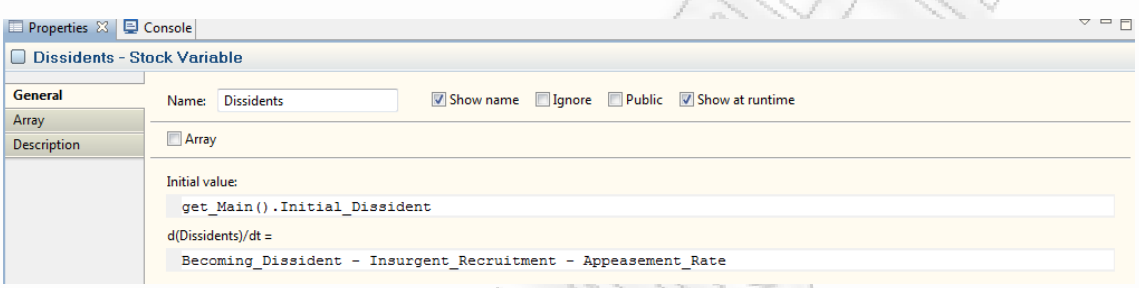
$$Removing\_Insurgents = Indicated\_Force\_Strength$$



Εικόνα 9-32(ζ): Γενικές ιδιότητες της ροής Insurgent_Recruitment, όπου

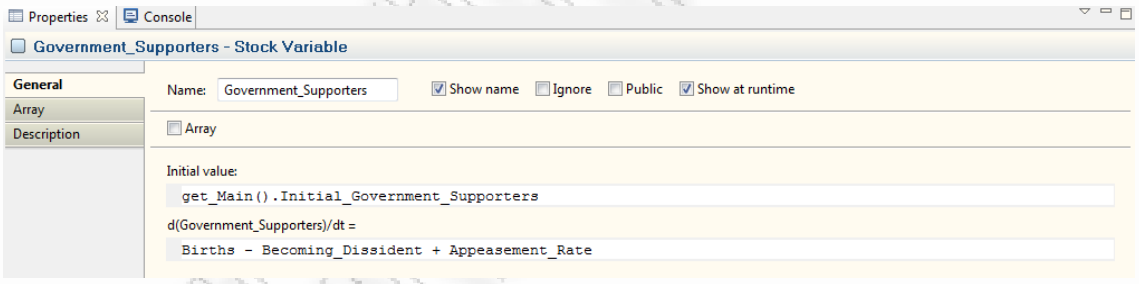
$$Insurgent\_Recruitment = \frac{Dissidents}{Avg\_Time\_as\_Dissident * (1 - Appeasement\_Fraction)}$$

Στις εικόνες 9-33(α) έως και 9-33(ε) παρουσιάζονται οι γενικές ιδιότητες των σημείων συσσώρευσης της κλάσης SDModel.



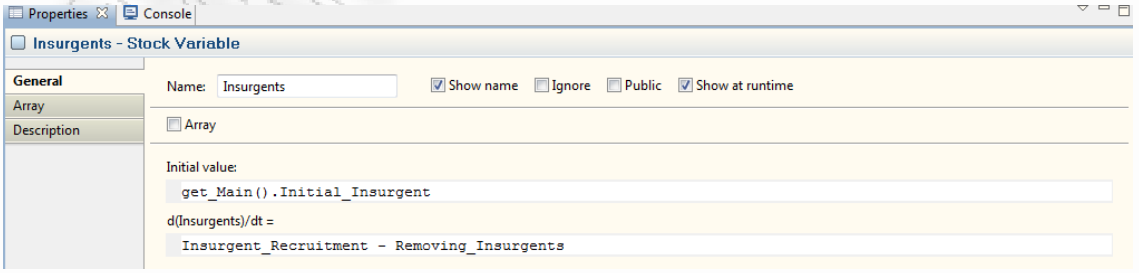
Εικόνα 9-33(α): Γενικές ιδιότητες του σημείου συσσώρευσης Dissidents, όπου

$$\frac{d(Dissidents)}{dt} = Becoming\_Dissident - Insurgent\_Recruitment - Appeasement\_Rate$$



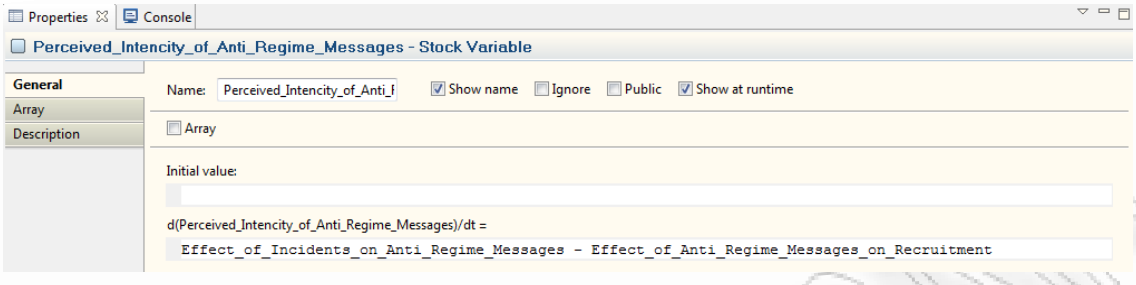
Εικόνα 9-33(β): Γενικές ιδιότητες του σημείου συσσώρευσης Government_Supporters, όπου

$$\frac{d(Government\_Supporters)}{dt} = Births - Becoming\_Dissident + Appeasement\_Rate$$



Εικόνα 9-33(γ): Γενικές ιδιότητες του σημείου συσσώρευσης Insurgents, όπου

$$\frac{d(Insurgents)}{dt} = Insurgent\_Recruitment - Removing\_Insurgents$$

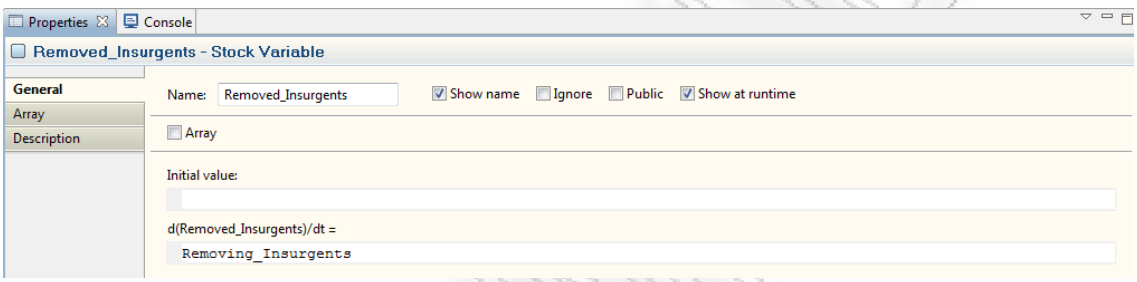


Εικόνα 9-33(δ): Το σημείο συσσώρευσης Perceived_Intencity_of_Anti_Regime_Messages και οι γενικές του ιδιότητες, όπου

$$\frac{d(Perceived\_Intencity\_of\_Anti\_Regime\_Messages)}{dt} = a - b$$

$$a = Effect\_of\_Incidents\_on\_Anti\_Regime\_Messages$$

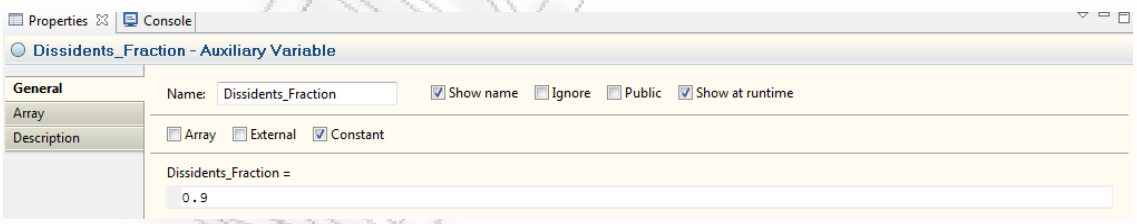
$$b = Effect\_of\_Anti\_Regime\_Messages\_on\_Recruitment$$



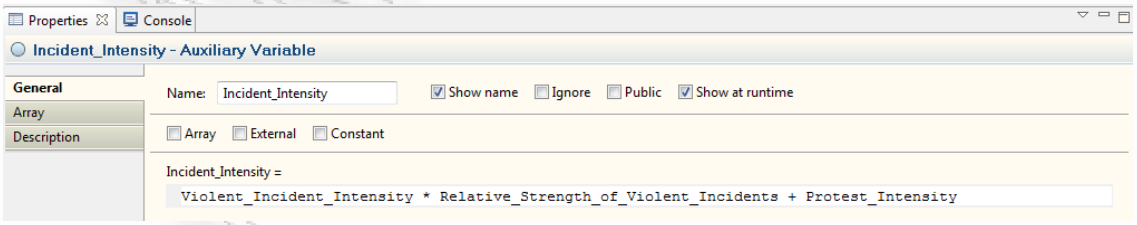
Εικόνα 9-33(ε): Γενικές ιδιότητες του σημείου συσσώρευσης Removed_Insurgents, όπου

$$\frac{d(Removed\_Insurgents)}{dt} = Removing\_Insurgents$$

Στις εικόνες 9-34(α) έως και 9-34(ιε) παρουσιάζονται οι γενικές ιδιότητες των βοηθητικών παραμέτρων της κλάσης SDModel.



Εικόνα 9-34(α): Γενικές ιδιότητες της βοηθητικής παραμέτρου Dissidents_Fraction.



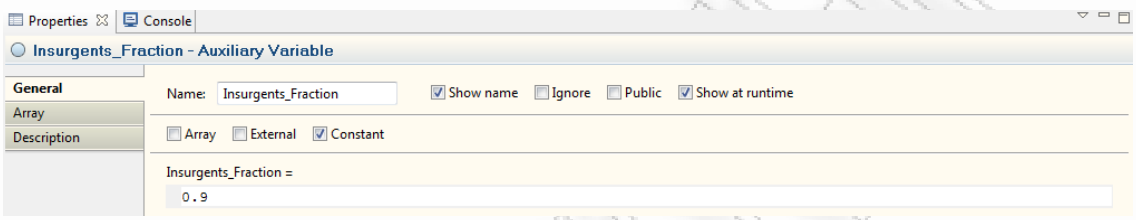
Εικόνα 9-34(β): Γενικές ιδιότητες της βοηθητικής παραμέτρου Incident_Intensity.

$$Incident\_Intensity = a + Protest\_Intensity$$

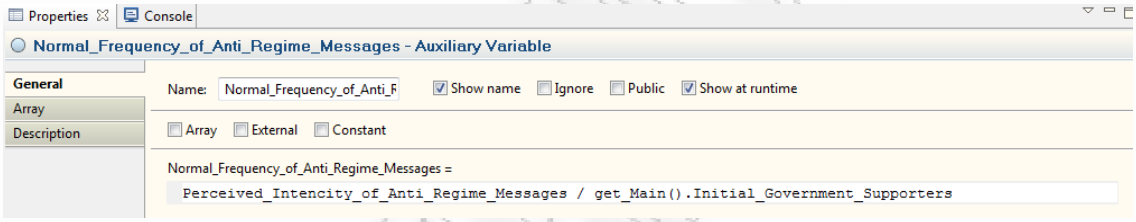
$$a = Violent\_Incident\_Intensity * Relative\_Strength\_of\_Violent\_Incidents$$

Εικόνα 9-34(γ): Γενικές ιδιότητες της βοηθητικής παραμέτρου Indicated_Force_Strength.

$$Indicated\_Force\_Strength = \frac{Removal\_Effectiveness * Insurgents}{Desired\_Time\_to\_Remove\_Insurgents}$$

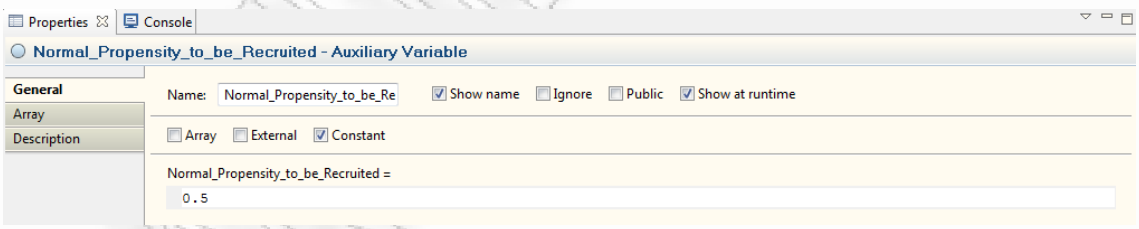


Εικόνα 9-34(δ): Γενικές ιδιότητες της βοηθητικής παραμέτρου Insurgents_Fraction.

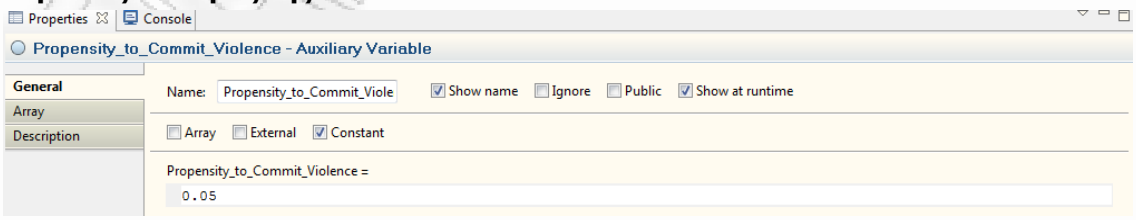


Εικόνα 9-34(ε): Η βοηθητική παράμετρος Normal_Frequency_of_Anti_Regime_Messages και οι γενικές ιδιότητες της, όπου

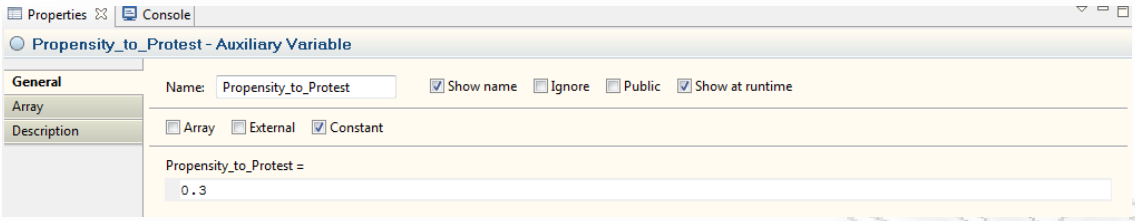
$$Normal\_Frequency\_of\_Anti\_Regime\_Messages = \frac{Perceived\_Intensity\_of\_Anti\_Regime\_Messages}{get\_Main().Initial\_Government\_Supporters}$$



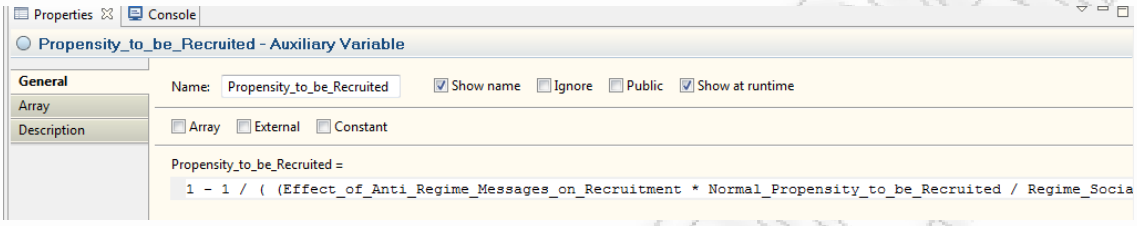
Εικόνα 9-34(στ): Η βοηθητική παράμετρος Normal_Propensity_to_be_Recruited και οι γενικές ιδιότητες της.



Εικόνα 9-34(ζ): Γενικές ιδιότητες της βοηθητικής παραμέτρου Propensity_to_Commit_Violence.



Εικόνα 9-34(η): Γενικές ιδιότητες της βοηθητικής παραμέτρου Propensity_to_Protest.



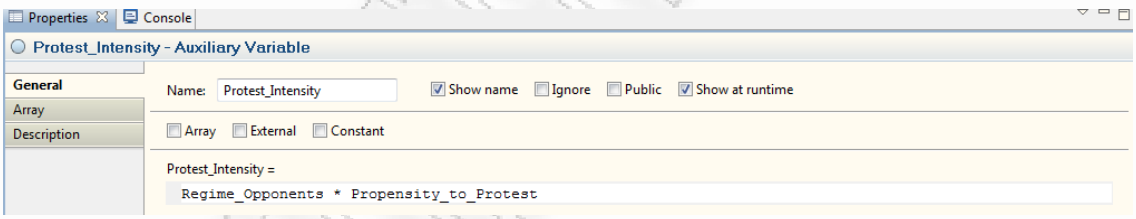
Εικόνα 9-34(θ): Γενικές ιδιότητες της βοηθητικής παραμέτρου Propensity_to_Recruited, όπου

$$\zeta_1 = Effect\_of\_Anti\_Regime\_Messages\_on\_Recruitment * Normal\_Propensity\_to\_be\_Recruited$$

$$\zeta_2 = \frac{Regime\_Social\_and\_Economic\_Efforts}{(Dissidents + Insurgents)}$$

$$\zeta_3 = \frac{\zeta_1}{\zeta_2}$$

$$Propensity\_to\_be\_Recruited = 1 - \frac{1}{(\zeta_3 * \zeta_3 + 1)}$$



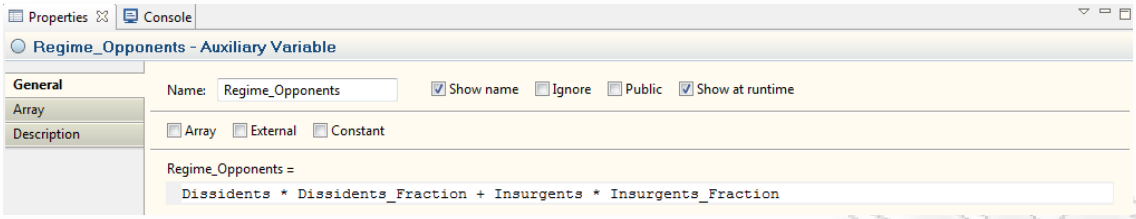
Εικόνα 9-34(ι): Γενικές ιδιότητες της βοηθητικής παραμέτρου Protest_Intensity.

Εικόνα 9-34(ια): Γενικές ιδιότητες της βοηθητικής παραμέτρου Recruits_Through_Social_Network, όπου

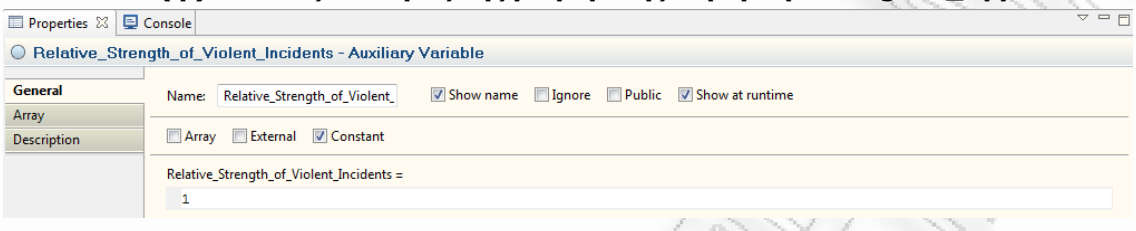
$$\eta_1 = Propensity\_to\_be\_Recruited * Government\_Supporters * Regime\_Opponents * Contact\_Rate$$

$$\eta_2 = Regime\_Opponents + Government\_Supporters$$

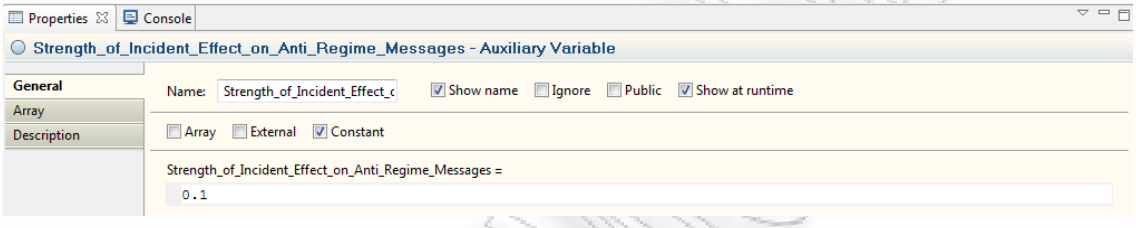
$$Recruits\_Through\_Social\_Network = \frac{\eta_1}{\eta_2}$$



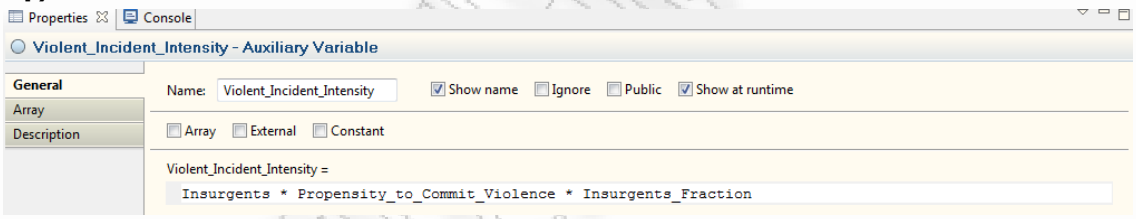
Εικόνα 9-34(ιβ): Γενικές ιδιότητες της βοηθητικής παραμέτρου Regime_Opponents.



Εικόνα 9-34(ιγ): Η βοηθητική παράμετρος Relative_Strength_of_Violent_Incidents και οι γενικές της ιδιότητες.



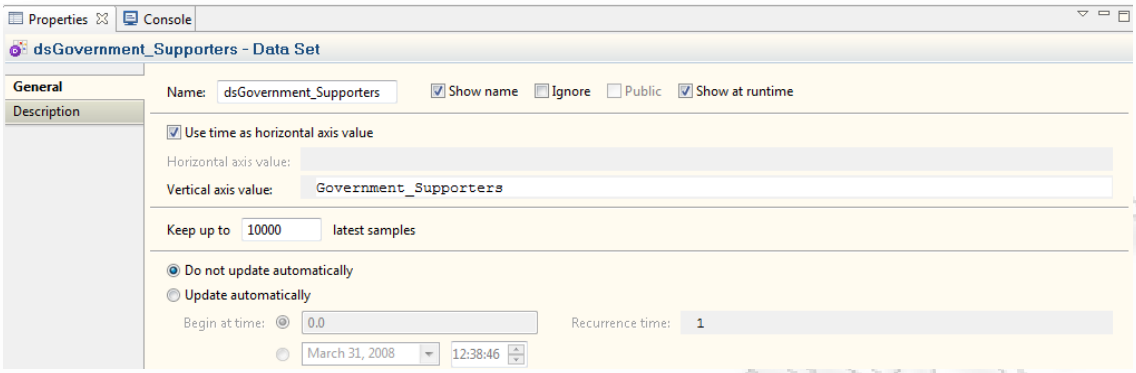
Εικόνα 9-34(ιδ): Η βοηθητική παράμετρος Strength_of_Incident_Effect_on_Anti_Regime_Messages και οι γενικές ιδιότητες της.



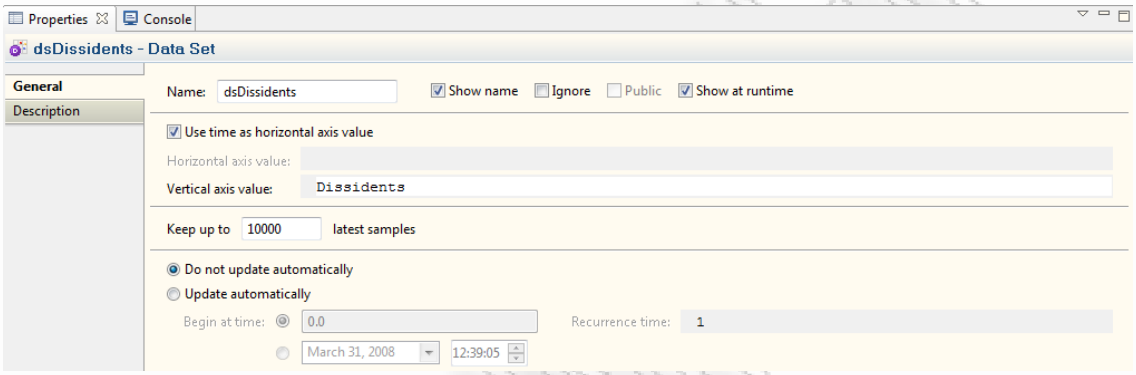
Εικόνα 9-34(ιε): Γενικές ιδιότητες της βοηθητικής παραμέτρου Violent_Incident_Intensity, όπου

$$Violent\_Incident\_Intensity = Insurgents * Propensity\_to\_Commit\_Violence * Insurgents\_Fraction$$

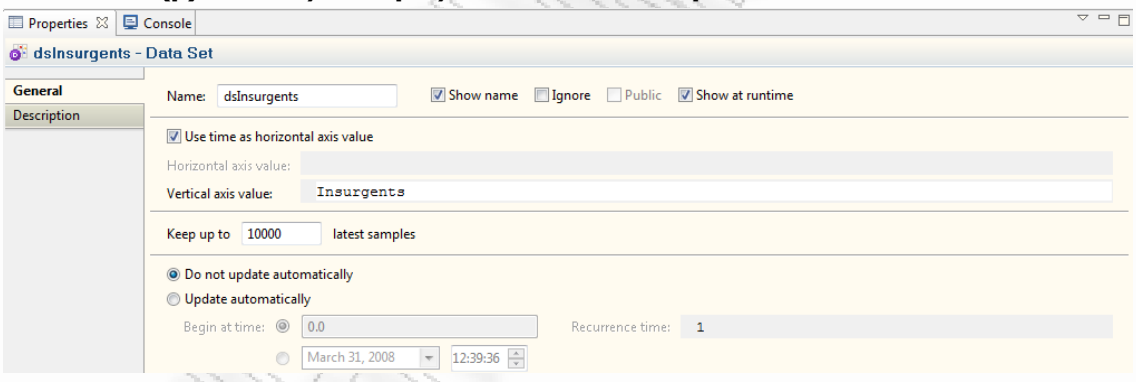
Στις εικόνες 9-35(α) έως και 9-35(δ) παρουσιάζονται οι γενικές ιδιότητες των συνόλων δεδομένων dsGovernment_Supporters, dsDissidents, dsInsurgents και dsRemoved_Insurgents, αντίστοιχα και στην εικόνα 9-36 οι γενικές ιδιότητες του χρονικού διαγράμματος plot1 της κλάσης SDModel.



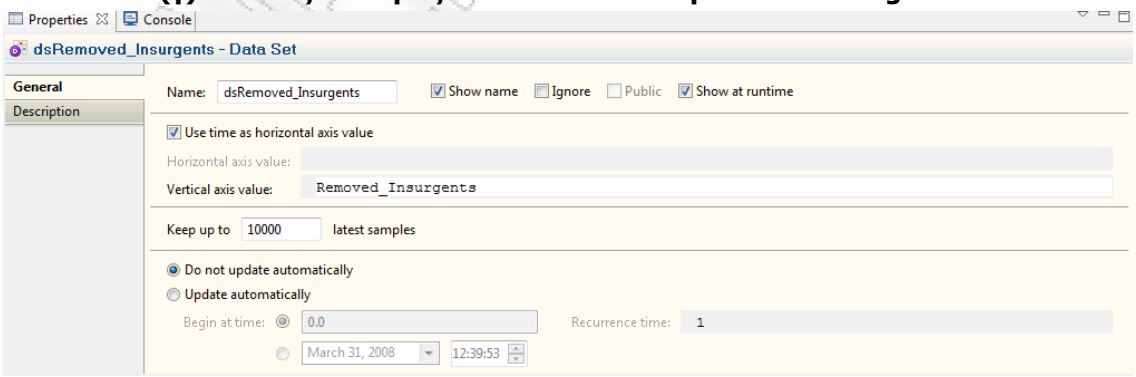
Εικόνα 9-35(α): Γενικές ιδιότητες του συνόλου δεδομένων dsGovernment_Supporters.



Εικόνα 9-35(β): Γενικές ιδιότητες του συνόλου δεδομένων dsDissidents.



Εικόνα 9-35(γ): Γενικές ιδιότητες του συνόλου δεδομένων dsInsurgents.



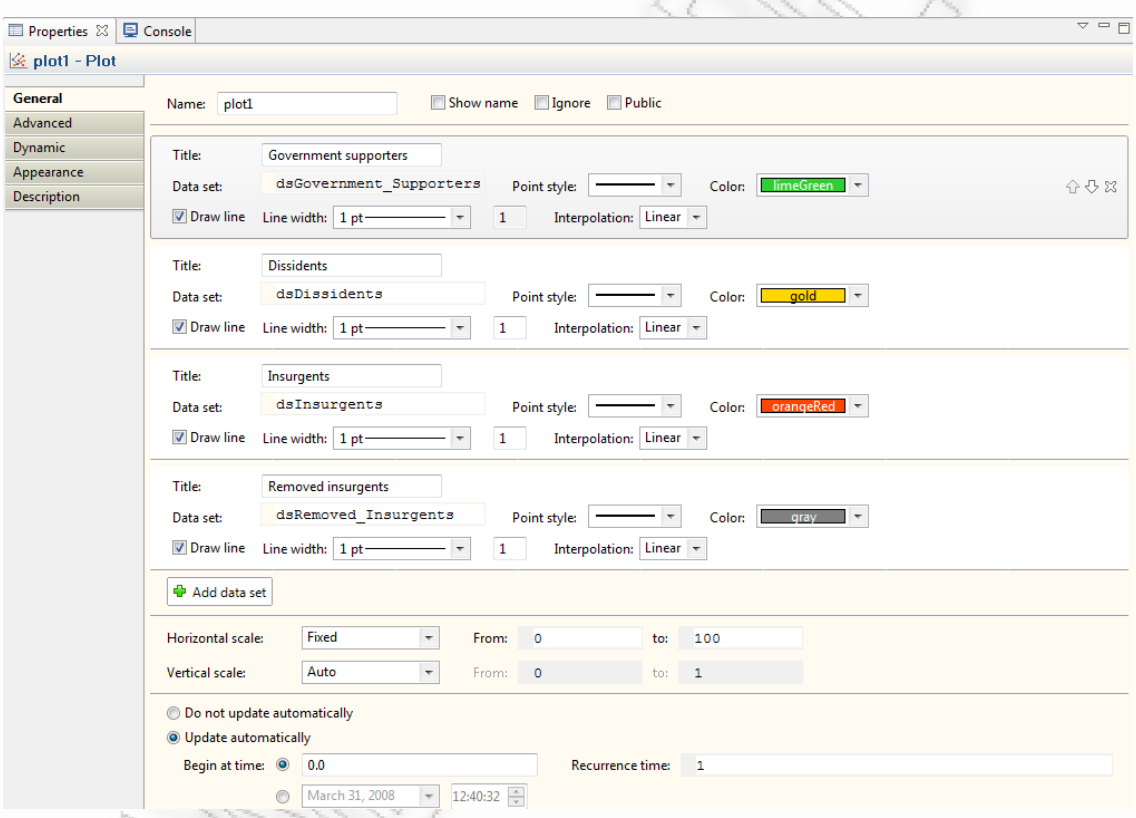
Εικόνα 9-35(δ): Γενικές ιδιότητες του συνόλου δεδομένων dsRemoved_Insurgents.

Όπως και στην περίπτωση των κλάσεων ABModel και Main, η ομάδα των συνδέσμων που χρησιμεύει για την μετάβαση μεταξύ των κλάσεων κατά την προσομοίωση εκτελούνται από τα παρακάτω τμήματα κώδικα:

ABModel
getEngine().getPresentation().setPresentable(((Main)getEngine().getRoot()).aM);
getPresentation().getPanel().setOffsets(0,0);

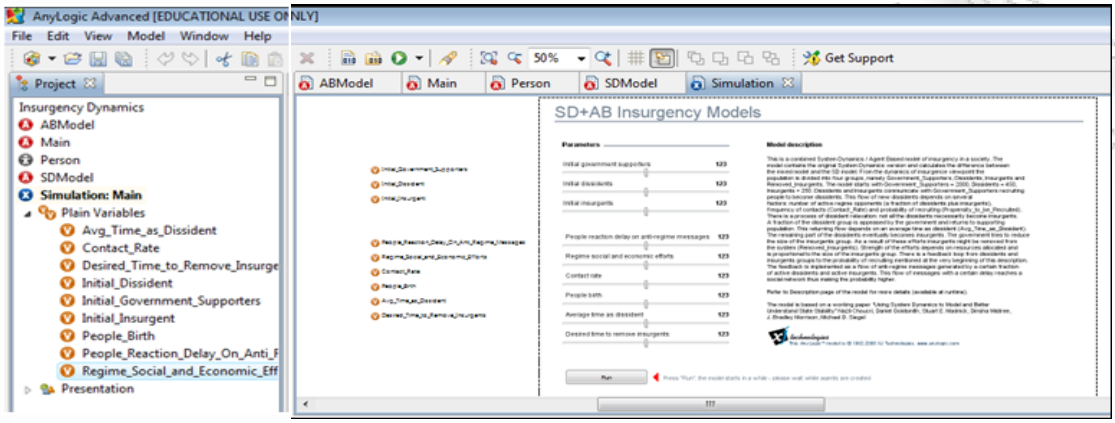
Comparison (Main)
getEngine().getPresentation().setPresentable(((Main)getEngine().getRoot()));
getPresentation().getPanel().setOffsets(0,0);

Description
getEngine().getPresentation().setPresentable(((Main)getEngine().getRoot()));
getPresentation().getPanel().setOffsets(0,1500);



Εικόνα 9-36: Γενικές ιδιότητες του διαγράμματος χρόνου plot1.

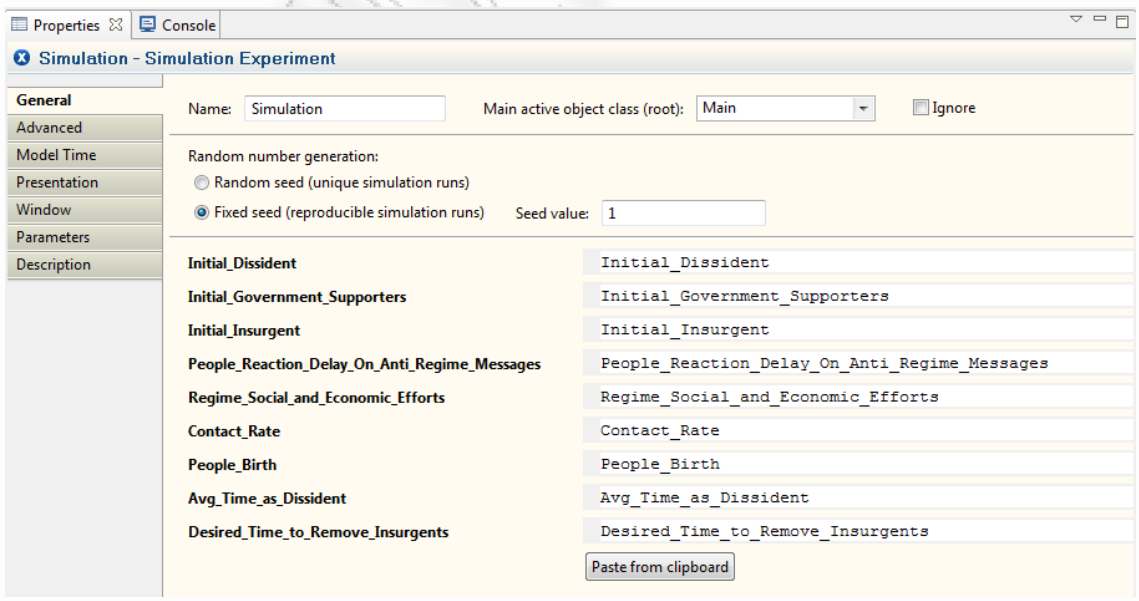
9.5 Η κλάση προσομοίωσης του έργου, Simulation:Main



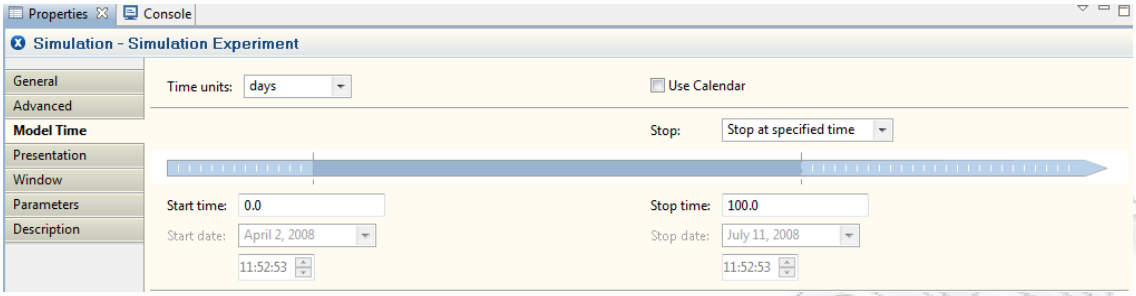
Εικόνα 9-37(α): Η κλάση της προσομοίωσης Simulation:Main.

Η κλάση, της προσομοίωσης του έργου Insurgency Dynamics, Simulation:Main στην εικόνα 9-37(α) αποτελείται από εννέα (9) απλές μεταβλητές, εικόνα 9-37(β), οι οποίες είναι οι: Avg_Time_as_Dissident, Contact_Rate, Desired_Time_to_Remove_Insurgents, Initial_Dissident, People_Reaction_Delay_On_Anti_Regime_Messages, Regime_Social_and_Economic_Efforts, Initial_Government_Supporters, People_Birth, και Initial_Insurgent. Την κλάση αποτελούν επίσης ένα (1) κομβίο για την έναρξη της προσομοίωσης, μια (1) εικόνα με το λογότυπο της εταιρείας στην οποία ανήκει το Λογισμικό, μερικά στοιχεία κειμένου και εννέα (9) ράβδοι κύλισης οι οποίοι είναι συνδεδεμένες με τις παραπάνω απλές μεταβλητές.

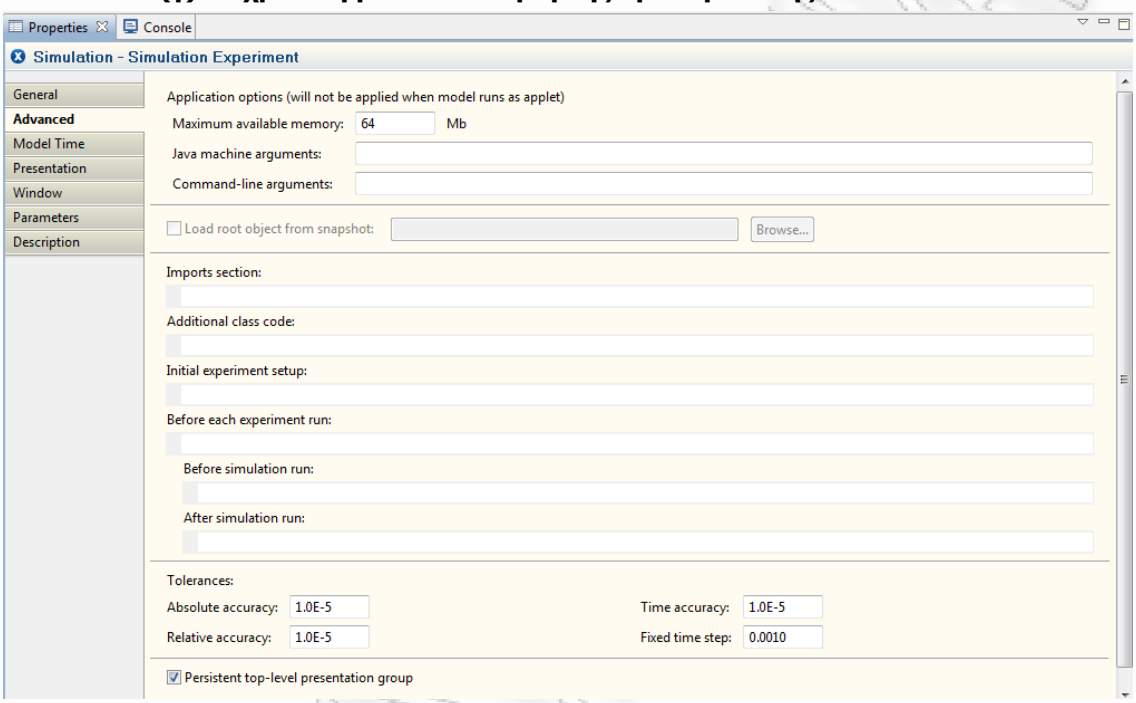
Η προσομοίωση ξεκινά με αρχικές τιμές τις: Dissidents = 450, Insurgents = 250 και Government_Supporters = 2000. Στις εικόνες 9-37(γ) έως και 9-37(στ) δίνονται οι ιδιότητες της κλάσης και οι προκαθορισμένες παράμετροι της προσομοίωσης ενώ στις εικόνες 9-38(α) έως και 9-38(θ) παρουσιάζονται οι ιδιότητες των απλών μεταβλητών.



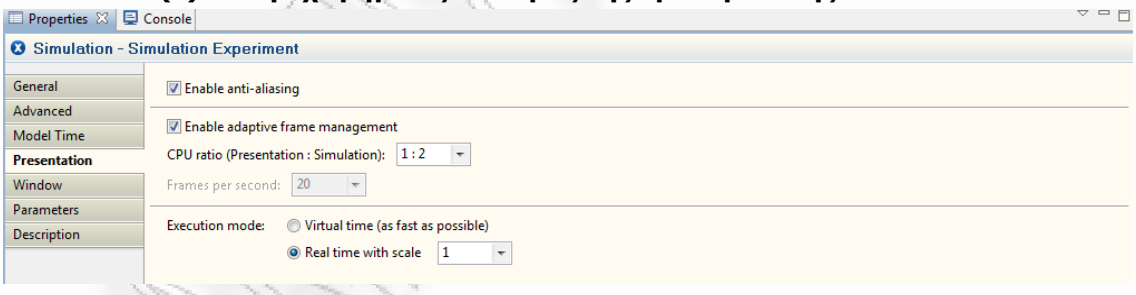
Εικόνα 9-37(β): Οι αρχικές τιμές των παραμέτρων προσομοίωσης στην κλάση της προσομοίωσης, συγκεντρωμένες.



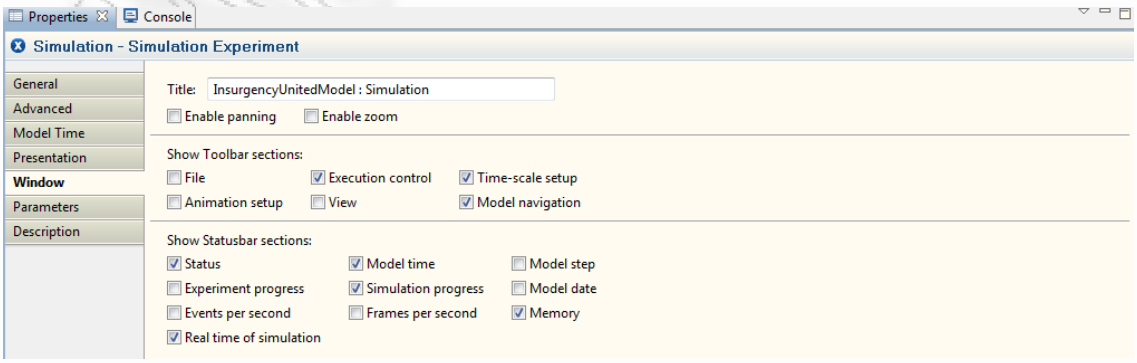
Εικόνα 9-37(γ): Η χρονική μοντελοποίηση της προσομοίωσης.



Εικόνα 9-37(δ): Οι προχωρημένες ιδιότητες της προσομοίωσης.

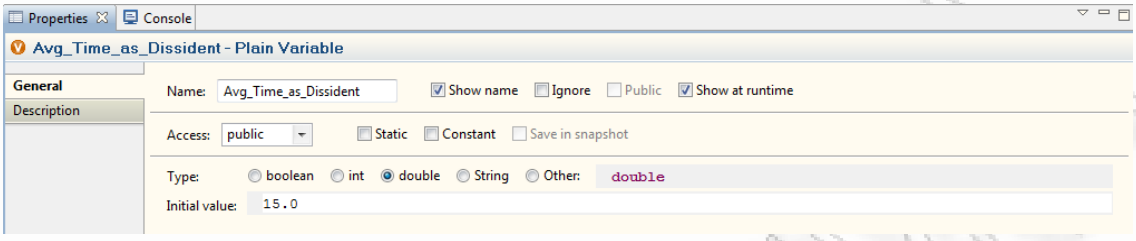


Εικόνα 9-37(ε): Οι ιδιότητες της παρουσίασης του μοντέλου κατά την προσομοίωση.

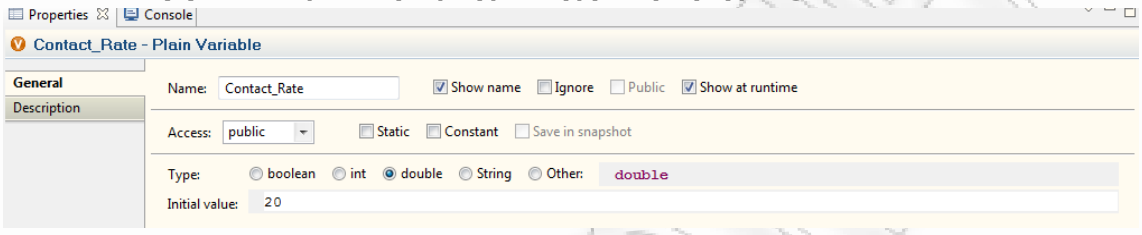


Εικόνα 9-37(στ): Οι ιδιότητες του παραθύρου της προσομοίωσης.

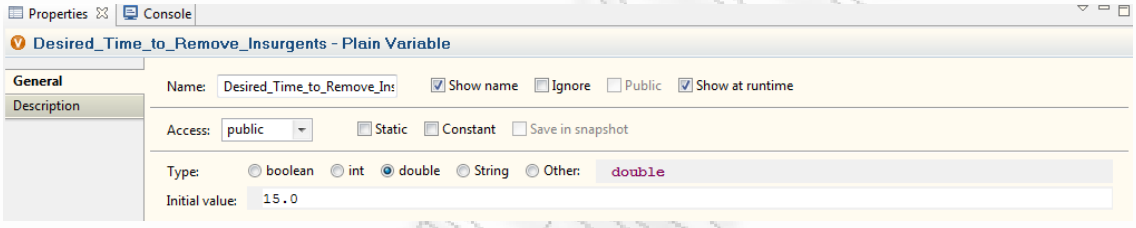
Στις εικόνες 9-39(α) έως και 9-39(θ) δίνονται οι δυναμικές ιδιότητες των στοιχείων κειμένου, οι οποίες συνδέονται με τις παραμέτρους προσομοίωσης αλλά και με τις ράβδους κύλισης.



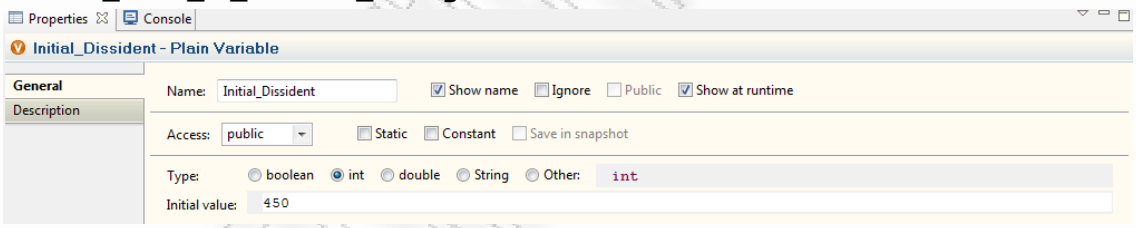
Εικόνα 9-38(α): Γενικές ιδιότητες της απλής μεταβλητής Avg_Time_as_Dissident.



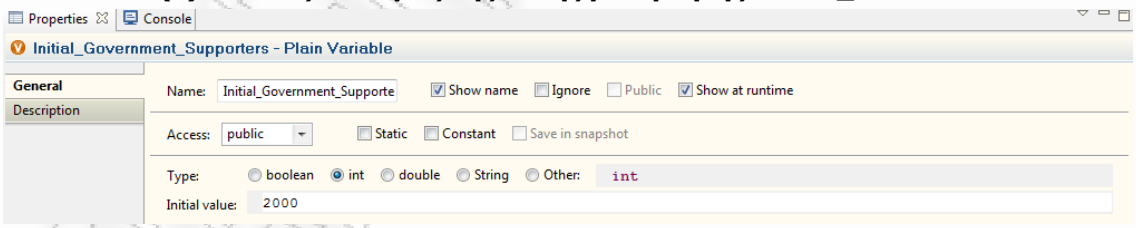
Εικόνα 9-38(β): Γενικές ιδιότητες της απλής μεταβλητής Contact_Rate.



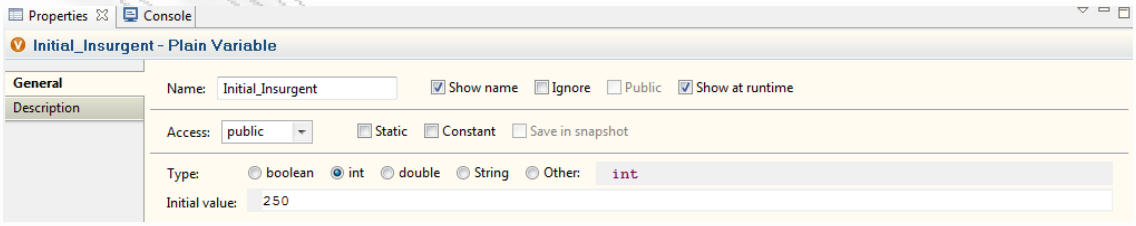
Εικόνα 9-38(γ): Γενικές ιδιότητες της απλής μεταβλητής Desired_Time_to_Remove_Insurgents.



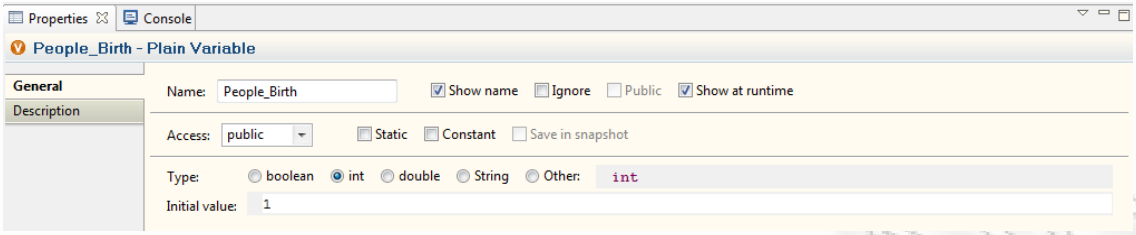
Εικόνα 9-38(δ): Γενικές ιδιότητες της απλής μεταβλητής Initial_Dissident.



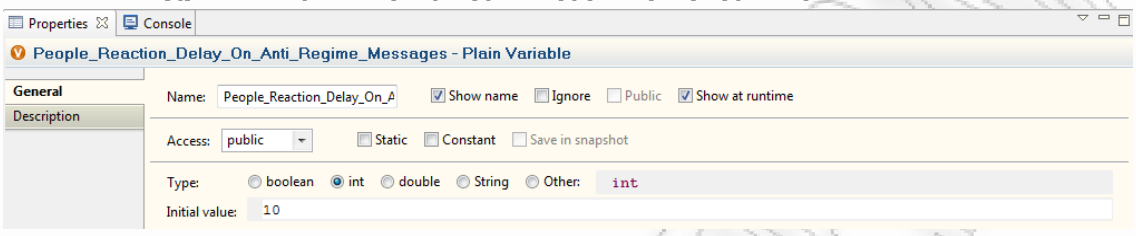
Εικόνα 9-38(ε): Γενικές ιδιότητες της απλής μεταβλητής Initial_Government_Supporters.



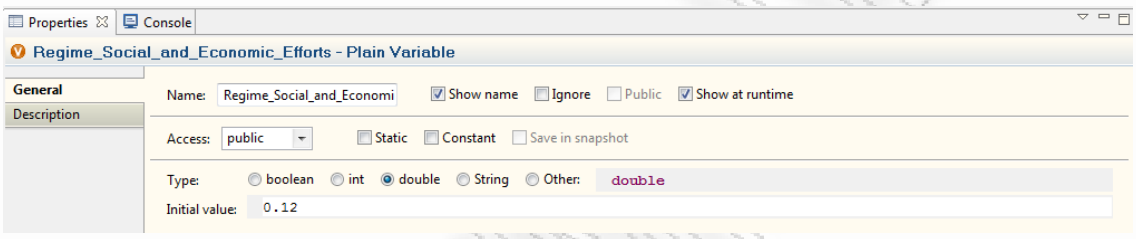
Εικόνα 9-38(στ): Γενικές ιδιότητες της απλής μεταβλητής Initial_Insurgent.



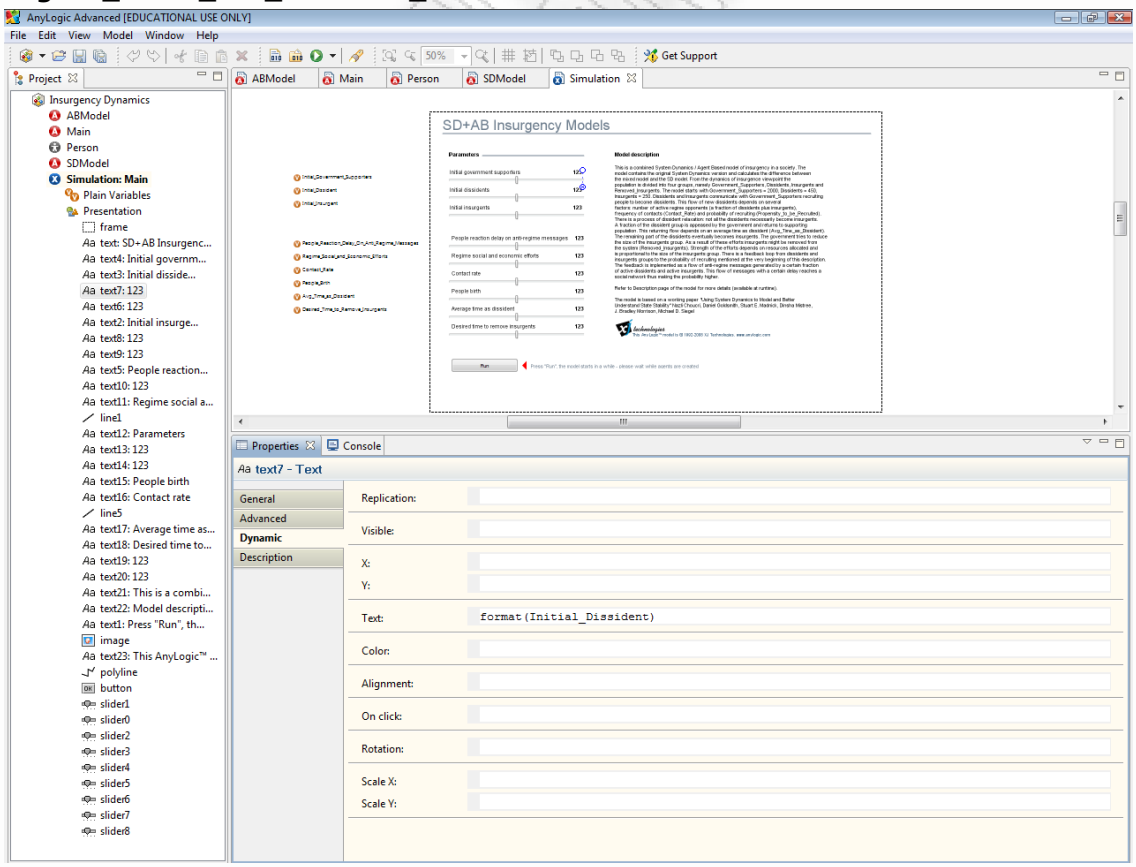
Εικόνα 9-38(ζ): Γενικές ιδιότητες της απλής μεταβλητής People_Birth.



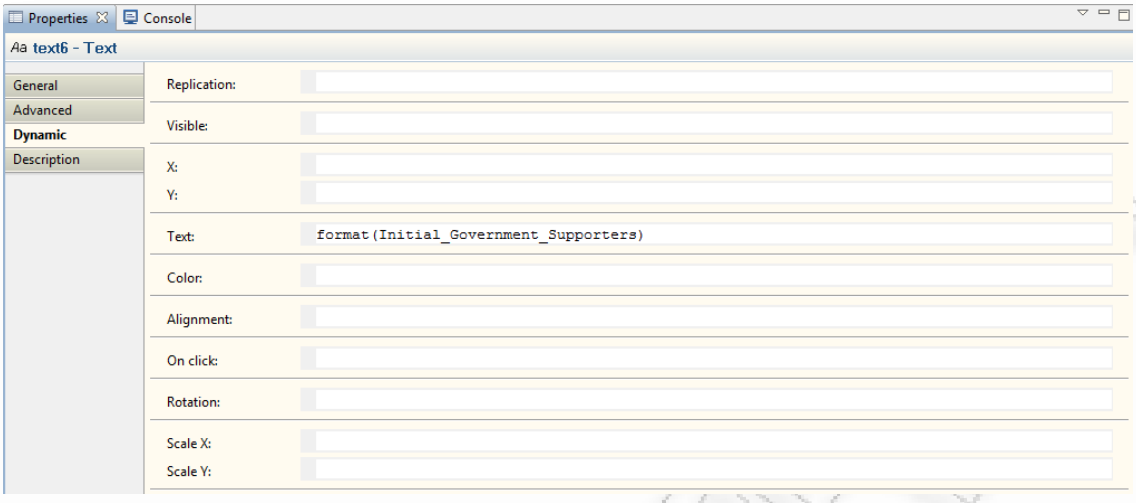
Εικόνα 9-38(η): Η απλή μεταβλητή People_Reaction_Delay_On_Anti_Regime_Messages και οι γενικές ιδιότητες της.



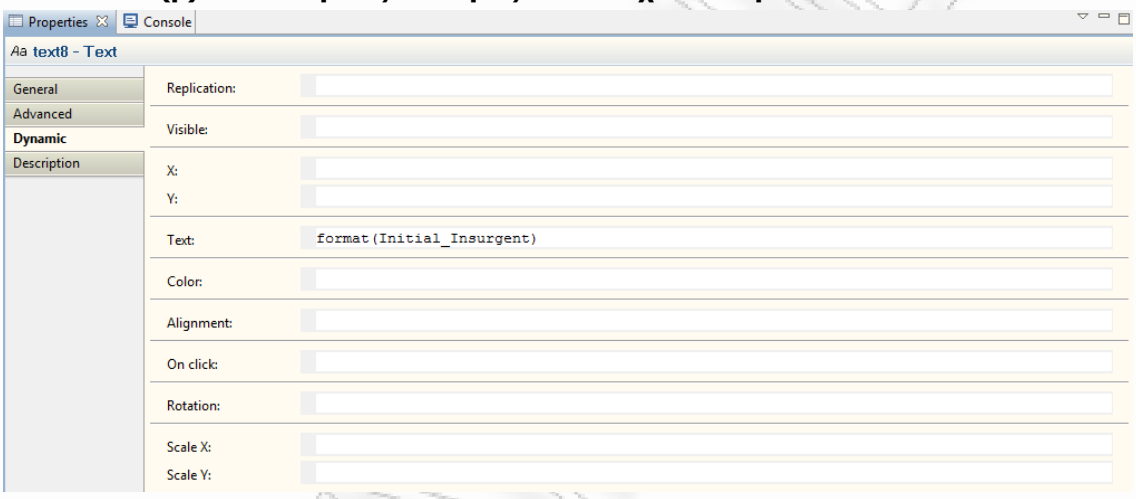
Εικόνα 9-38(θ): Γενικές ιδιότητες της απλής μεταβλητής Regime_Social_and_Economic_Efforts.



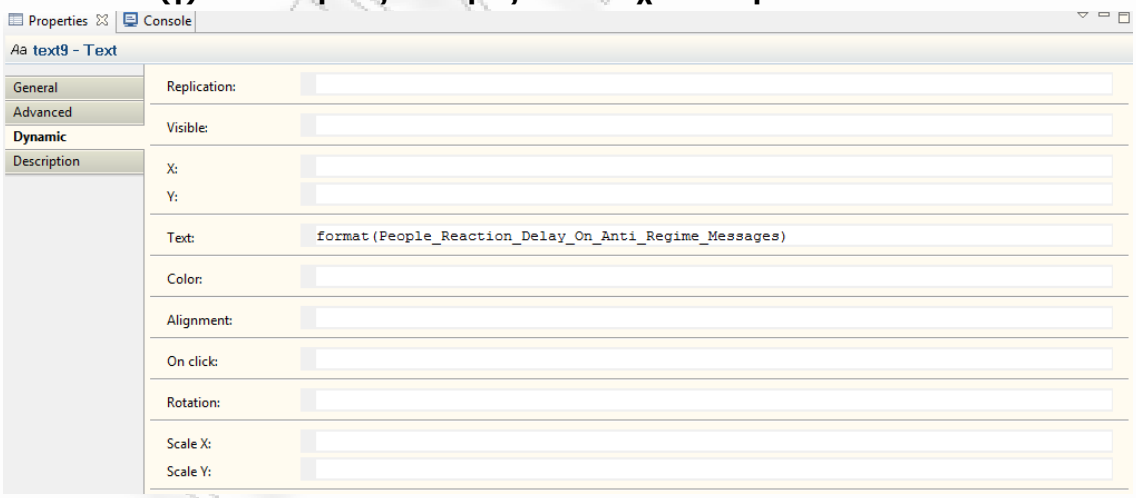
Εικόνα 9-39(α): Οι δυναμικές ιδιότητες του στοιχείου κειμένου text7 τύπου Text.



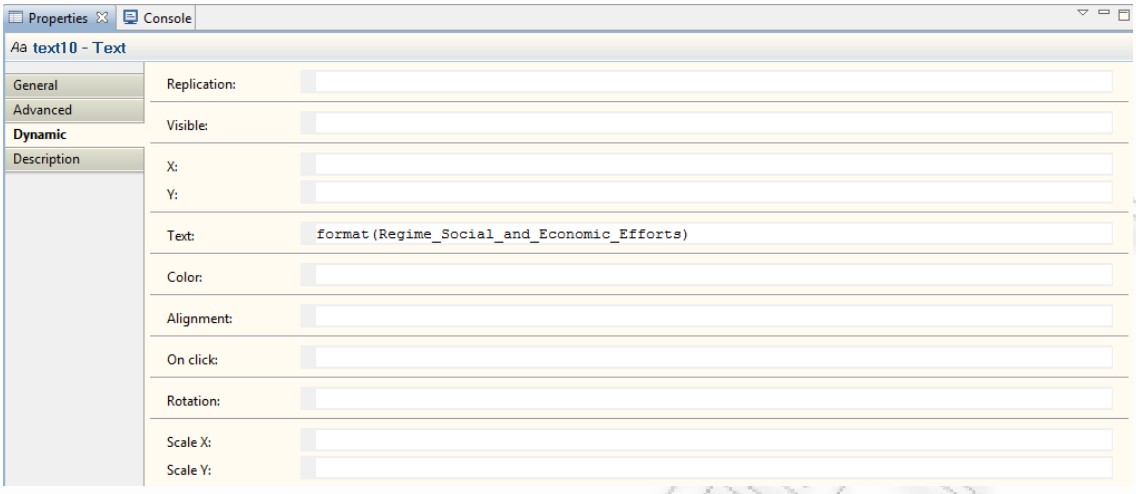
Εικόνα 9-39(β): Οι δυναμικές ιδιότητες του στοιχείου κειμένου text6 τύπου Text.



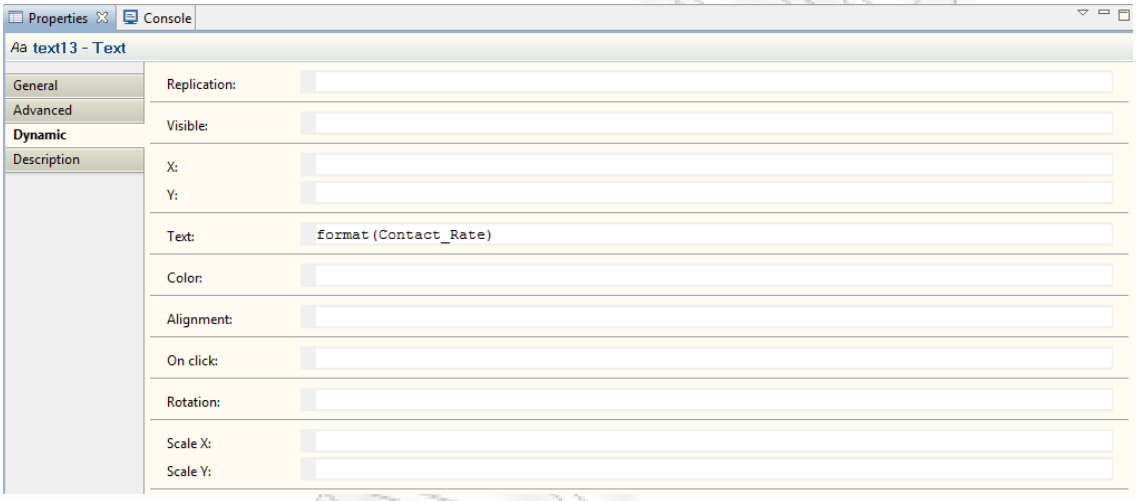
Εικόνα 9-39(γ): Οι δυναμικές ιδιότητες του στοιχείου κειμένου text8 τύπου Text.



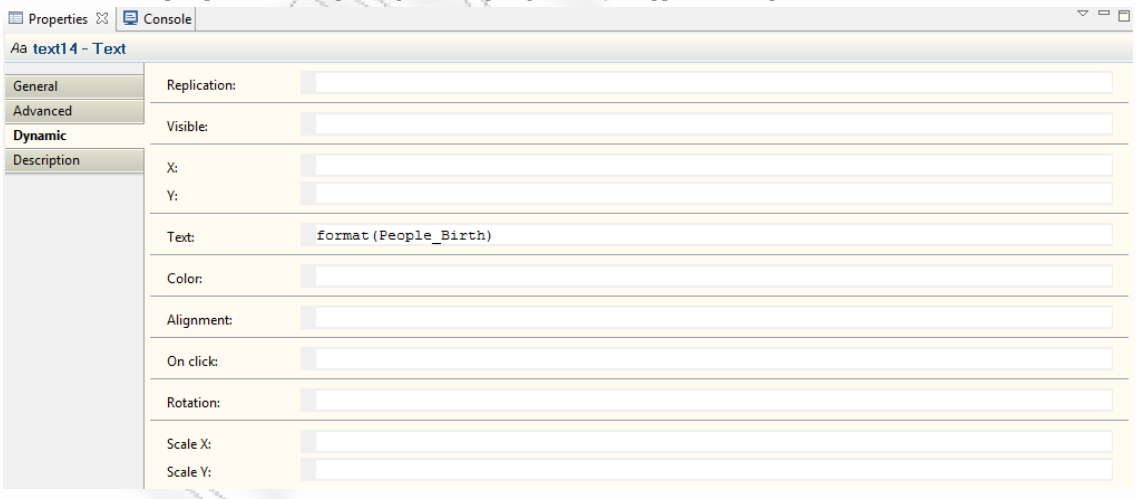
Εικόνα 9-39(δ): Οι δυναμικές ιδιότητες του στοιχείου κειμένου text9 τύπου Text.



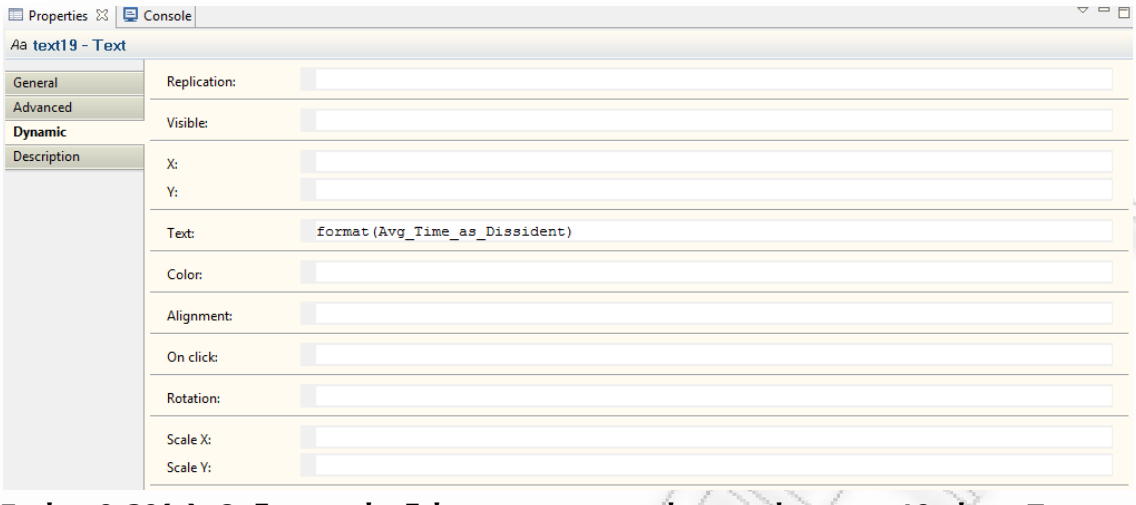
Εικόνα 9-39(ε): Οι δυναμικές ιδιότητες του στοιχείου κειμένου text10 τύπου Text.



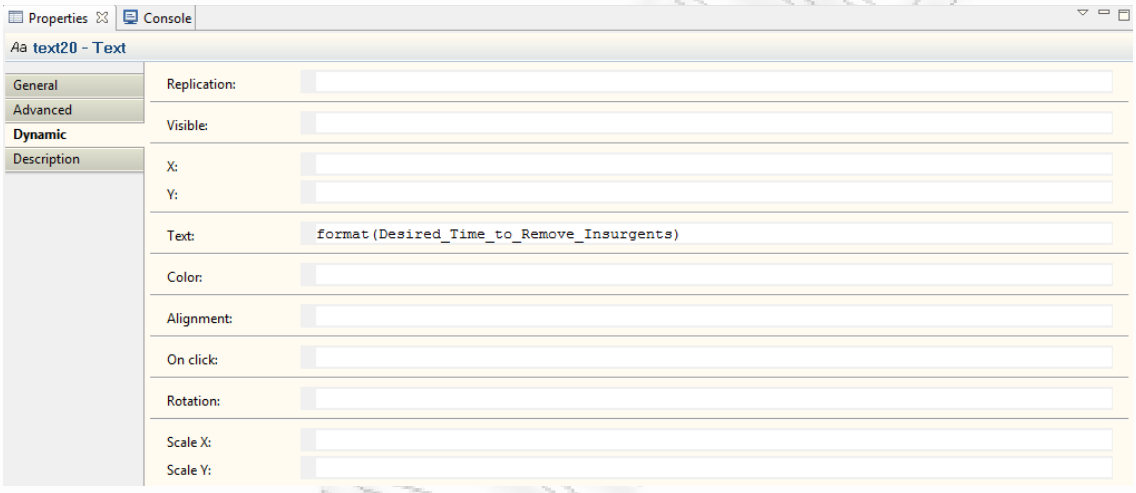
Εικόνα 9-39(στ): Οι δυναμικές ιδιότητες του στοιχείου κειμένου text13 τύπου Text.



Εικόνα 9-39(ζ): Οι δυναμικές ιδιότητες του στοιχείου κειμένου text14 τύπου Text.

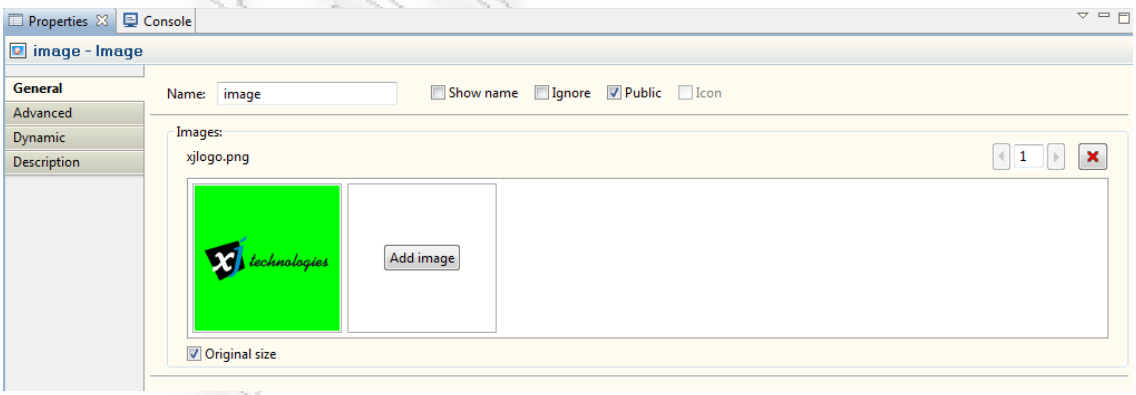


Εικόνα 9-39(η): Οι δυναμικές ιδιότητες του στοιχείου κειμένου text19 τύπου Text.

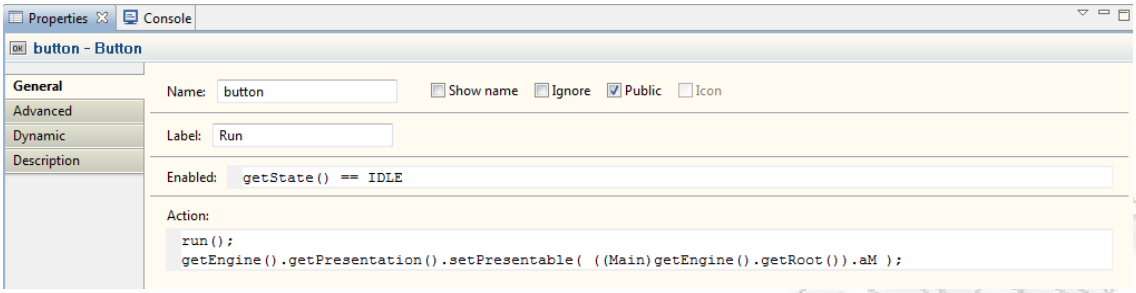


Εικόνα 9-39(θ): Οι δυναμικές ιδιότητες του στοιχείου κειμένου text20 τύπου Text.

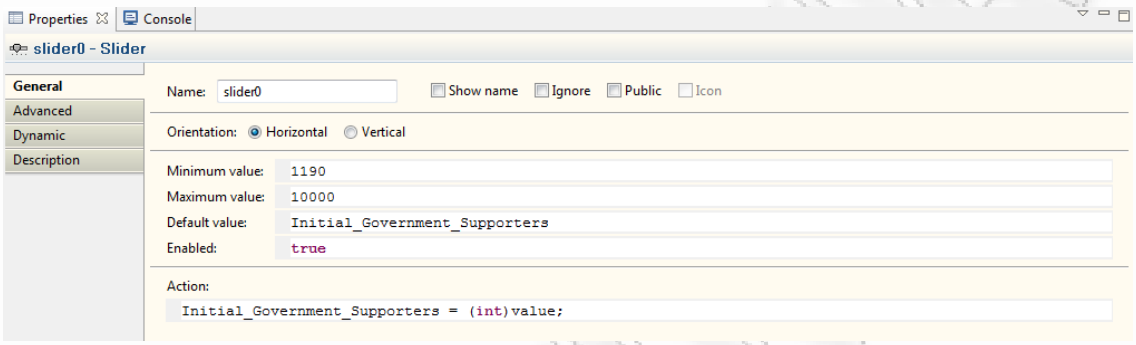
Στην εικόνα 9-40 εμφανίζονται οι γενικές ιδιότητες της εικόνας image τύπου Image, στην εικόνα 9-41 δίνονται οι γενικές ιδιότητες του κομβίου button τύπου Button και στις εικόνες 9-42(α) έως και 9-42(θ) παρουσιάζονται οι γενικές ιδιότητες των ράβδων κύλισης.



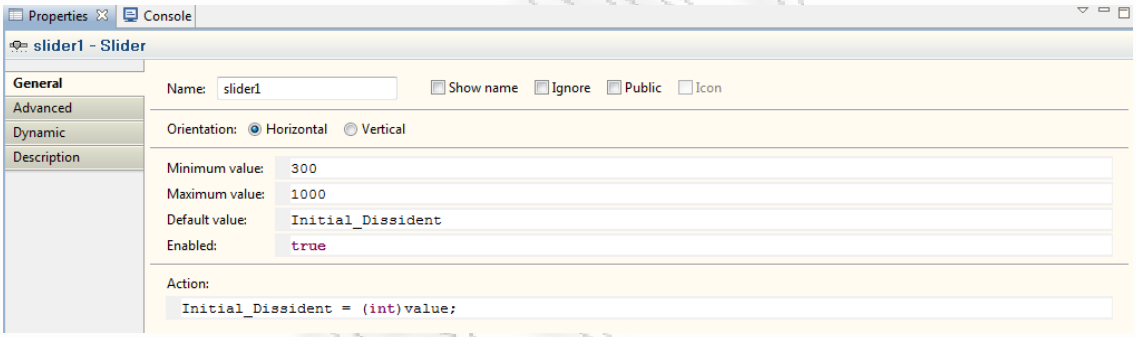
Εικόνα 9-40: Γενικές ιδιότητες της εικόνας image τύπου Image.



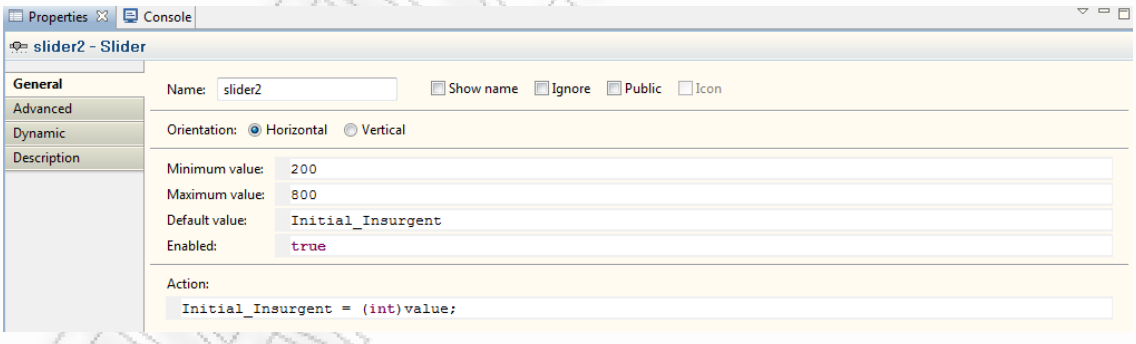
Εικόνα 9-41: Γενικές ιδιότητες του κομβίου button τύπου Button.



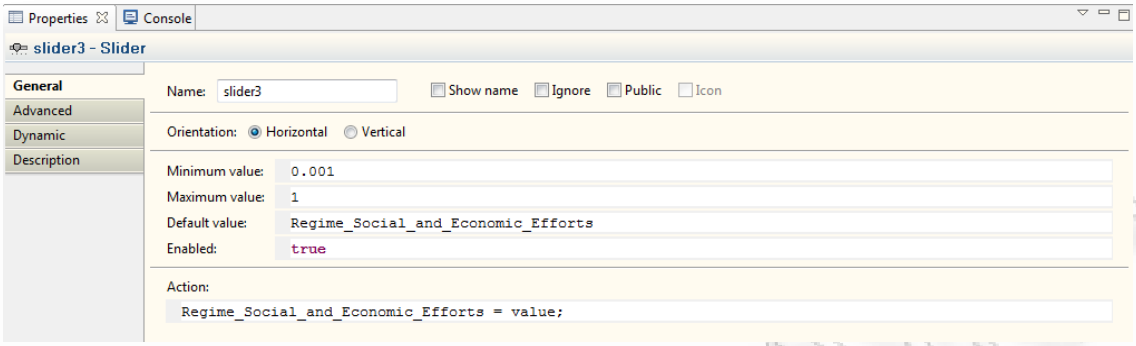
Εικόνα 9-42(α): Γενικές ιδιότητες της ράβδου κύλισης slider0.



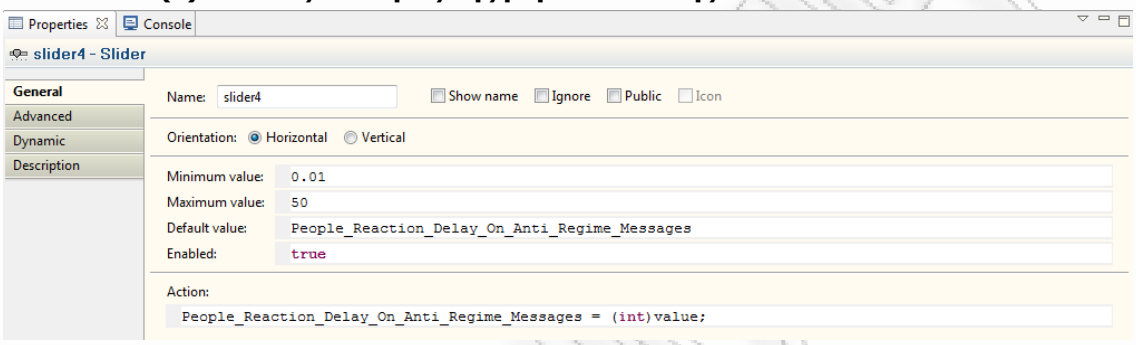
Εικόνα 9-42(β): Γενικές ιδιότητες της ράβδου κύλισης slider1.



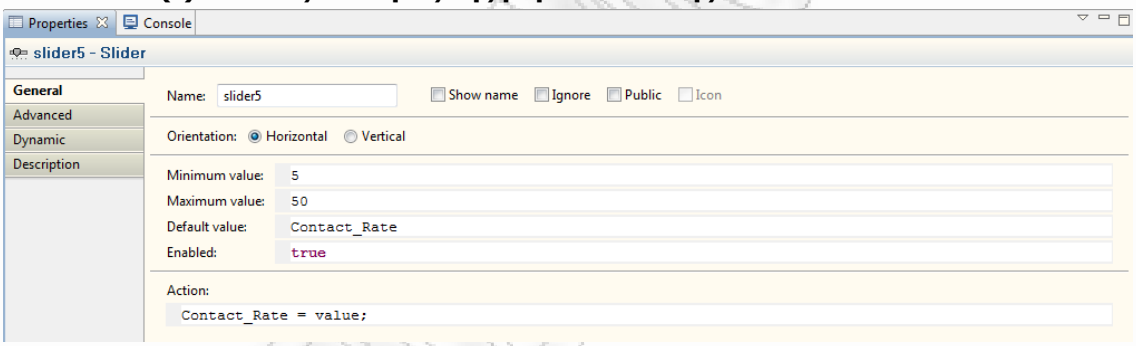
Εικόνα 9-42(γ): Γενικές ιδιότητες της ράβδου κύλισης slider2.



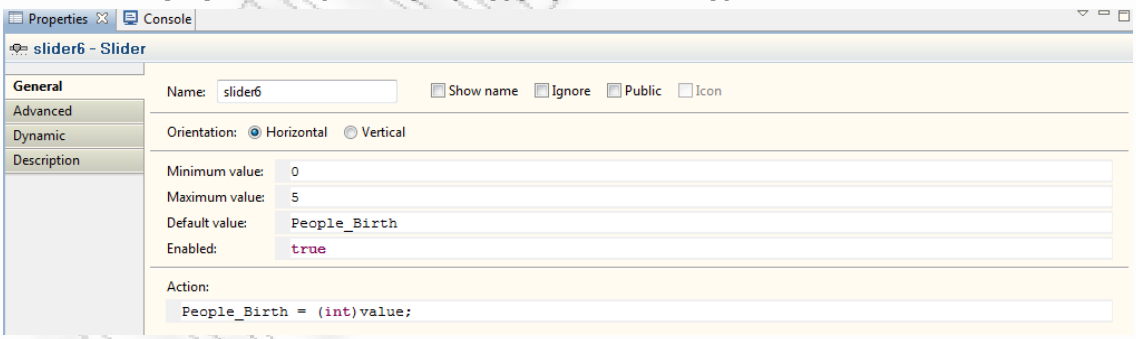
Εικόνα 9-42(δ): Γενικές ιδιότητες της ράβδου κύλισης slider3.



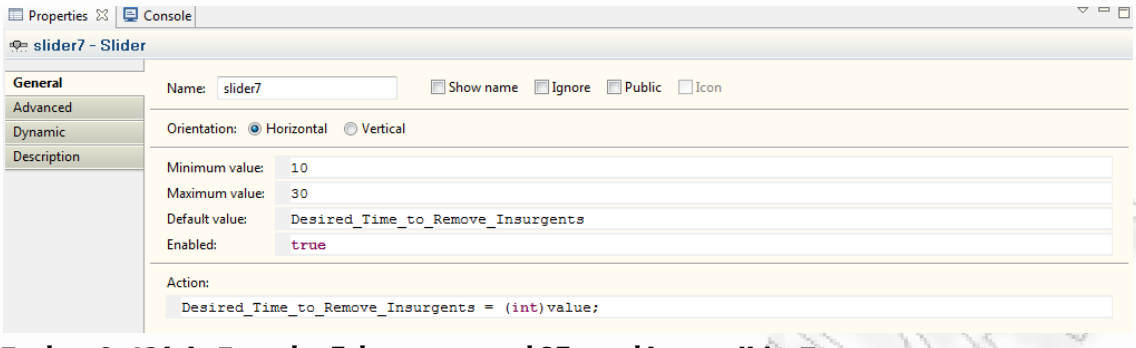
Εικόνα 9-42(ε): Γενικές ιδιότητες της ράβδου κύλισης slider4.



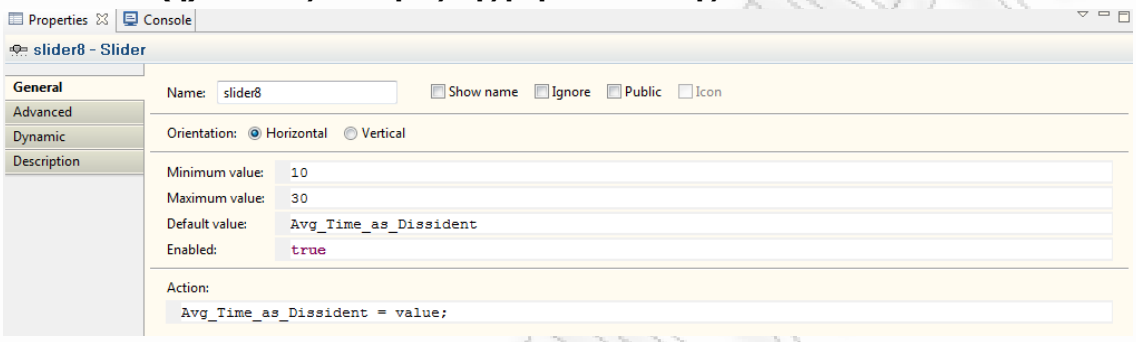
Εικόνα 9-42(στ): Γενικές ιδιότητες της ράβδου κύλισης slider5.



Εικόνα 9-42(ζ): Γενικές ιδιότητες της ράβδου κύλισης slider6.



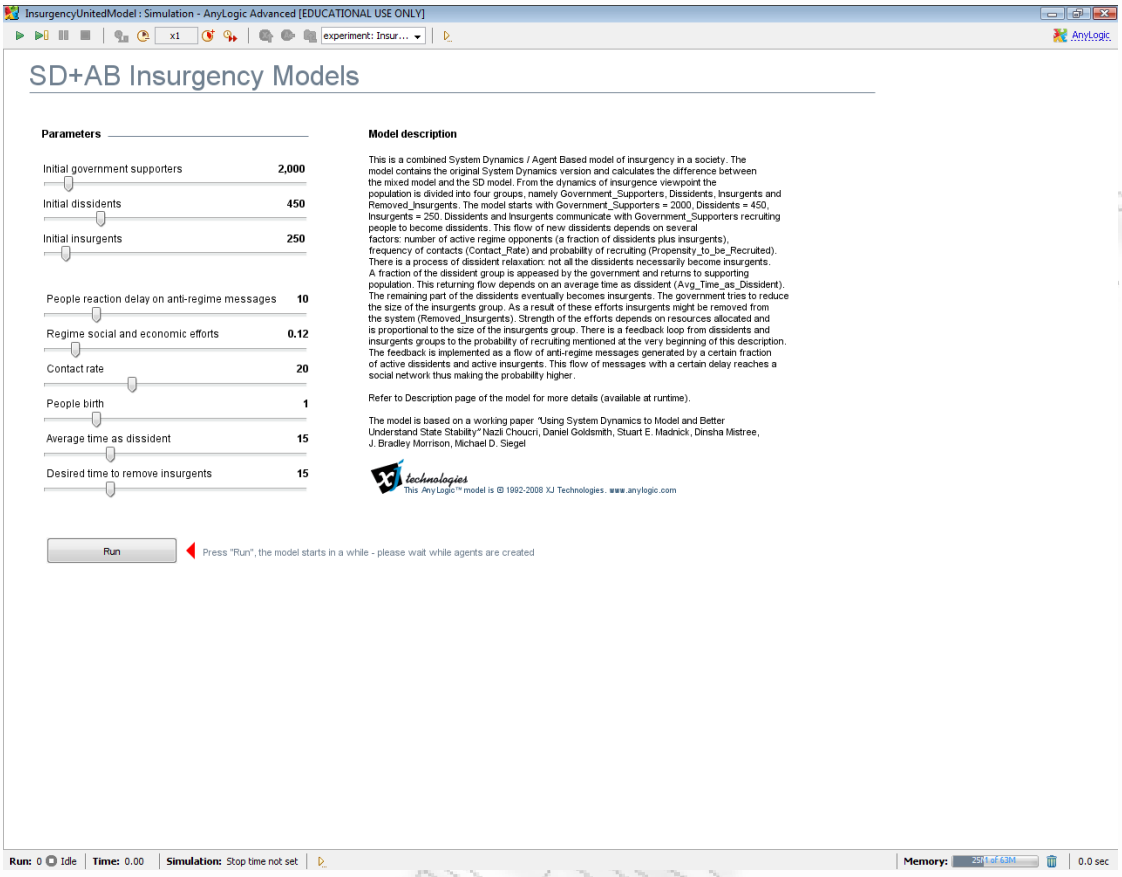
Εικόνα 9-42(η): Γενικές ιδιότητες της ράβδου κύλισης slider7.



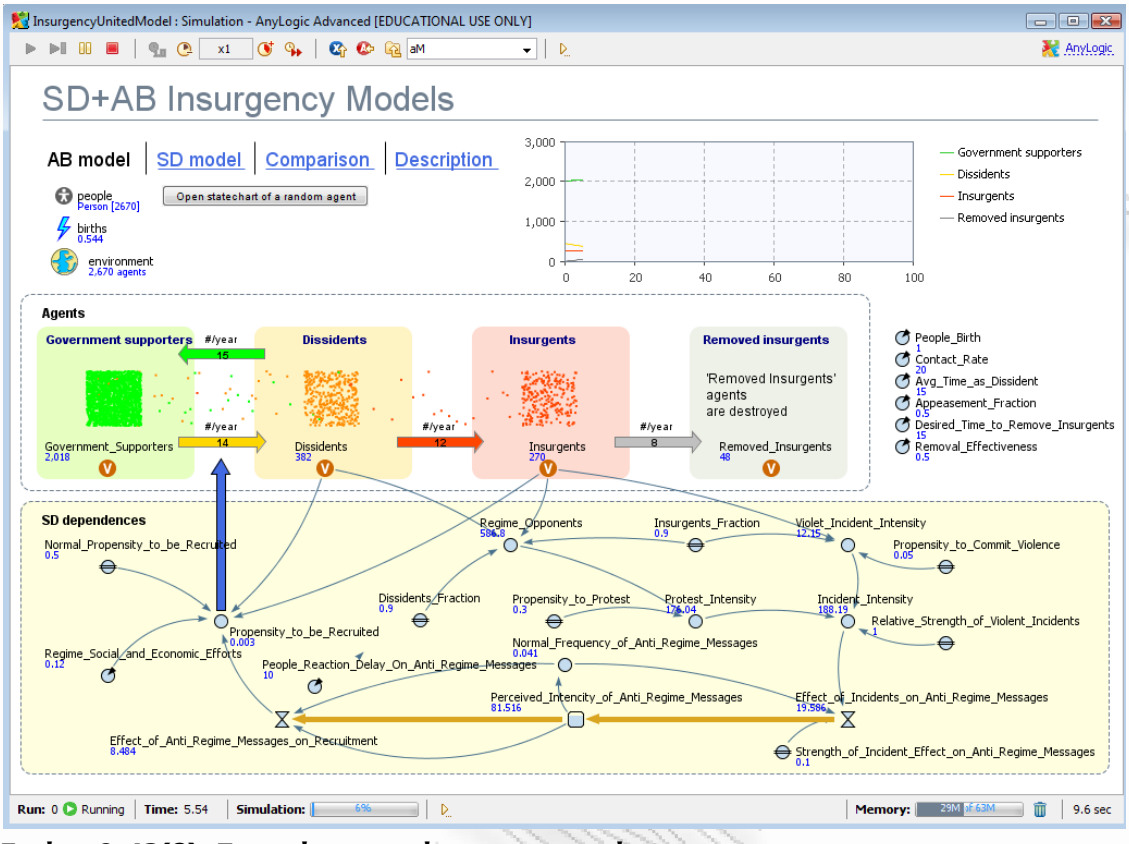
Εικόνα 9-42(θ): Γενικές ιδιότητες της ράβδου κύλισης slider8.

Ακολουθούν οι εικόνες 9-43(α) έως και 9-43(θ) με ενδεικτικά στιγμιότυπα από τη προσομοίωση.

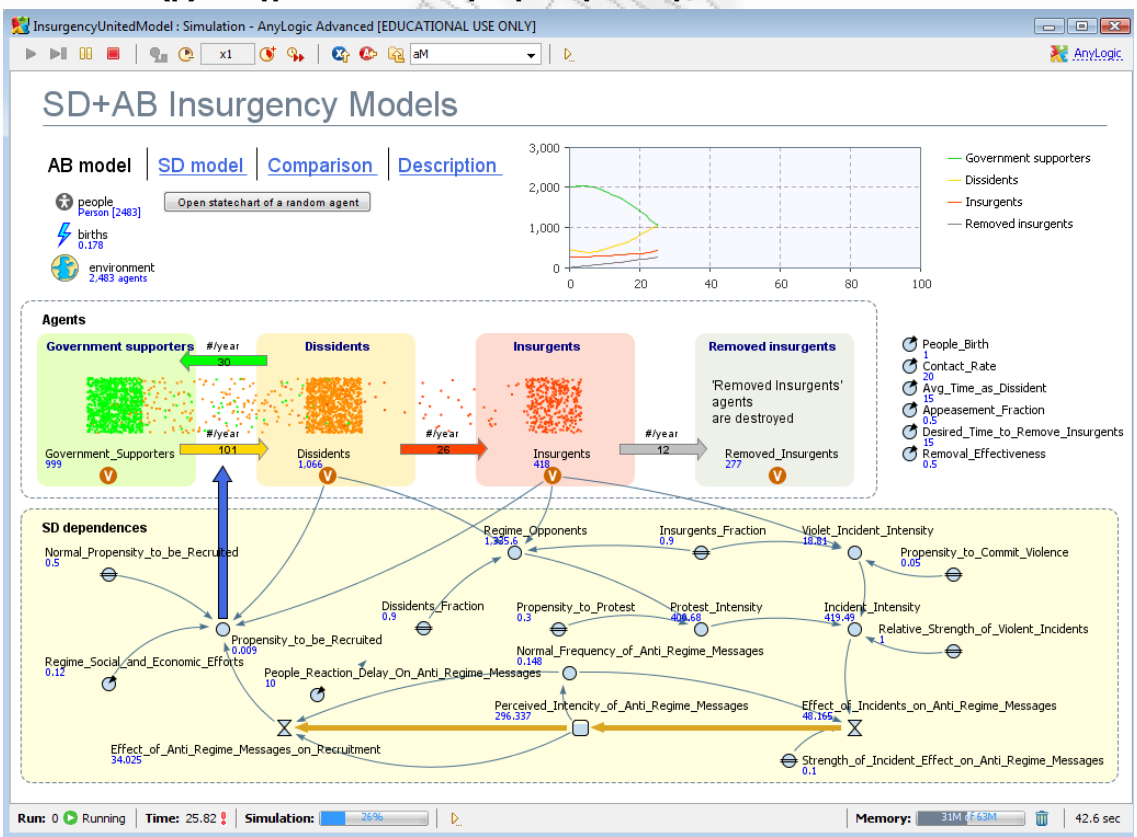
Στο Παράρτημα 7 βρίσκονται (α) τα διαγράμματα κλάσεων και (β) ο κώδικας του έργου Insurgency Dynamics.



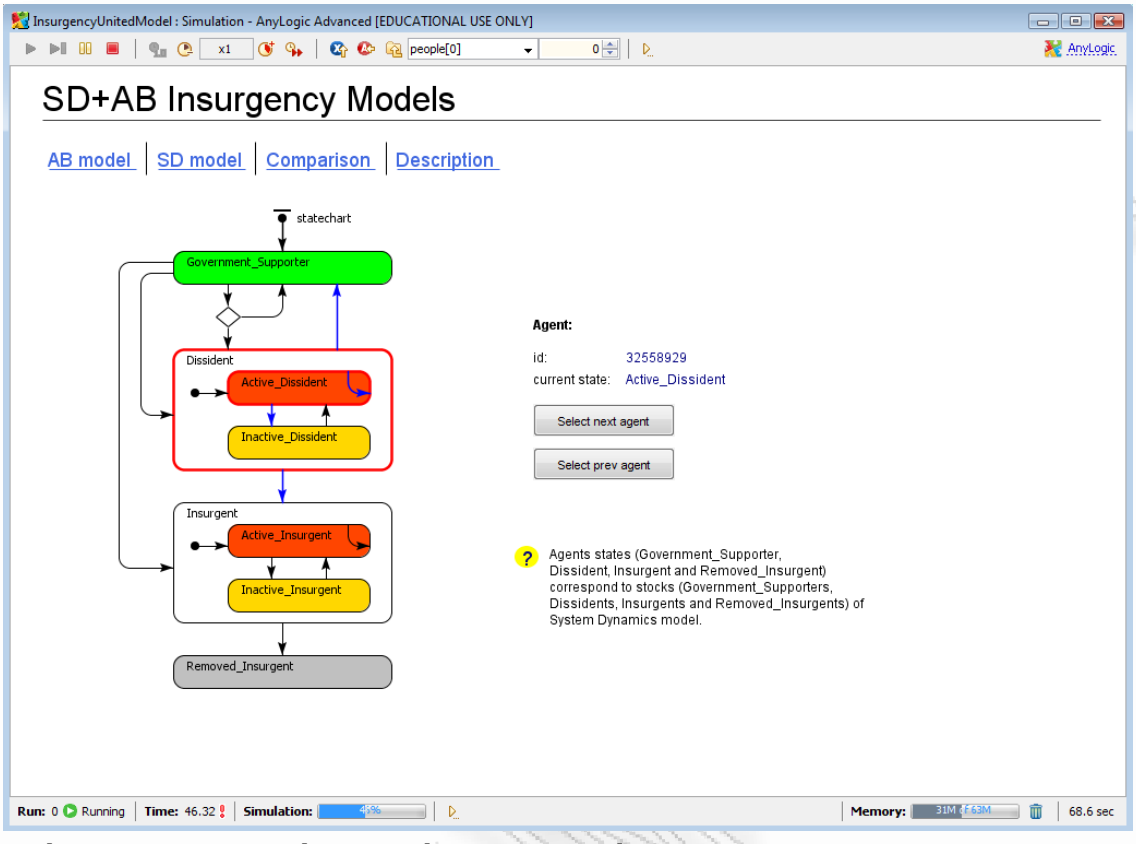
Εικόνα 9-43(α): Στιγμιότυπα από την προσομοίωση. Η αρχική εικόνα πριν την προσομοίωση.



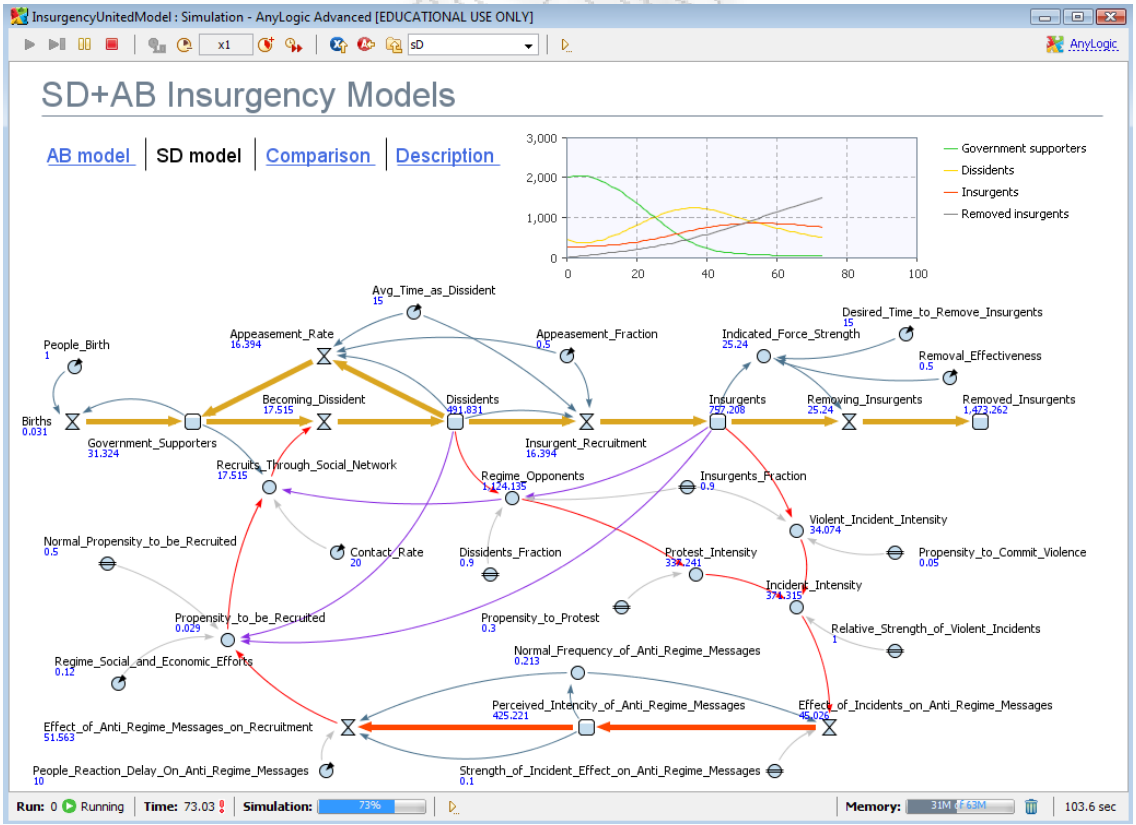
Εικόνα 9-43(β): Στιγμιότυπα από την προσομοίωση.



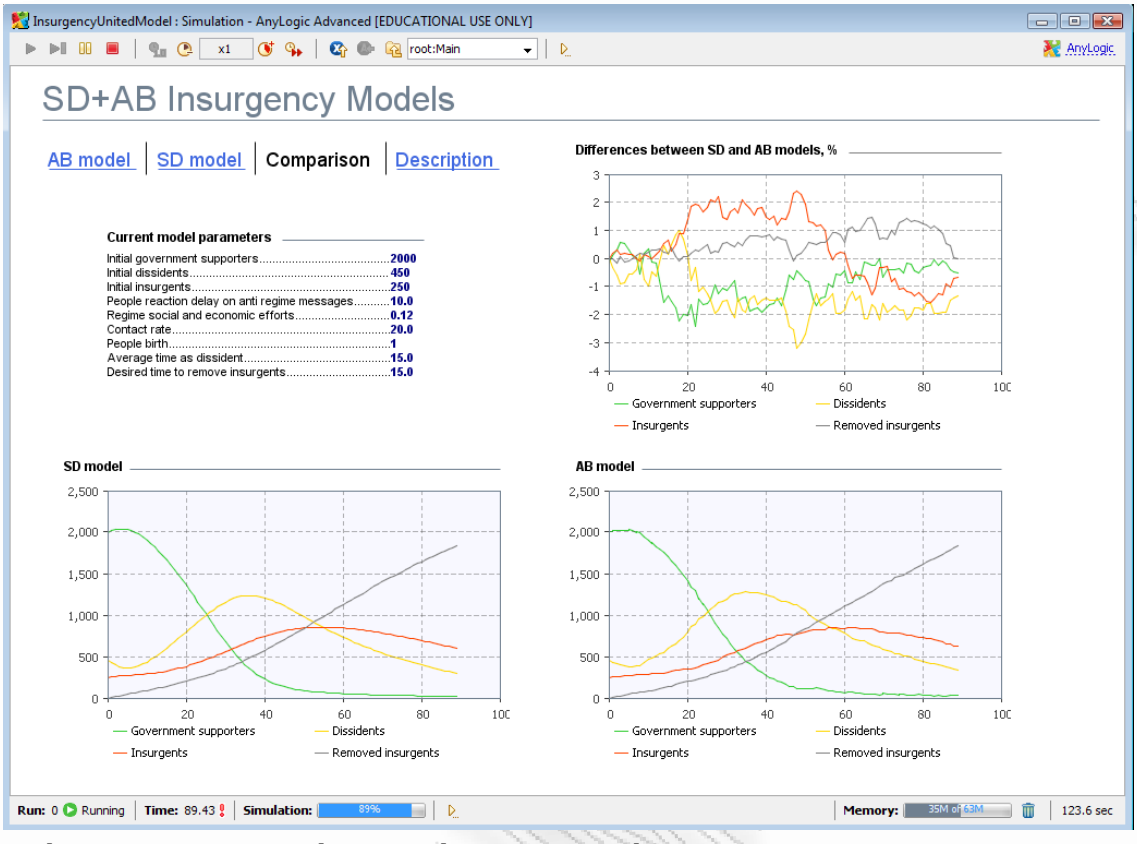
Εικόνα 9-43(γ): Στιγμιότυπα από την προσομοίωση.



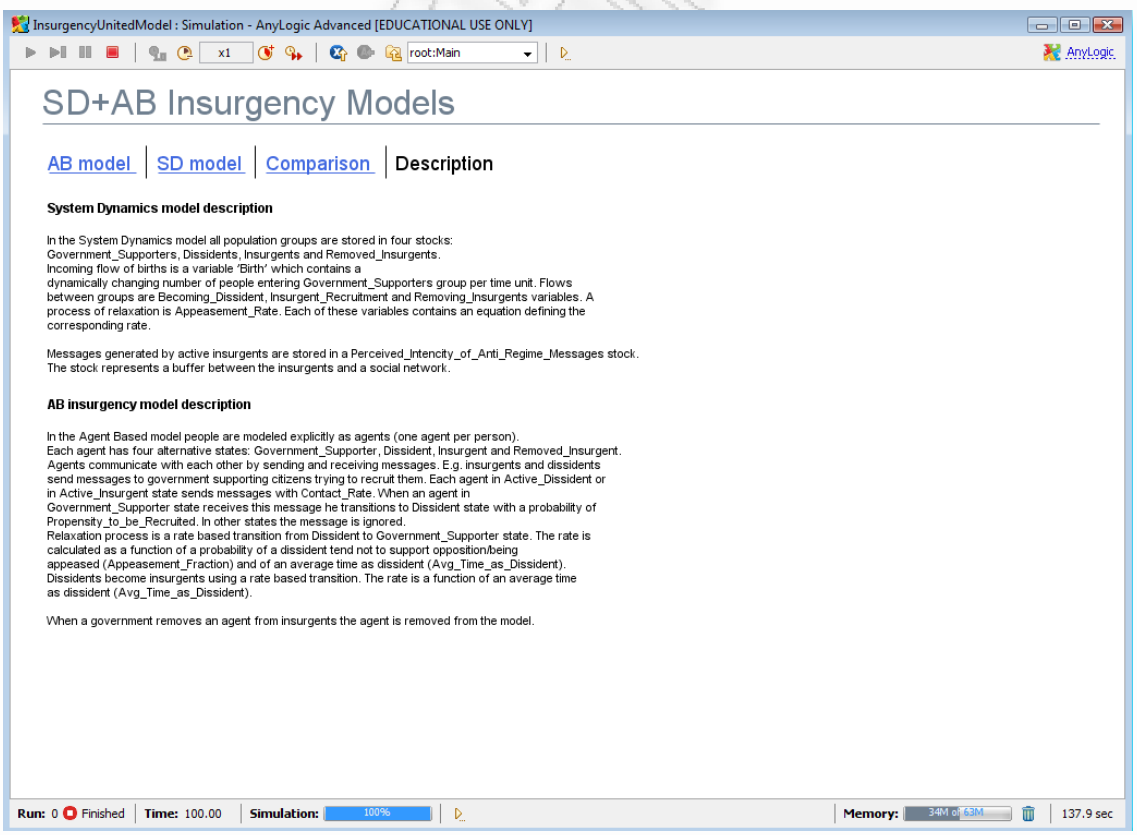
Εικόνα 9-43(δ): Στιγμιότυπα από την προσομοίωση.



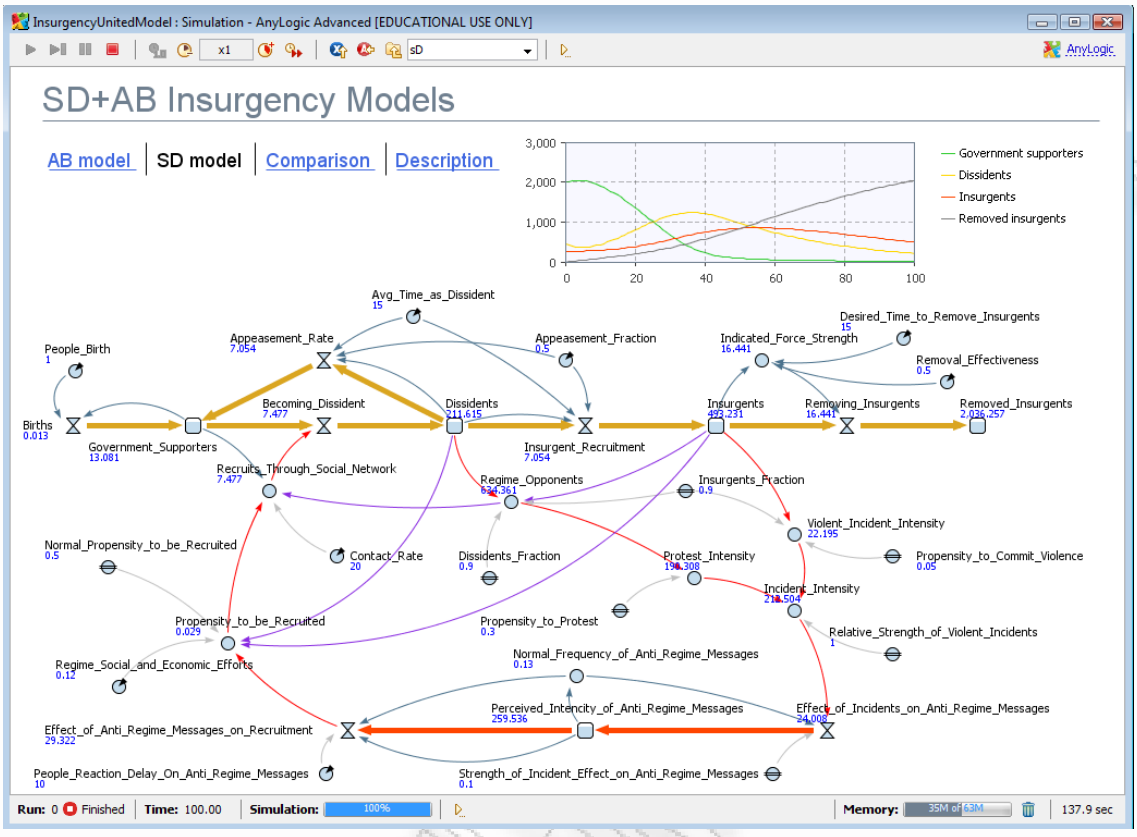
Εικόνα 9-43(ε): Στιγμιότυπα από την προσομοίωση.



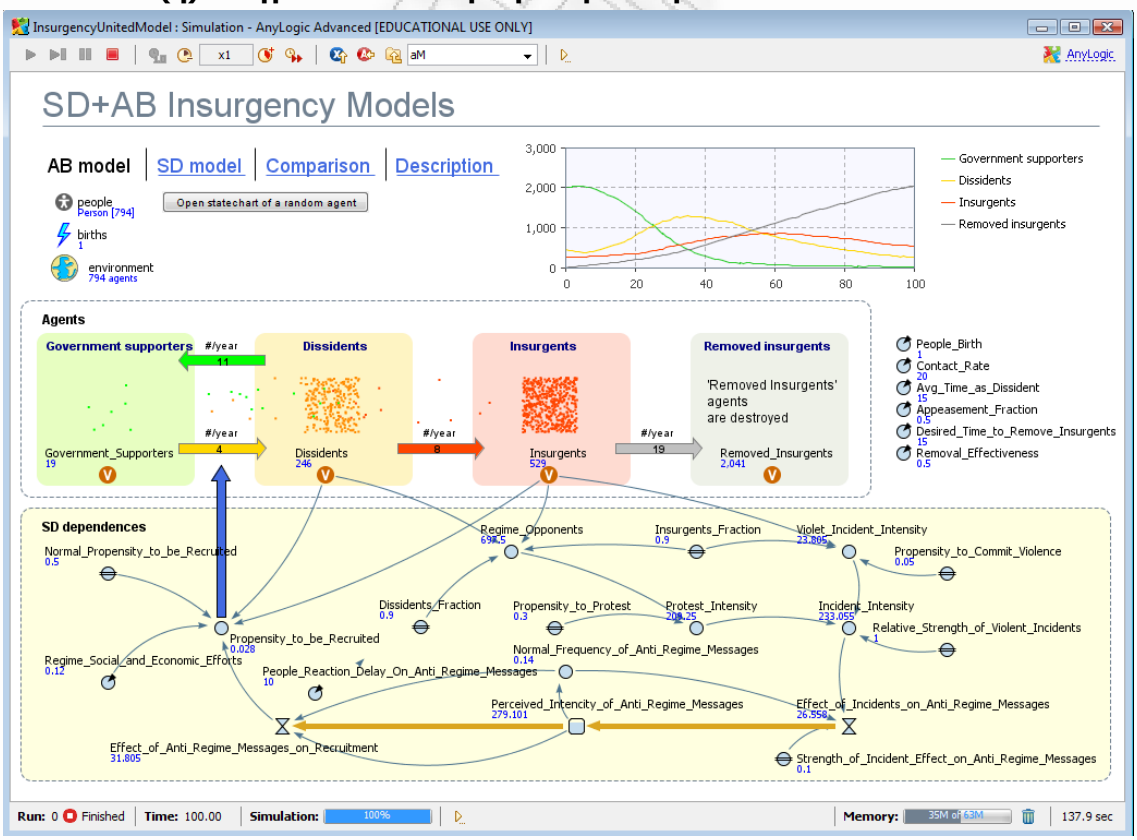
Εικόνα 9-43(στ): Στιγμιότυπα από την προσομοίωση.



Εικόνα 9-43(ζ): Στιγμιότυπα από την προσομοίωση.



Εικόνα 9-43(η): Στιγμιότυπα από την προσομοίωση.



Εικόνα 9-43(θ): Στιγμιότυπα από την προσομοίωση.

ΚΕΦΑΛΑΙΟ 10

Μοντελοποίηση και προσομοίωση του πρωτοκόλλου εκλογής αρχηγού σε ένα κατανεμημένο σύστημα υπολογιστών

Το παράδειγμα αυτού του τελευταίου κεφαλαίου της Μεταπτυχιακής Διατριβής μοντελοποιεί το πρωτόκολλο εκλογής αρχηγού, (Leader Election Protocol), σε ένα κατανεμημένο δίκτυο υπολογιστών μέσω των πρακτόρων λογισμικού.

Οι αλγόριθμοι επιλογής αρχηγού, με βάση την τοπολογία του δικτύου, χωρίζονται σε τρείς κατηγορίες, (α) σε αυτούς που χρησιμοποιούνται σε ισχυρά συνδεδεμένους γράφους, (β) σε αυτούς που χρησιμοποιούνται σε δακτυλίους μιας κατεύθυνσης και (γ) σε αυτούς που χρησιμοποιούνται σε τοπολογίες δένδρου.

Μια κατηγορία των κατανεμημένων υπολογιστικών συστημάτων είναι αυτά με αρχιτεκτονική εξυπηρετητή – πελάτη, (client - server). Σε ένα σύνολο υπολογιστών, ο υπολογιστής εξυπηρετητής είναι αυτός που προσφέρει υπηρεσίες στους υπολοίπους υπολογιστές πελάτες. Αν ο υπολογιστής εξυπηρετητής συμβεί να καταρρεύσει, (για οποιοδήποτε λόγο), τι μπορεί να συμβεί; Το σύστημα, το σύνολο των υπολογιστών σταματά να λειτουργεί;

Η απάντηση είναι ότι το σύστημα δεν πρέπει να σταματήσει τη λειτουργία του και για το λόγο αυτό θα πρέπει το συντομότερο δυνατό οι υπόλοιποι υπολογιστές πελάτες να «εκλέξουν» έναν άλλο υπολογιστή που θα αναλάβει το ρόλο του εξυπηρετητή με μεταξύ τους συμφωνία. Τι θα συμβεί όμως αν περισσότεροι του ενός υπολογιστές πελάτες αντιληφθούν ταυτόχρονα την απώλεια του υπολογιστή εξυπηρετητή; Θα κινήσει ο καθένας από αυτούς τις διαδικασίες εκλογής νέου αρχηγού, νέου υπολογιστή εξυπηρετητή;

Το πρόβλημα της εκλογής νέου αρχηγού σε τοπολογία δακτυλίου μιας κατεύθυνσης, έχει τεθεί πολλά χρόνια πριν από τον Gerard LeLann και από τότε αρκετοί αλγόριθμοι εμφανίστηκαν για την επίλυση του προβλήματος. Ο αλγόριθμος επιλογής αρχηγού πρέπει να πληρεί ορισμένες προϋποθέσεις, (α) κάθε υπολογιστής πρέπει να εκτελεί τον ίδιο αλγόριθμο, (β) η διαδικασία εκλογής μπορεί να κινείται από όλους τους υπολογιστές και (γ) ο αλγόριθμος θα πρέπει να φτάνει σε τελική κατάσταση όπου μόνο ένας θα γίνεται αρχηγός. Οι βασικές ιδιότητες των αλγορίθμων πρέπει να ισχύουν επίσης, δηλαδή ο αλγόριθμος πρέπει (α) να δίνει αποτέλεσμα σε πεπερασμένο χρόνο, (β) κάθε βήμα του αλγορίθμου πρέπει να εκτελείται και (γ) να δίνει τελική απόφαση.

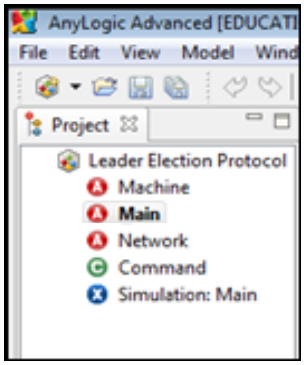
Για την υλοποίηση του αλγορίθμου, εκτός των παραπάνω, πρέπει να ισχύουν οι υποθέσεις: (α) δεν έχουμε κεντρικό ρολόι συγχρονισμού, (είναι βολικό αν και δεν είναι απαραίτητο), (β) κάθε υπολογιστής έχει μοναδικό αναγνωριστικό, το οποίο είναι διατεταγμένο σε σχέση με αυτά των υπολοίπων, (γ) η σύγκριση γίνεται στα αναγνωριστικά των υπολογιστών και (δ) κάθε μήνυμα μπορεί να μεταφέρει μέχρι έναν ορισμένο αριθμό αναγνωριστικών.

Επίσης, οι γενικές παραδοχές του μοντέλου για την εκλογή του αρχηγού είναι: (α) κάθε υπολογιστής γνωρίζει μόνο το δικό του αναγνωριστικό, (β) κάθε υπολογιστής στέλνει μηνύματα στον επόμενο του υπολογιστή, (γ) κάθε υπολογιστής λαμβάνει μηνύματα από τον προηγούμενο του υπολογιστή και (δ) αρχηγός εκλέγεται ο υπολογιστής με το μικρότερο αναγνωριστικό. Η διαδικασία του αλγορίθμου εκλογής αρχηγού όταν ξεκινά, έχει ως εξής: Κάθε υποψήφιος αρχηγός που λέγεται αρχικοποιητής, (initiator, είναι η διεργασία σε κάθε υπολογιστή που ξεκινά τοπικά την εκτέλεση του αλγορίθμου εκλογής αρχηγού αυτόματα μετά την ανίχνευση ότι ο αρχηγός κατέρρευσε), στέλνει ένα token με το αναγνωριστικό του σε όλους τους άλλους υπολογιστές του δακτυλίου. Οι υπολογιστές δεν επιτρέπεται να αρχικοποιηθούν αφού λάβουν ένα token από άλλους υπολογιστές. Κάθε υπολογιστής κρατά ιστορικό με τα αναγνωριστικά που πέρασαν. Όσοι υπολογιστές δεν είναι αρχικοποιητές, τα αναγνωριστικά τους αμέσως περνούν σε κατάσταση άρνησης εκλογής. Ο κάθε αρχικοποιητής, μόλις λάβει τα αναγνωριστικά εξετάζει αν

αυτό με τη μικρότερη τιμή ανήκει σε αυτόν. Αν όχι, αμέσως περνά σε κατάσταση άρνησης εκλογής ενώ αν ισχύει ότι το αναγνωριστικό που έχει τη μικρότερη τιμή του ανήκει αμέσως εκλέγεται αρχηγός και ο αλγόριθμος τερματίζεται.

Στο μοντέλο που θα εξετάσουμε, ένας αριθμός υπολογιστών είναι συνδεδεμένοι μεταξύ τους σε δίκτυο, σε τοπολογία δακτυλίου, το οποίο όμως είναι αναξιόπιστο με συνέπεια τα μηνύματα που διακινούνται μεταξύ των υπολογιστών να χάνονται, να επαναλαμβάνονται και να καθυστερούν με τυχαίο τρόπο. Οι υπολογιστές δεν είναι ιδανικοί, δεν λειτουργούν αδιάλειπτα, πράγμα που σημαίνει ότι μπορούν να καταρρεύσουν και να επανέλθουν, να επανεκκινήσουν.

Ο αλγόριθμος του πρωτοκόλλου εκλογής αρχηγού, εκτελείται από κάθε υπολογιστή που βρίσκεται σε λειτουργία και χρησιμεύει ώστε να γνωρίζουν κάθε στιγμή όλοι οι υπολογιστές που είναι σε λειτουργία ποιος έχει αναλάβει το ρόλο του αρχηγού, (Master). Αυτό επιτυγχάνεται με τη χρήση αλγορίθμων συγχρονισμού και αλγορίθμων εκλογής που βασίζονται σε σταθερά χρονικά όρια και σε τυχαία χρονικά όρια. Το μοντέλο, επίσης, επιτρέπει την επιβολή κατάρρευσης και επανεκκίνησης οποιουδήποτε υπολογιστή. Η εκλογή του αρχηγού παρουσιάζεται σε έγχρωμο γραφικό διάγραμμα.

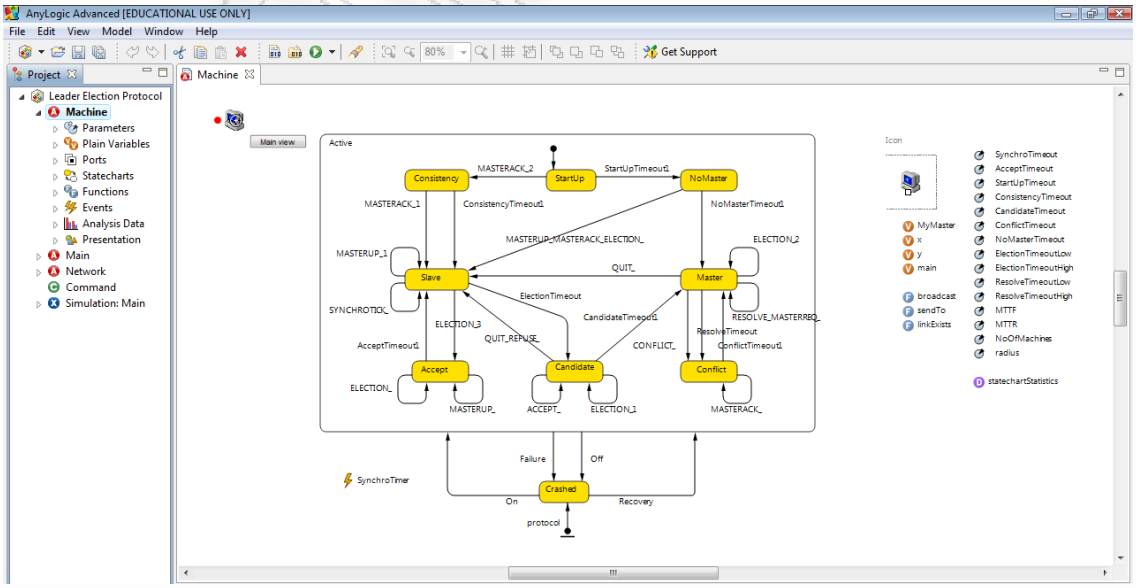


Εικόνα 10-1: Η ιεραρχία του έργου Leader Election Protocol.

Στην εικόνα 10-1 παρουσιάζεται η ιεραρχία του έργου Leader Election Protocol. Αποτελείται από τρεις (3) κλάσεις ενεργού αντικειμένου, μια (1) κλάση υλοποιημένη με κώδικα σε γλώσσα προγραμματισμού Java και μια (1) κλάση παρουσίασης της προσομοίωσης.

10.1 Η κλάση Machine

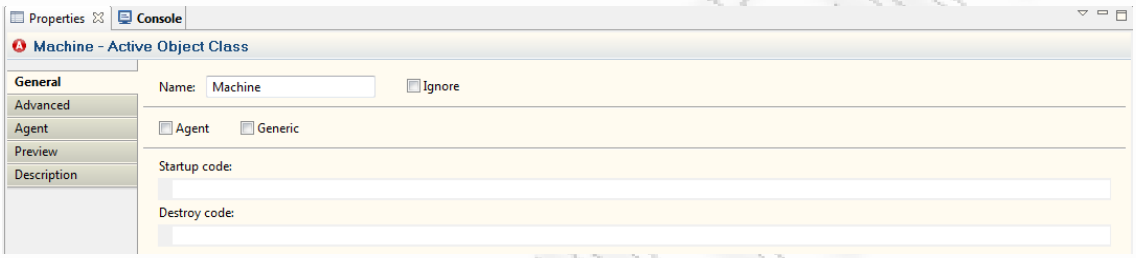
Η κλάση ενεργού αντικειμένου Machine είναι αυτή που υλοποιεί την αναπαράσταση του κάθε υπολογιστή. Δύο παρατηρήσεις πρέπει να γίνουν στο σημείο αυτό. Η πρώτη έχει να κάνει με το ότι υπάρχει συγχρονισμένος μετρητής χρόνου που υλοποιείται μέσω του γεγονότος SynchroTimer. Η δεύτερη παρατήρηση έχει να κάνει με το ότι η αντιστοιχία server - client της εισαγωγής εδώ γίνεται master – slave. Το διάγραμμα καταστάσεων ορίζει τον αλγόριθμο της εκλογής αρχηγού. Κατά την έναρξη της διαδικασίας του πρωτοκόλλου ελέγχεται αν ο υπολογιστής έχει καταρρεύσει ή όχι. Αν έχει καταρρεύσει τότε είτε ο υπολογιστής ξεκινά τη λειτουργία του, είτε επανεκκινεί μέσω διαδικασίας ανάκτησης. Στην περίπτωση που δεν έχει καταρρεύσει ο υπολογιστής θεωρείται ενεργός και μπορεί να συμμετάσχει στη διαδικασία επιλογής αρχηγού όταν αυτό απαιτηθεί.



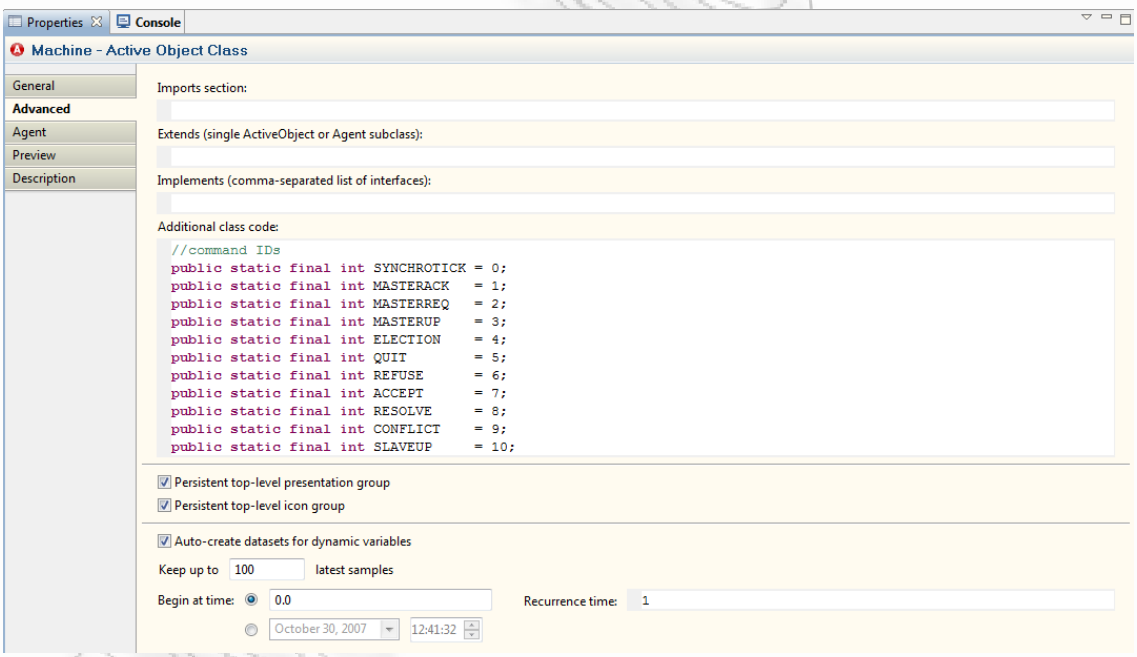
Εικόνα 10-2: Η κλάση Machine.

Όσο ο υπολογιστής είναι ενεργός, Active, και κατά την έναρξη της διαδικασίας επιλογής, κατάσταση StartUp, ελέγχεται η συνοχή του υπολογιστή, κατάσταση Consistency, και αν είναι αυτός ο αρχηγός, κατάσταση NoMaster. Ο υπολογιστής μπορεί να μεταβεί είτε στη κατάσταση Master και να γίνει αρχηγός είτε να γίνει απευθείας σκλάβος, κατάσταση Slave. Όταν ο αρχηγός πάψει να εκτελεί τα καθήκοντα του λόγω της ανάληψης της αρχηγίας από άλλο υπολογιστή, μεταβαίνει στην κατάσταση Slave. Όσο ο υπολογιστής βρίσκεται στη κατάσταση Slave μπορεί να επιχειρήσει να αλλάξει κατάσταση, κατάσταση Candidate και αν δεν το πετύχει να ξαναγίνει σκλάβος μέσω της κατάστασης Accept. Στη Slave κατάσταση Candidate ο υπολογιστής μπορεί να μεταβεί είτε στη κατάσταση Slave είτε στη κατάσταση Master. Αν υπάρχει άλλος υπολογιστής ως αρχηγός, γίνεται μετάβαση στη κατάσταση Conflict και επιλέγεται τελικά ο νέος αρχηγός. Η διαδικασία συνεχίζεται σύμφωνα με τις τιμές των παραμέτρων της κλάσης.

Στις εικόνες 10-3(α) έως και 10-3(β) παρουσιάζονται οι γενικές και οι προχωρημένες ιδιότητες της κλάσης Machine.

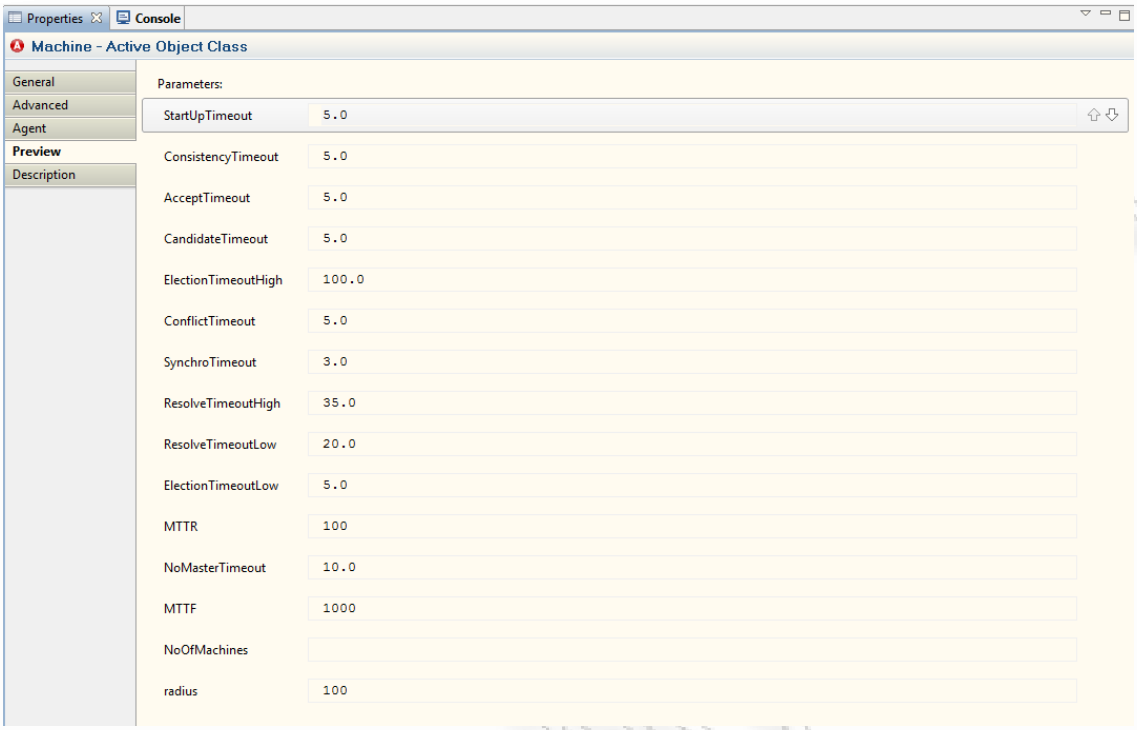


Εικόνα 10-3(α): Γενικές ιδιότητες της κλάσης Machine.

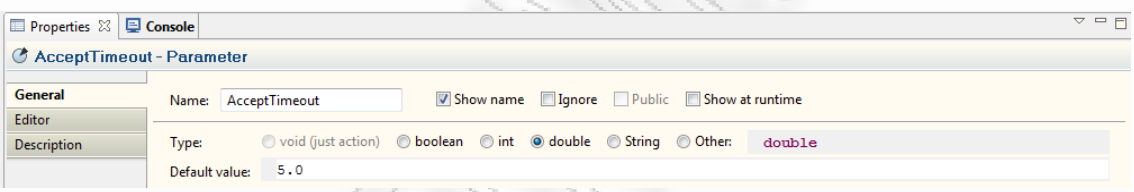


Εικόνα 10-3(β): Προχωρημένες ιδιότητες της κλάσης Machine.

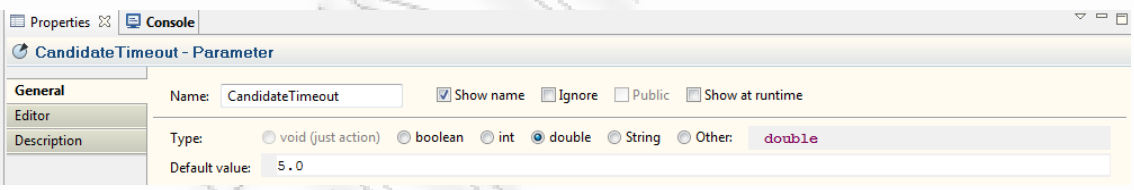
Στην εικόνα 10-3(γ) γίνεται η προεπισκόπιση των τιμών των παραμέτρων της κλάσης. Στις εικόνες 10-4(α) έως και 10-4(ιε) παρουσιάζονται οι γενικές ιδιότητες των παραμέτρων που ανήκουν στην κλάση. Στις εικόνες 10-5(α) έως και 10-5(δ) παρουσιάζονται οι ιδιότητες των απλών μεταβλητών: (α) MyMaster στην οποία καταχωρείται ο τρέχων αρχηγός, (β) main που λαμβάνει τιμές από τη συνάρτηση getOwner της κλάσης Main και (γ) x και y οι μεταβλητές που χρησιμεύουν για την απεικόνιση του δικτύου των υπολογιστών στην τοπολογία του δακτυλίου κατά την προσομοίωση. Στην εικόνα 10-6 δίνονται οι ιδιότητες της θύρας port, η οποία είναι αναγκαία για τη δημιουργία του δικτύου και της ενεργοποίησης του μηχανισμού ανταλλαγής μηνυμάτων του Λογισμικού.



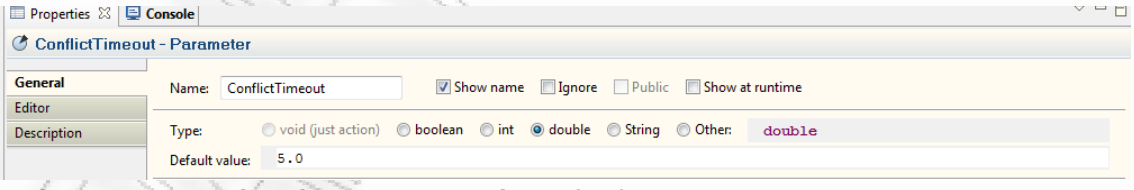
Εικόνα 10-3(γ): Προεπισκόπηση των παραμέτρων της κλάσης Machine.



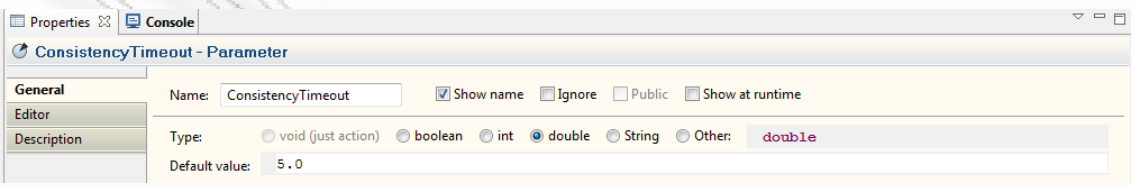
Εικόνα 10-4(α): Γενικές ιδιότητες της παραμέτρου AcceptTimeout.



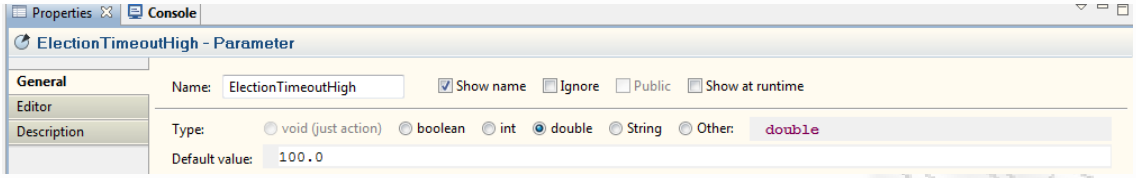
Εικόνα 10-4(β): Γενικές ιδιότητες της παραμέτρου CandidateTimeout.



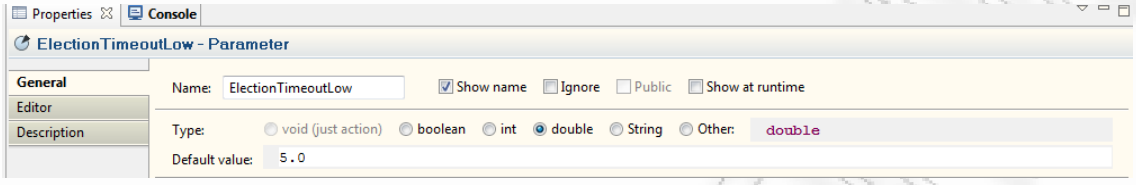
Εικόνα 10-4(γ): Γενικές ιδιότητες της παραμέτρου ConflictTimeout.



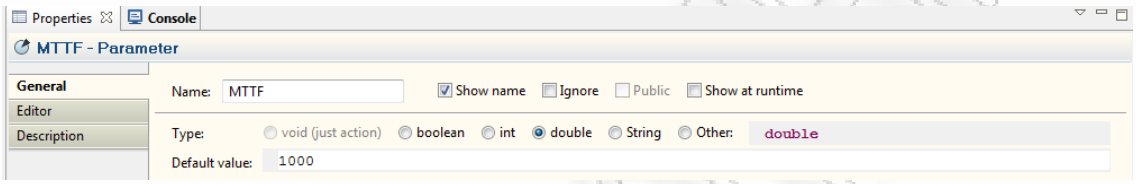
Εικόνα 10-4(δ): Γενικές ιδιότητες της παραμέτρου ConsistencyTimeout.



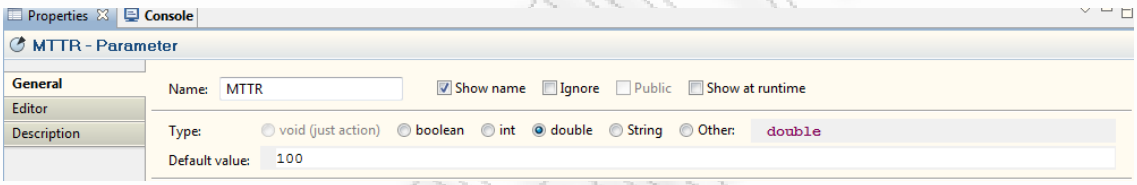
Εικόνα 10-4(ε): Γενικές ιδιότητες της παραμέτρου ElectionTimeoutHigh.



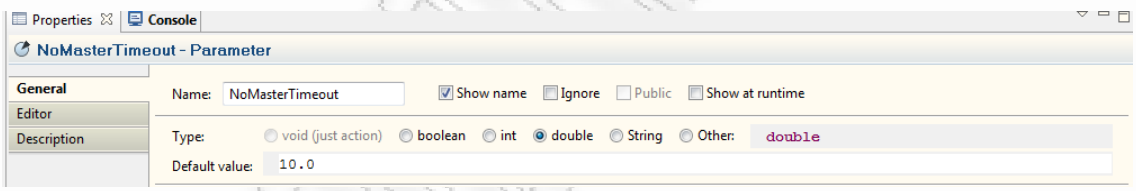
Εικόνα 10-4(στ): Γενικές ιδιότητες της παραμέτρου ElectionTimeoutLow.



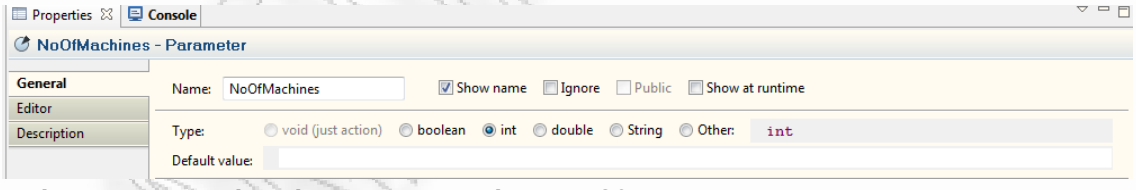
Εικόνα 10-4(ζ): Γενικές ιδιότητες της παραμέτρου MTTF.



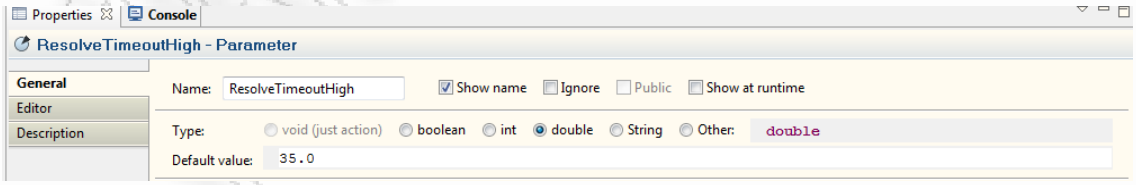
Εικόνα 10-4(η): Γενικές ιδιότητες της παραμέτρου MTTR.



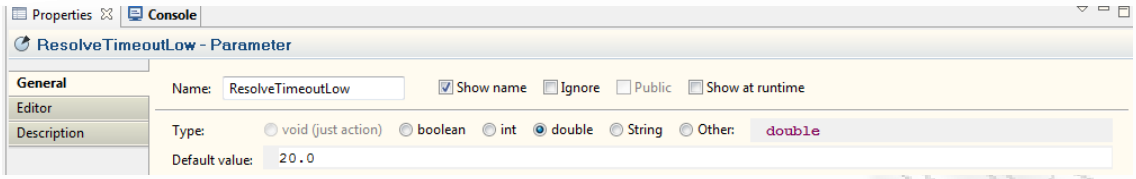
Εικόνα 10-4(θ): Γενικές ιδιότητες της παραμέτρου NoMasterTimeout.



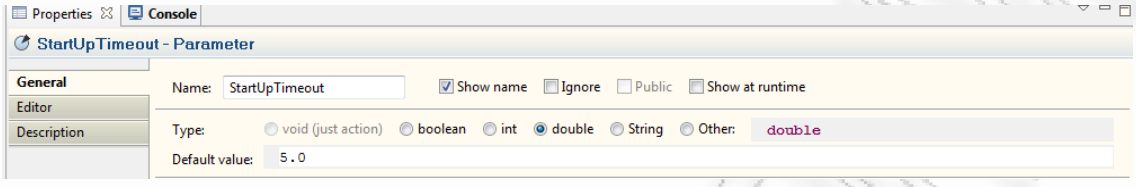
Εικόνα 10-4(ι): Γενικές ιδιότητες της παραμέτρου NoOfMachines.



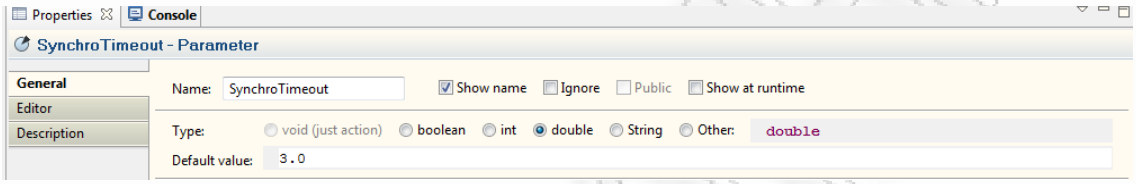
Εικόνα 10-4(ια): Γενικές ιδιότητες της παραμέτρου ResolveTimeoutHigh.



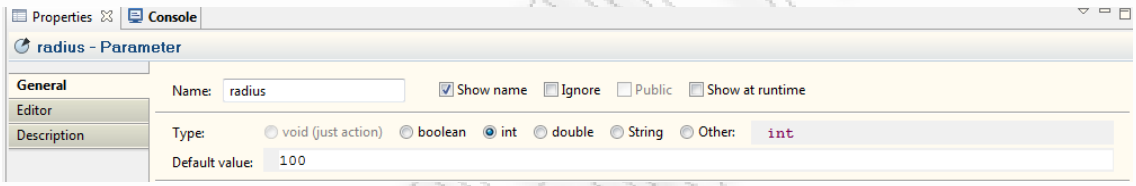
Εικόνα 10-4(ιβ): Γενικές ιδιότητες της παραμέτρου ResolveTimeoutLow.



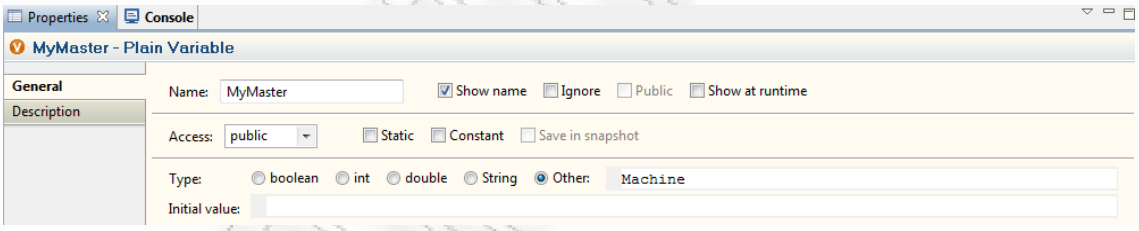
Εικόνα 10-4(ιγ): Γενικές ιδιότητες της παραμέτρου StartUpTimeout.



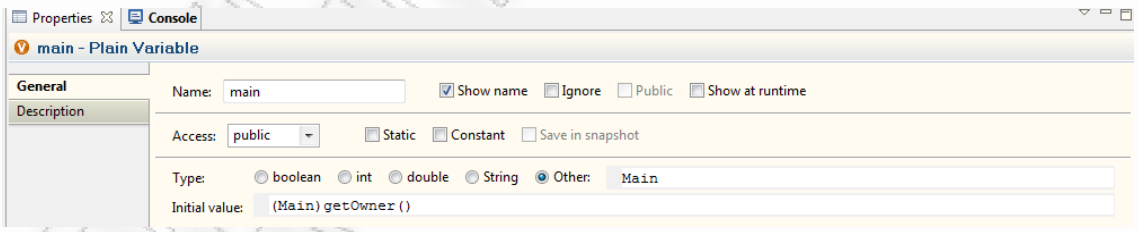
Εικόνα 10-4(ιδ): Γενικές ιδιότητες της παραμέτρου SynchroTimeout.



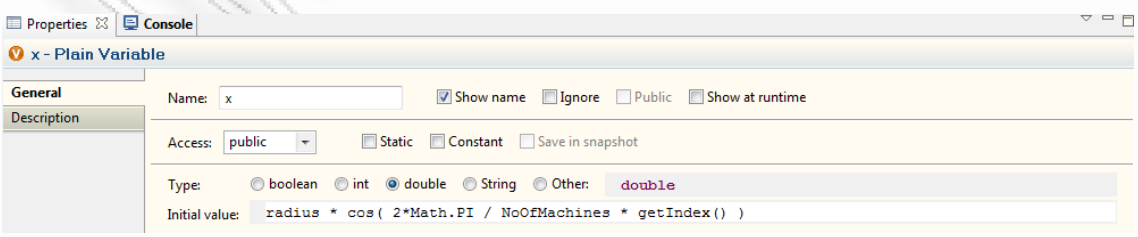
Εικόνα 10-4(ιε): Γενικές ιδιότητες της παραμέτρου radius.



Εικόνα 10-5(α): Γενικές ιδιότητες της απλής μεταβλητής MyMaster.

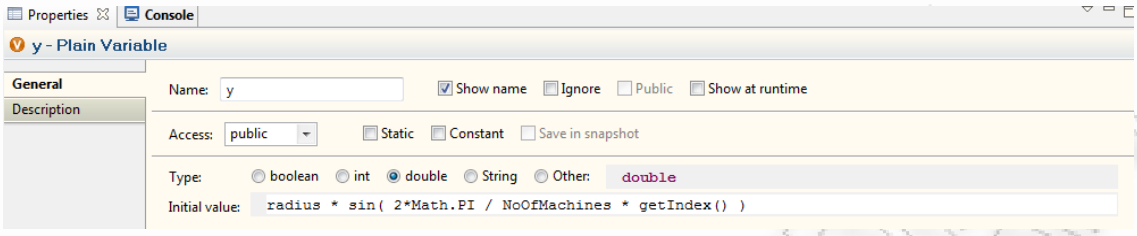


Εικόνα 10-5(β): Γενικές ιδιότητες της απλής μεταβλητής main.



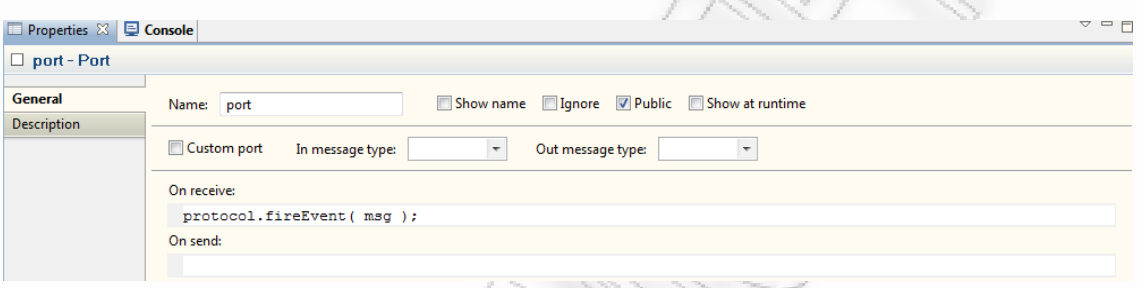
Εικόνα 10-5(γ): Γενικές ιδιότητες της απλής μεταβλητής x, με αρχική τιμή την

$$radius * cos(\frac{2 * Math.PI}{NoOfMachines * getIndex()})$$



Εικόνα 10-5(δ): Γενικές ιδιότητες της απλής μεταβλητής y, με αρχική τιμή την

$$radius * sin(\frac{2 * Math.PI}{NoOfMachines * getIndex()})$$

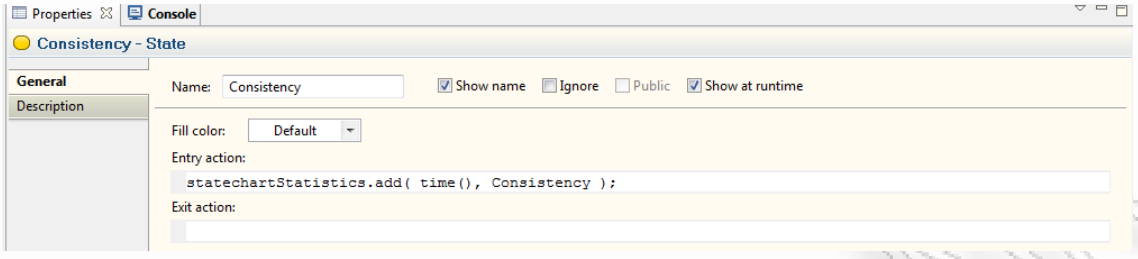


Εικόνα 10-6: Γενικές ιδιότητες της θύρας port.

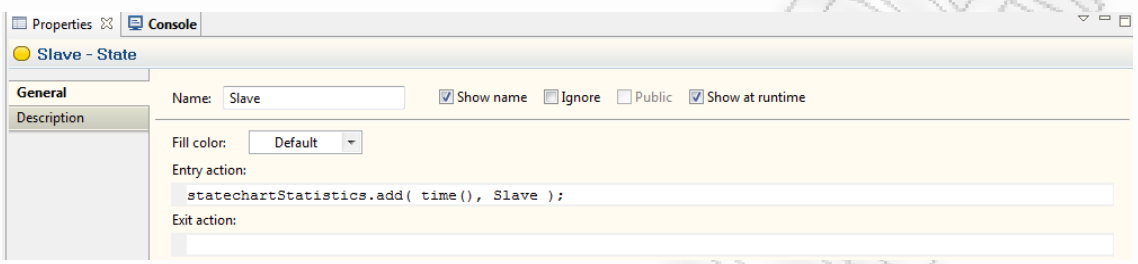
Στις εικόνες 10-7(α) έως και 10-7(θ) δίνονται οι γενικές ιδιότητες των καταστάσεων του διαγράμματος καταστάσεων της κλάσης και στις εικόνες 10-8(α) έως και 10-8(κη) παρουσιάζονται οι ιδιότητες των μεταβάσεων της κλάσης.

Εικόνα 10-7(α): Γενικές ιδιότητες της κατάστασης Crashed.

Εικόνα 10-7(β): Γενικές ιδιότητες της κατάστασης StartUp.

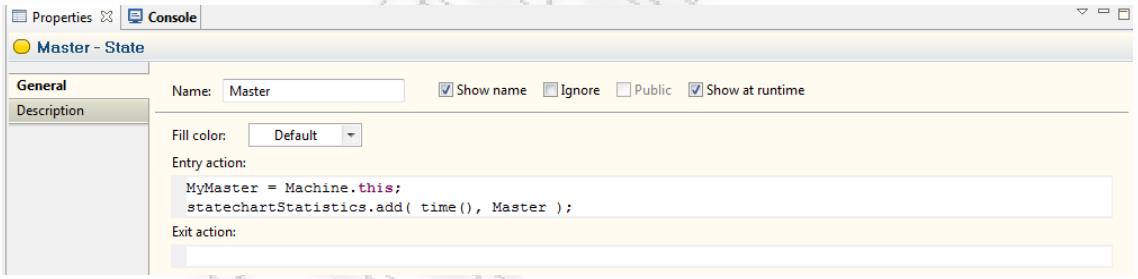


Εικόνα 10-7(γ): Γενικές ιδιότητες της κατάστασης Consistency.



Εικόνα 10-7(δ): Γενικές ιδιότητες της κατάστασης Slave.

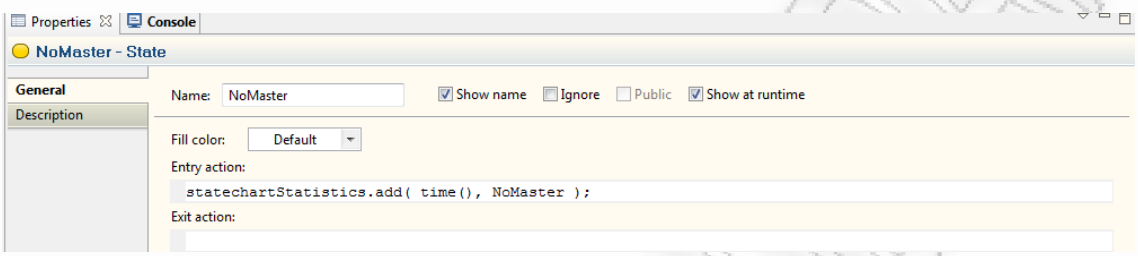
Εικόνα 10-7(ε): Γενικές ιδιότητες της κατάστασης Candidate.



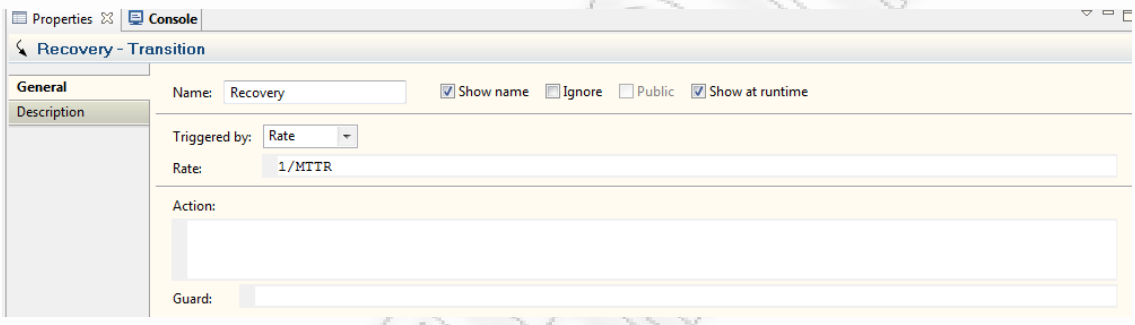
Εικόνα 10-7(στ): Γενικές ιδιότητες της κατάστασης Master.

Εικόνα 10-7(ζ): Γενικές ιδιότητες της κατάστασης Conflict.

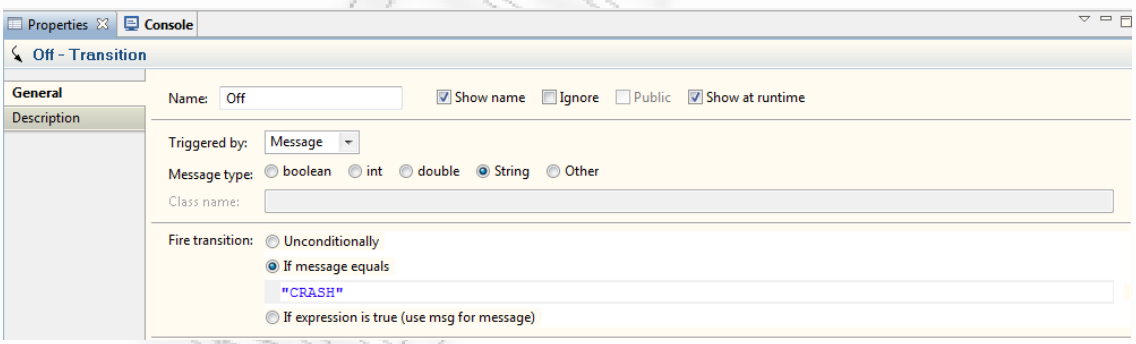
Εικόνα 10-7(η): Γενικές ιδιότητες της κατάστασης Accept.



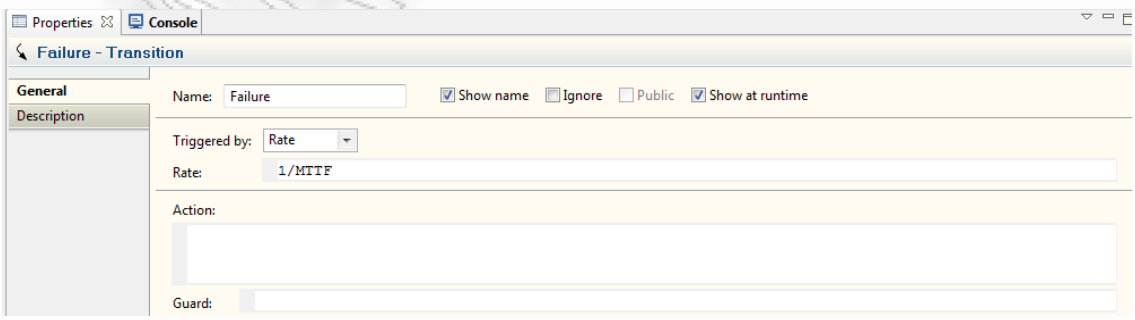
Εικόνα 10-7(θ): Γενικές ιδιότητες της κατάστασης NoMaster.



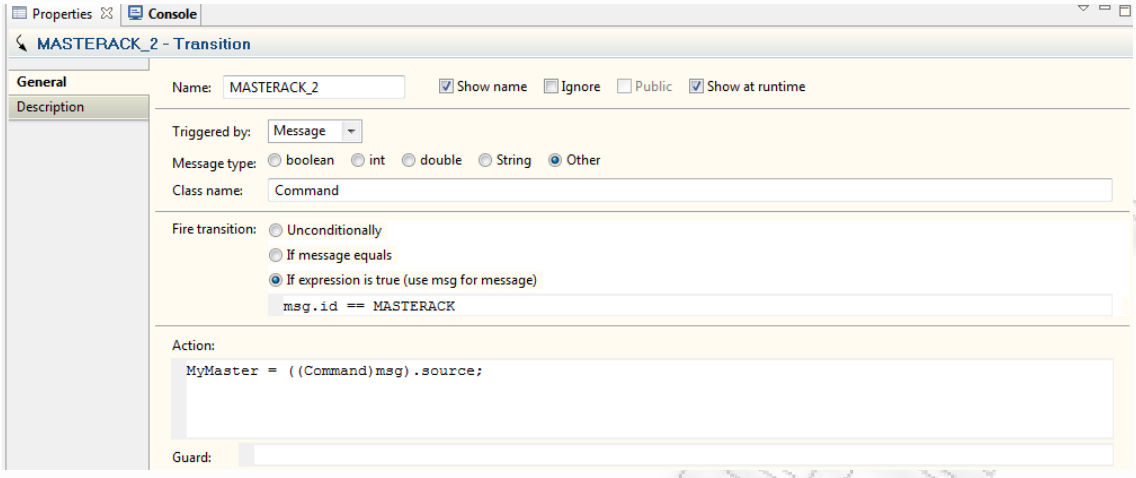
Εικόνα 10-8(α): Γενικές ιδιότητες της μετάβασης Recovery.



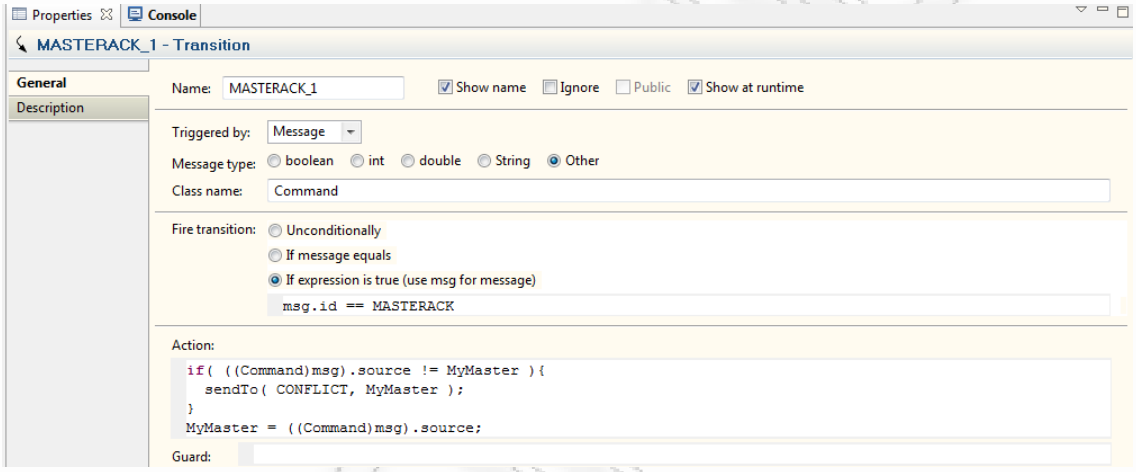
Εικόνα 10-8(β): Γενικές ιδιότητες της μετάβασης Off.



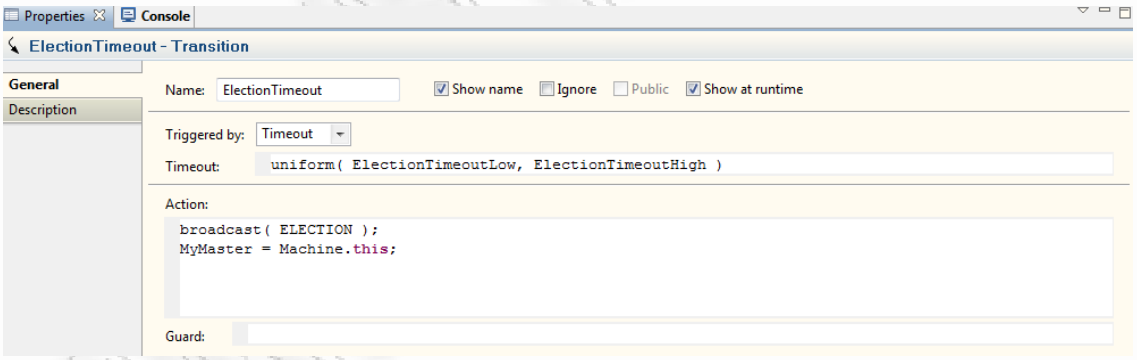
Εικόνα 10-8(γ): Γενικές ιδιότητες της μετάβασης Failure.



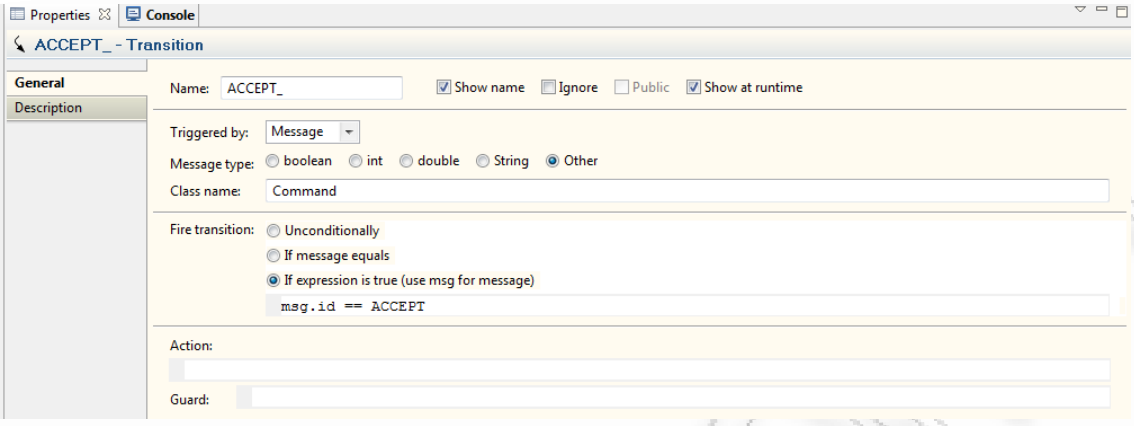
Εικόνα 10-8(δ): Γενικές ιδιότητες της μετάβασης MASTERACK_2.



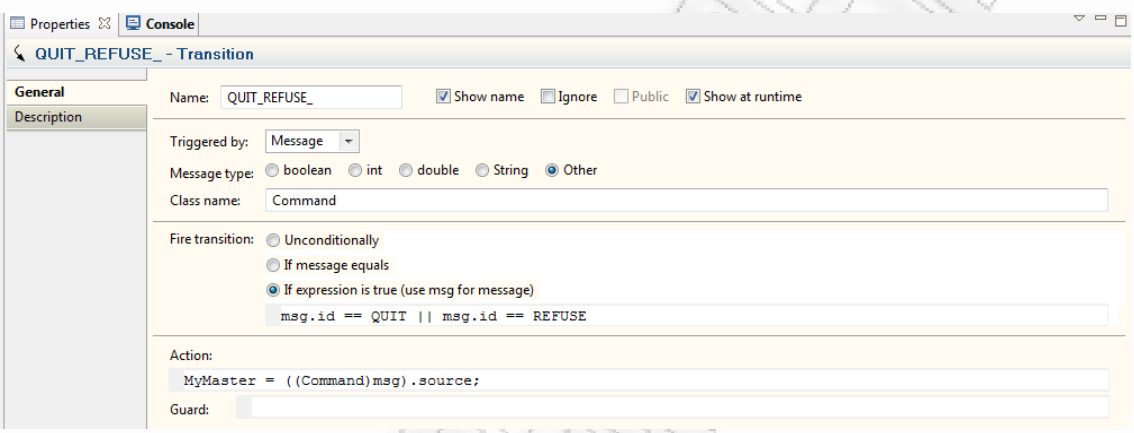
Εικόνα 10-8(ε): Γενικές ιδιότητες της μετάβασης MASTERACK_1.



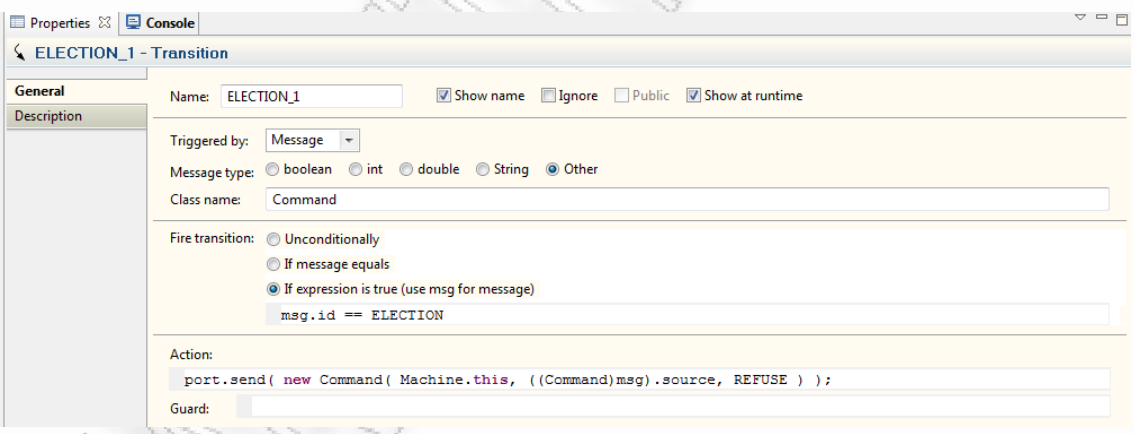
Εικόνα 10-8(στ): Γενικές ιδιότητες της μετάβασης ElectionTimeout.



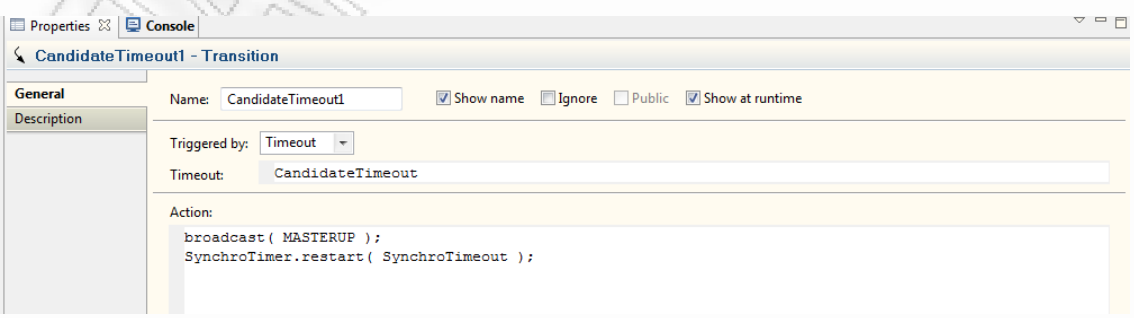
Εικόνα 10-8(ζ): Γενικές ιδιότητες της μετάβασης ACCEPT_.



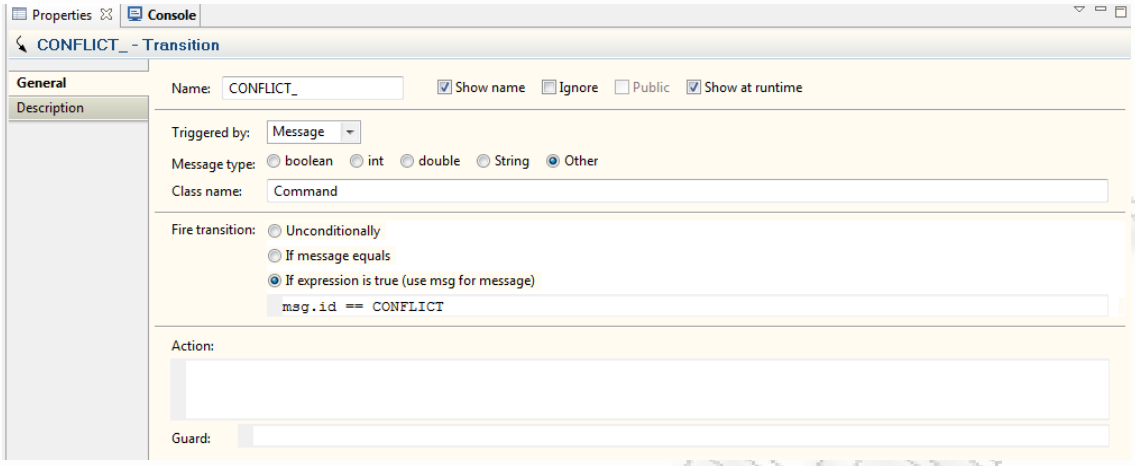
Εικόνα 10-8(η): Γενικές ιδιότητες της μετάβασης QUIT_REFUSE_.



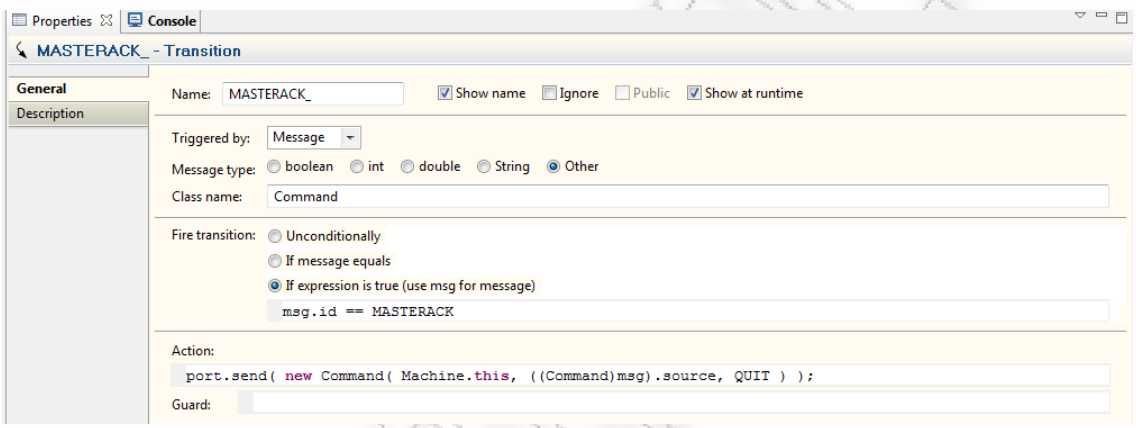
Εικόνα 10-8(θ): Γενικές ιδιότητες της μετάβασης ELECTION_1.



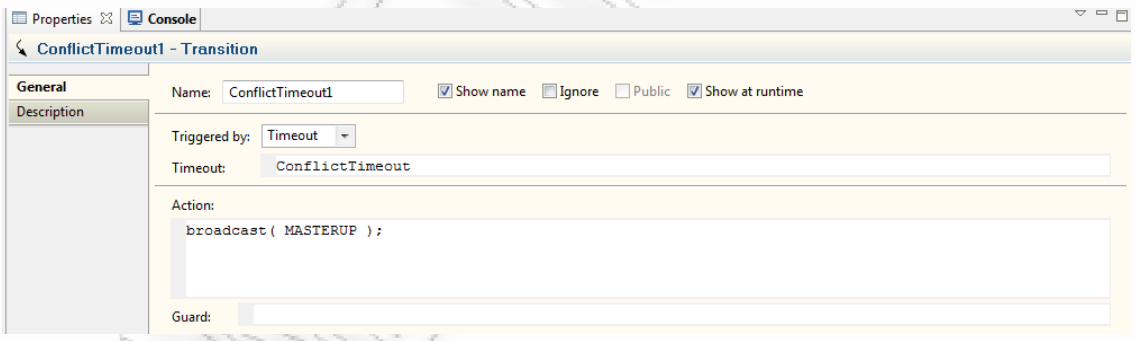
Εικόνα 10-8(ι): Γενικές ιδιότητες της μετάβασης CandidateTimeout1.



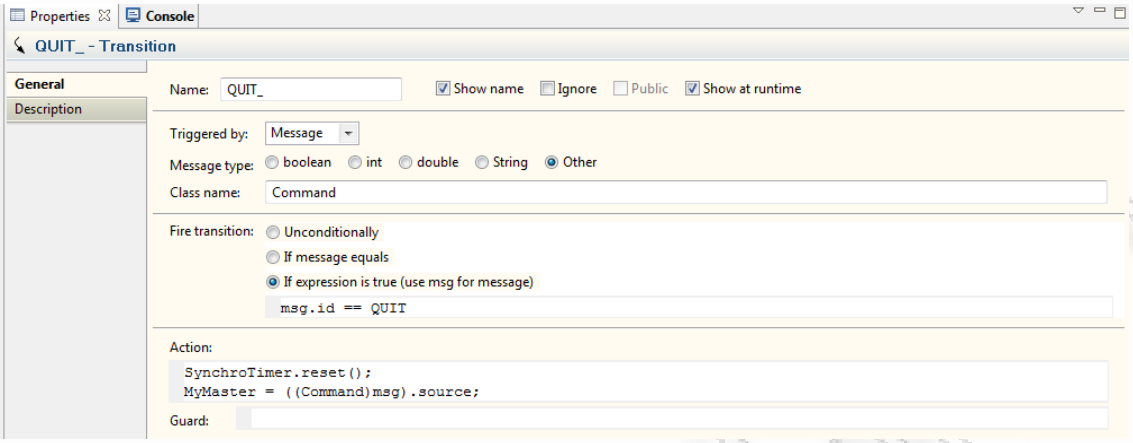
Εικόνα 10-8(ια): Γενικές ιδιότητες της μετάβασης CONFLICT_.



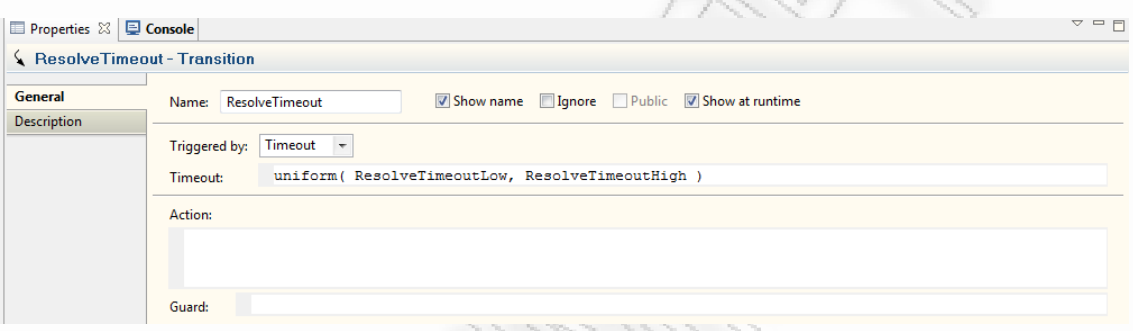
Εικόνα 10-8(ιβ): Γενικές ιδιότητες της μετάβασης MASTERACK_.



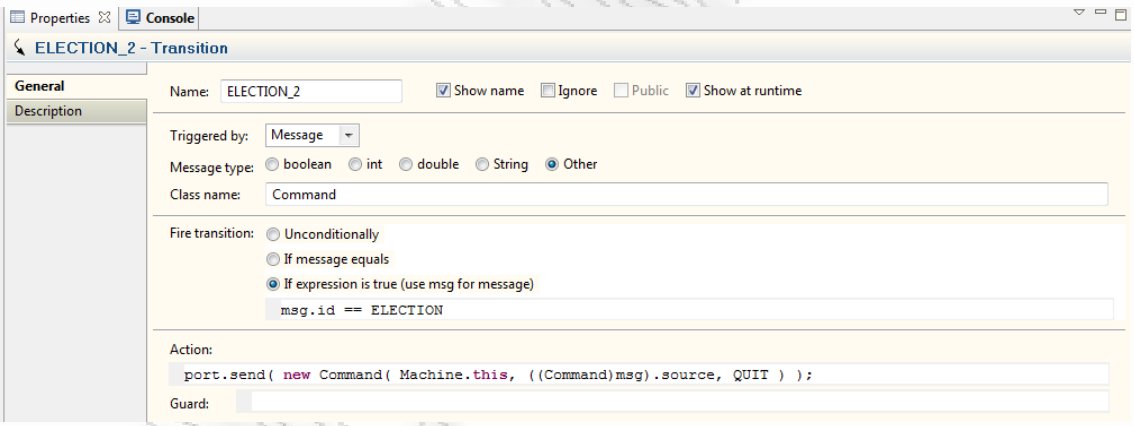
Εικόνα 10-8(ιγ): Γενικές ιδιότητες της μετάβασης ConflictTimeout1.



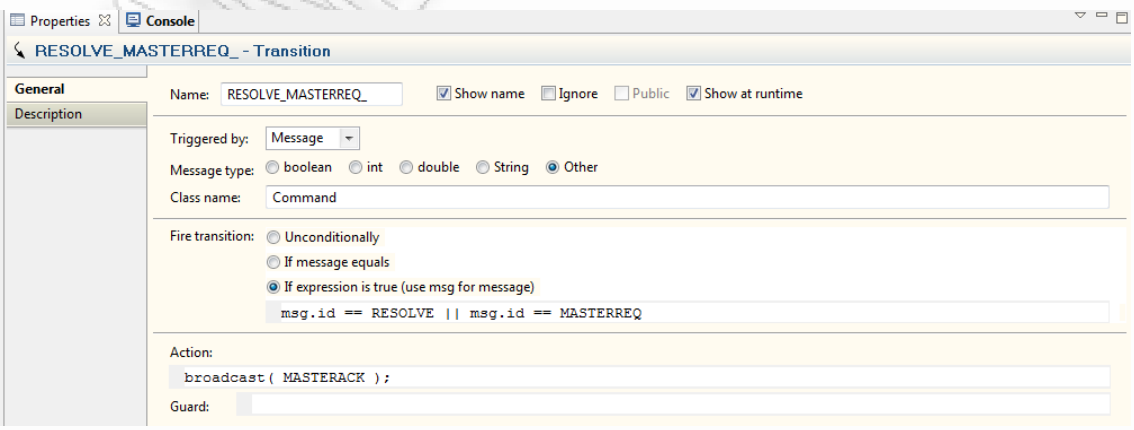
Εικόνα 10-8(ιδ): Γενικές ιδιότητες της μετάβασης QUIT_.



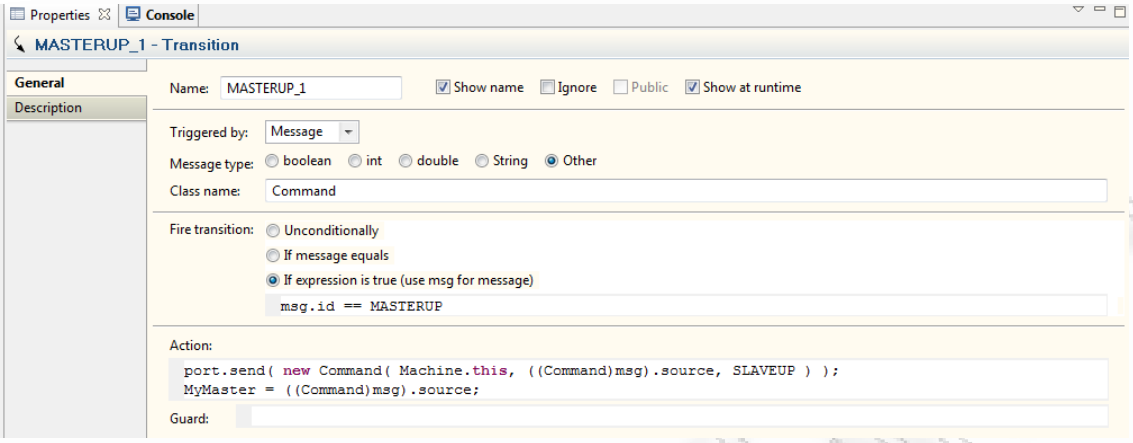
Εικόνα 10-8(ιε): Γενικές ιδιότητες της μετάβασης ResolveTimeout.



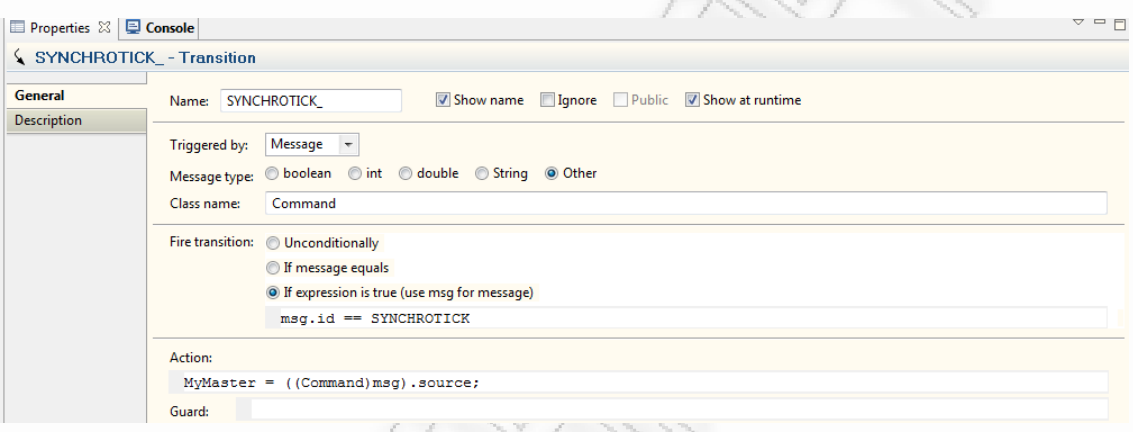
Εικόνα 10-8(ιστ): Γενικές ιδιότητες της μετάβασης ELECTION_2.



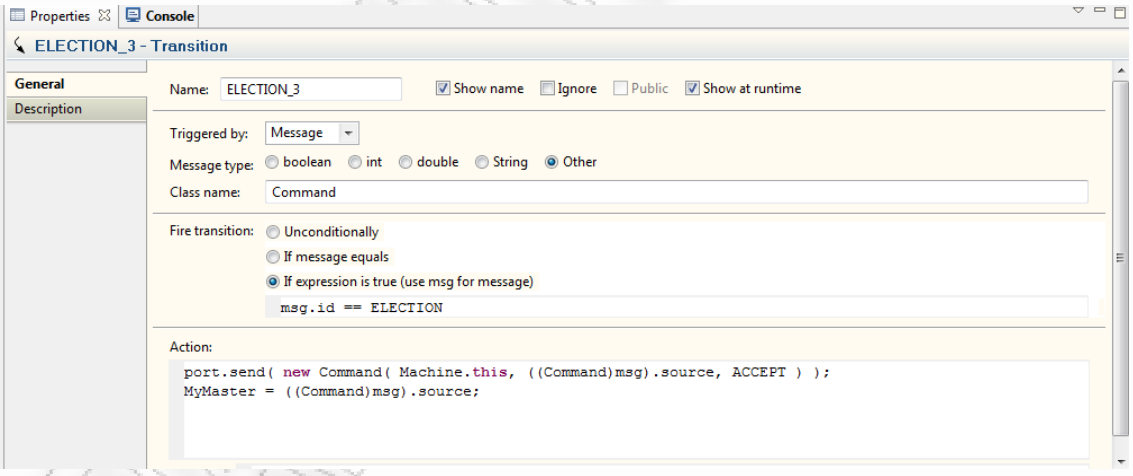
Εικόνα 10-8(ιζ): Γενικές ιδιότητες της μετάβασης RESOLVE_MASTERREQ_.



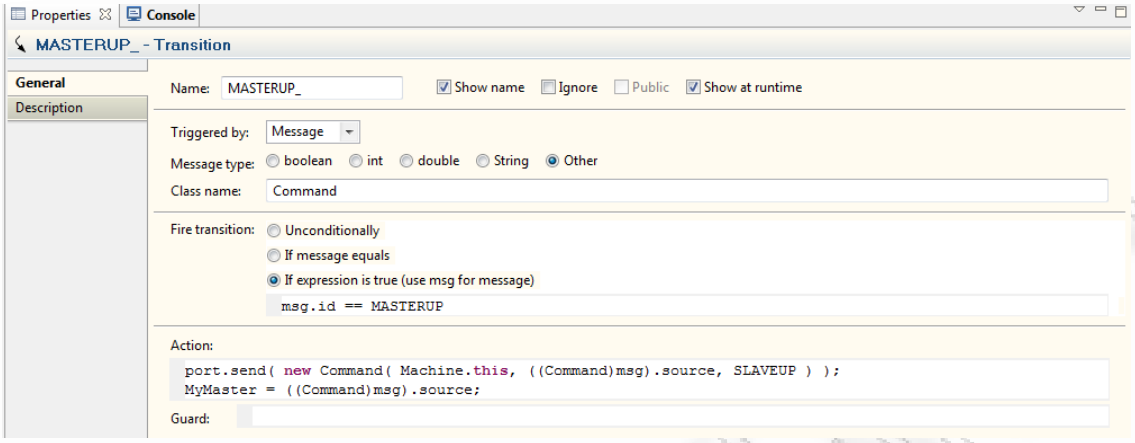
Εικόνα 10-8(ιη): Γενικές ιδιότητες της μετάβασης MASTERUP_1.



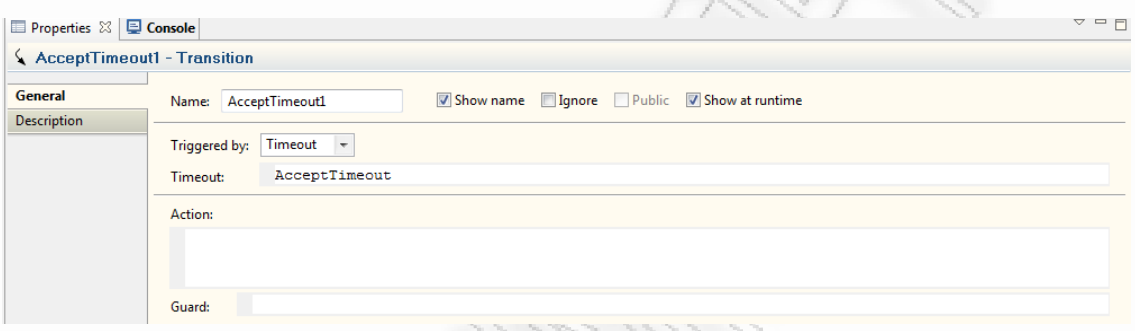
Εικόνα 10-8(ιθ): Γενικές ιδιότητες της μετάβασης SYNCHROTICK_.



Εικόνα 10-8(κ): Γενικές ιδιότητες της μετάβασης ELECTION_3.

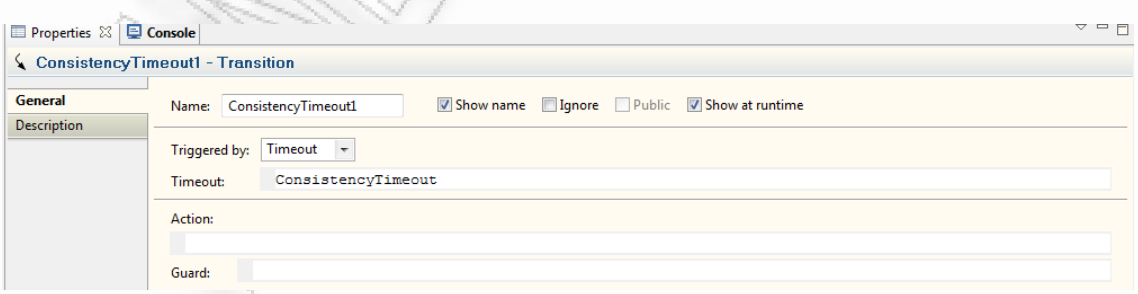


Εικόνα 10-8(κα): Γενικές ιδιότητες της μετάβασης MASTERUP_.

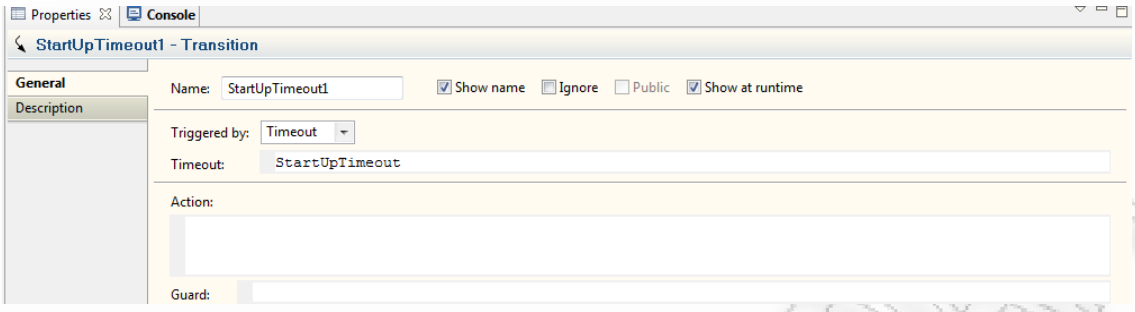


Εικόνα 10-8(κβ): Γενικές ιδιότητες της μετάβασης AcceptTimeout1.

Εικόνα 10-8(κγ): Γενικές ιδιότητες της μετάβασης ELECTION_.

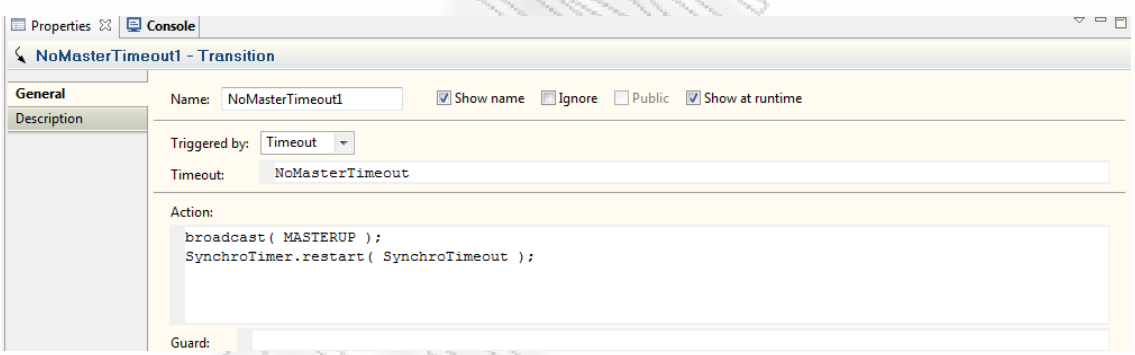


Εικόνα 10-8(κδ): Γενικές ιδιότητες της μετάβασης ConsistencyTimeout1.

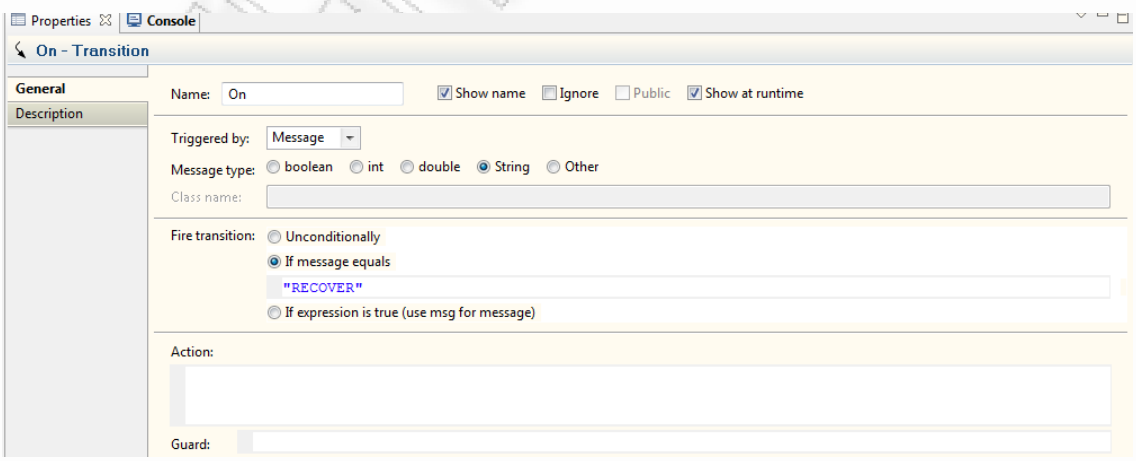


Εικόνα 10-8(κε): Γενικές ιδιότητες της μετάβασης StartUpTimeout1.

Εικόνα 10-8(κστ): Γενικές ιδιότητες της μετάβασης MASTERUP_MASTERACK_ELECTION_.

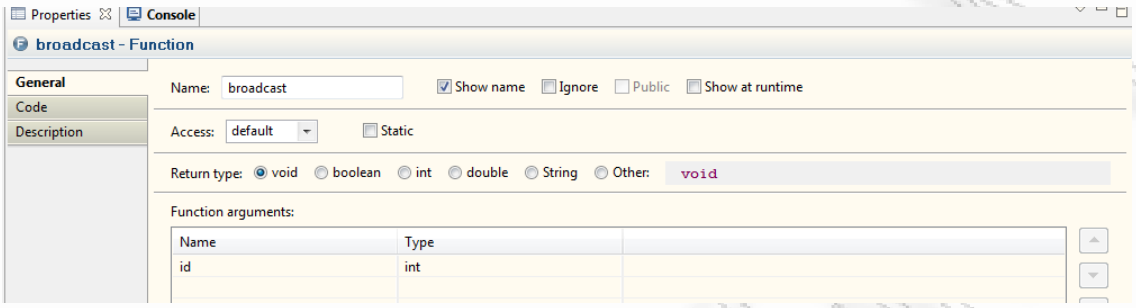


Εικόνα 10-8(κζ): Γενικές ιδιότητες της μετάβασης NoMasterTimeout1.

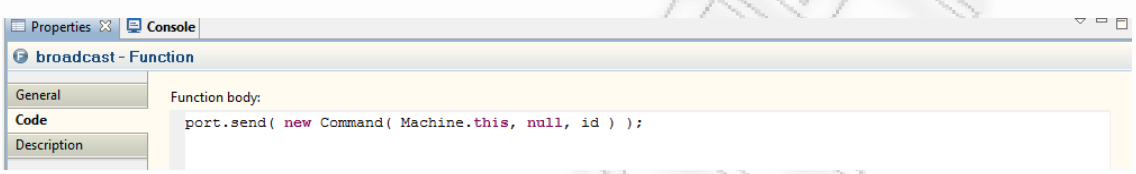


Εικόνα 10-8(κη): Γενικές ιδιότητες της μετάβασης On.

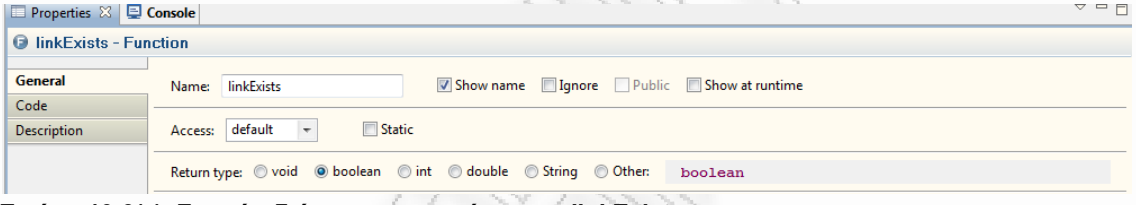
Η κλάση αποτελείται επίσης από επίσης από τρεις συναρτήσεις, Functions, οι οποίες σε συνεργασία με την κλάση Command συνδέουν τους υπολογιστές με τα αναγνωριστικά τους. Στις εικόνες 10-9(α) έως και 10-9(στ) δίνονται οι γενικές ιδιότητες και ο κώδικας λειτουργίας της κάθε μιας από τις τρεις συναρτήσεις.



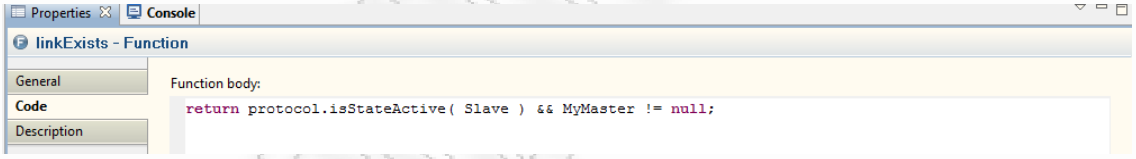
Εικόνα 10-9(α): Γενικές ιδιότητες της συνάρτησης broadcast.



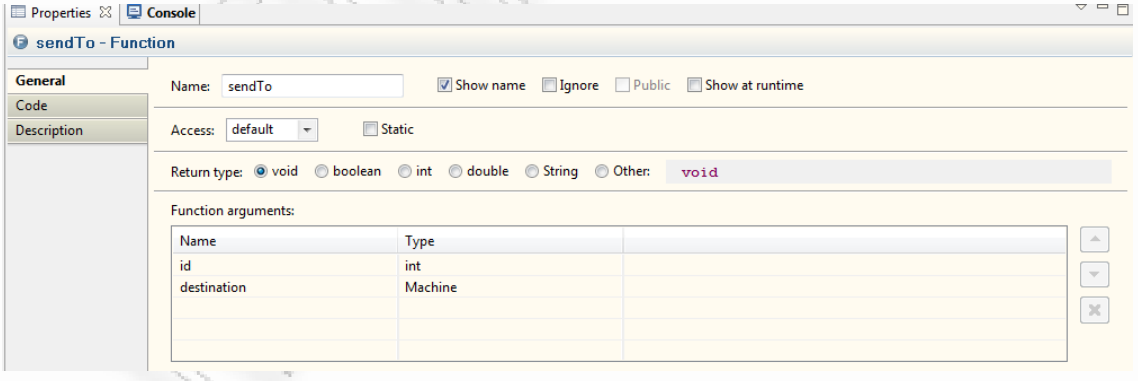
Εικόνα 10-9(β): Ο κώδικας της συνάρτησης broadcast.



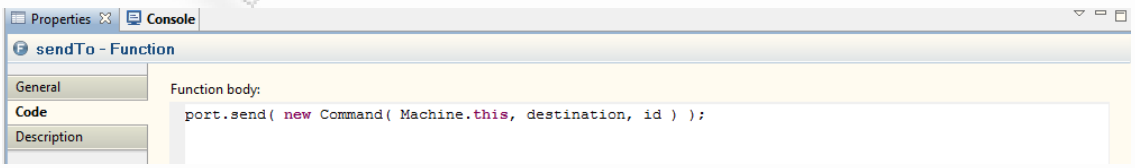
Εικόνα 10-9(γ): Γενικές ιδιότητες της συνάρτησης linkExists.



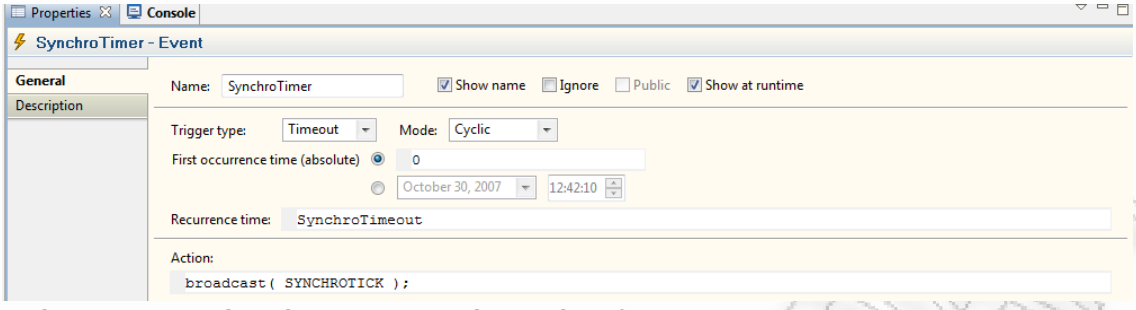
Εικόνα 10-9(δ): Ο κώδικας της συνάρτησης linkExists.



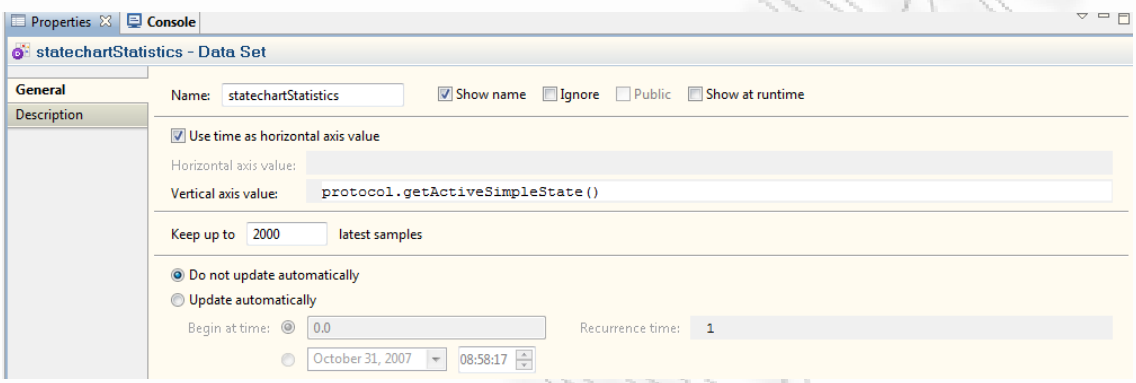
Εικόνα 10-9(ε): Γενικές ιδιότητες της συνάρτησης sendTo.



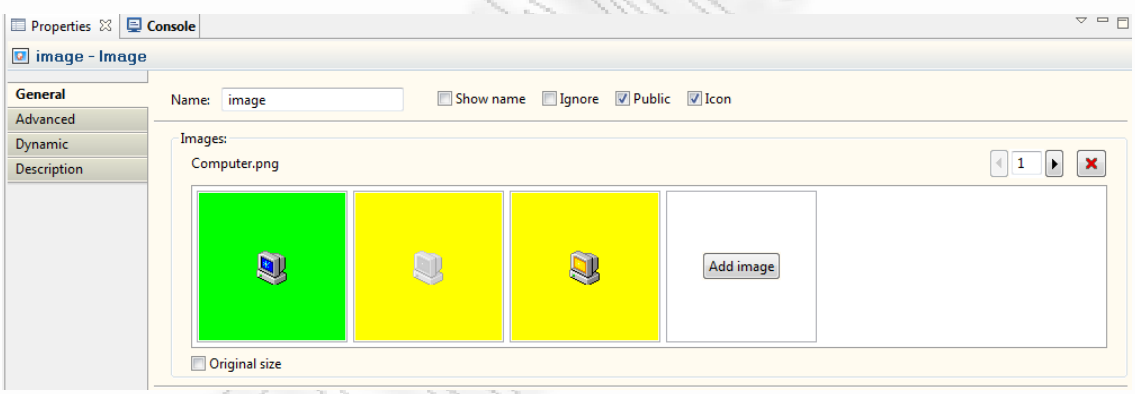
Εικόνα 10-9(στ): Ο κώδικας της συνάρτησης sendTo.



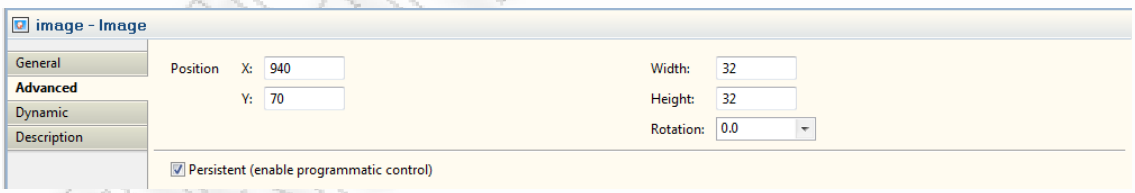
Εικόνα 10-10: Γενικές ιδιότητες του απλού γεγονότος SynchroTimer.



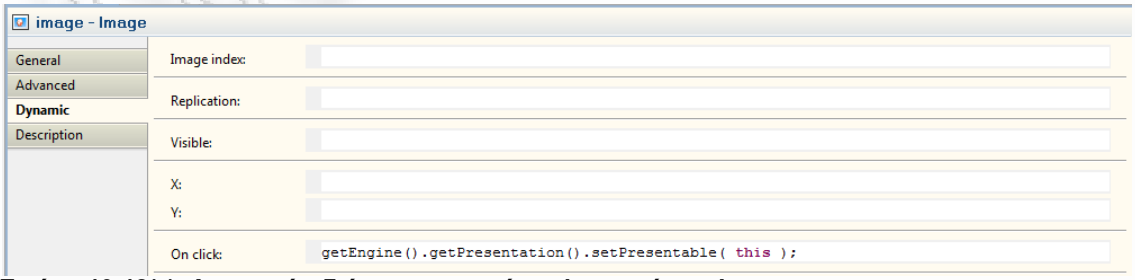
Εικόνα 10-11: Γενικές ιδιότητες του συνόλου δεδομένων statechartStatistics.



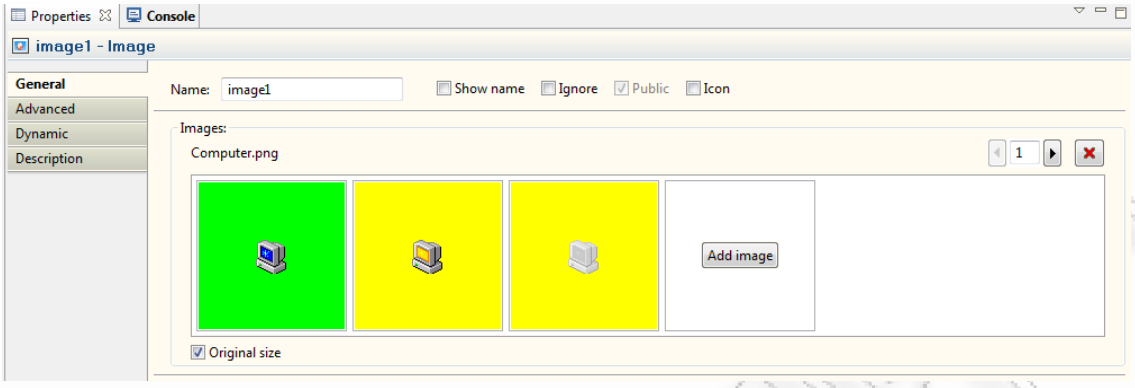
Εικόνα 10-12(α): Γενικές ιδιότητες της εικόνας image τύπου Image.



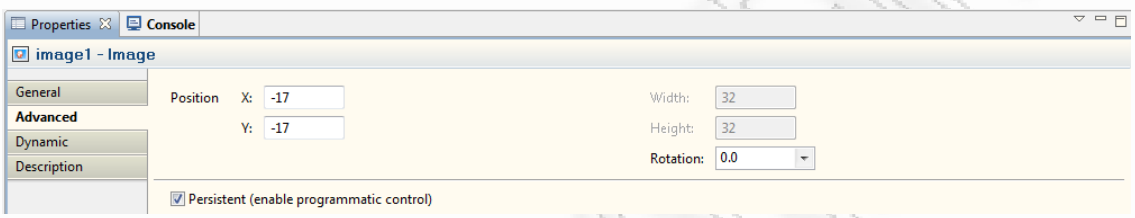
Εικόνα 10-12(β): Προχωρημένες ιδιότητες της εικόνας image τύπου Image.



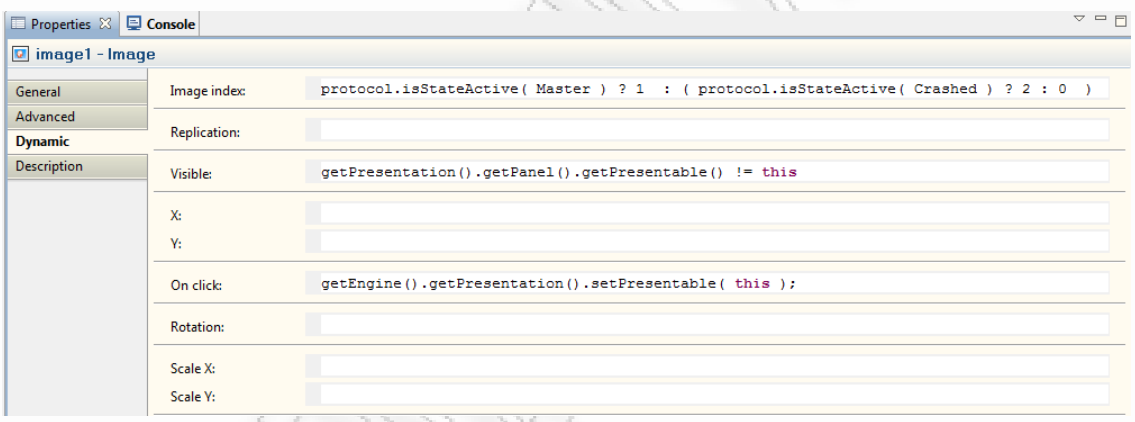
Εικόνα 10-12(γ): Δυναμικές ιδιότητες της εικόνας image τύπου Image.



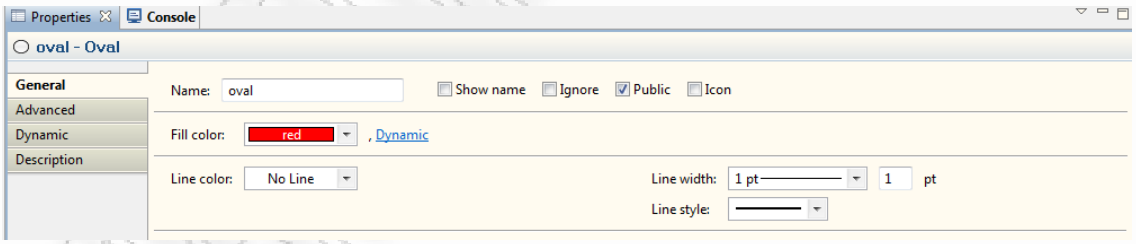
Εικόνα 10-12(δ): Γενικές ιδιότητες της εικόνας image1 τύπου Image.



Εικόνα 10-12(ε): Προχωρημένες ιδιότητες της εικόνας image1 τύπου Image.

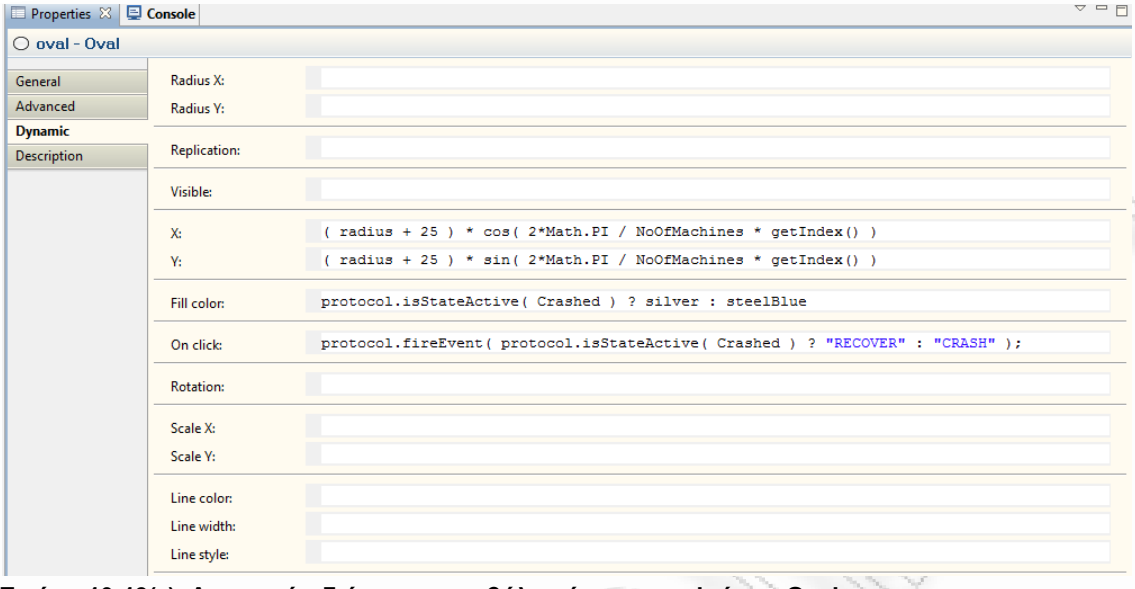


Εικόνα 10-12(στ): Δυναμικές ιδιότητες της εικόνας image1 τύπου Image.

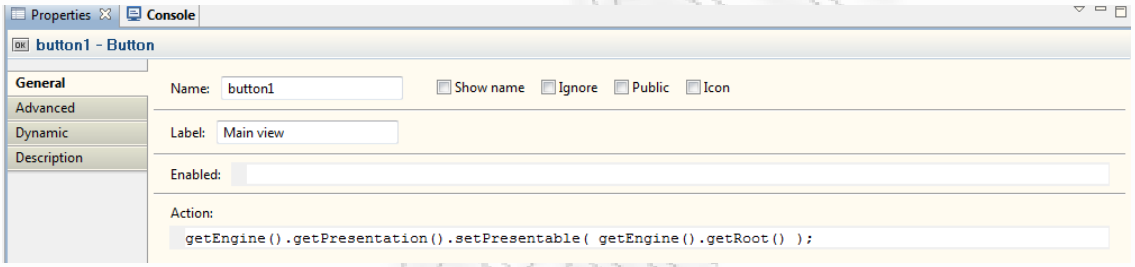


Εικόνα 10-13(α): Γενικές ιδιότητες του οβάλ σχήματος oval τύπου Oval.

Εικόνα 10-13(β): Προχωρημένες ιδιότητες του οβάλ σχήματος oval τύπου Oval.



Εικόνα 10-13(γ): Δυναμικές ιδιότητες του οβάλ σχήματος oval τύπου Oval.

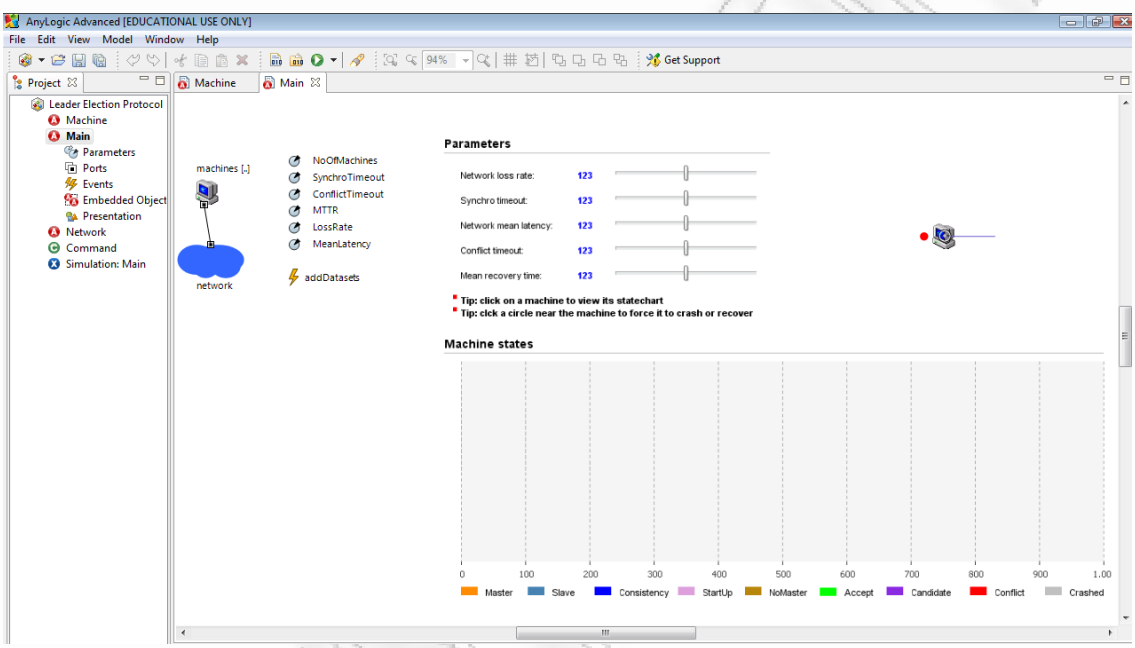


Εικόνα 10-14: Γενικές ιδιότητες του κομβίου button1.

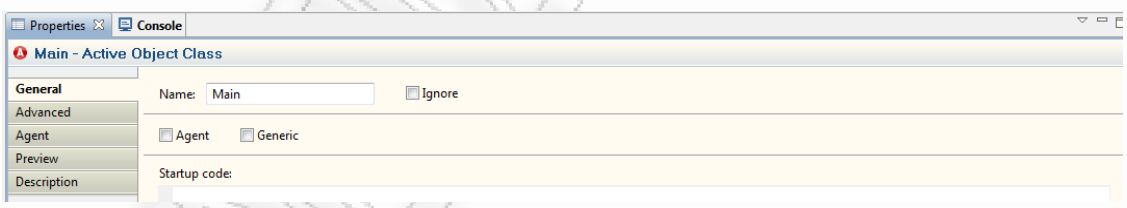
Στην εικόνα 10-10 παρουσιάζονται οι γενικές ιδιότητες του γεγονότος SynchroTimer ενώ στην εικόνα 10-11 δίνονται οι γενικές ιδιότητες του συνόλου δεδομένων. Στις εικόνες 10-12(α) έως και 10-12(στ) παρουσιάζονται οι γενικές, οι προχωρημένες και οι δυναμικές ιδιότητες των στοιχείων εικόνων image και image1 τύπου Image. Μέσω των εικόνων φαίνεται στην προσομοίωση αν οι υπολογιστές είναι ενεργοί ή ανενεργοί. Το στοιχείο σχήματος οβάλ βρίσκεται επάνω και αριστερά στην εικόνα 10-2(α), αριστερά από την εικόνα του υπολογιστή. Στις εικόνες 10-13(α) έως και 10-13(γ) δίνονται οι γενικές, οι προχωρημένες και οι δυναμικές ιδιότητες του στοιχείου oval τύπου Oval. Τέλος, στην εικόνα 10-14 παρουσιάζονται οι γενικές ιδιότητες του κομβίου με το οποίο μεταφερόμαστε στην κλάση Main.

10.2 Η κλάση Main

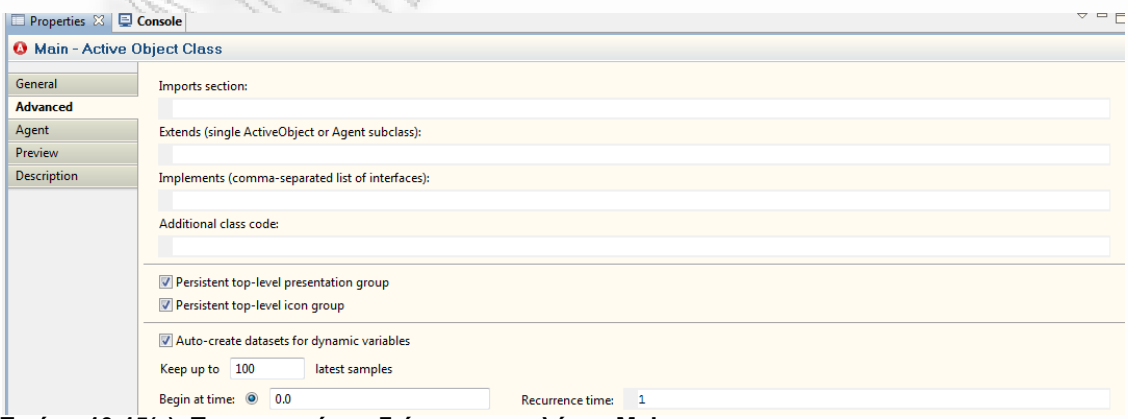
Η κλάση αποτελείται από έξι (6) παραμέτρους, ένα (1) γεγονός, πέντε (5) ράβδους κύλισης, ένα (1) δίκτυο τύπου Network, ένα αντικείμενο υπολογιστών τύπου Machine, ένα (1) διάγραμμα γραφικών εμφάνισης των καταστάσεων των υπολογιστών και ένα (1) ενσωματωμένο αντικείμενο εμφάνισης του υπολογιστή. Στις εικόνες 10-15(α) έως και 10-15(γ) εμφανίζεται η κλάση όπως έχει υλοποιηθεί στο γραφικό συντάκτη και οι γενικές και προχωρημένες ιδιότητες της, αντίστοιχα. Στην εικόνα 10-15(δ) γίνεται η προεπισκόπιση των παραμέτρων της κλάσης και στις εικόνες 10-16(α) έως και 10-16(στ) παρουσιάζονται οι γενικές ιδιότητες των παραμέτρων.



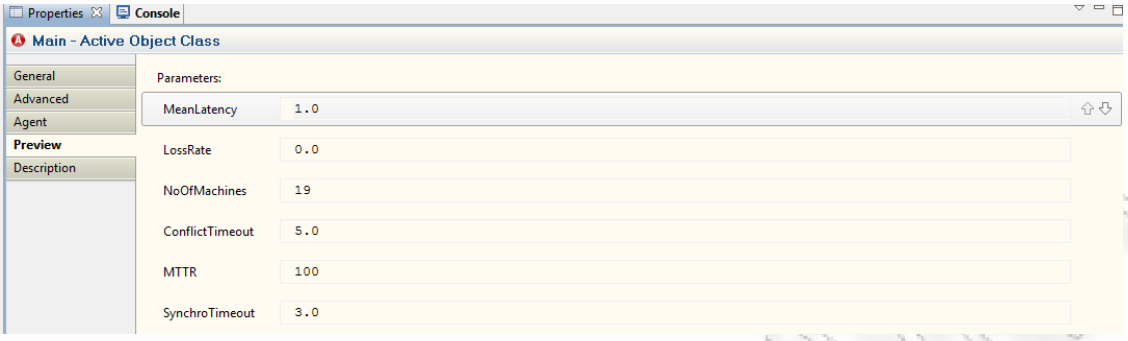
Εικόνα 10-15(α): Η κλάση Main.



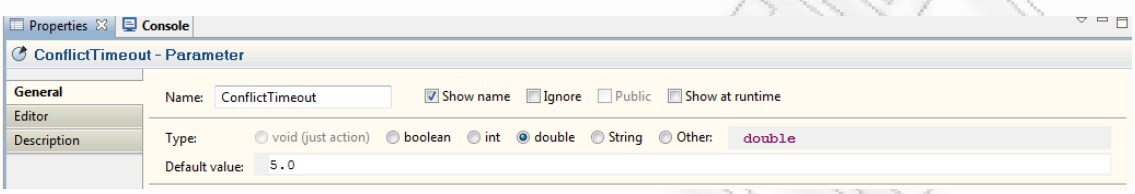
Εικόνα 10-15(β): Γενικές ιδιότητες της κλάσης Main.



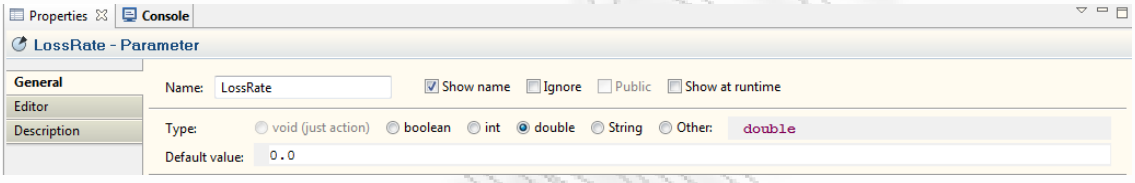
Εικόνα 10-15(γ): Προχωρημένες ιδιότητες της κλάσης Main.



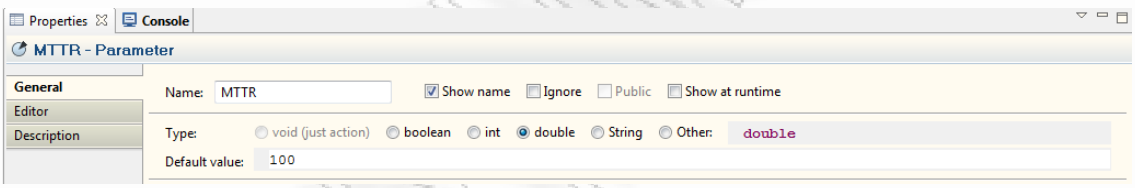
Εικόνα 10-15(δ): Προεπισκόπιση των παραμέτρων της κλάσης Main.



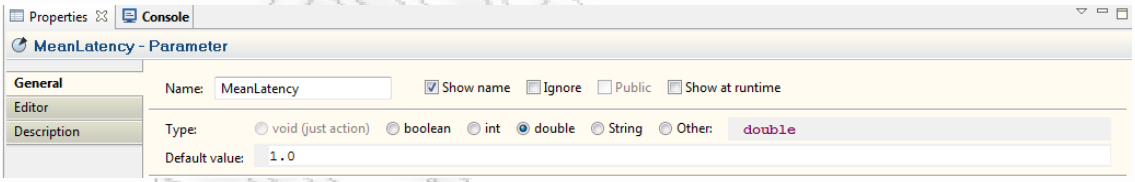
Εικόνα 10-16(α): Γενικές ιδιότητες της παραμέτρου ConflictTimeout.



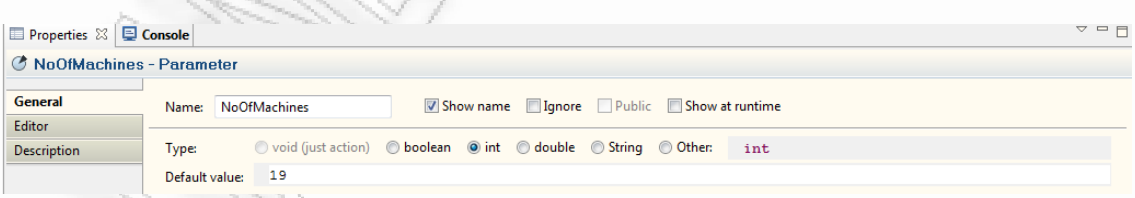
Εικόνα 10-16(β): Γενικές ιδιότητες της παραμέτρου LossRate.



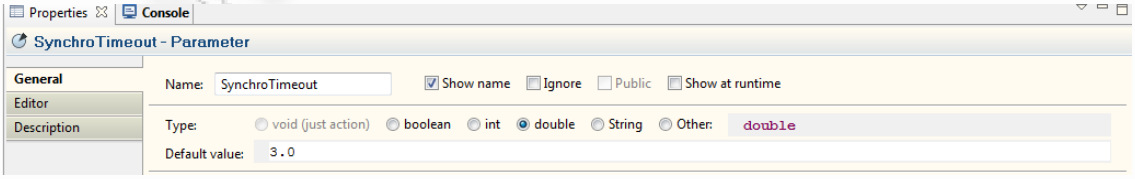
Εικόνα 10-16(γ): Γενικές ιδιότητες της παραμέτρου MTTR.



Εικόνα 10-16(δ): Γενικές ιδιότητες της παραμέτρου MeanLatency.

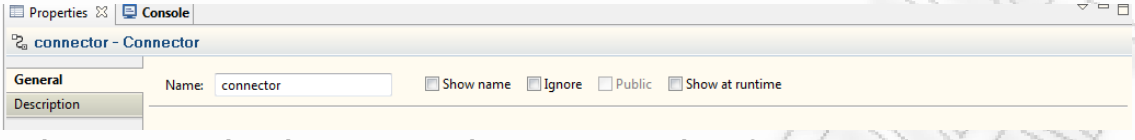


Εικόνα 10-16(ε): Γενικές ιδιότητες της παραμέτρου NoOfMachines.

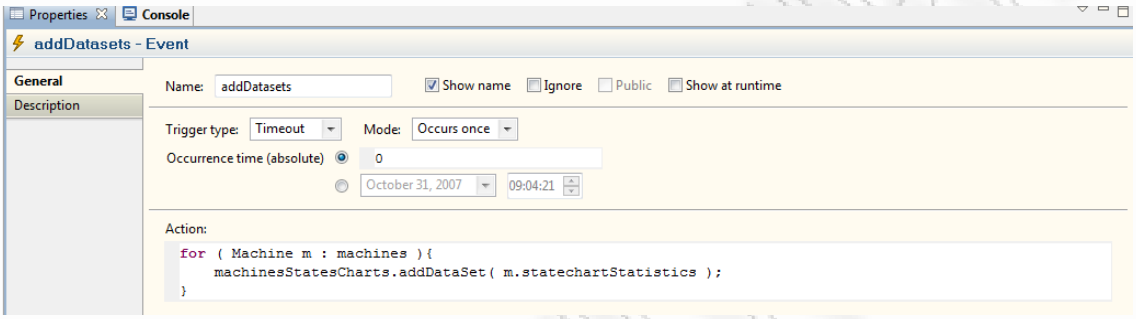


Εικόνα 10-16(στ): Γενικές ιδιότητες της παραμέτρου SynchroTimeout.

Η εικόνα 10-17 δεν έχει καμία πληροφορία να προσφέρει. Το στοιχείο διασύνδεσης connector τύπου Connector είναι το ευθύγραμμο τμήμα που συνδέει το εικονίδιο του δικτύου, (σύννεφο), με το εικονίδιο του υπολογιστή. Στην εικόνα 10-18 παρουσιάζονται οι γενικές ιδιότητες του γεγονότος addDatasets.

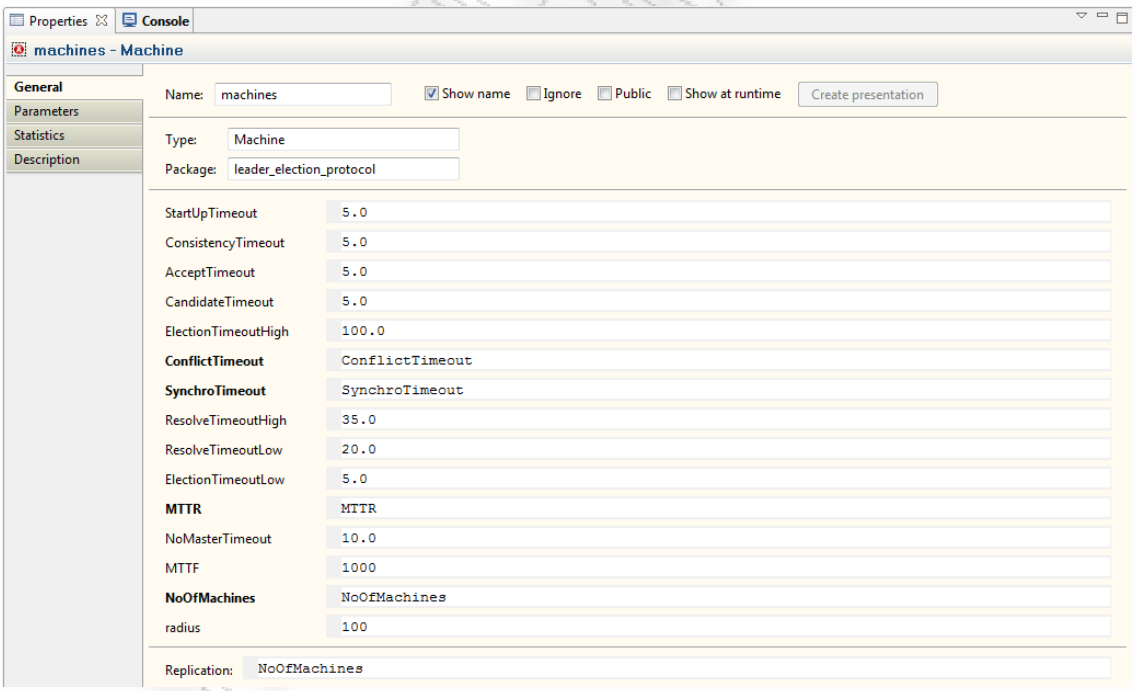


Εικόνα 10-17: Γενικές ιδιότητες του συνδέσμου connector τύπου Connector.

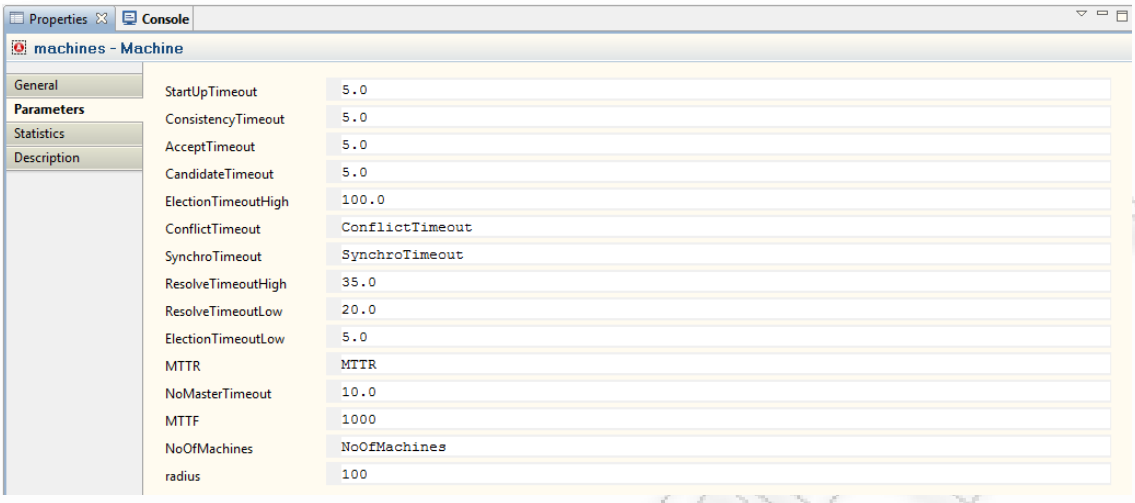


Εικόνα 10-18: Γενικές ιδιότητες του γεγονότος addDatasets.

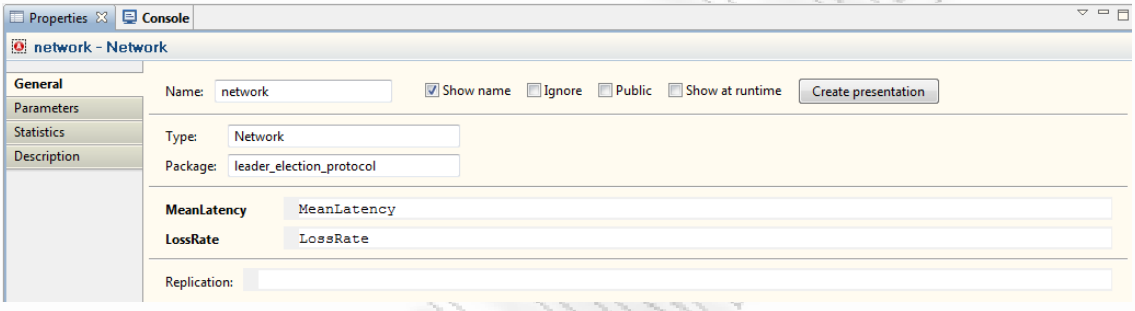
Η εικόνες 10-19(α) έως και 10-19(δ) εμφανίζουν τις γενικές ιδιότητες του ενεργού αντικειμένου machines τύπου Machine και του ενεργού αντικειμένου network τύπου Network. Η εικόνα 10-19(ε) δείχνει τις γενικές ιδιότητες του ενεργού αντικειμένου machines_presentation.



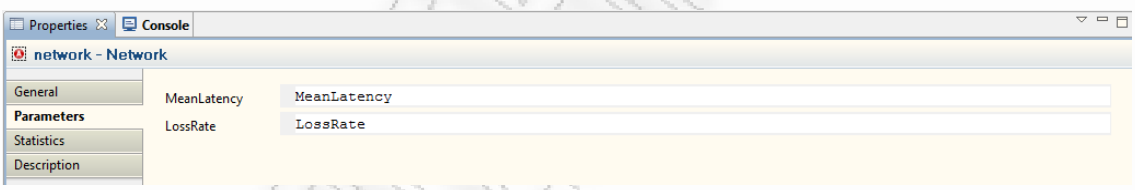
Εικόνα 10-19(α): Γενικές ιδιότητες του αντικειμένου machines τύπου Machine.



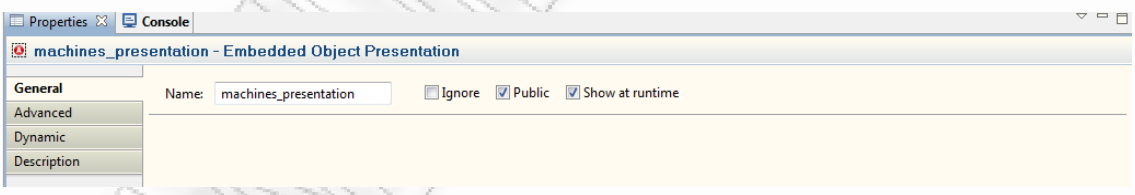
Εικόνα 10-19(β): Παράμετροι του αντικειμένου machines τύπου Machine.



Εικόνα 10-19(γ): Γενικές ιδιότητες του αντικειμένου network τύπου Network.



Εικόνα 10-19(δ): Παράμετροι του αντικειμένου network τύπου Network.

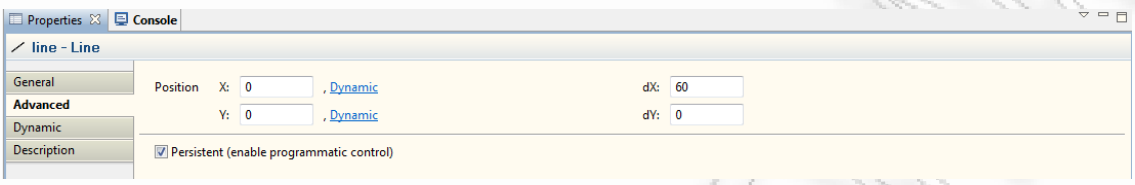


Εικόνα 10-19(ε): Γενικές ιδιότητες του αντικειμένου machines_presentation.

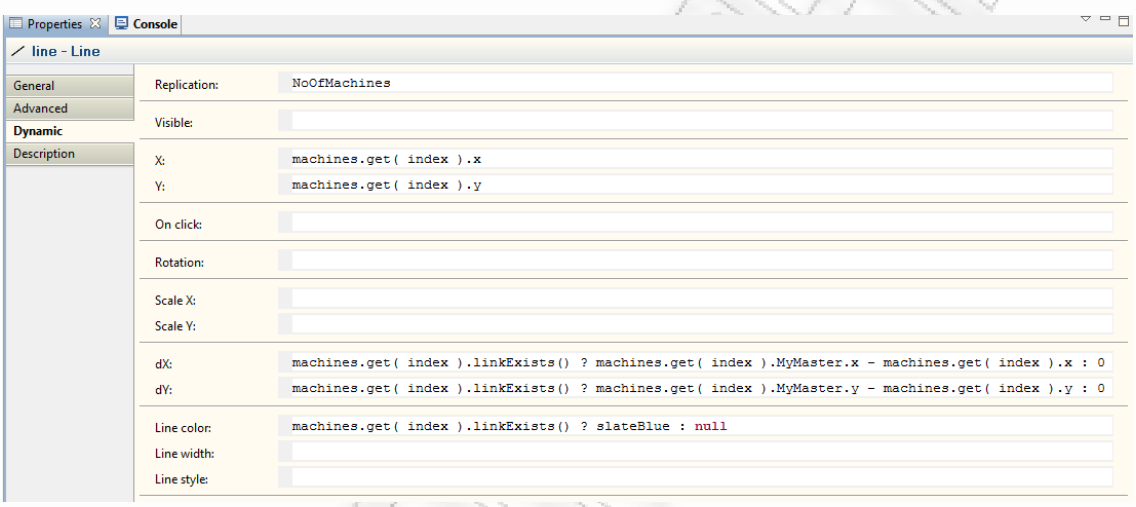
Στις εικόνες 10-20(α) έως και 10-20(γ) δίνονται οι ιδιότητες του στοιχείου γραμμής line τύπου Line. Η γραμμή αυτή συνδέεται με το εικονίδιο του υπολογιστή που βρίσκεται δεξιά στην εικόνα 10-15(α). Οι δυναμικές ιδιότητες των στοιχείων κειμένου που συνδέονται με τις ράβδους κύλισης εμφανίζονται στις εικόνες 10-21(β) έως και 10-21(στ). Στην εικόνα 10-21(α) δίνονται οι γενικές για ένα στοιχείο αλλά ισχύει το ίδιο και για τα υπόλοιπα.

Στις εικόνες 10-22(α) έως και 10-22(ε) παρουσιάζονται οι γενικές ιδιότητες των ράβδων κύλισης, ενώ στην εικόνα 10-23 οι γενικές ιδιότητες του διαγράμματος γραφικών που εμφανίζει τις καταστάσεις των υπολογιστών του δικτύου.

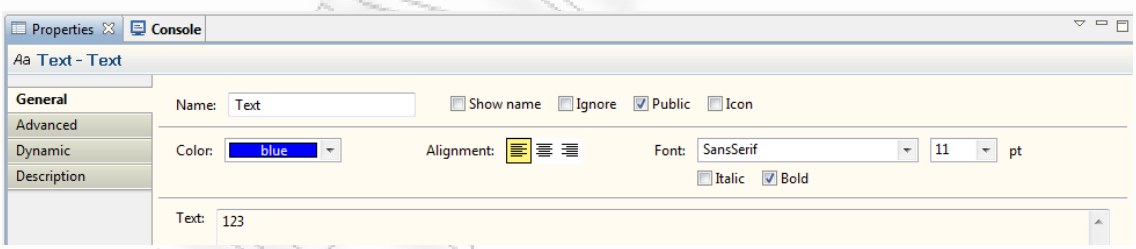
Εικόνα 10-20(α): Γενικές ιδιότητες της γραμμής line τύπου Line.



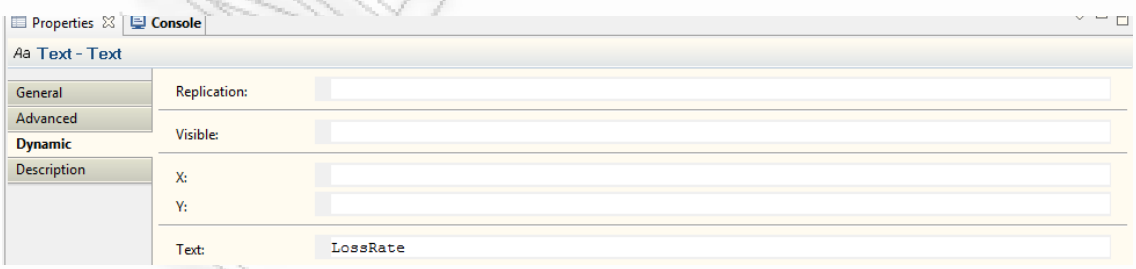
Εικόνα 10-20(β): Προχωρημένες ιδιότητες της γραμμής line τύπου Line.



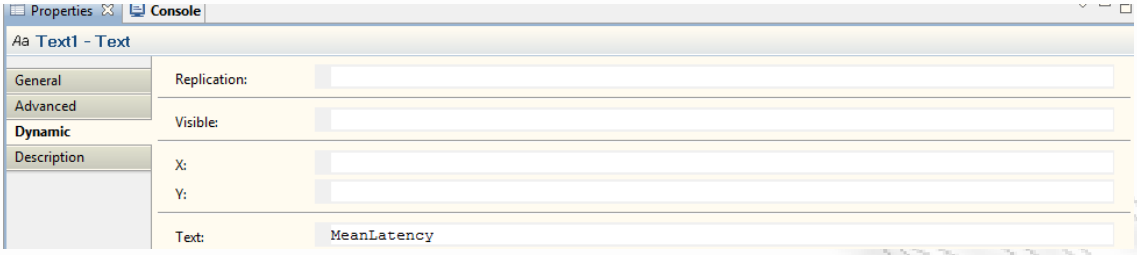
Εικόνα 10-20(γ): Δυναμικές ιδιότητες της γραμμής line τύπου Line.



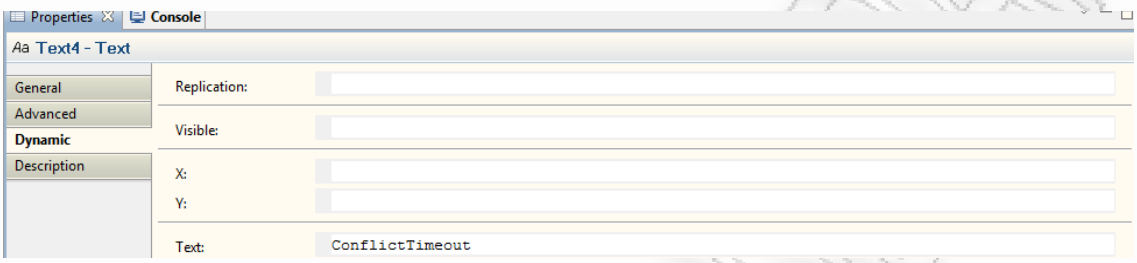
Εικόνα 10-21(α): Γενικές ιδιότητες του κειμένου Text τύπου Text.



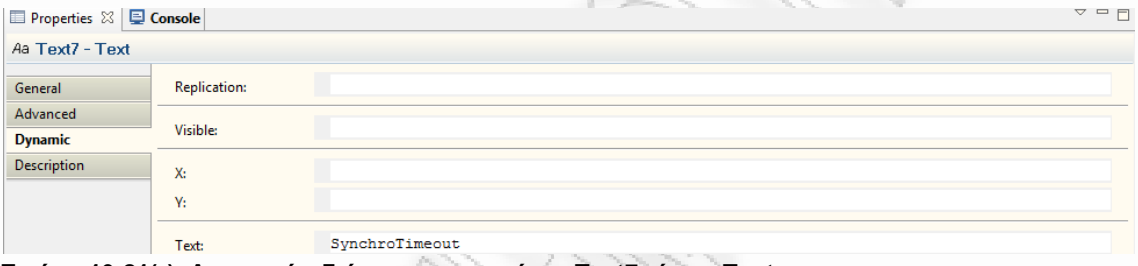
Εικόνα 10-21(β): Δυναμικές ιδιότητες του κειμένου Text τύπου Text.



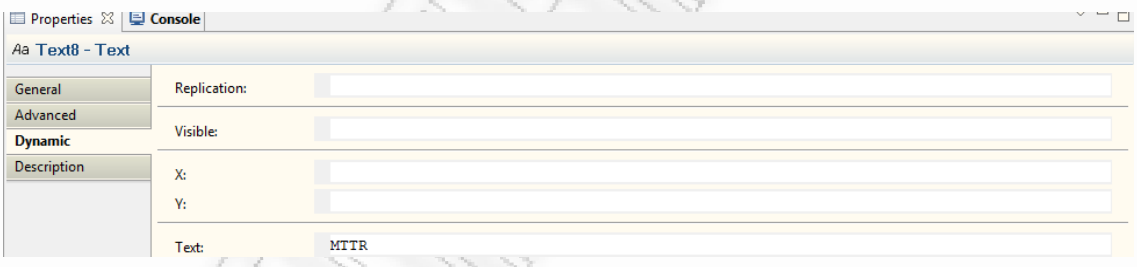
Εικόνα 10-21(γ): Δυναμικές ιδιότητες του κειμένου Text1 τύπου Text.



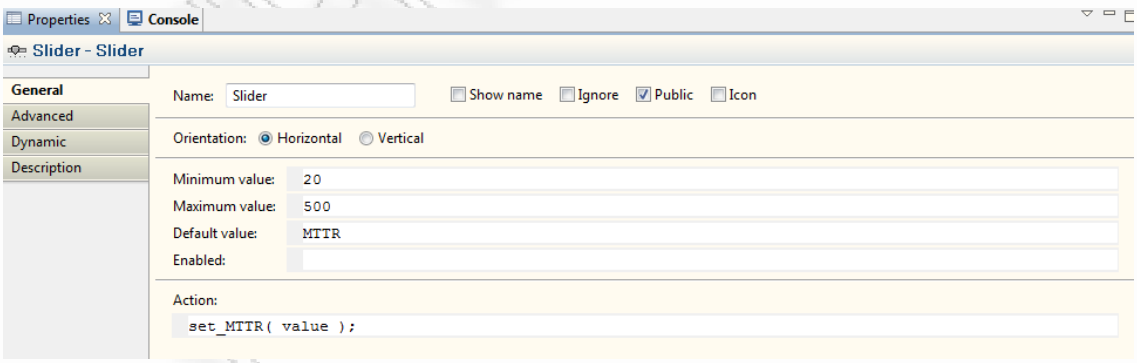
Εικόνα 10-21(δ): Δυναμικές ιδιότητες του κειμένου Text4 τύπου Text.



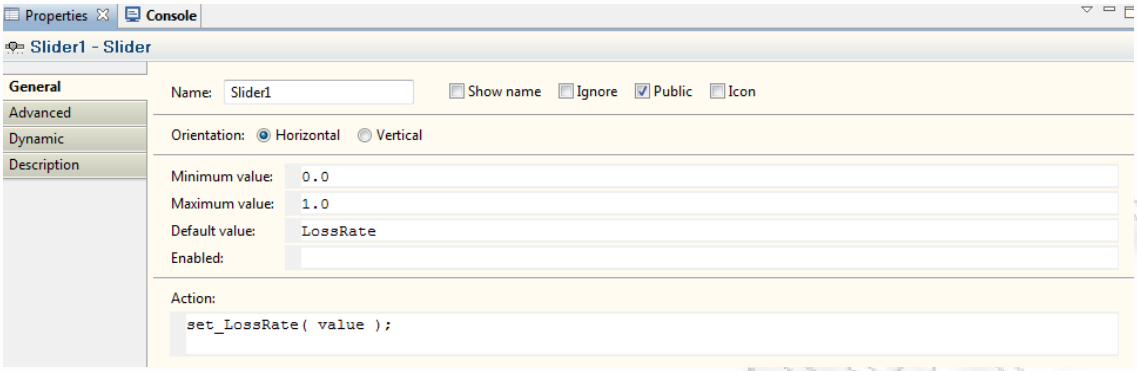
Εικόνα 10-21(ε): Δυναμικές ιδιότητες του κειμένου Text7 τύπου Text.



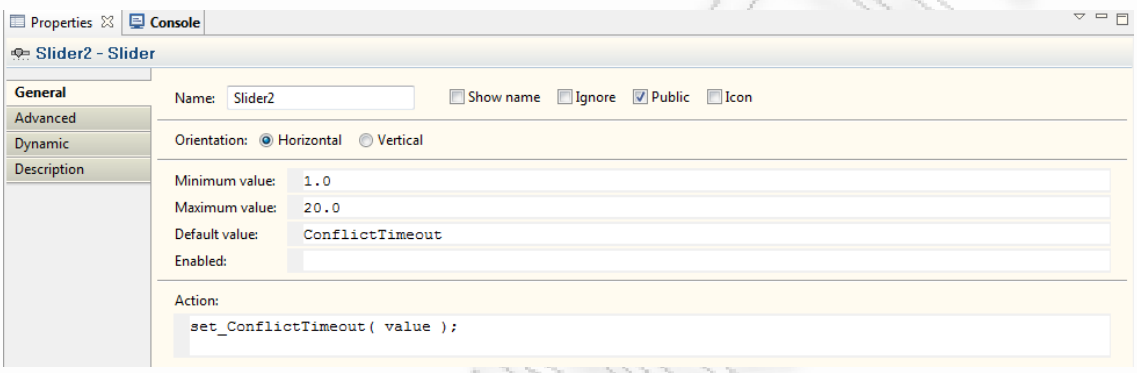
Εικόνα 10-21(στ): Δυναμικές ιδιότητες του κειμένου Text8 τύπου Text.



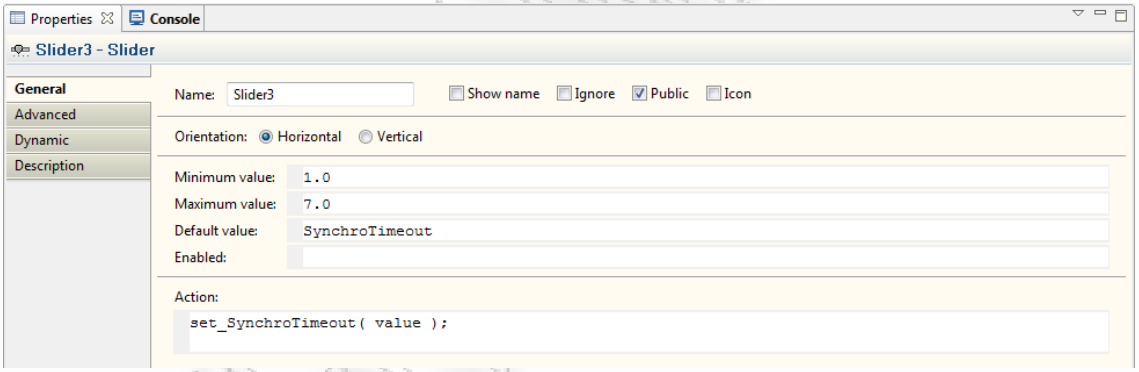
Εικόνα 10-22(α): Γενικές ιδιότητες της ράβδου κύλισης Slider.



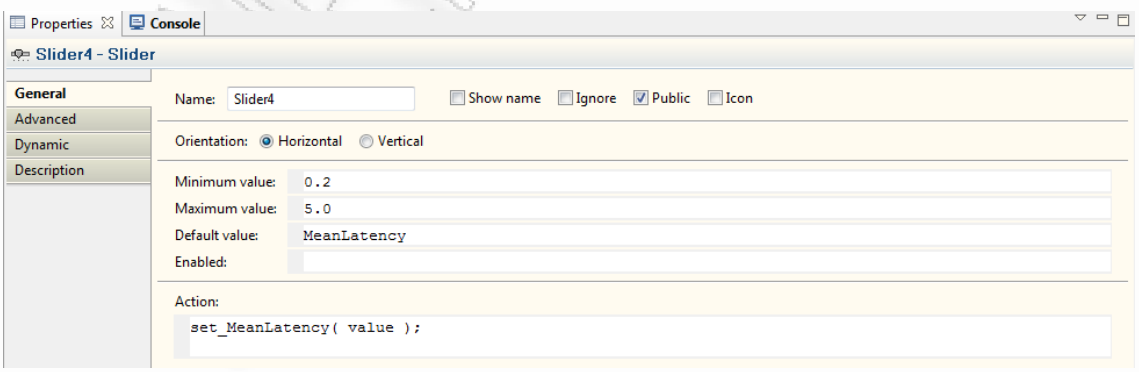
Εικόνα 10-22(β): Γενικές ιδιότητες της ράβδου κύλισης Slider1.



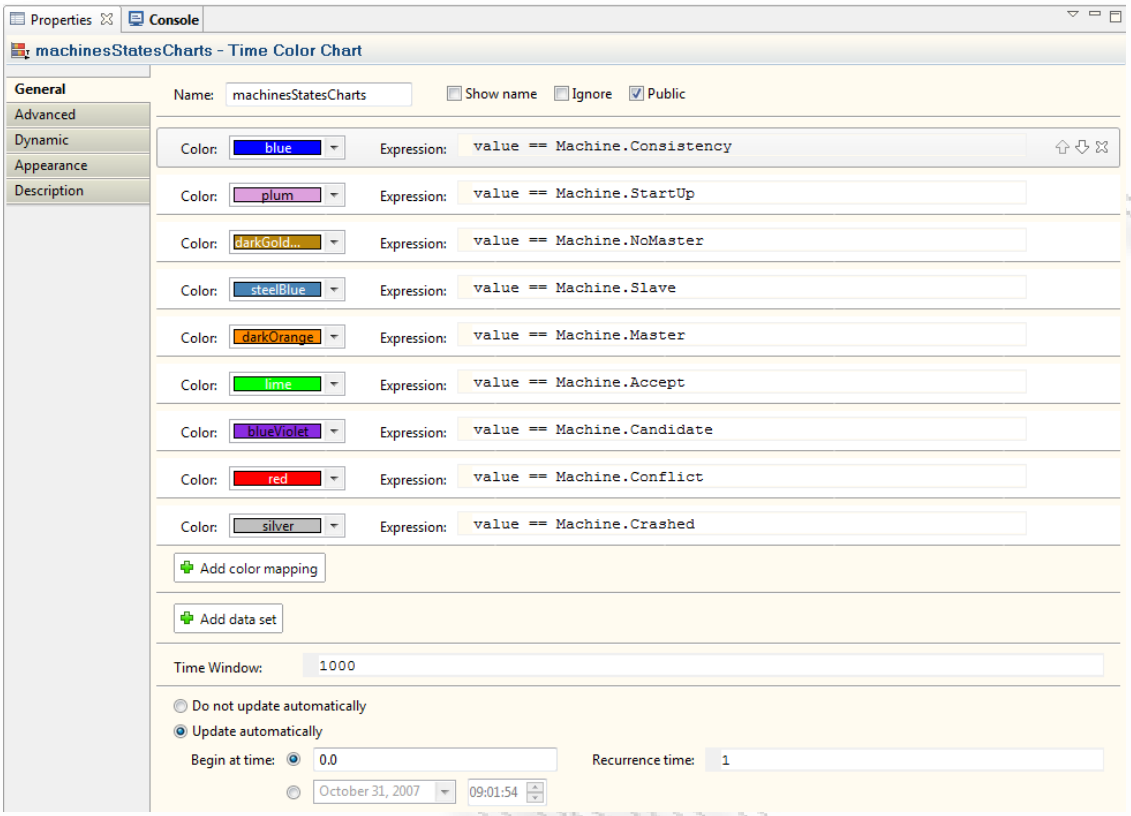
Εικόνα 10-22(γ): Γενικές ιδιότητες της ράβδου κύλισης Slider2.



Εικόνα 10-22(δ): Γενικές ιδιότητες της ράβδου κύλισης Slider3.



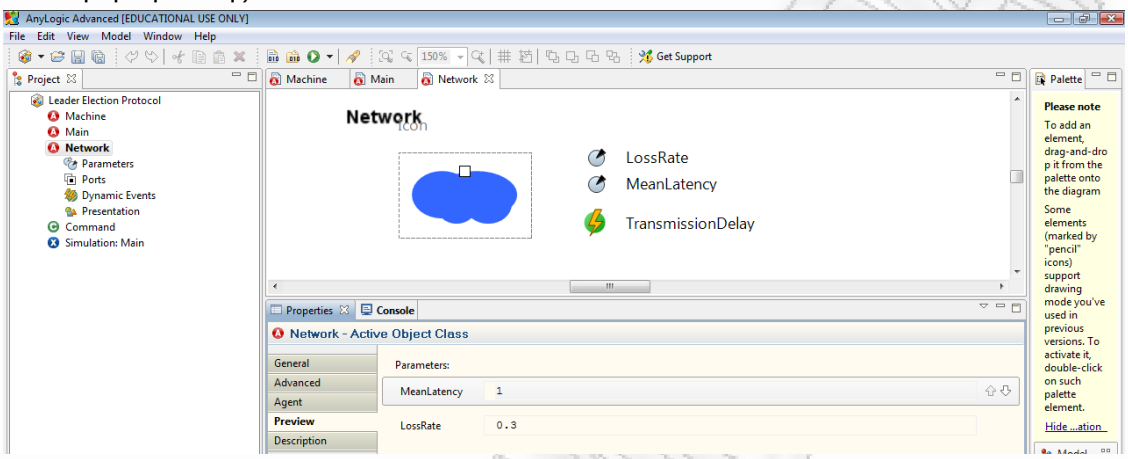
Εικόνα 10-22(ε): Γενικές ιδιότητες της ράβδου κύλισης Slider4.



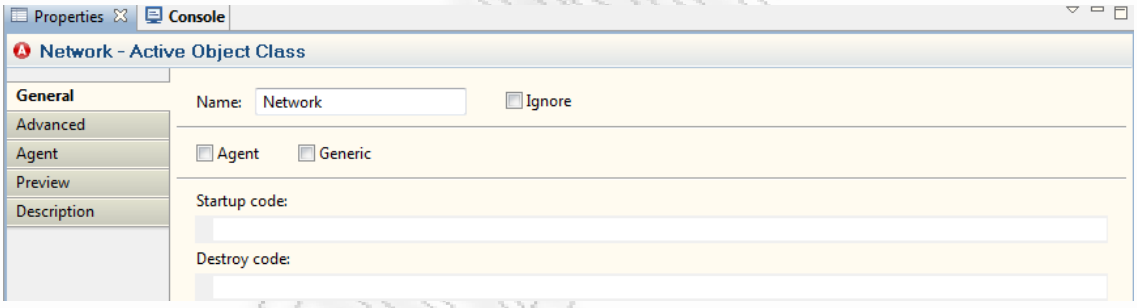
Εικόνα 10-23: Γενικές ιδιότητες του έγχρωμου διαγράμματος χρόνου machinesStatesCharts.

10.3 Η κλάση Network

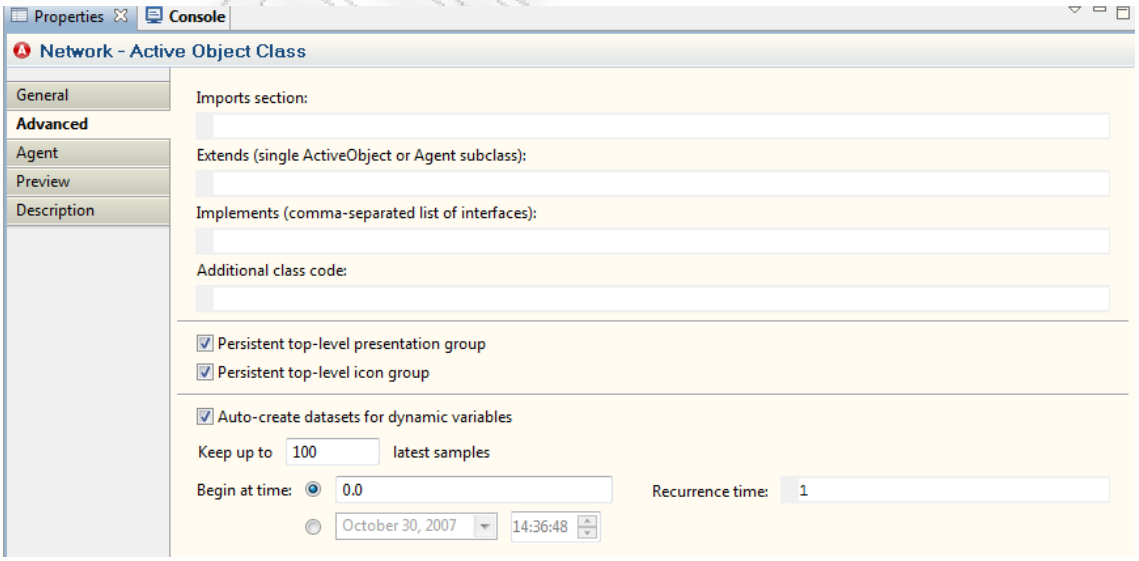
Η κλάση Network αποτελείται από ένα (1) εικονίδιο στο οποίο είναι τοποθετημένη μια (1) θύρα, δύο (2) παραμέτρους και ένα (1) δυναμικό γεγονός. Οι παράμετροι και το δυναμικό γεγονός μοντελοποιούν τα εσωτερικά χαρακτηριστικά του δικτύου, όπως δηλώνουν τα ονόματα τους. Στις εικόνες 10-24(α) έως και 10-24(γ) δίνονται οι ιδιότητες της κλάσης και δίνονται και οι τιμές των παραμέτρων της.



Εικόνα 10-24(α): Η κλάση Network και η προεπισκόπιση των παραμέτρων της.

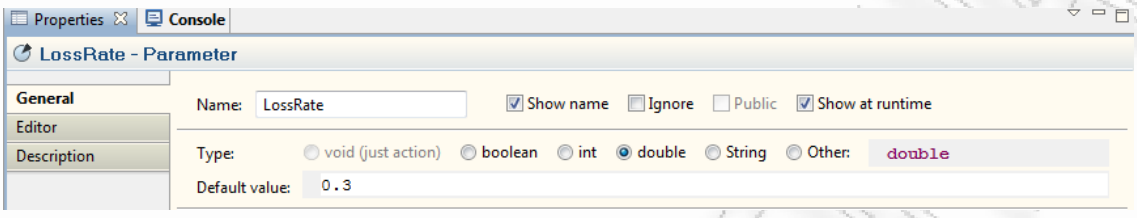


Εικόνα 10-24(β): Γενικές ιδιότητες της κλάσης Network.

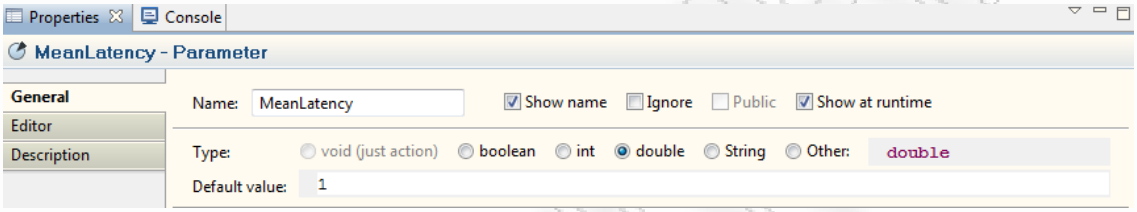


Εικόνα 10-24(γ): Προχωρημένες ιδιότητες της κλάσης Network.

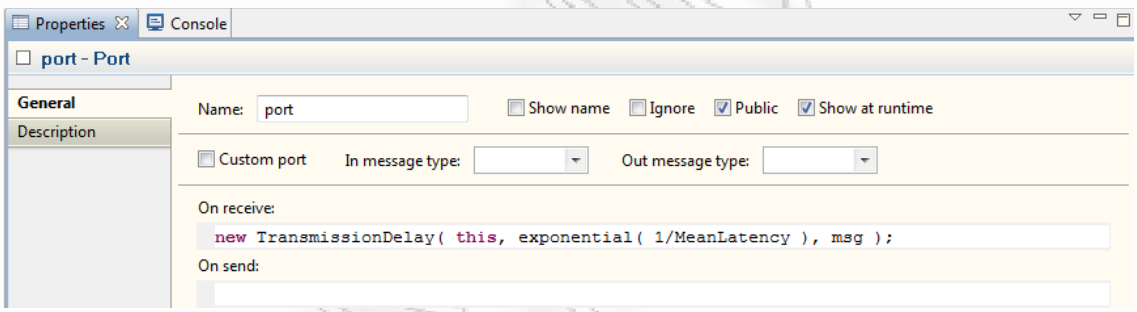
Στις εικόνες 10-25(α) έως και 10-25(δ) εμφανίζονται οι γενικές ιδιότητες των παραμέτρων, της θύρας και του δυναμικού γεγονότος. Αν η τιμή της παραμέτρου LossRate είναι μικρότερη ή ίση από την τρέχουσα τιμή μια ομοιόμορφης κατανομής, τότε η θύρα στέλνει ένα νέο μήνυμα μέσω του εσωτερικού μηχανισμού του Λογισμικού πίσω στο δίκτυο. Στις εικόνες 10-26(α) έως και 10-26(δ) δίνονται οι δυναμικές ιδιότητες των στοιχείων σχημάτων οβάλ που δημιουργούν το σύννεφο στο εικονίδιο του δικτύου.



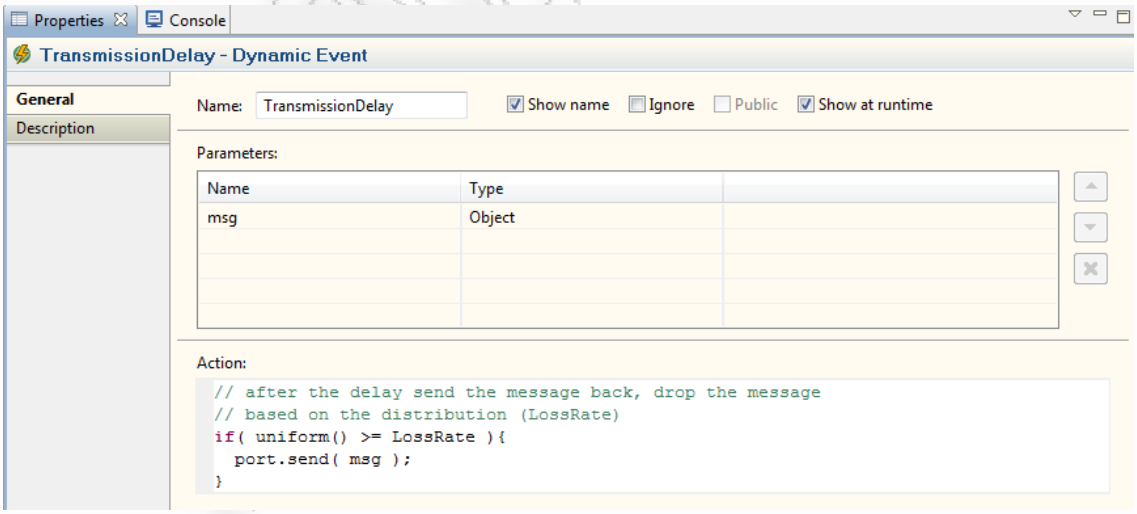
Εικόνα 10-25(α): Γενικές ιδιότητες της παραμέτρου LossRate.



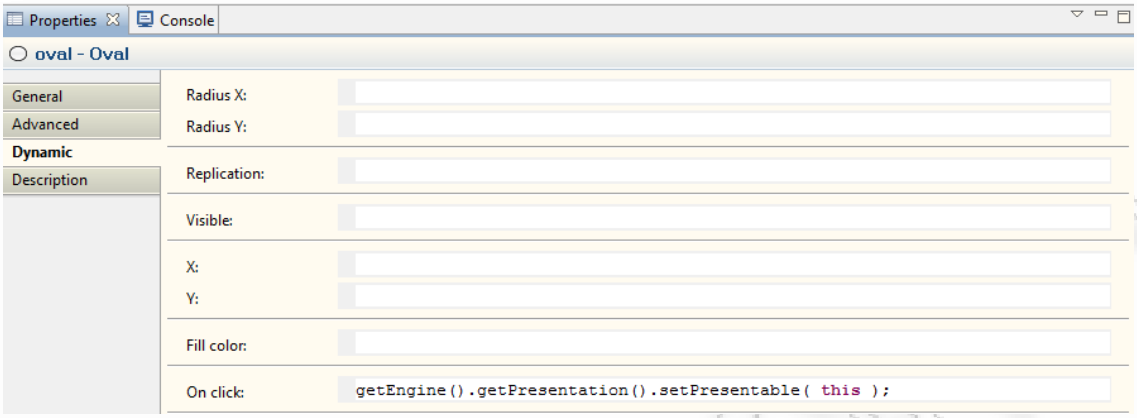
Εικόνα 10-25(β): Γενικές ιδιότητες της παραμέτρου MeanLatency.



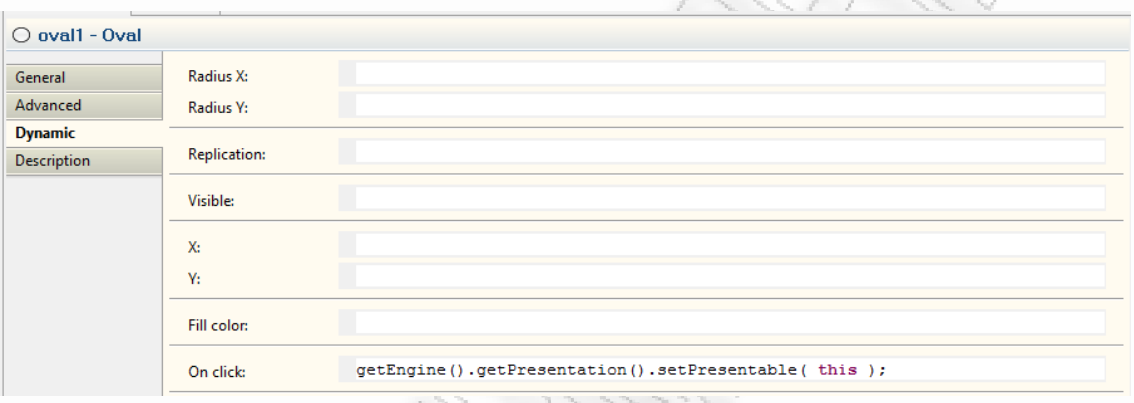
Εικόνα 10-25(γ): Γενικές ιδιότητες της θύρας port.



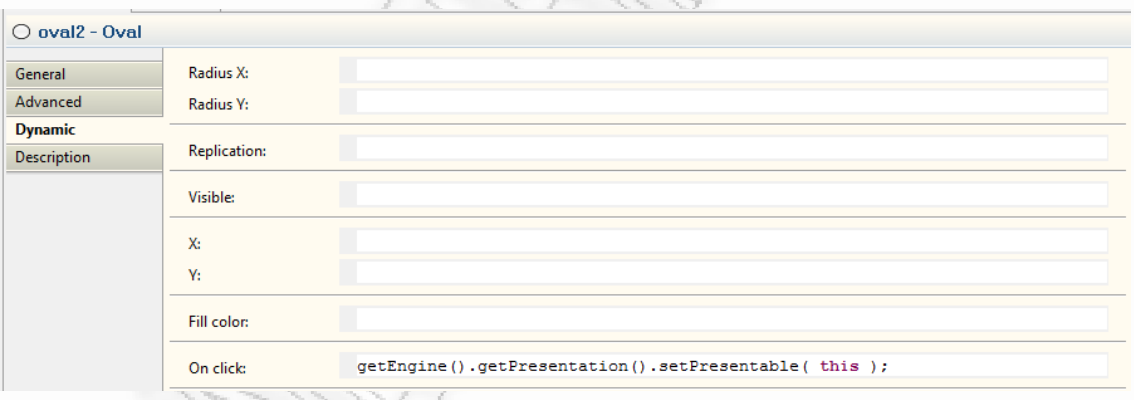
Εικόνα 10-25(δ): Γενικές ιδιότητες του δυναμικού γεγονότος TransmissionDelay.



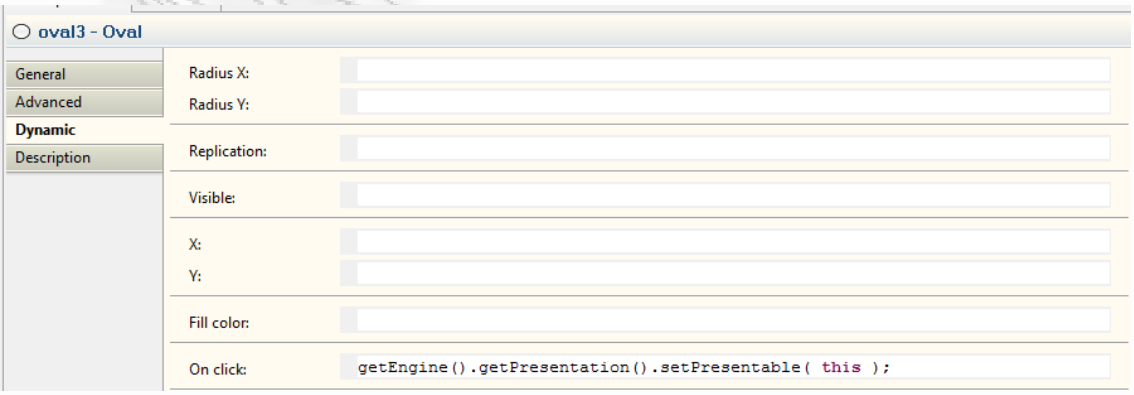
Εικόνα 10-26(α): Δυναμικές ιδιότητες του σχήματος oval τύπου Oval.



Εικόνα 10-26(β): Δυναμικές ιδιότητες του σχήματος oval1 τύπου Oval.



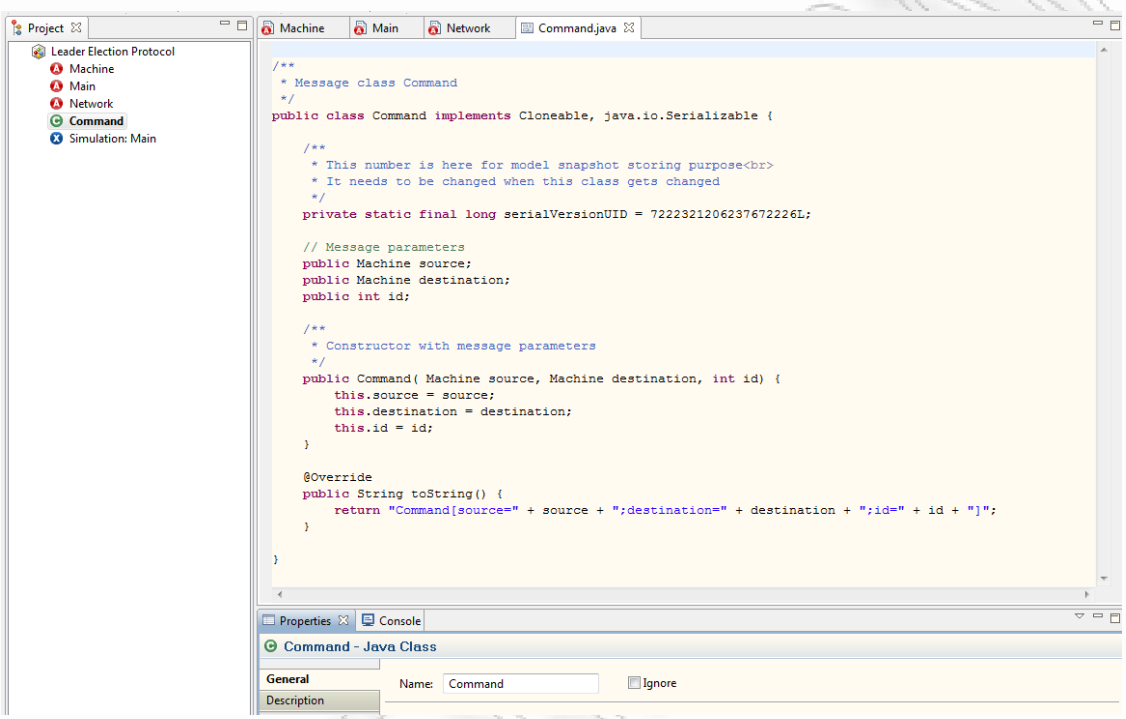
Εικόνα 10-26(γ): Δυναμικές ιδιότητες του σχήματος oval2 τύπου Oval.



Εικόνα 10-26(δ): Δυναμικές ιδιότητες του σχήματος oval3 τύπου Oval.

10.4 Η κλάση Command

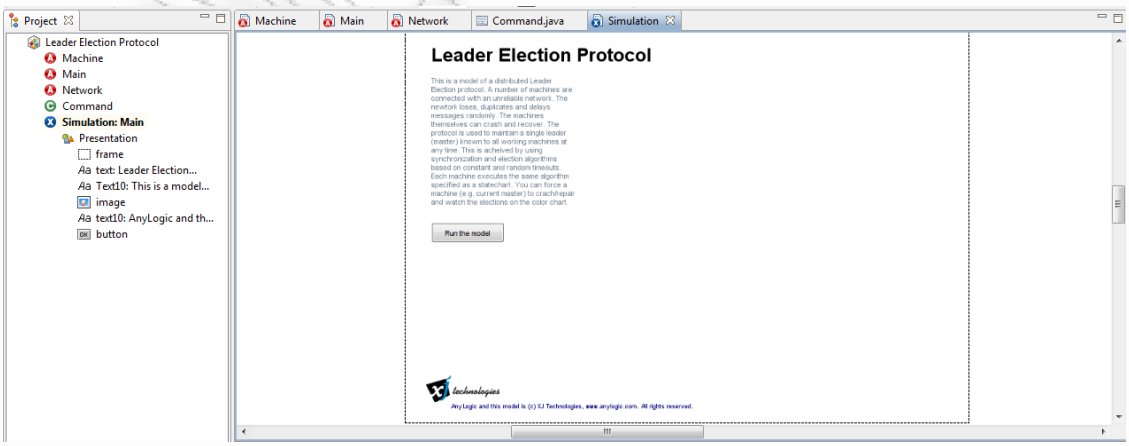
Η κλάση Command, εικόνα 10-27, που είναι μια κλάση Java, ορίζει την εντολή που λαμβάνει ο κάθε υπολογιστής, κάθε instance της κλάσης ενεργού αντικειμένου Machine, για την μεταφορά του αναγνωριστικού,id, μέσω token από ένα υπολογιστή, source, σε άλλον, destination.



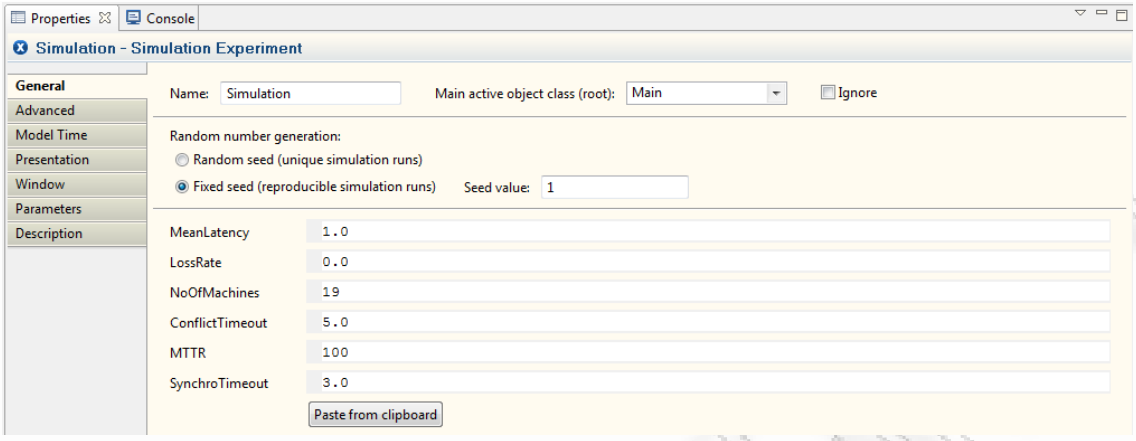
Εικόνα 10-27: Η κλάση Command.java με τον κώδικα της.

10.5 Η κλάση προσομοίωσης του έργου, Simulation:Main

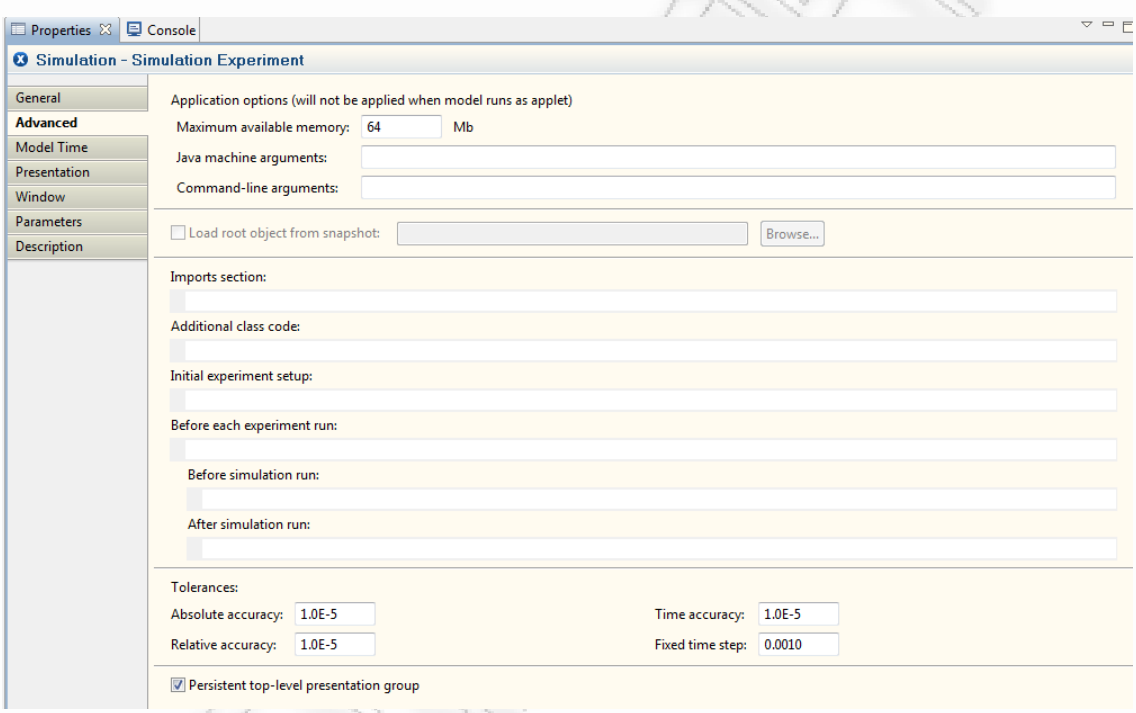
Η κλάση αυτή χρησιμεύει για την προσομοίωση του μοντέλου που υλοποιήθηκε μέσω του έργου Leader Election Protocol. Στις εικόνες 10-28(α) έως και 10-28(στ) παρουσιάζονται οι ιδιότητες κλάσης και οι παράμετροι της προσομοίωσης που λαμβάνονται υπόψη από το Λογισμικό. Στην εικόνα 10-28(ζ) δίνονται οι αρχικές τιμές των παραμέτρων που χρησιμοποιούνται στην προσομοίωση.



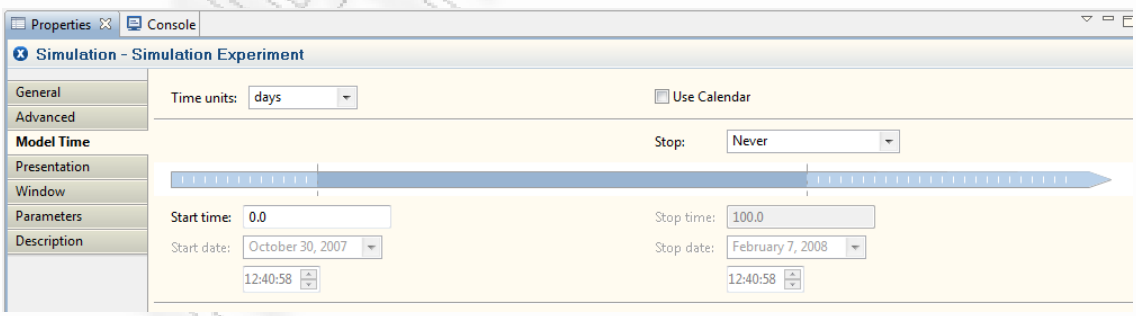
Εικόνα 10-28(α): Η κλάση προσομοίωσης, Simulation:Main.



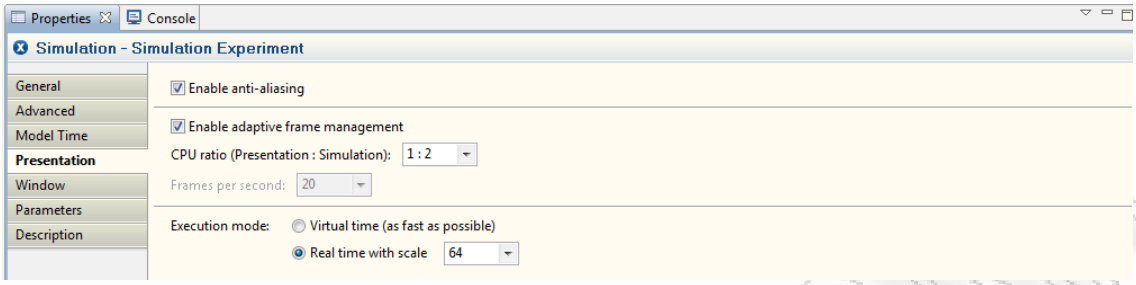
Εικόνα 10-28(β): Γενικές ιδιότητες της κλάσης Simulation:Main.



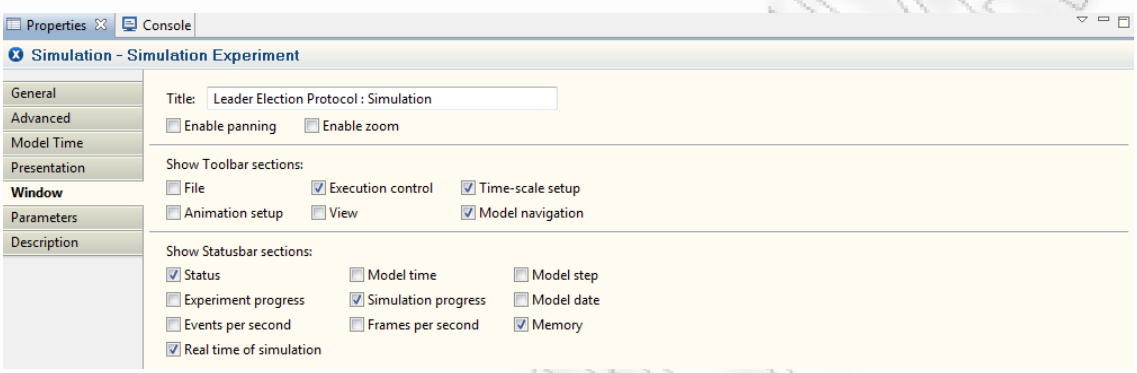
Εικόνα 10-28(γ): Προχωρημένες ιδιότητες της κλάσης Simulation:Main.



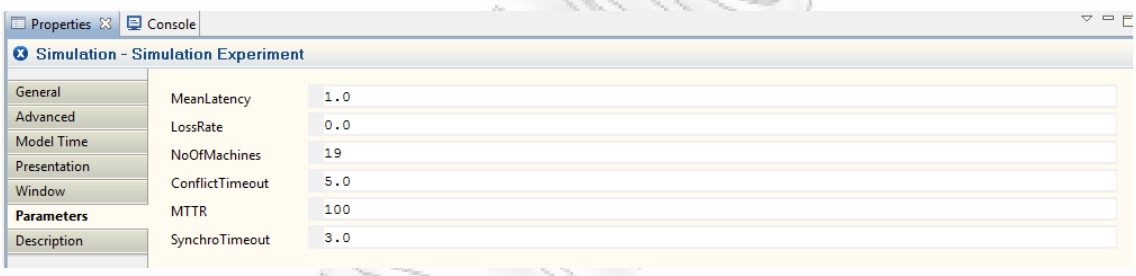
Εικόνα 10-28(δ): Μοντελοποίηση χρόνου της κλάσης Simulation:Main.



Εικόνα 10-28(ε): Ιδιότητες παρουσίασης της κλάσης Simulation:Main.

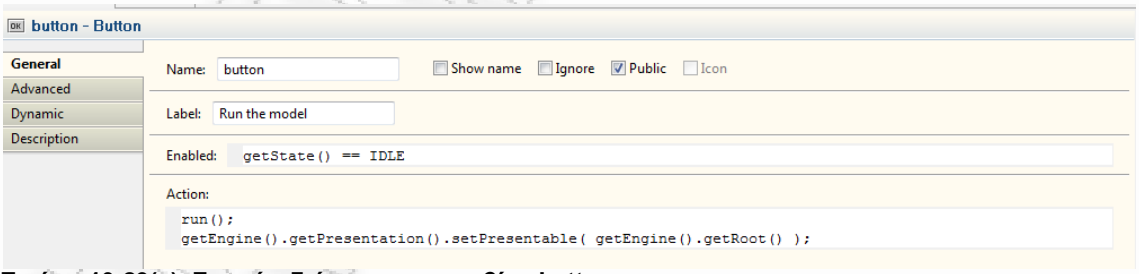


Εικόνα 10-28(στ): Ιδιότητες του παραθύρου της κλάσης Simulation:Main.

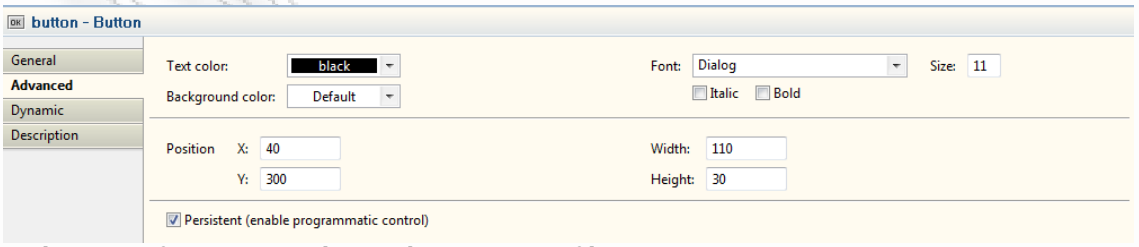


Εικόνα 10-28(ζ): Προεπισκόπιση των παραμέτρων της κλάσης Simulation:Main.

Το πρώτο παράθυρο της προσομοίωσης αποτελείται από ένα ενημερωτικό κείμενο για το μοντέλο, ένα κόμβιο που πατώντας σε αυτό ξεκινά η προσομοίωση και περιέχει το λογότυπο της εταιρείας. Οι ιδιότητες του κομβίου παρουσιάζονται στις εικόνες 10-29(α) και 10-29(β).

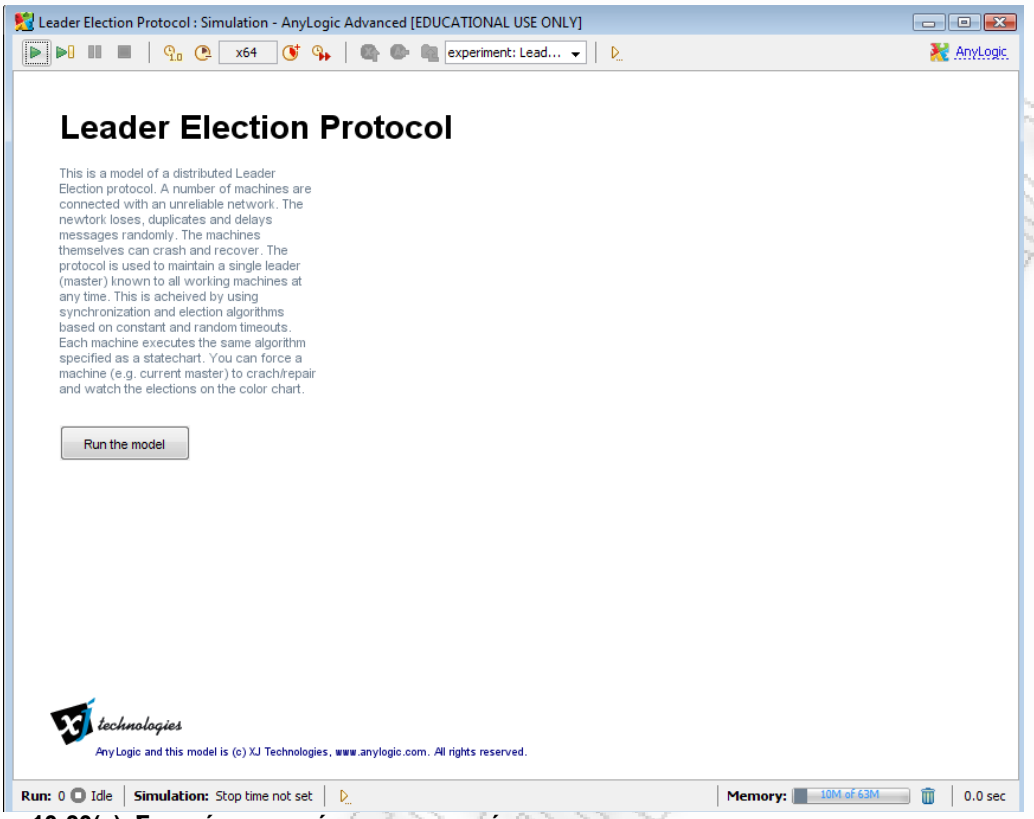


Εικόνα 10-29(α): Γενικές ιδιότητες του κομβίου button.

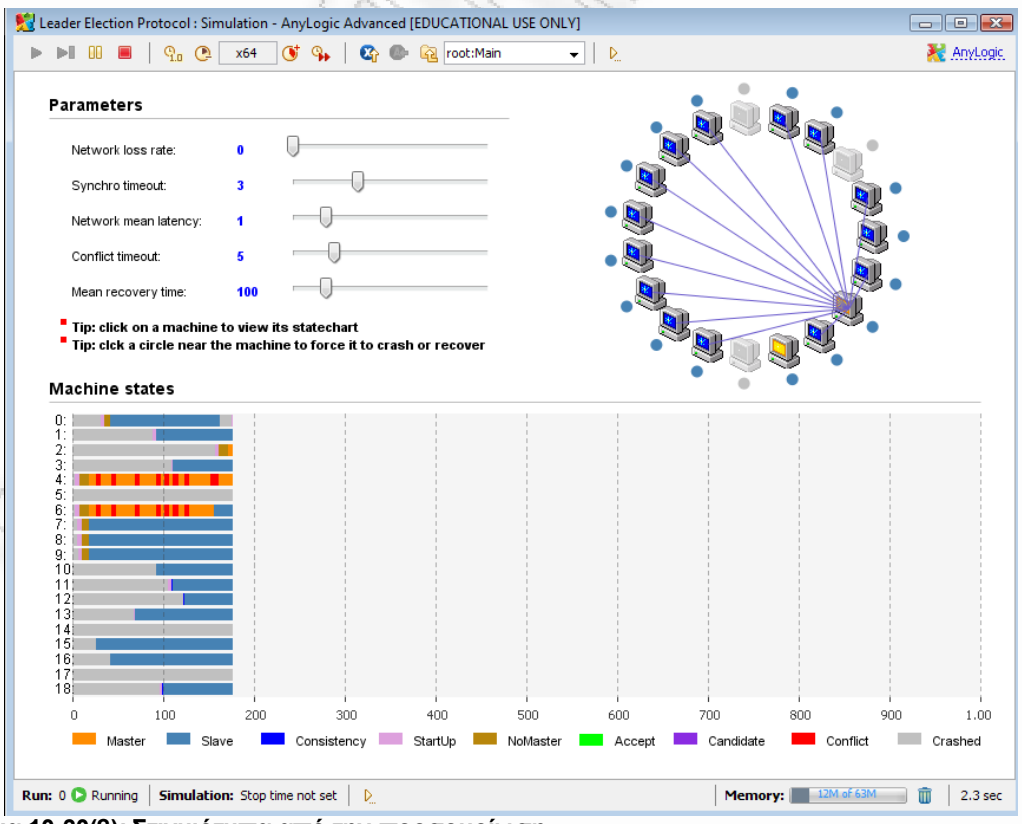


Εικόνα 10-29(β): Προχωρημένες ιδιότητες του κομβίου button.

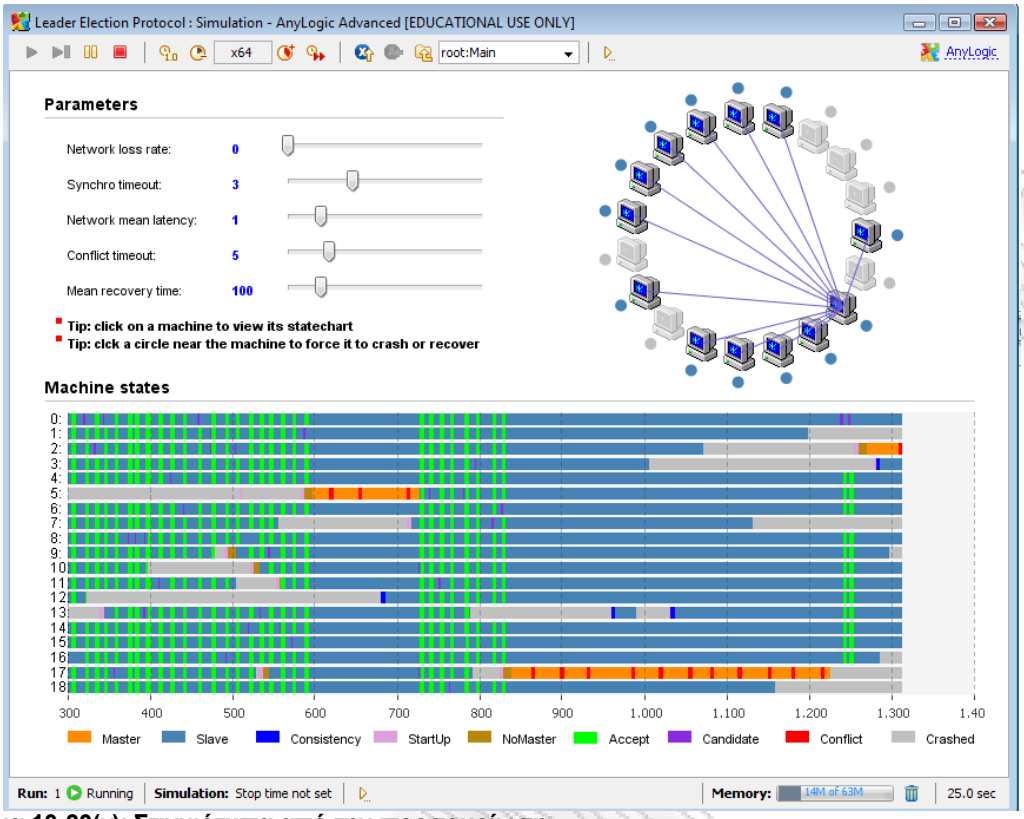
Στη συνέχεια, εικόνες 10-30(α) έως και 10-30(ζ), παρουσιάζονται στιγμιότυπα από την προσομοίωση.



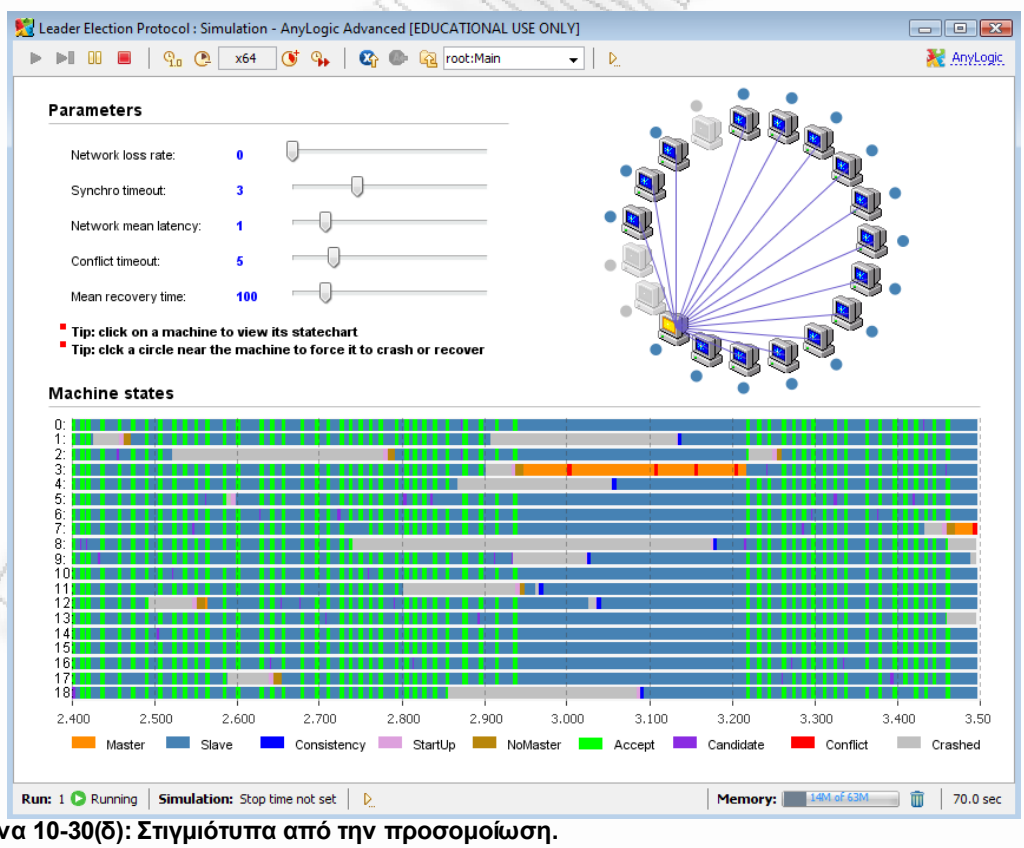
Εικόνα 10-30(α): Στιγμιότυπα από την προσομοίωση.



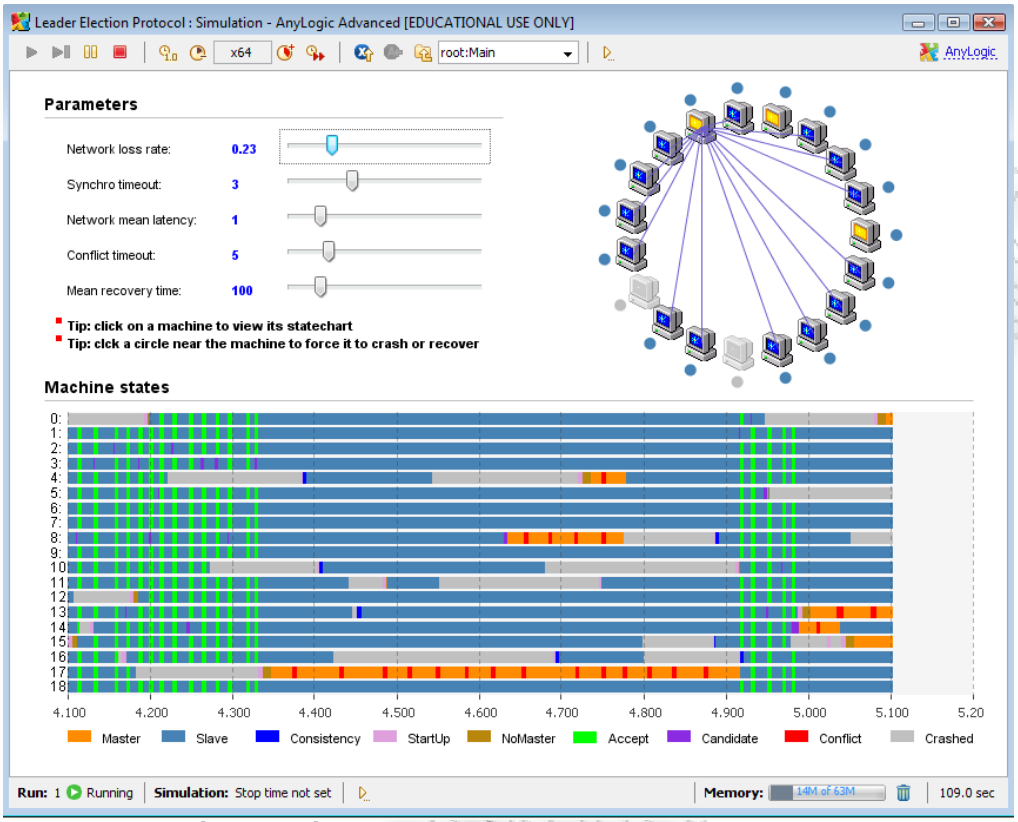
Εικόνα 10-30(β): Στιγμιότυπα από την προσομοίωση.



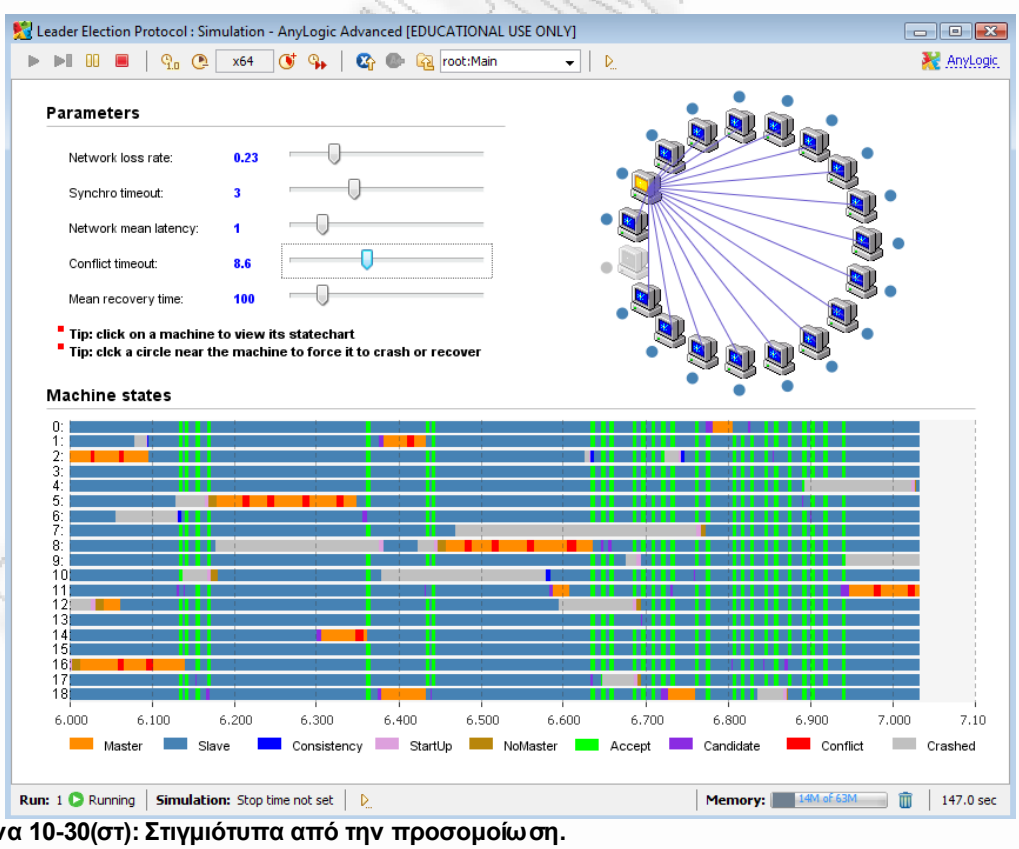
Εικόνα 10-30(γ): Στιγμιότυπα από την προσομοίωση.



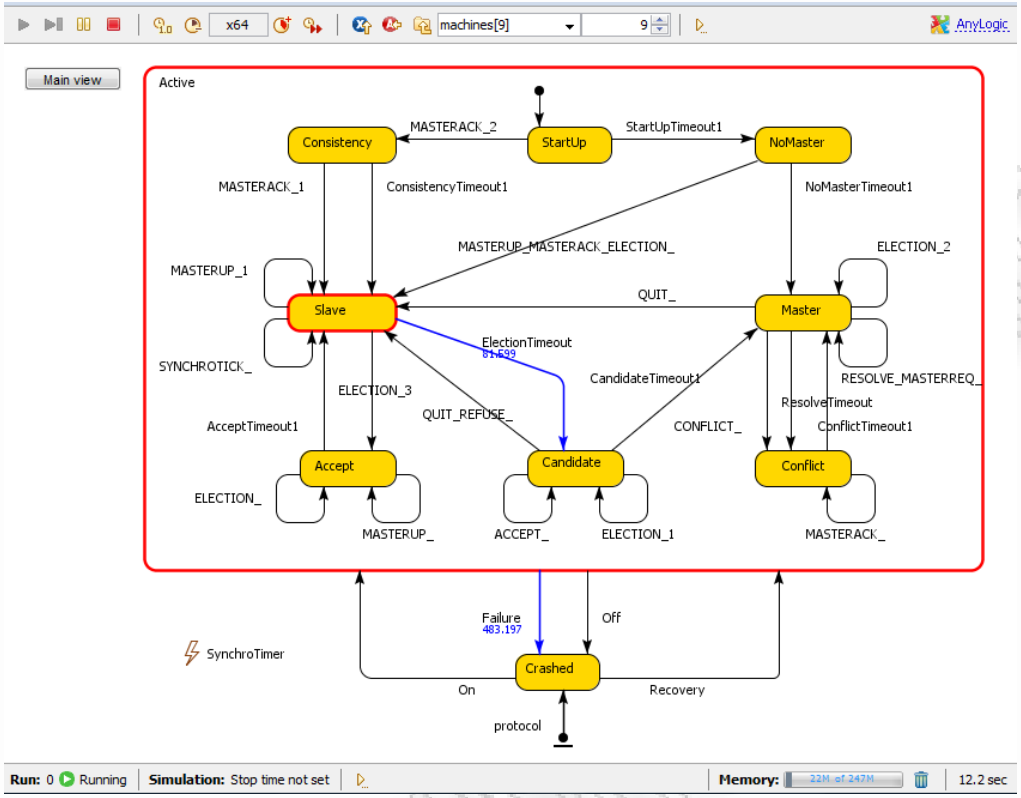
Εικόνα 10-30(δ): Στιγμιότυπα από την προσομοίωση.



Εικόνα 10-30(ε): Στιγμιότυπα από την προσομοίωση.



Εικόνα 10-30(στ): Στιγμιότυπα από την προσομοίωση.



Εικόνα 10-30(ζ): Στιγμιότυπα από την προσομοίωση.

Τα διαγράμματα κλάσεων και ο πλήρης κώδικας του παραδείγματος βρίσκονται στο Παράρτημα 8.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Η παρούσα Μεταπτυχιακή Διατριβή είχε ως βασικό σκοπό την παρουσίαση ορισμένων εφαρμογών από μια πληθώρα παραδειγμάτων μοντέλων Συστημικής Δυναμικής τα οποία έχουν υλοποιηθεί μέσω του λογισμικού AnyLogic της εταιρείας XJTek.

Η παρουσίαση των μοντέλων Συστημικής Δυναμικής περιελάμβανε την έκθεση του θεωρητικού υπόβαθρου του προς εξέταση ζητήματος, την ανάλυση του προβλήματος στα στοιχεία και τις παραμέτρους που το διαμορφώνουν, στη σύνθεση του μοντέλου μέσω του Λογισμικού και στην παρουσίαση των αποτελεσμάτων της προσομοίωσης του μοντέλου στο περιβάλλον ανάπτυξης του Λογισμικού. Η ενδελεχής περιγραφή και η παρουσίαση των στοιχείων που συνθέτουν το περιβάλλον ανάπτυξης του Λογισμικού δεν ήταν ο πρωταρχικός στόχος αυτής της Διατριβής.

Τα μοντέλα, που παρουσιάσαμε, προσομοιώνουν εφαρμογές από διαφορετικές περιοχές επιστημονικού ενδιαφέροντος και συγκεκριμένα από το χώρο της Υγείας, από το χώρο των Οικοσυστημάτων, από το χώρο της Κοινωνιολογίας και της Κοινωνικής Δυναμικής και από το χώρο της Πληροφορικής και των Τηλεπικοινωνιών. Οι λόγοι που επιλέξαμε τα συγκεκριμένα μοντέλα Συστημικής Δυναμικής έχουν σχέση: α) με την απλότητα, ορισμένα από τα μοντέλα που παρουσιάστηκαν ήταν ιδανικά για την εισαγωγή του αρχάριου υποψηφίου χρήστη του Λογισμικού στην φιλοσοφία, στις δυνατότητες και στον τρόπο δημιουργίας των μοντέλων μέσω του περιβάλλοντος ανάπτυξης του AnyLogic, β) με την σταδιακή αύξηση της δυσκολίας των παραδειγμάτων δόθηκαν επιπλέον πληροφορίες στον υποψήφιο χρήστη, γ) παρουσιάστηκαν όλες οι μεθοδολογίες μοντελοποίησης και δ) οι εφαρμογές που παρουσιάστηκαν μοντελοποιούν και προσομοιώνουν σύγχρονα ζητήματα. Το Λογισμικό βέβαια ανταποκρίνεται στις απαιτήσεις πολλών άλλων επιστημονικών πεδίων, πράγμα που ο αναγνώστης μπορεί να επιβεβαιώσει επισκεπτόμενος τον ιστοχώρο της εταιρείας. Οι σχετικές πληροφορίες που βρίσκονται εκεί το αποδεικνύουν.

Τα πλεονεκτήματα από τη χρήση του Λογισμικού είναι τρία κατά τη γνώμη μας και είναι εξίσου σημαντικά. Το πρώτο, αφορά στην βιωσιμότητα του περιβάλλοντος ανάπτυξης των μοντέλων Συστημικής Δυναμικής και των τελικών εφαρμογών, είναι η διαρκής υποστήριξη και η συνεχόμενη εξέλιξη του Λογισμικού από την εταιρεία που το έχει δημιουργήσει. Το δεύτερο πλεονέκτημα, το σημαντικότερο από πλευράς Συστημικής Ανάλυσης, είναι πως AnyLogic παραμένει το μοναδικό μέχρι σήμερα λογισμικό που έχει τη δυνατότητα να εκτελεί υβριδικές προσομοιώσεις, συνδυάζοντας τις τρείς (3) γνωστές κατηγορίες - μεθοδολογίες μοντελοποίησης. Τέλος, το τρίτο πλεονέκτημα του είναι ο γραφικός τρόπος αναπαράστασης των μοντέλων και η χρήση του αντικειμενοστραφούς προγραμματισμού για τη δημιουργία των μοντέλων. Τα παραπάνω χαρακτηριστικά σίγουρα δεν περνούν απαρατήρητα.

Το Λογισμικό προσφέρεται για περαιτέρω ερευνητική εργασία, πολύ ενδιαφέρουσα κατά την γνώμη μας, σε δύο βασικές κατευθύνσεις. Η πρώτη κατεύθυνση αφορά στη καθαρή εφαρμογή της Συστημικής Ανάλυσης σε νέα πεδία εφαρμογής μιας που η δημιουργία νέων μοντέλων προσφέρει στην κατανόηση της πολυπλοκότητας των φαινομένων και των διαδικασιών. Ένας τομέας που θα μπορούσε να βρει εφαρμογή το Λογισμικό είναι η εκπαίδευση. Ερωτήματα σχετικά με τη διαχείριση εκπαιδευτικού προσωπικού και εξοπλισμού όπως επίσης και ζητήματα χρονοπρογραμματισμού εκπαιδευτικών εργασιών σε σχολικές μονάδες θα μπορούσαν να μοντελοποιηθούν μέσω του λογισμικού AnyLogic. Για λόγους πληρότητας, η σύγκριση των εξαγομένων αποτελεσμάτων με τα αποτελέσματα που δίνουν άλλα λογισμικά ή άλλες (μη γραμμικές ή στοχαστικές) διαδικασίες εκτίμησης είναι απαραίτητη. Η δεύτερη ερευνητική κατεύθυνση αφορά στην προσπάθεια για την καλυτέρευση, τη βελτίωση και την ανάπτυξη, του ίδιου του λογισμικού, (α) εστιάζοντας στη δημιουργία νέων επεκτάσεων για την καλύτερη λειτουργία και τη φορητότητα του λογισμικού και (β) τροποποιώντας προς το βέλτιστο τον πυρήνα του λογισμικού, (σε συνεργασία με την εταιρεία XJTek).

Παράρτημα 1

UML – Διαγράμματα κλάσεων

SIR
+ContactRate: double +Infectivity: double +TotalPopulation: double +AverageIllnessDuration: double +InfectionRate: doube +RecoveryRate: double +Susceptible: double +Infectious: double +Recovered: double ~integrationManager _xjal: ActiveObjectIntegrtionManager(3, 0, 2) = new ActiveObjectIntegrtionManager(3, 0, 2) +_chart_autoUpdateEvent_xjal: EventTimeout = new EventTimeout(this) +_chartStack_autoUpdateEvent_xjal: EventTimeout = new EventTimeout(this) +_autoCreateDS_xjal: EventTimeout = new EventTimeout(this) +_ds_InfectionRate: DataSet = new DataSet(100) {@Override public void update() {add(time(), sir.this.InfectionRate); }} +_ds_RecoveryRate: DataSet = new DataSet(100) {@Override public void update() {add(time(), sir.this.RecoveryRate); }} +_ds_Susceptible: DataSet = new DataSet(100) {@Override public void update() {add(time(), sir.this.Susceptible); }} +_ds_Infectious: DataSet = new DataSet(100) {@Override public void update() {add(time(), sir.this.Infectious); }} +_ds_Recovered: DataSet = new DataSet(100) {@Override public void update() {add(time(), sir.this. Recovered); }} +dsSusceptible: DataSet = new DataSet(101) {@Override public void update() {add(time(), _dsSusceptible_YValue(); }} +dsInfectious: DataSet = new DataSet(101) {@Override public void update() {add(time(), _dsInfectious_YValue(); }} +dsRecovered: DataSet = new DataSet(101) {@Override public void update() {add(time(), _dsRecovered_YValue(); }} ~button: int=1 ~rect: int=2 ~roundRect: int=3 ~roundRect1: int=4 ~roundRect2: int=5 ~text: int=6 ~line: int=7 ~text1: int=8 ~line1: int=9 ~_chart: int=10 ~_chartStack: int=11 ~_presentation_content: int[*] = new int[] {button, rect, roundRect, roundRect1, roundRect2, text, line, text1, line1, _chart, _chartStack, } ~presentation: int=0 ~icon: int=-1 ~chart: PieChart ~chartStack: TimeStackChart
<< create>> +Main(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection) +_ContactRate_DefaultValue_xjal(): double +set_ContactRate(ContactRate: double) ~onChange_ContactRate() +_Infectivity_DefaultValue_xjal(): double +set_Infectivity(Infectivity: double) ~onChange_Infectivity() +_TotalPopulation_DefaultValue_xjal(): double +set_ TotalPopulation(TotalPopulation: double) ~onChange_ TotalPopulation() +_AverageIllnessDuration_DefaultValue_xjal(): double +set_ AverageIllnessDuration(AverageIllnessDuration: double) ~onChange_AverageIllnessDuration() +getScalarPhaseVector(_d: double, _a: double) +putScalarPhaseVector(_d: double, _a: double) +assignInitialConditions() +assignInitialConditions(_variableNameCode: int) +_InfectionRate_Formula(): double +_RecoveryRate_Formula(): double +_Infectious_Expression(): double

+_Infectious_Initial_Value(): double
+_Recovered_Expression(): double
+_Susceptible_Expression(): double
+_Susceptible_Initial_Value(): double
+formulasExecute()
+formulasExecute(_variableNameCode: int)
+getScalarRightPart(_dr: double, _ar:double)
+getIntegrationManager(): ActiveObjectIntegrationManager
+getNameOf(_e: EventTimeout): String
+getModeOf(_e: EventTimeout): int
+getFirstOccurrenceTime(_e: EventTimeout): double
+evaluateTimeoutOf(_e: EventTimeout): double
+executeActionOf(_e: EventTimeout)
-_dsSusceptible_YValue(): double
-_dsInfectious_YValue(): double
-_dsRecovered_YValue(): double
+executeShapeControlAction(_shape: int, index: int)
+getShapeGroupContent(_shape: int): int
-_chart_DataItem0Value(): double
-_chart_DataItem1Value(): double
-_chart_DataItem2Value(): double
+getNameOfShape(_shape: int): String
+getShapeType(_shape: int): int
+getShapeX(_shape: int, index: int): double
+getShapeY(_shape: int, index: int): double
+getShapeLineColor(_shape: int, index: int): Color
+getShapeFillColor(_shape: int, index: int): Color
+getShapeLineStyle(_shape: int, index: int): int
+getShapeLineWidth(_shape: int, index: int): double
+getShapeLineDx(_shape: int, index: int): double
+getShapeLineDy(_shape: int, index: int): double
+getShapeWidth(_shape: int, index: int): double
+getShapeHeigh(_shape: int, index: int): double
+getShapeRadiousX(_shape: int, index: int): double
+getShapeRadiousY(_shape: int, index: int): double
+getShapeText(_shape: int, index: int): Object
+getShapeFont(_shape: int, index: int): Font
+getShapeTextAlignement(_shape: int, index: int): int
+getPersistentShape(_shape: int): Object
+drawModeElements(_panel: Panel, _g: Graphics2D, publicOnly: boolean)
+onClickModelAt(panel: Panel, x: double, y: double, clickCount: int, publicOnly: Boolean): Boolean
+create()
+start()
+onDestroy()

Κώδικας Κεφαλαίου 3 – Έργο SIR

Main.java

```
package sir;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import java.awt.geom.Arc2D;

public class Main extends ActiveObject
{
 // Parameters

 public double ContactRate;

 /**
  * Returns default value for parameter <code>ContactRate</code>.
  * <i>This method should not be called by user</i>
  */
 public double _ContactRate_DefaultValue_xjal() {
  return 5 ;
 }

 public void set_ContactRate(
double ContactRate ) {
  if (ContactRate == this.ContactRate) {
   return;
  }
  this.ContactRate = ContactRate;
  onChange_ContactRate();
  onChange();
 }

 void onChange_ContactRate() {
 }

 public double Infectivity;

 /**
  * Returns default value for parameter <code>Infectivity</code>.
  * <i>This method should not be called by user</i>
  */
 public double _Infectivity_DefaultValue_xjal() {
  return 0.05 ;
 }

 public void set_Infectivity(
double Infectivity ) {
  if (Infectivity == this.Infectivity) {
   return;
  }
  this.Infectivity = Infectivity;
  onChange_Infectivity();
  onChange();
 }

 void onChange_Infectivity() {
 }

 public double TotalPopulation;

 /**
  * Returns default value for parameter
<code>TotalPopulation</code>.
  * <i>This method should not be called by user</i>
  */
 public double _TotalPopulation_DefaultValue_xjal() {
  return 1000 ;
 }

 public void set_TotalPopulation(
double TotalPopulation ) {
  if (TotalPopulation == this.TotalPopulation) {
   return;
  }
  this.TotalPopulation = TotalPopulation;
  onChange_TotalPopulation();
```

```
    onChange();
  }

  void onChange_TotalPopulation() {
  }


  public
double AverageIllnessDuration;

  /**
   *       Returns       default       value       for       parameter
<code>AverageIllnessDuration</code>.
   * <i>This method should not be called by user</i>
   */
  public
double _AverageIllnessDuration_DefaultValue_xjal() {
    return
15
;
  }

  public void set_AverageIllnessDuration(
double AverageIllnessDuration ) {
    if (AverageIllnessDuration == this.AverageIllnessDuration) {
      return;
    }
    this.AverageIllnessDuration = AverageIllnessDuration;
    onChange_AverageIllnessDuration();
    onChange();
  }

  void onChange_AverageIllnessDuration() {
  }

  // Plain Variables


  // Collection Variables

  // Flow/Auxiliary Variables
  public double InfectionRate;
  public double RecoveryRate;

  // Stock Variables
  public double Susceptible;
  public double Infectious;
  public double Recovered;

  /**
   * Writes model variables into given arrays
   */
  public void getScalarPhaseVector(double[] _d, double[] _a){
    _d[0] = Susceptible;
    _d[1] = Infectious;
    _d[2] = Recovered;
  }

  /**
   * Writes given arrays to model variables
   */
  public void putScalarPhaseVector(double[] _d, double[] _a){
    Susceptible = _d[0];
    Infectious = _d[1];
    Recovered = _d[2];
  }

  public void assignInitialConditions(){
    Susceptible = _Susceptible_Initial_Value();
    Infectious = _Infectious_Initial_Value();
    InfectionRate = _InfectionRate_Formula();
    RecoveryRate = _RecoveryRate_Formula();
  }

  public void assignInitialConditions( int _variableNameCode ) {
    switch (_variableNameCode) {
      case 1460724841:
        Susceptible = _Susceptible_Initial_Value();
        break;
      case 1349127161:
        Infectious = _Infectious_Initial_Value();
        break;
      case -464053453:
        InfectionRate = _InfectionRate_Formula();
        break;
      case -2127773131:
        RecoveryRate = _RecoveryRate_Formula();
        break;
      case -1293709085:
        break;
      default:
        break;
    }
  }

  public double _Infectious_Expression() {
    return InfectionRate - RecoveryRate ;
  }

  public double _RecoveryRate_Formula() {
    return Infectious / AverageIllnessDuration ;
  }

  public double _Infectious_Initial_Value() {
    return 1 ;
  }

  public double _InfectionRate_Formula() {
    return
Infectious * ContactRate * Susceptible / TotalPopulation * Infectivity
;
  }

  public double _Susceptible_Expression() {
    return
-InfectionRate
;
  }

  public double _Susceptible_Initial_Value() {
    return
TotalPopulation - 1
;
  }

  public double _Recovered_Expression() {
    return
RecoveryRate
;
  }
```

```
public void formulasExecute(){
  InfectionRate = _InfectionRate_Formula();
  RecoveryRate = _RecoveryRate_Formula();
}

public void formulasExecute( int _variableNameCode ) {
  switch (_variableNameCode) {
    case -464053453:
      InfectionRate = _InfectionRate_Formula();
      break;
    case -2127773131:
      RecoveryRate = _RecoveryRate_Formula();
      break;
    default:
      break;
  }
}

public void getScalarRightPart( double[] _dr, double[] _ar ) {
  _dr[ 0 ] = _Susceptible_Expression();
  _dr[ 1 ] = _Infectious_Expression();
  _dr[ 2 ] = _Recovered_Expression();
}

static ActiveObjectIntegrationManager integrationManager_xjal = new
ActiveObjectIntegrationManager( 3, 0, 2 );

public ActiveObjectIntegrationManager getIntegrationManager(){
  return integrationManager_xjal;
}

// Events

public EventTimeout _InfectiousDS_autoUpdateEvent_xjal = new
EventTimeout(this);
public EventTimeout _chart_autoUpdateEvent_xjal = new
EventTimeout(this);
public EventTimeout _chartStack_autoUpdateEvent_xjal = new
EventTimeout(this);
public EventTimeout _autoCreatedDS_xjal = new EventTimeout(this);


@Override
public String getNameOf( EventTimeout _e ) {
  if ( _e == _InfectiousDS_autoUpdateEvent_xjal ) return "InfectiousDS
auto update event";
  if ( _e == _chart_autoUpdateEvent_xjal ) return "chart auto update
event";
  if ( _e == _chartStack_autoUpdateEvent_xjal ) return "chartStack
auto update event";
  if ( _e == _autoCreatedDS_xjal ) return "Auto-created DataSets auto
update event";
   return super.getNameOf( _e );
}

@Override
public int getModeOf( EventTimeout _e ) {
  if ( _e == _InfectiousDS_autoUpdateEvent_xjal ) return
EVENT_TIMEOUT_MODE_CYCLIC;
  if ( _e == _chart_autoUpdateEvent_xjal ) return
EVENT_TIMEOUT_MODE_CYCLIC;
  if ( _e == _chartStack_autoUpdateEvent_xjal ) return
EVENT_TIMEOUT_MODE_CYCLIC;
  if ( _e == _autoCreatedDS_xjal ) return
EVENT_TIMEOUT_MODE_CYCLIC;
  return super.getModeOf( _e );
}

@Override
public double getFirstOccurrenceTime( EventTimeout _e ) {
  if (
     _e == _InfectiousDS_autoUpdateEvent_xjal
     || _e == _chart_autoUpdateEvent_xjal
     || _e == _chartStack_autoUpdateEvent_xjal
     || _e == _autoCreatedDS_xjal
   ) return getEngine().getStartTime();
  return super.getFirstOccurrenceTime( _e );
}

@Override
public double evaluateTimeoutOf( EventTimeout _e ) {
  if ( _e == _InfectiousDS_autoUpdateEvent_xjal ) return
1
;
  if ( _e == _chart_autoUpdateEvent_xjal ) return
1
;
  if ( _e == _chartStack_autoUpdateEvent_xjal ) return
1
;
  if ( _e == _autoCreatedDS_xjal ) return
1
;
  return super.evaluateTimeoutOf( _e );
}

@Override
public void executeActionOf( EventTimeout _e ) {
  if ( _e == _InfectiousDS_autoUpdateEvent_xjal ) {
    InfectiousDS.update();
    return;
  }
  if ( _e == _chart_autoUpdateEvent_xjal ) {
    chart.updateData();
    return;
  }
  if ( _e == _chartStack_autoUpdateEvent_xjal ) {
    chartStack.updateData();
    return;
  }
  if ( _e == _autoCreatedDS_xjal ) {
    _ds_InfectionRate.update();
    _ds_RecoveryRate.update();
    _ds_Susceptible.update();
    _ds_Infectious.update();
    _ds_Recovered.update();
  }
  super.executeActionOf( _e );
}
// Analysis Data Elements
/**
 * Auto-created data set(s) for InfectionRate
 */
public DataSet _ds_InfectionRate = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
    if ( time() == _lastUpdateTime ) { return; }
    add( time(), Main.this.InfectionRate );
    _lastUpdateTime = time();
  }
};
/**
```

```
  * Auto-created data set(s) for RecoveryRate
  */
 public DataSet _ds_RecoveryRate = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), Main.this.RecoveryRate );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Susceptible
  */
 public DataSet _ds_Susceptible = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), Main.this.Susceptible );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Infectious
  */
 public DataSet _ds_Infectious = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), Main.this.Infectious );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Recovered
  */
 public DataSet _ds_Recovered = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), Main.this.Recovered );
   _lastUpdateTime = time();
  }
 };
 public DataSet _chartStack_expression0_dataSet_xjal = new DataSet(
101 ) {
  double _lastUpdateX = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateX ) { return; }
   double _a =
Susceptible
;
   add( time(), _a );
   _lastUpdateX = time();
  }
 };
 public DataSet _chartStack_expression1_dataSet_xjal = new DataSet(
101 ) {
  double _lastUpdateX = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateX ) { return; }
   double _a =
Infectious
;
   add( time(), _a );
   _lastUpdateX = time();
  }
 };
 public DataSet _chartStack_expression2_dataSet_xjal = new DataSet(
101 ) {
  double _lastUpdateX = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateX ) { return; }
   double _a =
Recovered
;
   add( time(), _a );
   _lastUpdateX = time();
  }
 };
 public DataSet InfectiousDS = new DataSet( 101 ) {
  double _lastUpdateX = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateX ) { return; }
   add( time(), _InfectiousDS_YValue() );
   _lastUpdateX = time();
  }
 };

 /**
  * <i>This method should not be called by user</i>
  */
 private double _InfectiousDS_YValue() {
  return
Infectious
;
  }

 static final Font _button_Font = new Font("Dialog", 0, 11 );
 static final Font _text_Font = new Font("SansSerif", 1, 12 );
 static final Font _text1_Font = _text_Font;
 static final Color _roundRect_FillColor = new Color( 0xFFDDE2E9, true
);
 static final Color _roundRect1_FillColor = new Color( 0xFFDDE2E9, true
);
 static final Color _roundRect2_FillColor = new Color( 0xFFDDE2E9, true
);
 static final Color _roundRect3_FillColor = new Color( 0xFFDDE2E9, true
);
 static final int button = 1;
 static final int rect = 2;
 static final int roundRect = 3;
 static final int roundRect1 = 4;
 static final int roundRect2 = 5;
 static final int text = 6;
 static final int line = 7;
 static final int text1 = 8;
 static final int line1 = 9;
 static final int roundRect3 = 10;
 static final int _chart = 11;
 static final int _chartStack = 12;


 /**
  * Top-level presentation group content
  */
```

```
 static final int[] _presentation_content = new int[] {
  button,
  rect,
  roundRect,
  roundRect1,
  roundRect2,
  text,
  line,
  text1,
  line1,
  roundRect3,
  _chart,
  _chartStack,
 };

 /**
  * Top-level presentation group id
  */
 static final int presentation = 0;

 /**
  * Top-level icon group id
  */
 static final int icon = -1;


 @Override
 public void executeShapeControlAction( int _shape, int index ) {
  switch( _shape ) {
   case button: {
TimePlot plot = ((Simulation)getEngine().getExperiment()).plot;
plot.addDataSet( InfectiousDS,
            "Infectious when CR = " + ContactRate + " Inf = " +
Infectivity + " AID = " + AverageIllnessDuration,
            spectrumColor( plot.getCount(), 5 ), true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE );
getEngine().getExperiment().stop();
;}
    break;
   default:
    super.executeShapeControlAction( _shape, index );
    break;
  }
 }

 @Override
 public int[] getShapeGroupContent( int _shape ){
   switch( _shape ) {
    case presentation: return _presentation_content;
    default: return super.getShapeGroupContent( _shape );
   }
 }


 /**
  * <i>This method should not be called by user</i>
  */
 private double _chart_DataItem0Value() {
  return
Susceptible
;
 }

 /**
  * <i>This method should not be called by user</i>
  */
 private double _chart_DataItem1Value() {
  return
Infectious
;
 }

 /**
  * <i>This method should not be called by user</i>
  */
 private double _chart_DataItem2Value() {
  return
Recovered
;
 }

 @Override
 public String getNameOfShape( int _shape ) {
  switch( _shape ) {
   case rect: return "rect";
   case roundRect: return "roundRect";
   case roundRect1: return "roundRect1";
   case roundRect2: return "roundRect2";
   case text: return "text";
   case line: return "line";
   case text1: return "text1";
   case line1: return "line1";
   case roundRect3: return "roundRect3";
   case button: return "button";
   default: return super.getNameOfShape( _shape );
  }
 }

 @Override
 public int getShapeType( int _shape ) {
  switch( _shape ) {
   case presentation: return SHAPE_GROUP;
   case icon: return SHAPE_GROUP;
   case rect: return SHAPE_RECTANGLE;
   case roundRect: return SHAPE_ROUNDED_RECTANGLE;
   case roundRect1: return SHAPE_ROUNDED_RECTANGLE;
   case roundRect2: return SHAPE_ROUNDED_RECTANGLE;
   case text: return SHAPE_TEXT;
   case line: return SHAPE_LINE;
   case text1: return SHAPE_TEXT;
   case line1: return SHAPE_LINE;
   case roundRect3: return SHAPE_ROUNDED_RECTANGLE;
   case button: return SHAPE_BUTTON;
   default: return super.getShapeType( _shape );
  }
 }

 @Override
 public double getShapeX( int _shape, int index ) {
  switch( _shape ) {
   case rect: return 0;
   case roundRect: return 120;
   case roundRect1: return 90;
   case roundRect2: return 490;
   case text: return 50;
   case line: return 400;
   case text1: return 50;
   case line1: return 360;
   case roundRect3: return 350;
   case button: return 580;
   default: return super.getShapeX( _shape, index );
```

```
  }
 }

 @Override
 public double getShapeY( int _shape, int index ) {
  switch( _shape ) {
   case rect: return 0;
   case roundRect: return 310;
   case roundRect1: return 415;
   case roundRect2: return 420;
   case text: return 50;
   case line: return 60;
   case text1: return 280;
   case line1: return 290;
   case roundRect3: return 480;
   case button: return 240;
   default: return super.getShapeY( _shape, index );
  }
 }

 @Override
 public Color getShapeLineColor( int _shape, int index ) {
  switch( _shape ) {
   // shapes
   case rect: return null;
   case roundRect: return null;
   case roundRect1: return null;
   case roundRect2: return null;
   case line: return gray;
   case line1: return gray;
   case roundRect3: return null;
   // controls (text color)
   case button: return black;
   // charts (border color)
   default: return super.getShapeLineColor( _shape, index );
  }
 }

 @Override
 public Color getShapeFillColor( int _shape, int index ) {
  switch( _shape ) {
   case rect: return lightSteelBlue;
   case roundRect: return _roundRect_FillColor;
   case roundRect1: return _roundRect1_FillColor;
   case roundRect2: return _roundRect2_FillColor;
   case text: return gray;
   case text1: return gray;
   case roundRect3: return _roundRect3_FillColor;
   case button: return controlDefault;
   default: return super.getShapeFillColor( _shape, index );
  }
 }

 @Override
 public int getShapeLineStyle( int _shape, int index ) {
  switch( _shape ) {
   case rect: return LINE_STYLE_SOLID;
   case roundRect: return LINE_STYLE_SOLID;
   case roundRect1: return LINE_STYLE_SOLID;
   case roundRect2: return LINE_STYLE_SOLID;
   case line: return LINE_STYLE_SOLID;
   case line1: return LINE_STYLE_SOLID;
   case roundRect3: return LINE_STYLE_SOLID;
   default: return super.getShapeLineStyle( _shape, index );
  }
 }

 @Override
 public double getShapeLineWidth( int _shape, int index ) {
  switch( _shape ) {
   case line: return 0;
   case line1: return 0;
   default: return super.getShapeLineWidth( _shape, index );
  }
 }

 @Override
 public double getShapeLineDx( int _shape, int index ) {
  switch( _shape ) {
   case line: return 330;
   case line1: return 370;
   default: return super.getShapeLineDx( _shape, index );
  }
 }

 @Override
 public double getShapeLineDy( int _shape, int index ) {
  switch( _shape ) {
   case line: return 0;
   case line1: return 0;
   default: return super.getShapeLineDy( _shape, index );
  }
 }

 @Override
 public double getShapeWidth( int _shape, int index ) {
  switch( _shape ) {
   case rect: return 800;
   case roundRect: return 490;
   case roundRect1: return 160;
   case roundRect2: return 180;
   case roundRect3: return 180;
   case button: return 150;
   default: return super.getShapeWidth( _shape, index );
  }
 }

 @Override
 public double getShapeHeight( int _shape, int index ) {
  switch( _shape ) {
   case rect: return 600;
   case roundRect: return 80;
   case roundRect1: return 150;
   case roundRect2: return 40;
   case roundRect3: return 40;
   case button: return 20;
   default: return super.getShapeHeight( _shape, index );
  }
 }

 @Override
 public double getShapeRadiusX( int _shape, int index ) {
  switch( _shape ) {
   case roundRect: return 10;
   case roundRect1: return 10;
   case roundRect2: return 10;
   case roundRect3: return 10;
   default: return super.getShapeRadiusX( _shape, index );
  }
 }
```

```
 @Override
 public double getShapeRadiusY( int _shape, int index ) {
  switch( _shape ) {
   case roundRect: return 10;
   case roundRect1: return 10;
   case roundRect2: return 10;
   case roundRect3: return 10;
   default: return super.getShapeRadiusY( _shape, index );
  }
 }
```

```
 @Override
 public Object getShapeText( int _shape, int index ) {
  switch( _shape ) {
   case button: return "Save results and Finish";
   case text: return "Percent of Susceptible, Infectious and Recovered
population";
   case text1: return "Stock & Flow Diagram of the System Dynamics
Model";
   default: return super.getShapeText( _shape, index );
  }
 }
```

```
 @Override
 public Font getShapeFont( int _shape, int index ) {
  switch( _shape ) {
   case text: return _text_Font;
   case text1: return _text1_Font;
   case button: return _button_Font;
   default: return super.getShapeFont( _shape, index );
  }
 }

 @Override
 public int getShapeTextAlignment( int _shape, int index ) {
  switch( _shape ) {
   case text: return ALIGNMENT_LEFT;
   case text1: return ALIGNMENT_LEFT;
   default: return super.getShapeTextAlignment( _shape, index );
  }
 }
```

```
 PieChart chart;
 TimeStackChart chartStack;

 @Override
 public Object getPersistentShape( int _shape ) {
  switch(_shape){

   case _chart: return chart;
   case _chartStack: return chartStack;
   default: return null;
  }
 }
```

```
 static final Arc2D.Double arc_PD_1153899673370 = new
Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1153899673370, 170,360,
270, 360, 22.11849430599321 );}
 static final Arc2D.Double arc_PD_1153899673371 = new
Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1153899673371, 370,360,
270, 360, 23.4945613519697 );}
 static final Arc2D.Double arc_PD_1153899770400 = new
Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1153899770400, 370,360,
460, 360, 25.237557774322468 );}

 @Override
 public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
  if (!_publicOnly) {
   drawPlainDependencyArc( _panel, _g, arc_PD_1153899673370, null
);
  }
  if (!_publicOnly) {
   drawPlainDependencyArc( _panel, _g, arc_PD_1153899673371, null
);
  }
  if (!_publicOnly) {
   drawPlainDependencyLine( _panel, _g, 158, 434, 253, 371, null );
  }
  if (!_publicOnly) {
   drawPlainDependencyLine( _panel, _g, 223, 510, 264, 379, null );
  }
  if (!_publicOnly) {
   drawPlainDependencyLine( _panel, _g, 176, 482, 258, 376, null );
  }
  if (!_publicOnly) {
   drawPlainDependencyArc( _panel, _g, arc_PD_1153899770400, null
);
  }
  if (!_publicOnly) {
   drawPlainDependencyLine( _panel, _g, 505, 432, 471, 377, null );
  }
  if (!_publicOnly) {
   drawFlowStockDependencyLine( _panel, _g, 185, 360, 252, 360, null
);
  }
  if (!_publicOnly) {
   drawFlowStockDependencyLine( _panel, _g, 285, 360, 352, 360, null
);
  }
  if (!_publicOnly) {
   drawFlowStockDependencyLine( _panel, _g, 385, 360, 442, 360, null
);
  }
  if (!_publicOnly) {
   drawFlowStockDependencyLine( _panel, _g, 475, 360, 532, 360, null
);
  }
  if (!_publicOnly) {
   drawParameter( _panel, _g, 150, 440, -50, 15, "ContactRate",
ContactRate, false, false );
  }
  if (!_publicOnly) {
   drawParameter( _panel, _g, 170, 490, -45, 15, "Infectivity",
Infectivity, false, false );
  }
  if (!_publicOnly) {
   drawParameter( _panel, _g, 220, 520, -70, 15, "TotalPopulation",
TotalPopulation, false, false );
  }
  if (!_publicOnly) {
   drawParameter( _panel, _g, 510, 440, 10, 0,
"AverageIllnessDuration", AverageIllnessDuration, false, false );
  }
  if (!_publicOnly) {
   drawStock( _panel, _g, 170, 360, -30, -30, "Susceptible",
Susceptible, false );
  }
```

```
    if (!_publicOnly) {
      drawStock( _panel, _g, 370, 360, -30, -30, "Infectious", Infectious, false );
    }
    if (!_publicOnly) {
      drawStock( _panel, _g, 550, 360, -20, -30, "Recovered", Recovered, false );
    }
    if (!_publicOnly) {
      drawFlow( _panel, _g, 270, 360, -30, -30, "InfectionRate", InfectionRate, false, false );
    }
    if (!_publicOnly) {
      drawFlow( _panel, _g, 460, 360, -30, -30, "RecoveryRate", RecoveryRate, false, false );
    }
    if (!_publicOnly) {
      drawDataset( _panel, _g, 370, 500, 10, 0, "InfectiousDS", InfectiousDS );
    }
  }

  @Override
  public boolean onClickModelAt( Panel panel, double x, double y, int clickCount, boolean publicOnly ) {
    if( !publicOnly && modelElementContains(x, y, 150, 440) ) {
      panel.addInspect( 150, 440, this, "ContactRate" );
      return true;
    }
    if( !publicOnly && modelElementContains(x, y, 170, 490) ) {
      panel.addInspect( 170, 490, this, "Infectivity" );
      return true;
    }
    if( !publicOnly && modelElementContains(x, y, 220, 520) ) {
      panel.addInspect( 220, 520, this, "TotalPopulation" );
      return true;
    }
    if( !publicOnly && modelElementContains(x, y, 510, 440) ) {
      panel.addInspect( 510, 440, this, "AverageIllnessDuration" );
      return true;
    }
    if( !publicOnly && modelElementContains(x, y, 270, 360) ) {
      panel.addInspect( 270, 360, this, "InfectionRate" );
      return true;
    }
    if( !publicOnly && modelElementContains(x, y, 460, 360) ) {
      panel.addInspect( 460, 360, this, "RecoveryRate" );
      return true;
    }
    if( !publicOnly && modelElementContains(x, y, 170, 360) ) {
      panel.addInspect( 170, 360, this, "Susceptible" );
      return true;
    }
    if( !publicOnly && modelElementContains(x, y, 370, 360) ) {
      panel.addInspect( 370, 360, this, "Infectious" );
      return true;
    }
    if( !publicOnly && modelElementContains(x, y, 550, 360) ) {
      panel.addInspect( 550, 360, this, "Recovered" );
      return true;
    }
    if( !publicOnly && modelElementContains(x, y, 370, 500) ) {
      panel.addInspect( 370, 500, this, "InfectiousDS" );
      return true;
    }
    return false;
  }

  /**
   * Constructor
   */
  public Main( Engine engine, ActiveObject owner, ActiveObjectCollection<? extends Main> collection ) {
    super( engine, owner, collection );
    // Registering in Engine continuous part
    getEngine().registerActiveObjectWithEquations( this );
  }

  @Override
  public void create() {
    // Dynamic initialization of persistent elements
    {
    List<DataItem> _items = new ArrayList<DataItem>( 3 );
    _items.add( new DataItem() {
      @Override
      public void update() {
        setValue( _chart_DataItem0Value() );
      }
    });
    _items.add( new DataItem() {
      @Override
      public void update() {
        setValue( _chart_DataItem1Value() );
      }
    });
    _items.add( new DataItem() {
      @Override
      public void update() {
        setValue( _chart_DataItem2Value() );
      }
    });
    List<String> _titles = new ArrayList<String>( 3 );
    _titles.add( "Susceptible" );
    _titles.add( "Infectious" );
    _titles.add( "Recovered" );
    List<Color> _colors = new ArrayList<Color>( 3 );
    _colors.add( gold );
    _colors.add( red );
    _colors.add( gray );
    chart = new PieChart(
                                Main.this, true, 40, 70,
                                        120, 120,
          null, null,
          10, 10,
                                        100, 100, white, null, black,
          0, Chart.NONE,
          _items, _titles, _colors
    );
    }
    {
    DataSet _item;
    List<DataSet> _items = new ArrayList<DataSet>( 3 );
    _items.add( _chartStack_expression0_dataSet_xjal );
    _items.add( _chartStack_expression1_dataSet_xjal );
    _items.add( _chartStack_expression2_dataSet_xjal );
    List<String> _titles = new ArrayList<String>( 3 );
    _titles.add( "Susceptible" );
    _titles.add( "Infectious" );
    _titles.add( "Recovered" );
    List<Color> _colors = new ArrayList<Color>( 3 );
    _colors.add( gold );
    _colors.add( red );
    _colors.add( gray );
```

```
  chartStack = new TimeStackChart(
                                Main.this, true, 170, 70,
                                        580, 160,
        null, null,
        20, 10,
                                        540, 100, white, dimGray, black,
        20, Chart.SOUTH,

100 , null, Chart.SCALE_100_PERCENT
        , 0, Chart.GRID_DEFAULT, Chart.GRID_DEFAULT,
        darkGray, darkGray, _items, _titles, _colors
  );
  }
            // Port connectors with non-replicated objects
  // Creating replicated embedded objects
  assignInitialConditions();
  onCreate();
 }

 @Override
 public void start() {
  _InfectiousDS_autoUpdateEvent_xjal.start();
  _chart_autoUpdateEvent_xjal.start();
  _chartStack_autoUpdateEvent_xjal.start();
  _autoCreatedDS_xjal.start();
  onStartup();
 }


 public void onDestroy() {
  super.onDestroy();
  _InfectiousDS_autoUpdateEvent_xjal.onDestroy();
  _chart_autoUpdateEvent_xjal.onDestroy();
  _chartStack_autoUpdateEvent_xjal.onDestroy();
  _autoCreatedDS_xjal.onDestroy();
  // Unregistering in Engine continuous part
  getEngine().unregisterActiveObjectWithEquations( this );
 }


}
```

Simulation.java

```
package sir;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import javax.swing.JApplet;

public class Simulation extends ExperimentSimulation<Main> {
 // Plain Variables
  public
double
 ContactRate =
5 ;
  public
double
 Infectivity =
0.05 ;
  public
double
 AverageIllnessDuration =
15 ;

 static final Font _button_Font = new Font("Dialog", 0, 12 );
 static final Font _text2_Font = new Font("SansSerif", 0, 12 );
 static final Font _text3_Font = _text2_Font;
 static final Font _text4_Font = new Font("SansSerif", 1, 12 );
 static final Font _text5_Font = _text2_Font;
 static final Font _text6_Font = _text2_Font;
 static final Font _text7_Font = _text2_Font;
 static final Font _text8_Font = _text2_Font;
 static final Font _text9_Font = new Font("SansSerif", 0, 32 );
 static final Font _text_Font = _text4_Font;
 static final Font _text10_Font = new Font("SansSerif", 0, 9 );
 static final int button = 1;
 static final int slider = 2;
 static final int slider1 = 3;
 static final int slider2 = 4;
 static final int rect = 5;
 static final int text2 = 6;
 static final int text3 = 7;
 static final int text4 = 8;
 static final int line = 9;
 static final int text5 = 10;
 static final int text6 = 11;
 static final int text7 = 12;
```

```
  static final int text8 = 13;
  static final int text9 = 14;
  static final int line1 = 15;
  static final int text = 16;
  static final int line2 = 17;
  static final int image = 18;
  static final int text10 = 19;
  static final int _plot = 20;

   /**
   * Top-level presentation group content
   */
  static final int[] _presentation_content = new int[] {
    button,
    slider,
    slider1,
    slider2,
    rect,
    text2,
    text3,
    text4,
    line,
    text5,
    text6,
    text7,
    text8,
    text9,
    line1,
    text,
    line2,
    image,
    text10,
    _plot,
  };

   /**
   * Top-level presentation group id
   */
  static final int presentation = 0;

   /**
   * Top-level icon group id
   */
  static final int icon = -1;


   @Override
  public void executeShapeControlAction( int _shape, int index ) {
    switch( _shape ) {
      case button: {
run();

getEngine().getPresentation().setPresentable( getEngine().getRoot() );
;}
        break;
      default:
        super.executeShapeControlAction( _shape, index );
        break;
    }
  }

   @Override
  public void executeShapeControlAction( int _shape, int index, double
value ) {
    switch( _shape ) {
      case slider:
        ContactRate = value;
        break;
      case slider1:
        Infectivity = value;
        break;
      case slider2:
        AverageIllnessDuration = value;
        break;
      default:
        super.executeShapeControlAction( _shape, index, value );
        break;
    }
  }

   @Override
  public int getShapeControlValueType( int _shape, int index ) {
    switch( _shape ) {
      case slider: return ShapeControl.TYPE_DOUBLE;
      case slider1: return ShapeControl.TYPE_DOUBLE;
      case slider2: return ShapeControl.TYPE_DOUBLE;
      default: return super.getShapeControlValueType( _shape, index );
    }
  }

   @Override
  public double getShapeControlMinimum( int _shape, int index ) {
    switch( _shape ) {
      case slider: return
0
;
      case slider1: return
0
;
      case slider2: return
0
;
      default: return super.getShapeControlMinimum( _shape, index );
    }
  }

   @Override
  public double getShapeControlMaximum( int _shape, int index ) {
    switch( _shape ) {
      case slider: return
20
;
      case slider1: return
1
;
      case slider2: return
100
;
      default: return super.getShapeControlMaximum( _shape, index );
    }
  }

   @Override
  public double getShapeControlDefaultValueDouble( int _shape, int index
) {
    switch(_shape) {
      case slider: return
ContactRate
;
      case slider1: return
Infectivity
;
```

```
    case slider2: return
AverageIllnessDuration
;
    default: return super.getShapeControlDefaultValueDouble( _shape,
index );
   }
 }

 @Override
 public int[] getShapeGroupContent( int _shape ){
    switch( _shape ) {
      case presentation: return _presentation_content;
      default: return super.getShapeGroupContent( _shape );
    }
 }


 @Override
 public String getNameOfShape( int _shape ) {
  switch( _shape ) {
    case rect: return "rect";
    case text2: return "text2";
    case text3: return "text3";
    case text4: return "text4";
    case line: return "line";
    case text5: return "text5";
    case text6: return "text6";
    case text7: return "text7";
    case text8: return "text8";
    case text9: return "text9";
    case line1: return "line1";
    case text: return "text";
    case line2: return "line2";
    case image: return "image";
    case text10: return "text10";
    case button: return "button";
    case slider: return "slider";
    case slider1: return "slider1";
    case slider2: return "slider2";
    default: return super.getNameOfShape( _shape );
  }
 }

 @Override
 public int getShapeType( int _shape ) {
  switch( _shape ) {
    case presentation: return SHAPE_GROUP;
    case icon: return SHAPE_GROUP;
    case rect: return SHAPE_RECTANGLE;
    case text2: return SHAPE_TEXT;
    case text3: return SHAPE_TEXT;
    case text4: return SHAPE_TEXT;
    case line: return SHAPE_LINE;
    case text5: return SHAPE_TEXT;
    case text6: return SHAPE_TEXT;
    case text7: return SHAPE_TEXT;
    case text8: return SHAPE_TEXT;
    case text9: return SHAPE_TEXT;
    case line1: return SHAPE_LINE;
    case text: return SHAPE_TEXT;
    case line2: return SHAPE_LINE;
    case image: return SHAPE_IMAGE;
    case text10: return SHAPE_TEXT;
    case button: return SHAPE_BUTTON;
    case slider: return SHAPE_SLIDER;
    case slider1: return SHAPE_SLIDER;
    case slider2: return SHAPE_SLIDER;
    default: return super.getShapeType( _shape );
  }
 }

 @Override
 public double getShapeX( int _shape, int index ) {
  switch( _shape ) {
    case rect: return 0;
    case text2: return 30;
    case text3: return 335;
    case text4: return 30;
    case line: return 105;
    case text5: return 29;
    case text6: return 334;
    case text7: return 29;
    case text8: return 334;
    case text9: return 30;
    case line1: return 30;
    case text: return 420;
    case line2: return 550;
    case image: return 24;
    case text10: return 64;
    case button: return 30;
    case slider: return 25;
    case slider1: return 24;
    case slider2: return 24;
    default: return super.getShapeX( _shape, index );
  }
 }

 @Override
 public double getShapeY( int _shape, int index ) {
  switch( _shape ) {
    case rect: return 0;
    case text2: return 125;
    case text3: return 125;
    case text4: return 90;
    case line: return 100;
    case text5: return 185;
    case text6: return 185;
    case text7: return 250;
    case text8: return 250;
    case text9: return 20;
    case line1: return 60;
    case text: return 90;
    case line2: return 100;
    case image: return 530;
    case text10: return 570;
    case button: return 340;
    case slider: return 140;
    case slider1: return 200;
    case slider2: return 260;
    default: return super.getShapeY( _shape, index );
  }
 }

 @Override
 public Color getShapeLineColor( int _shape, int index ) {
  switch( _shape ) {
    // shapes
    case rect: return null;
    case line: return gray;
    case line1: return navy;
    case line2: return gray;
```

```
    // controls (text color)
    case button: return black;
    // charts (border color)
    default: return super.getShapeLineColor( _shape, index );
  }
 }

 @Override
 public Color getShapeFillColor( int _shape, int index ) {
  switch( _shape ) {
    case rect: return lightSteelBlue;
    case text2: return black;
    case text3: return blue;
    case text4: return black;
    case text5: return black;
    case text6: return blue;
    case text7: return black;
    case text8: return blue;
    case text9: return navy;
    case text: return black;
    case text10: return navy;
    case button: return controlDefault;
    case slider: return transparent;
    case slider1: return transparent;
    case slider2: return transparent;
    default: return super.getShapeFillColor( _shape, index );
  }
 }

 @Override
 public int getShapeLineStyle( int _shape, int index ) {
  switch( _shape ) {
    case rect: return LINE_STYLE_SOLID;
    case line: return LINE_STYLE_SOLID;
    case line1: return LINE_STYLE_SOLID;
    case line2: return LINE_STYLE_SOLID;
    default: return super.getShapeLineStyle( _shape, index );
  }
 }

 @Override
 public double getShapeLineWidth( int _shape, int index ) {
  switch( _shape ) {
    case line: return 0;
    case line2: return 0;
    default: return super.getShapeLineWidth( _shape, index );
  }
 }

 @Override
 public double getShapeLineDx( int _shape, int index ) {
  switch( _shape ) {
    case line: return 230;
    case line1: return 740;
    case line2: return 220;
    default: return super.getShapeLineDx( _shape, index );
  }
 }

 @Override
 public double getShapeLineDy( int _shape, int index ) {
  switch( _shape ) {
    case line: return 0;
    case line1: return 0;
    case line2: return 0;
    default: return super.getShapeLineDy( _shape, index );
  }
 }

 @Override
 public double getShapeWidth( int _shape, int index ) {
  switch( _shape ) {
    case rect: return 800;
    case image: return 116;
    case button: return 300;
    case slider: return 316;
    case slider1: return 316;
    case slider2: return 316;
    default: return super.getShapeWidth( _shape, index );
  }
 }

 @Override
 public double getShapeHeight( int _shape, int index ) {
  switch( _shape ) {
    case rect: return 600;
    case image: return 40;
    case button: return 30;
    case slider: return 40;
    case slider1: return 40;
    case slider2: return 40;
    default: return super.getShapeHeight( _shape, index );
  }
 }


 // Images files
 static final String[] _image_filenames = {
   "xjlogo.png",
 };

 @Override
 public String getShapePackagePrefix( int shape ) {
  switch( shape ) {
    case image: return "/sir/";
    default: return super.getShapePackagePrefix( shape );
  }
 }

 @Override
 public String[] getShapeImageFileNames( int shape, int index ) {
  switch( shape ) {
    case image: return _image_filenames;
    default: return super.getShapeImageFileNames( shape, index );
  }
 }


 @Override
 public Object getShapeText( int _shape, int index ) {
  switch( _shape ) {
    case button: return "Run the model and switch to Simulation view";
    case text2: return "Contact Rate";
    case text3: return
format( ContactRate )
;
    case text4: return "Parameters";
    case text5: return "Infectivity";
    case text6: return
format( Infectivity )
```

```
;
    case text7: return "Average Illness Duration";
    case text8: return
format( AverageIllnessDuration )
;
    case text9: return "Susceptible Infectious Recovered Simulation
Model";
    case text: return "Simulation run output";
    case text10: return "AnyLogic and this model is (c) XJ Technologies,
www.anylogic.com. All rights reserved.";
    default: return super.getShapeText( _shape, index );
   }
  }


  @Override
  public Font getShapeFont( int _shape, int index ) {
   switch( _shape ) {
    case text2: return _text2_Font;
    case text3: return _text3_Font;
    case text4: return _text4_Font;
    case text5: return _text5_Font;
    case text6: return _text6_Font;
    case text7: return _text7_Font;
    case text8: return _text8_Font;
    case text9: return _text9_Font;
    case text: return _text_Font;
    case text10: return _text10_Font;
    case button: return _button_Font;
    default: return super.getShapeFont( _shape, index );
   }
  }

  @Override
  public int getShapeTextAlignment( int _shape, int index ) {
   switch( _shape ) {
    case text2: return ALIGNMENT_LEFT;
    case text3: return ALIGNMENT_RIGHT;
    case text4: return ALIGNMENT_LEFT;
    case text5: return ALIGNMENT_LEFT;
    case text6: return ALIGNMENT_RIGHT;
    case text7: return ALIGNMENT_LEFT;
    case text8: return ALIGNMENT_RIGHT;
    case text9: return ALIGNMENT_LEFT;
    case text: return ALIGNMENT_LEFT;
    case text10: return ALIGNMENT_LEFT;
    default: return super.getShapeTextAlignment( _shape, index );
   }
  }

  @Override
  public boolean isShapeControlEnabled( int _shape, int index ) {
   switch( _shape ) {
    case button: return
getEngine().getState() == Engine.IDLE
;
    default: return super.isShapeControlEnabled( _shape, index );
   }
  }

  @Override
  public boolean isShapeControlVertical( int _shape, int index ) {
   switch( _shape ) {
    case slider: return false;
    case slider1: return false;
    case slider2: return false;
    default: return super.isShapeControlVertical( _shape, index );
   }
  }


  TimePlot plot;

  @Override
  public Object getPersistentShape( int _shape ) {
   switch(_shape){

    case _plot: return plot;
    default: return null;
   }
  }


  @Override
  public int getWindowWidth() {
   return 800;
  }

  @Override
  public int getWindowHeight() {
   return 600;
  }

  /**
   * Applet class to run experiment as java applet
   */
  public static class Applet extends JApplet {

    Simulation ex;

    @Override
    public void init() {
     ex = new Simulation();
     ex.setup( this );
    }

    @Override
    public void destroy() {
     ex.close();
    }

  }

  @Override
  public void initDefaultRandomNumberGenerator(Engine engine) {
   engine.getDefaultRandomGenerator().setSeed( 1 );
  }

  @Override
  public Main createRoot( Engine engine ) {
   // Create the root object
   return new Main( engine, null, null );
  }

  @Override
  public void setupRootParameters( Main root, boolean
callOnChangeActions ) {
    double ContactRate_xjal =
ContactRate
;
    if (callOnChangeActions) {
     root.set_ContactRate( ContactRate_xjal );
```

```
    } else {
      root.ContactRate = ContactRate_xjal;
    }
    double Infectivity_xjal =
Infectivity
;
    if (callOnChangeActions) {
      root.set_Infectivity( Infectivity_xjal );
    } else {
      root.Infectivity = Infectivity_xjal;
    }
    double                    TotalPopulation_xjal                    =
root._TotalPopulation_DefaultValue_xjal();
    if (callOnChangeActions) {
      root.set_TotalPopulation( TotalPopulation_xjal );
    } else {
      root.TotalPopulation = TotalPopulation_xjal;
    }
    double AverageIllnessDuration_xjal =
AverageIllnessDuration
;
    if (callOnChangeActions) {
      root.set_AverageIllnessDuration( AverageIllnessDuration_xjal );
    } else {
      root.AverageIllnessDuration = AverageIllnessDuration_xjal;
    }
  }

  /**
   * Engine setup
   */
  @Override
  public void setupEngine(Engine engine) {
    engine.setATOL( 1.0E-5 );
    engine.setRTOL( 1.0E-5 );
    engine.setTTOL( 1.0E-5 );
    engine.setHTOL( 0.0010 );
    engine.setSolverODE( Engine.SOLVER_ODE_EULER );
    engine.setSolverNAE( Engine.SOLVER_NAE_MODIFIED_NEWTON );
    engine.setSolverDAE( Engine.SOLVER_DAE_RK45_NEWTON );
    engine.setVMethods( 16032 );
    engine.setStartTime( 0.0 );
    engine.setTimeUnit( TIME_UNIT_DAY );
    engine.setStopTime( 100.0 );
    engine.setRealTimeMode( false );
  }

  /**
   * Experiment setup
   */
  @Override
  public void setup( JApplet applet ) {
    setName( "Simulation" );

    // Dynamic initialization of persistent elements
    {
    DataSet _item;
    List<DataSet> _items = new ArrayList<DataSet>( 0 );
    List<String> _titles = new ArrayList<String>( 0 );
    List<Chart2DPlot.Appearance>        _appearances        =        new
ArrayList<Chart2DPlot.Appearance>( 0 );
    plot = new TimePlot(
                                Simulation.this, true, 380, 110,
                                        400, 440,
          null, null,
          40, 20,
                                                    340, 190, white, black, black,
          210, Chart.SOUTH,

100
          , null, Chart.SCALE_AUTO,
          0, 0, Chart.GRID_DEFAULT, Chart.GRID_DEFAULT,
          darkGray, darkGray, _items, _titles, _appearances
    );
    }
    // Setup presentation
    Presentation _p = new Presentation( this, applet != null ?
Presentation.MODE_APPLET :
        Presentation.MODE_APPLICATION, applet );
    _p.start();

    Panel _panel = _p.getPanel();
    ToolBar _tb = _p.getToolBar();
    StatusBar _sb = _p.getStatusBar();

    _panel.setZoomEnabled( false );
    _panel.setPanningEnabled( false );
    _panel.setFrameManagementAdaptive( false );
    _panel.setFrameRate( 20.0 );

    _sb.setSectionVisible( StatusBar.DATE, false );
    _sb.setSectionVisible( StatusBar.EPS, false );
    _sb.setSectionVisible( StatusBar.EXPERIMENT, false );
    _sb.setSectionVisible( StatusBar.FPS, false );
    _sb.setSectionVisible( StatusBar.MEMORY, false );
    _sb.setSectionVisible( StatusBar.SECONDS, true );
    _sb.setSectionVisible( StatusBar.SIMULATION, true );
    _sb.setSectionVisible( StatusBar.STATUS, true );
    _sb.setSectionVisible( StatusBar.STEP, false );
    _sb.setSectionVisible( StatusBar.TIME, true );
    _tb.setSectionVisible( ToolBar.ANIMATION, false );
    _tb.setSectionVisible( ToolBar.EXECUTION, true );
    _tb.setSectionVisible( ToolBar.FILE, false );
    _tb.setSectionVisible( ToolBar.NAVIGATION, false );
    _tb.setSectionVisible( ToolBar.TIME_SCALE, true );
    _tb.setSectionVisible( ToolBar.VIEW, false );
  }

}
```

Παράρτημα 2

UML - Διαγράμματα κλάσεων

Person
+statechart: Statechart = new Statechart(this, short(2)) +Susceptible: short=0 +Infectious: short=1 +Recovered: short=2 +Recovery: TransitionRate = new TransitionRate(this) +Contact: TransitionRate = new TransitionRate(this) +Infection: TransitionMessage = new TransitionMessage(this) ~pixel: int=1 ~_presentation_content: int[*] = new int[] { pixel } ~presentation: int=0 ~icon: int=-1 ~_Infection_pointsX: int[*] = {170, 170} ~_Infection_pointsY: int[*] = {80, 130} ~_Recovery_pointsX: int[*] = {170, 170} ~_Recovery_pointsY: int[*] = {180, 250} ~_Contact_pointsX: int[*] = {210, 210, 230} ~_Contact_pointsY: int[*] = {180, 160, 160}
<<create>>+Person(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection) +getNameOf(_s: Statechart): String +executeActionOf(_s: Statechart) +getNameOfState(_state: short): String +enterState(_state: short, _destination: boolean) +exitState(_state: short, _t: Transition, _source: boolean, _statechart: Statechart) +getNameOf(_t: TransitionRate): String +getStatechartOf(_t: TransitionRate): Statechart +executeActionOf(_t: TransitionRate) +evaluateRateOf(_t: TransitionRate): double +getNameOf(_t: TransitionMessage): String +getStatechartOf(_t: TransitionMessage): Statechart +executeActionOf(_t: TransitionMessage, _msg: Object) +testMessageOf(_t: TransitionMessage, _msg: Object): boolean +getShapeGroupContent(_shape: int): int +getNameOfShape(_shape: int): String +getShapeType(_shape: int): int +getShapeX(_shape: int, index: int): double +getShapeY(_shape: int, index: int): double +getShapeFillColor(_shape: int, index: int): Color +drawModeElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean) +onClickModeAt(panel: Panel, x: double, y: double, clickCount: int, publicOnly: boolean): boolean +onReceive(msg: Object, sender: Agent) +create() +start() +get_Main(): sirab +onChange() +onDestroy()

sirab
+AverageIllnessDuration: double +ContactRate: double +InfectionProbability: double +TotalPopulation: int +nSusceptible: double +nInfectious: double +nRecovered: double +people: ActiveObjectArrayList = new ActiveObjectArrayList() ~rect3: int=1 ~text: int=2

~text1: int=3 ~text2: int=4 ~text3: int=5 ~text4: int=6 ~text5: int=7 ~rect: int=8 ~rect1: int=9 ~rect2: int=10 ~people_presentation: int=11 ~_presentation_content: int[*] = new int[] {rect3, text, text1, text2, text3, text4, text5, rect, rect1, rect2, people_presentation} ~presentation: int=0 ~icon: int=-1 +environment: Environment = new Environment(this)
<<create>> +sirab(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection) +_AverageIllnessDuration_DefaultValue_xjal(): double +set_AverageIllnessDuration(AverageIllnessDuration: double) ~onChange_AverageIllnessDuration() +_ContactRate_DefaultValue_xjal(): double +set_ContactRate(ContactRate: double) ~onChange_ContactRate() +_InfectionProbability_DefaultValue_xjal(): double +set_InfectionProbability(InfectionProbability: double) ~onChange_InfectionProbability() +_TotalPopulation_DefaultValue_xjal(): int +set_TotalPopulation(TotalPopulation: int) ~onChange_TotalPopulation() +getNameOf(ao: ActiveObject): String +getNameOf(aolist: ActiveObjectCollection): String +add_people(): Person +remove_people(object: Person): boolean -instantiate_people_xjal(index: int): Person -setupParameters_people_xjal(object: Person, index: int) -create_people_xjal(object: Person, index: int) +getShapeGroupContent(_shape: int): int +getNameOfShape(_shape: int): String +getShapeType(_shape: int): int +getShapeReplication(_shape: int): int +getShapeX(_shape: int, index: int): double +getShapeY(_shape: int, index: int): double +getShapeEmbeddedObject(_shape: int): Object +getShapeLineColor(_shape: int, index: int): Color +getShapeFillColor(_shape: int, index: int): Color +getShapeLineStyle(_shape: int, index: int): int +getShapeWidth(_shape: int, index: int): double +getShapeHeight(_shape: int, index: int): double +getShapeText(_shape: int, index: int): Object +getShapeFont(_shape: int, index: int): Font +getShapeTextAlignment(_shape: int, index: int): int +create() +start() +onStartup() +getEmbeddedObjects(): List.Object +onDestroy()

Κώδικας Κεφαλαίου 4 – Έργο SIR Agent Based

Main.java

```
package sir_agent_based;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import java.awt.geom.Arc2D;

public class Main extends ActiveObject
{
 // Parameters

 public double AverageIllnessDuration;

 /**
  * Returns default value for parameter
<code>AverageIllnessDuration</code>.
  * <i>This method should not be called by user</i>
  */
 public
double _AverageIllnessDuration_DefaultValue_xjal() {
  return 15 ;
 }

 public void set_AverageIllnessDuration(
double AverageIllnessDuration ) {
  if (AverageIllnessDuration == this.AverageIllnessDuration) {
   return;
  }
  this.AverageIllnessDuration = AverageIllnessDuration;
  onChange_AverageIllnessDuration();
  onChange();
 }

 void onChange_AverageIllnessDuration() {
 }

 public double ContactRate;

 /**
  * Returns default value for parameter <code>ContactRate</code>.
  * <i>This method should not be called by user</i>
  */
 public
double _ContactRate_DefaultValue_xjal() {
  return 1 ;
 }

 public void set_ContactRate(
double ContactRate ) {
  if (ContactRate == this.ContactRate) {
   return;
  }
  this.ContactRate = ContactRate;
  onChange_ContactRate();
  onChange();
 }

 void onChange_ContactRate() {
 }

 public double InfectionProbability;

 /**
  * Returns default value for parameter
<code>InfectionProbability</code>.
  * <i>This method should not be called by user</i>
  */
 public
double _InfectionProbability_DefaultValue_xjal() {
  return 0.5 ;
 }
```

```
  public void set_InfectionProbability(
double  InfectionProbability ) {
    if (InfectionProbability == this.InfectionProbability) {
      return;
    }
    this.InfectionProbability = InfectionProbability;
    onChange_InfectionProbability();
    onChange();
  }

  void onChange_InfectionProbability() {
  }


  public int  TotalPopulation;

  /**
   *        Returns        default        value        for        parameter
<code>TotalPopulation</code>.
   * <i>This method should not be called by user</i>
   */
  public
int  _TotalPopulation_DefaultValue_xjal() {
    return 250000 ;
  }

  public void set_TotalPopulation(
int  TotalPopulation ) {
    if (TotalPopulation == this.TotalPopulation) {
      return;
    }
    this.TotalPopulation = TotalPopulation;
    onChange_TotalPopulation();
    onChange();
  }

  void onChange_TotalPopulation() {
  }

  // Plain Variables

  public double  nSusceptible;
  public double  nInfectious;
  public double  nRecovered;

  // Collection Variables

  // Flow/Auxiliary Variables

  // Stock Variables

  // Embedded Objects

  public String getNameOf( ActiveObject ao ) {
    return null;
  }

  public        ActiveObjectArrayList<Person>        people        =        new
ActiveObjectArrayList<Person>();

  public String getNameOf( ActiveObjectCollection<?> aolist ) {
    if( aolist == people ) return "people";
    return null;
  }

  /**
   * This method creates and adds new embedded object in the
replicated embedded object collection people<br>
   * @return newly created embedded object
   */
  public Person add_people() {
    int index = people.size();
    Person object = instantiate_people_xjal( index );
    setupParameters_people_xjal( object, index );
    create_people_xjal( object, index );
    object.start();
    return object;
  }

  /**
   * This method removes the given embedded object from the replicated
embedded object collection people<br>
   * The given object is destroyed, but not immediately in common case.
   * @param object the active object - element of replicated embedded
object people - which should be removed
   * @return <code>true</code> if object was removed successfully,
<code>false</code> if it doesn't belong to people
   */
  public boolean remove_people( Person object ) {
    if( ! people._remove( object ) ){
      return false;
    }
    object.setDestroyed();
    return true;
  }

  /**
   * Creates an embedded object instance and adds it to the end of
replicated embedded object list<br>
   * <i>This method should not be called by user</i>
   */
  private Person instantiate_people_xjal( final int index ) {Person object
= new Person( getEngine(), this, people );

    people._add(object);

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_people_xjal(Person object, final int index
) {
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_people_xjal(Person object, final int index ) {
    object.setEnvironment(
environment
);
    object.create();

    // Port connections
  }

  static final Font _text_Font = new Font("SansSerif", 1, 12 );
  static final Font _text1_Font = _text_Font;
```

```
static final Font _text2_Font = _text_Font;
static final Font _text3_Font = _text_Font;
static final Font _text4_Font = _text_Font;
static final Font _text5_Font = _text_Font;
static final Color _rect3_FillColor = new Color( 0xFFC7C7A5, true );
static final int rect3 = 1;
static final int text = 2;
static final int text1 = 3;
static final int text2 = 4;
static final int text3 = 5;
static final int text4 = 6;
static final int text5 = 7;
static final int rect = 8;
static final int rect1 = 9;
static final int rect2 = 10;
static final int people_presentation = 11;


/**
 * Top-level presentation group content
 */
static final int[] _presentation_content = new int[] {
  rect3,
  text,
  text1,
  text2,
  text3,
  text4,
  text5,
  rect,
  rect1,
  rect2,
  people_presentation,
};

/**
 * Top-level presentation group id
 */
static final int presentation = 0;

/**
 * Top-level icon group id
 */
static final int icon = -1;


@Override
public int[] getShapeGroupContent( int _shape ){
    switch( _shape ) {
      case presentation: return _presentation_content;
      default: return super.getShapeGroupContent( _shape );
    }
}


@Override
public String getNameOfShape( int _shape ) {
  switch( _shape ) {
    case rect3: return "rect3";
    case text: return "text";
    case text1: return "text1";
    case text2: return "text2";
    case text3: return "text3";
    case text4: return "text4";
    case text5: return "text5";
    case rect: return "rect";
    case rect1: return "rect1";
    case rect2: return "rect2";
    case people_presentation: return "people_presentation";
    default: return super.getNameOfShape( _shape );
  }
}

@Override
public int getShapeType( int _shape ) {
  switch( _shape ) {
    case presentation: return SHAPE_GROUP;
    case icon: return SHAPE_GROUP;
    case rect3: return SHAPE_RECTANGLE;
    case text: return SHAPE_TEXT;
    case text1: return SHAPE_TEXT;
    case text2: return SHAPE_TEXT;
    case text3: return SHAPE_TEXT;
    case text4: return SHAPE_TEXT;
    case text5: return SHAPE_TEXT;
    case rect: return SHAPE_RECTANGLE;
    case rect1: return SHAPE_RECTANGLE;
    case rect2: return SHAPE_RECTANGLE;
    case people_presentation: return SHAPE_EMBEDDED_OBJECT;
    default: return super.getShapeType( _shape );
  }
}

@Override
public int getShapeReplication( int _shape ) {
  switch( _shape ) {
    case people_presentation: return
people.size()
;
    default: return super.getShapeReplication( _shape );
  }
}

@Override
public double getShapeX( int _shape, int index ) {
  switch( _shape ) {
    case rect3: return 0;
    case text: return 65;
    case text1: return 140;
    case text2: return 250;
    case text3: return 315;
    case text4: return 430;
    case text5: return 500;
    case rect: return 50;
    case rect1: return 235;
    case rect2: return 415;
    case people_presentation: return 50;
    default: return super.getShapeX( _shape, index );
  }
}

@Override
public double getShapeY( int _shape, int index ) {
  switch( _shape ) {
    case rect3: return 0;
    case text: return 20;
    case text1: return 20;
    case text2: return 20;
    case text3: return 20;
    case text4: return 20;
```

```
    case text5: return 20;
    case rect: return 25;
    case rect1: return 25;
    case rect2: return 25;
    case people_presentation: return 50;
    default: return super.getShapeY( _shape, index );
   }
  }

  @Override
  public Object getShapeEmbeddedObject( int _shape ) {
   switch( _shape ) {
    case people_presentation: return people;
    default: return super.getShapeEmbeddedObject( _shape );
   }
  }

  @Override
  public Color getShapeLineColor( int _shape, int index ) {
   switch( _shape ) {
    // shapes
    case rect3: return null;
    case rect: return null;
    case rect1: return null;
    case rect2: return null;
    // controls (text color)
    // charts (border color)
    default: return super.getShapeLineColor( _shape, index );
   }
  }

  @Override
  public Color getShapeFillColor( int _shape, int index ) {
   switch( _shape ) {
    case rect3: return _rect3_FillColor;
    case text: return black;
    case text1: return blue;
    case text2: return black;
    case text3: return blue;
    case text4: return black;
    case text5: return blue;
    case rect: return
GOLD
;
    case rect1: return red;
    case rect2: return gray;
    default: return super.getShapeFillColor( _shape, index );
   }
  }

  @Override
  public int getShapeLineStyle( int _shape, int index ) {
   switch( _shape ) {
    case rect3: return LINE_STYLE_SOLID;
    case rect: return LINE_STYLE_SOLID;
    case rect1: return LINE_STYLE_SOLID;
    case rect2: return LINE_STYLE_SOLID;
    default: return super.getShapeLineStyle( _shape, index );
   }
  }

  @Override
  public double getShapeWidth( int _shape, int index ) {
   switch( _shape ) {
    case rect3: return 600;
    case rect: return 5;
    case rect1: return 5;
    case rect2: return 5;
    default: return super.getShapeWidth( _shape, index );
   }
  }

  @Override
  public double getShapeHeight( int _shape, int index ) {
   switch( _shape ) {
    case rect3: return 600;
    case rect: return 5;
    case rect1: return 5;
    case rect2: return 5;
    default: return super.getShapeHeight( _shape, index );
   }
  }


  @Override
  public Object getShapeText( int _shape, int index ) {
   switch( _shape ) {
    case text: return "Susceptible:";
    case text1: return format( nSusceptible ) ;
    case text2: return "Infectious:";
    case text3: return format( nInfectious ) ;
    case text4: return "Recovered:";
    case text5: return format( nRecovered ) ;
    default: return super.getShapeText( _shape, index );
   }
  }


  @Override
  public Font getShapeFont( int _shape, int index ) {
   switch( _shape ) {
    case text: return _text_Font;
    case text1: return _text1_Font;
    case text2: return _text2_Font;
    case text3: return _text3_Font;
    case text4: return _text4_Font;
    case text5: return _text5_Font;
    default: return super.getShapeFont( _shape, index );
   }
  }

  @Override
  public int getShapeTextAlignment( int _shape, int index ) {
   switch( _shape ) {
    case text: return ALIGNMENT_LEFT;
    case text1: return ALIGNMENT_LEFT;
    case text2: return ALIGNMENT_LEFT;
    case text3: return ALIGNMENT_LEFT;
    case text4: return ALIGNMENT_LEFT;
    case text5: return ALIGNMENT_LEFT;
    default: return super.getShapeTextAlignment( _shape, index );
   }
  }

  // Environments
  public final Environment environment = new Environment( this );

  /**
   * Constructor
   */
```

```
 public Main( Engine engine, ActiveObject owner,
ActiveObjectCollection<? extends Main> collection ) {
   super( engine, owner, collection );
 }

 @Override
 public void create() {
  // Creating embedded object instances
  for ( int i = 0; i <
TotalPopulation
; i++ ) {
    instantiate_people_xjal( i );
  }
  // Dynamic initialization of persistent elements
  // Environments setup
  environment.disableSteps();
  environment.setSpaceDiscrete(
500 ,
500 ,
500 ,
500 , Environment.NEIGHBORHOOD_MOORE );
  environment.setNetworkUserDefined();
  environment.setLayoutType( Environment.LAYOUT_ARRANGED );
            // Port connectors with non-replicated objects
  // Creating replicated embedded objects
  for ( int i = 0; i < people.size(); i++ ) {
    setupParameters_people_xjal( people.get(i), i );
    create_people_xjal( people.get(i), i );
  }
  assignInitialConditions();
  onCreate();
 }

 @Override
 public void start() {
  for (ActiveObject embeddedObject : people){
   embeddedObject.start();
  }
  onStartup();
 }

 public void onStartup() {
  super.onStartup();

environment.deliverToRandom( "Infection" );
 }

 public List<Object> getEmbeddedObjects() {
  LinkedList<Object> list = new LinkedList<Object>();
  list.add( people );
  return list;
 }

 public void onDestroy() {
  super.onDestroy();
  environment.onDestroy();
  for (ActiveObject embeddedObject : people) {
   embeddedObject.onDestroy();
  }
 }
}
```

Person.java

```
package sir_agent_based;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import java.awt.geom.Arc2D;

public class Person extends Agent
{
 // Plain Variables

 public Color  color;

 // Collection Variables

 // Flow/Auxiliary Variables

 // Stock Variables

 // Statecharts
 public Statechart statechart = new Statechart( this, (short)2 );

 @Override
 public String getNameOf( Statechart _s ) {
```

```
  if(_s == this.statechart) return "statechart";
  return super.getNameOf( _s );
 }

 @Override
 public void executeActionOf( Statechart _s ) {
  if( _s == this.statechart ) {
   enterState( Susceptible, true );
   return;
  }
  super.executeActionOf( _s );
 }

 // States of all statecharts

 public static final short Susceptible = 0;
 public static final short Infectious = 1;
 public static final short Recovered = 2;


 @Override
 public String getNameOfState( short _state ) {
  switch( _state ) {
   case Susceptible: return "Susceptible";
   case Infectious: return "Infectious";
   case Recovered: return "Recovered";
   default: return super.getNameOfState( _state );
  }
 }

 @Override
 public void enterState( short _state, boolean _destination ) {
  switch( _state ) {
   case Susceptible: // (Simple state (not composite))
    statechart.setActiveState( Susceptible );
    {
get_Main().nSusceptible++;
color = GOLD;
;}
    Infection.start();
    return;
   case Infectious: // (Simple state (not composite))
    statechart.setActiveState( Infectious );
    {
get_Main().nInfectious++;
color = RED;
;}
    Recovery.start();
    Contact.start();
    return;
   case Recovered: // (Simple state (not composite))
    statechart.setActiveState( Recovered );
    {
get_Main().nRecovered++;
color = GRAY;
;}
    return;
   default:
    super.enterState( _state, _destination );
    return;
  }
 }

 @Override
 public void exitState( short _state, Transition _t, boolean _source,
Statechart _statechart) {
  switch( _state ) {
   case Susceptible: // (Simple state (not composite))
    if ( !_source || _t != Infection) Infection.cancel();
    {
get_Main().nSusceptible--;
;}
    return;
   case Infectious: // (Simple state (not composite))
    if ( !_source || _t != Recovery) Recovery.cancel();
    if ( !_source || _t != Contact) Contact.cancel();
    {
get_Main().nInfectious--;
;}
    return;
   case Recovered: // (Simple state (not composite))
    {
get_Main().nRecovered--;
;}
    return;
   default:
    super.exitState( _state, _t, _source, _statechart);
    return;
  }
 }

 public TransitionRate Recovery = new TransitionRate( this );
 public TransitionRate Contact = new TransitionRate( this );

 @Override
 public String getNameOf( TransitionRate _t ) {
  if ( _t == Recovery ) return "Recovery";
  if ( _t == Contact ) return "Contact";
  return super.getNameOf( _t );
 }

 @Override
 public Statechart getStatechartOf( TransitionRate _t ) {
  if ( _t == Recovery ) return statechart;
  if ( _t == Contact ) return statechart;
  return super.getStatechartOf( _t );
 }

 @Override
 public void executeActionOf( TransitionRate _t ) {
  if ( _t == Recovery ) {
   exitState( Infectious, _t, true, statechart );
     enterState( Recovered, true );
   return;
  }
  if ( _t == Contact ) {
   {
send( "Infection", RANDOM_NEIGHBOR );

;}
   Contact.start();
   return;
  }
  super.executeActionOf( _t );
 }
 @Override
 public double evaluateRateOf( TransitionRate _t ) {
  if ( _t == Recovery ) return
1 / get_Main().AverageIllnessDuration
;
  if ( _t == Contact ) return
```

```
get_Main().ContactRate / get_Main().InfectionProbability
;
    return super.evaluateRateOf( _t );
  }

  public TransitionMessage Infection = new TransitionMessage( this );

  @Override
  public String getNameOf( TransitionMessage _t ) {
   if ( _t == Infection ) return "Infection";
   return super.getNameOf( _t );
  }

  @Override
  public Statechart getStatechartOf( TransitionMessage _t ) {
   if ( _t == Infection ) return statechart;
   return super.getStatechartOf( _t );
  }

  @Override
  public void executeActionOf( TransitionMessage _t, Object _msg ) {
   if ( _t == Infection ) {
     exitState( Susceptible, _t, true, statechart );
        enterState( Infectious, true );
     return;
   }
   super.executeActionOf( _t, _msg );
  }
  @Override
  public boolean testMessageOf( TransitionMessage _t, Object _msg ) {
   if ( _t == Infection ) {
     Object msg = (Object) _msg;
     return true;
   }
   return super.testMessageOf( _t, _msg );
  }
  static final int pixel = 1;


  /**
   * Top-level presentation group content
   */
  static final int[] _presentation_content = new int[] {
   pixel,
  };

  /**
   * Top-level presentation group id
   */
  static final int presentation = 0;

  /**
   * Top-level icon group id
   */
  static final int icon = -1;


  @Override
  public int[] getShapeGroupContent( int _shape ){
     switch( _shape ) {
       case presentation: return _presentation_content;
       default: return super.getShapeGroupContent( _shape );
     }
  }

  @Override
  public String getNameOfShape( int _shape ) {
   switch( _shape ) {
     case pixel: return "pixel";
     default: return super.getNameOfShape( _shape );
   }
  }

  @Override
  public int getShapeType( int _shape ) {
   switch( _shape ) {
     case presentation: return SHAPE_GROUP;
     case icon: return SHAPE_GROUP;
     case pixel: return SHAPE_PIXEL;
     default: return super.getShapeType( _shape );
   }
  }

  @Override
  public double getShapeX( int _shape, int index ) {
   switch( _shape ) {
     case pixel: return 0;
     default: return super.getShapeX( _shape, index );
   }
  }

  @Override
  public double getShapeY( int _shape, int index ) {
   switch( _shape ) {
     case pixel: return 0;
     default: return super.getShapeY( _shape, index );
    }
  }

  @Override
  public Color getShapeFillColor( int _shape, int index ) {
   switch( _shape ) {
     case pixel: return
color
;
     default: return super.getShapeFillColor( _shape, index );
    }
  }

  static final int[] _Infection_pointsX = {170, 170, };
  static final int[] _Infection_pointsY = {80, 130, };

  static final int[] _Recovery_pointsX = {170, 170, };
  static final int[] _Recovery_pointsY = {180, 250, };

  static final int[] _Contact_pointsX = {210, 210, 230, };
  static final int[] _Contact_pointsY = {180, 160, 160, };

  static final Color _Recovered_FillColor = new Color( 0xFFA9A9A9, true
);

  @Override
  public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
    if (!_publicOnly) {
      drawState( _panel, _g, 130, 50, 100, 30, 10, 10, "Susceptible", gold,
Susceptible, statechart );
    }
    if (!_publicOnly) {
      drawState( _panel, _g, 130, 130, 100, 50, 10, 10, "Infectious",
orangeRed, Infectious, statechart );
```

```
    }
    if (!_publicOnly) {
      drawState( _panel, _g, 130, 250, 100, 30, 10, 10, "Recovered",
_Recovered_FillColor, Recovered, statechart );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _Infection_pointsX, _Infection_pointsY,
180, 100, "Infection", Infection );
    }
    if (!_publicOnly) {
      drawStatechartEntryPoint( _panel, _g, 170, 30, 170, 50, 185, 23,
"statechart", statechart );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _Recovery_pointsX, _Recovery_pointsY,
180, 200, "Recovery", Recovery );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _Contact_pointsX, _Contact_pointsY,
160, 160, "Contact", Contact );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, 50, 50, 20, 0, "color", color, false );
    }
  }

  @Override
  public boolean onClickModelAt( Panel panel, double x, double y, int
clickCount, boolean publicOnly ) {
    if( !publicOnly && modelElementContains(x, y, 50, 50) ) {
      panel.addInspect( 50, 50, this, "color" );
      return true;
    }
    return false;
  }

  @Override
  public void onReceive( Object msg, Agent sender ) {

statechart.receiveMessage( msg );
  }

  /**
   * Constructor
   */
  public Person( Engine engine, ActiveObject owner,
ActiveObjectCollection<? extends Person> collection ) {
    super( engine, owner, collection );
  }

  @Override
  public void create() {
    // Dynamic initialization of persistent elements
                // Port connectors with non-replicated objects
    // Creating replicated embedded objects
    assignInitialConditions();
    onCreate();
  }

  @Override
  public void start() {
    statechart.start();
    onStartup();
  }

// User API ----------------------------------
  public Main get_Main() {
    ActiveObject owner = getOwner();
    if ( owner instanceof Main ) return (Main) owner;
    return null;
  }

  // Reaction on changes -----------------------
  public void onChange() {
    statechart.onChange();
  }

  public void onDestroy() {
    super.onDestroy();
    statechart.onDestroy();
  }
}
```

Simulation.java

```
package sir_agent_based;

import java.sql.Connection;
import java.sql.SQLException;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
```

```
import com.xj.anylogic.engine.presentation.*;
import javax.swing.JApplet;

public class Simulation extends ExperimentSimulation<Main> {
  static final Font _button_Font = new Font("Dialog", 0, 12 );
  static final Font _text_Font = new Font("SansSerif", 1, 28 );
  static final Font _text1_Font = _text_Font;
  static final Font _text2_Font = _text_Font;
  static final Font _text3_Font = _text_Font;
  static final Color _rect_FillColor = new Color( 0xFFC7C7A5, true );
  static final Color _rect1_FillColor = new Color( 0xFFECECE0, true );
  static final int button = 1;
  static final int rect = 2;
  static final int rect1 = 3;
  static final int text = 4;
  static final int text1 = 5;
  static final int text2 = 6;
  static final int text3 = 7;
  static final int poly = 8;
  static final int poly1 = 9;

  /**
   * Top-level presentation group content
   */
  static final int[] _presentation_content = new int[] {
    button,
    rect,
    rect1,
    text,
    text1,
    text2,
    text3,
    poly,
    poly1,
  };

  /**
   * Top-level presentation group id
   */
  static final int presentation = 0;

  /**
   * Top-level icon group id
   */
  static final int icon = -1;

  static final double[] _poly_pointsDX_xjal() {
    return new double[] { 0, 10, 10, 20, 10, 10, 0,  };
  }
  static final double[] _poly_pointsDY_xjal() {
    return new double[] { 0, 0, -5, 5, 15, 10, 10,  };
  }
  static final double[] _poly1_pointsDX_xjal() {
    return new double[] { 0, 10, 10, 20, 10, 10, 0,  };
  }
  static final double[] _poly1_pointsDY_xjal() {
    return new double[] { 0, 0, -5, 5, 15, 10, 10,  };
  }

  @Override
  public void executeShapeControlAction( int _shape, int index ) {
    switch( _shape ) {
      case button: {
run();
getEngine().getPresentation().setPresentable( getEngine().getRoot() );
;}
        break;
      default:
        super.executeShapeControlAction( _shape, index );
        break;
    }
  }

  @Override
  public int[] getShapeGroupContent( int _shape ){
    switch( _shape ) {
      case presentation: return _presentation_content;
      default: return super.getShapeGroupContent( _shape );
    }
  }
  @Override
  public String getNameOfShape( int _shape ) {
    switch( _shape ) {
      case rect: return "rect";
      case rect1: return "rect1";
      case text: return "text";
      case text1: return "text1";
      case text2: return "text2";
      case text3: return "text3";
      case poly: return "poly";
      case poly1: return "poly1";
      case button: return "button";
      default: return super.getNameOfShape( _shape );
    }
  }

  @Override
  public int getShapeType( int _shape ) {
    switch( _shape ) {
      case presentation: return SHAPE_GROUP;
      case icon: return SHAPE_GROUP;
      case rect: return SHAPE_RECTANGLE;
      case rect1: return SHAPE_RECTANGLE;
      case text: return SHAPE_TEXT;
      case text1: return SHAPE_TEXT;
      case text2: return SHAPE_TEXT;
      case text3: return SHAPE_TEXT;
      case poly: return SHAPE_POLYLINE;
      case poly1: return SHAPE_POLYLINE;
      case button: return SHAPE_BUTTON;
      default: return super.getShapeType( _shape );
    }
  }

  @Override
  public double getShapeX( int _shape, int index ) {
    switch( _shape ) {
      case rect: return 0;
      case rect1: return 0;
      case text: return 20;
      case text1: return 230;
      case text2: return 420;
      case text3: return 20;
      case poly: return 290;
      case poly1: return 330;
      case button: return 20;
      default: return super.getShapeX( _shape, index );
    }
  }
```

```
 @Override
 public double getShapeY( int _shape, int index ) {
  switch( _shape ) {
   case rect: return 0;
   case rect1: return 60;
   case text: return 60;
   case text1: return 60;
   case text2: return 60;
   case text3: return 10;
   case poly: return 130;
   case poly1: return 130;
   case button: return 120;
   default: return super.getShapeY( _shape, index );
  }
 }
 @Override
 public Color getShapeLineColor( int _shape, int index ) {
  switch( _shape ) {
   // shapes
   case rect: return null;
   case rect1: return null;
   // controls (text color)
   case button: return black;
   // charts (border color)
   default: return super.getShapeLineColor( _shape, index );
  }
 }

 @Override
 public Color getShapeFillColor( int _shape, int index ) {
  switch( _shape ) {
   case rect: return _rect_FillColor;
   case rect1: return _rect1_FillColor;
   case text: return
GOLD
;
   case text1: return red;
   case text2: return gray;
   case text3: return black;
   case poly: return null;
   case poly1: return null;
   case button: return controlDefault;
   default: return super.getShapeFillColor( _shape, index );
  }
 }

 @Override
 public int getShapeLineStyle( int _shape, int index ) {
  switch( _shape ) {
   case rect: return LINE_STYLE_SOLID;
   case rect1: return LINE_STYLE_SOLID;
   case poly: return LINE_STYLE_SOLID;
   case poly1: return LINE_STYLE_SOLID;
   default: return super.getShapeLineStyle( _shape, index );
  }
 }

 @Override
 public double getShapeLineWidth( int _shape, int index ) {
  switch( _shape ) {
   case poly: return 3;
   case poly1: return 3;
   default: return super.getShapeLineWidth( _shape, index );
  }
 }

 @Override
 public double getShapeWidth( int _shape, int index ) {
  switch( _shape ) {
   case rect: return 600;
   case rect1: return 600;
   case button: return 250;
   default: return super.getShapeWidth( _shape, index );
  }
 }

 @Override
 public double getShapeHeight( int _shape, int index ) {
  switch( _shape ) {
   case rect: return 600;
   case rect1: return 40;
   case button: return 30;
   default: return super.getShapeHeight( _shape, index );
  }
 }

 @Override
 public int getShapeNPoints( int _shape, int index ) {
  switch( _shape ) {
   case poly: return 7;
   case poly1: return 7;
   default: return super.getShapeNPoints( _shape, index );
  }
 }

 private static final double[] _poly_pointsDX = _poly_pointsDX_xjal();
 private static final double[] _poly_pointsDY = _poly_pointsDY_xjal();
 private static final double[] _poly1_pointsDX = _poly1_pointsDX_xjal();
 private static final double[] _poly1_pointsDY = _poly1_pointsDY_xjal();

 @Override
 public double[] getShapePointsDx( int _shape, int index ) {
  int npoints;
  double[] dx;
  switch( _shape ) {
   case poly: return _poly_pointsDX;
   case poly1: return _poly1_pointsDX;
   default: return super.getShapePointsDx( _shape, index );
  }
 }

 @Override
 public double[] getShapePointsDy( int _shape, int index ) {
  int npoints;
  double[] dy;
  switch( _shape ) {
   case poly: return _poly_pointsDY;
   case poly1: return _poly1_pointsDY;
   default: return super.getShapePointsDy( _shape, index );
  }
 }

@Override
 public Object getShapeText( int _shape, int index ) {
  switch( _shape ) {
   case button: return "Run the model and switch to Main view";
   case text: return "Susceptible";
   case text1: return "Infectious";
   case text2: return "Recovered";
   case text3: return "SIR";
```

```
    default: return super.getShapeText( _shape, index );
   }
  }

  @Override
  public Font getShapeFont( int _shape, int index ) {
   switch( _shape ) {
    case text: return _text_Font;
    case text1: return _text1_Font;
    case text2: return _text2_Font;
    case text3: return _text3_Font;
    case button: return _button_Font;
    default: return super.getShapeFont( _shape, index );
   }
  }
  @Override
  public int getShapeTextAlignment( int _shape, int index ) {
   switch( _shape ) {
    case text: return ALIGNMENT_LEFT;
    case text1: return ALIGNMENT_LEFT;
    case text2: return ALIGNMENT_LEFT;
    case text3: return ALIGNMENT_LEFT;
    default: return super.getShapeTextAlignment( _shape, index );
   }
  }

 @Override
  public int getWindowWidth() {
   return 600;
  }

  @Override
  public int getWindowHeight() {
   return 600;
  }

  /**
   * Applet class to run experiment as java applet
   */
  public static class Applet extends JApplet {

   Simulation ex;

   @Override
   public void init() {
    ex = new Simulation();
    ex.setup( this );
   }

   @Override
   public void destroy() {
    ex.close();
   }

  }

  @Override
  public void initDefaultRandomNumberGenerator(Engine engine) {
   engine.getDefaultRandomGenerator().setSeed( 1 );
  }

  @Override
  public Main createRoot( Engine engine ) {
   // Create the root object
   return new Main( engine, null, null );
  }

  @Override
  public void setupRootParameters( Main root, boolean
callOnChangeActions ) {
    double AverageIllnessDuration_xjal =
root._AverageIllnessDuration_DefaultValue_xjal();
    if (callOnChangeActions) {
     root.set_AverageIllnessDuration( AverageIllnessDuration_xjal );
    } else {
     root.AverageIllnessDuration = AverageIllnessDuration_xjal;
    }
    double ContactRate_xjal = root._ContactRate_DefaultValue_xjal();
    if (callOnChangeActions) {
     root.set_ContactRate( ContactRate_xjal );
    } else {
     root.ContactRate = ContactRate_xjal;
    }
    double InfectionProbability_xjal =
root._InfectionProbability_DefaultValue_xjal();
    if (callOnChangeActions) {
     root.set_InfectionProbability( InfectionProbability_xjal );
    } else {
     root.InfectionProbability = InfectionProbability_xjal;
    }
    int TotalPopulation_xjal = root._TotalPopulation_DefaultValue_xjal();
    if (callOnChangeActions) {
     root.set_TotalPopulation( TotalPopulation_xjal );
    } else {
     root.TotalPopulation = TotalPopulation_xjal;
    }
   }

   /**
    * Engine setup
    */
   @Override
   public void setupEngine(Engine engine) {
    engine.setATOL( 1.0E-5 );
    engine.setRTOL( 1.0E-5 );
    engine.setTTOL( 1.0E-5 );
    engine.setHTOL( 0.01 );
    engine.setSolverODE( Engine.SOLVER_ODE_EULER );
    engine.setSolverNAE( Engine.SOLVER_NAE_MODIFIED_NEWTON );
    engine.setSolverDAE( Engine.SOLVER_DAE_RK45_NEWTON );
    engine.setVMethods( 16032 );

    engine.setStartTime( 0.0 );
    engine.setTimeUnit( TIME_UNIT_DAY );
    engine.setRealTimeMode( false );
   }

   /**
    * Experiment setup
    */
   @Override
   public void setup( JApplet applet ) {
    setName( "SIR Agent Based : Simulation" );

    // Dynamic initialization of persistent elements
    // Setup presentation
    Presentation _p = new Presentation( this, applet != null ?
Presentation.MODE_APPLET :
       Presentation.MODE_APPLICATION, applet );
    _p.start();

    Panel _panel = _p.getPanel();
```

```
    ToolBar _tb = _p.getToolBar();
    StatusBar _sb = _p.getStatusBar();

    _panel.setZoomEnabled( false );
    _panel.setPanningEnabled( false );
    _panel.setFrameManagementBalance( 2.0 );
    _panel.setAntiAliasingEnabled( false );

    _sb.setSectionVisible( StatusBar.DATE, false );
    _sb.setSectionVisible( StatusBar.EPS, true );
    _sb.setSectionVisible( StatusBar.EXPERIMENT, false );
    _sb.setSectionVisible( StatusBar.FPS, true );
    _sb.setSectionVisible( StatusBar.MEMORY, true );
    _sb.setSectionVisible( StatusBar.SECONDS, true );
    _sb.setSectionVisible( StatusBar.SIMULATION, false );
    _sb.setSectionVisible( StatusBar.STATUS, true );
    _sb.setSectionVisible( StatusBar.STEP, true );
    _sb.setSectionVisible( StatusBar.TIME, false );
    _tb.setSectionVisible( ToolBar.ANIMATION, false );
    _tb.setSectionVisible( ToolBar.EXECUTION, true );
    _tb.setSectionVisible( ToolBar.FILE, false );
    _tb.setSectionVisible( ToolBar.NAVIGATION, false );
    _tb.setSectionVisible( ToolBar.TIME_SCALE, true );
    _tb.setSectionVisible( ToolBar.VIEW, false );
  }

}
```

Παράρτημα 3

UML – Διαγράμματα κλάσεων

predator
+Area: double
+HareNatality: double
+LynxNatality: double
+HareBirths: double
+HareDeaths: double
+LynxDeaths: double
+LynxBirths: double
+HareDensity: double
+Hares: double
+Lynx: double
~integrationManager_xjal: ActiveObjectIntegrationManager = new ActiveObjectIntegrationManager(2, 0, 5)
+HareInjectEvent: EventTimeout = new EventTimeout(this)
+LynxInjectEvent: EventTimeout = new EventTimeout(this)
+_plot_autoUpdateEvent_xjal: EventTimeout = new EventTimeout(this)
+_autoCreatedDS_xjal: EventTimeout = new EventTimeout(this)
-LynxMortality_Arguments_xjal: double[*] = LynxMortality_Arguments_xjal()
-LynxMortality_Values_xjal: double[*] = LynxMortality_Values_xjal()
+LynxMortality: TableFunction = new TableFunction(LynxMortality_Arguments_xjal, LynxMortality_Values_xjal, _ableFunction.INTERPOLATION_SPLINE, 1, TableFunction.OUTOFRANGE_NEAREST, 0, 0)
+ds_HareBirths:DataSet = new DataSet(100) { double_lastUpdateTime = Double.NaN; @Override public void update() { if(time() ==_lastUpdateTime){return;} add(time(), predator.this.HareBirths); _lastUpdateTime = time(); }}
+ds_HareDeaths:DataSet = new DataSet(100) { double_lastUpdateTime = Double.NaN; @Override public void update() { if(time() ==_lastUpdateTime){return;} add(time(), predator.this.HareDeaths); _lastUpdateTime = time(); }}
+ds_LynxBirths:DataSet = new DataSet(100) { double_lastUpdateTime = Double.NaN; @Override public void update() { if(time() ==_lastUpdateTime){return;} add(time(), predator.this.LynxBirths); _lastUpdateTime = time(); }}
+ds_LynxDeaths:DataSet = new DataSet(100) { double_lastUpdateTime = Double.NaN; @Override public void update() { if(time() ==_lastUpdateTime){return;} add(time(), predator.this. LynxDeaths); _lastUpdateTime = time(); }}
+ds_HareDensity:DataSet = new DataSet(100) { double_lastUpdateTime = Double.NaN; @Override public void update() { if(time() ==_lastUpdateTime){return;} add(time(), predator.this.HareDensity); _lastUpdateTime = time(); }}
+ds_Hares:DataSet = new DataSet(100) { double_lastUpdateTime = Double.NaN; @Override public void update() { if(time() ==_lastUpdateTime){return;} add(time(), predator.this.Hares); _lastUpdateTime = time(); }}
+ds_Lynx:DataSet = new DataSet(100) { double_lastUpdateTime = Double.NaN; @Override public void update() { if(time() ==_lastUpdateTime){return;} add(time(), predator.this.Lynx); _lastUpdateTime = time(); }}
+plot_expression0_dataSet_xjal: DataSet = new DataSet(200) {double_lastUpdate_X = Double.NaN; @Override public void update() { if (time() == _lastUpdateX) {return;} double_a = Hares; add(time(), a); _lastUpdateX = time(); }}
+plot_expression1_dataSet_xjal: DataSet = new DataSet(200) {double_lastUpdate_X = Double.NaN; @Override public void update() { if (time() == _lastUpdateX) {return;} double_a = Lynx * 100; add(time(), a); _lastUpdateX = time(); }}
~slider: int=1
~slider1: int=2
~slider4: int=3
~_button: int=4
~_button1: int=5
~poly: int=6
~poly1: int=7
~poly2: int=8
~text: int=9
~line: int=10
~text1: int=11
~plot: int=12
~_presentation_content: int[*] = new int[] { slider, slider1, sider4, _button, _button1, poly, poly1, poly2, text, line, text1, plot}
~presentation: int=0
~icon: int=-1
~_poly_pointsDX: double[*] = _poly_pointsDX_xjal()
~_poly_pointsDY: double[*] = _poly_pointsDY_xjal()
~_poly1_pointsDX: double[*] = _poly1_pointsDX_xjal()
~_poly1_pointsDY: double[*] = _poly1_pointsDY_xjal()
~_poly2_pointsDX: double[*] = _poly2_pointsDX_xjal()
~_poly2_pointsDY: double[*] = _poly2_pointsDY_xjal()
~button: ShapeButton
~button1: ShapeButton

<<create>> +predator(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection)
+_Area_DefaultValue_xjal(): double
+set_Area(Area: double)
~onChange_Area()
+_HareNatality_DefaultValue_xjal()
+set_HareNatality(HareNatality: double)
~onChange_HareNatality
+_LynxNatality_DefaultValue_xjal()
+set_LynxNatality(LynxNatality: double)
~onChange_LynxNatality
+getScalarPhaseVector(_d:double, _a:double)
+putScalarPhaseVector(_d:double, _a:double)
+assignInitialConditions()
+assignInitialConditions(_variableNameCode: int)
+_HareBirths_Formula(): double
+_Hares_Expression(): double
+_LynxBirths_Formula(): double
+_Lynx_Expression(): double
+_Hares_Initial_Value(): double
+_Lynx_Initial_Value(): double
+_HareDensity_Formula(): double
+_LynxDeaths_Formula(): double
+_HareDeaths_Formula(): double
+formulasExecute()
+formulasExecute(_variableNameCode: int)
+getScalarRightPart(_dr:double, _ar:double)
+getIntegrationManager(): ActiveObjectIntegrationManager
+getNameOf(_e: EventTimeout): String
+getModeOf(_e: EventTimeout): int
+getFirstOccurrenceTime(_e: EventTimeout): double
+evaluateTimeoutOf(_e: EventTimeout): double
+executeActionOf(_e: EventTimeout)
-_LynxMortality_Arguments_xjal(): double
-_LynxMortality_Values_xjal(): double
+LynxMortality(x: double): double
~_poly_pointsDX_xjal(): double
~_poly_pointsDY_xjal(): double
~_poly1_pointsDX_xjal(): double
~_poly1_pointsDY_xjal(): double
~_poly2_pointsDX_xjal(): double
~_poly2_pointsDY_xjal(): double
+executeShapeControlAction(_shape:int, index:int)
+executeShapeControlAction(_shape:int, index:int, value: double)
+getShapeControlValueType(_shape:int, index:int): int
+getShapeControlMinimum(_shape:int, index:int): double
+getShapeControlMaximum(_shape:int, index:int): double
+getShapeControlDefaultValueDouble(_shape:int, index:int): double
+getShapeGroupContent(_shape: int): int
+getNameOfShape(_shape: int): String
+getShapeType(_shape:int): int
+getShapeX(_shape:int, index:int): double
+getShapeY(_shape:int, index:int): double
+getShapeLineColor(_shape:int, index:int): Color
+getShapeFillColor(_shape:int, index:int): Color
+getShapeLineStyle(_shape:int, index:int): int
+getShapeLineDx(_shape:int, index:int): double
+getShapeLineDy(_shape:int, index:int): double
+getShapeWidth(_shape:int, index:int): double
+getShapeHeight(_shape:int, index:int): double
+getShapeNPoints(_shape:int, index:int): int
+getShapePointsDx(_shape:int, index:int): double
+getShapePointsDy(_shape:int, index:int): double
+getShapeText(_shape:int, index:int): Object
+getShapeFont(_shape:int, index:int): Font
+getShapeTextAlignment(_shape:int, index:int): int

+getShapeControlVertical(_shape:int, index:int): Boolean
+getShapeChartDataTitles(_shape:int, index:int): ArrayList
+getShapeChartProperties(_shape:int, index:int): Chart.Properties
+getShapeChartDataSets(_shape:int, index:int): ArrayList
+getShapeChartDataPlotOptions(_shape:int, index:int): ArrayList
+getPersistentShape(_shape:int): Object
+drawModelElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean)
+onClickModelAt(panel: Panel, x: double, y: double, clickCount: int, publicOnly: boolean): boolean
+create()
+start()
+onDestroy()

Κώδικας Κεφαλαίου 5 – Έργο Predator Prey

Model.java

```
package predator_prey;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import java.awt.geom.Arc2D;

public class Model extends ActiveObject
{
 // Parameters

 public double Area;

 /**
  * Returns default value for parameter <code>Area</code>.
  * <i>This method should not be called by user</i>
  */
 public double _Area_DefaultValue_xjal() {
  return 100;
 }

 public void set_Area(
double Area ) {
  if (Area == this.Area) {
   return;
  }
  this.Area = Area;
  onChange_Area();
  onChange();
 }

 void onChange_Area() {
 }

 public double HareNatality;

 /**
  * Returns default value for parameter <code>HareNatality</code>.
  * <i>This method should not be called by user</i>
  */
 public
double _HareNatality_DefaultValue_xjal() {
  return 1.25 ;
 }

 public void set_HareNatality(
double HareNatality ) {
  if (HareNatality == this.HareNatality) {
   return;
  }
  this.HareNatality = HareNatality;
  onChange_HareNatality();
  onChange();
 }

 void onChange_HareNatality() {
 }

 public double LynxNatality;

 /**
  * Returns default value for parameter <code>LynxNatality</code>.
  * <i>This method should not be called by user</i>
  */
 public
double _LynxNatality_DefaultValue_xjal() {
  return 0.25 ;
 }

 public void set_LynxNatality(
double LynxNatality ) {
  if (LynxNatality == this.LynxNatality) {
   return;  }
```

```
  this.LynxNatality = LynxNatality;
  onChange_LynxNatality();
  onChange();
 }

 void onChange_LynxNatality() {
 }

 // Plain Variables

 // Collection Variables

 // Flow/Auxiliary Variables
 public double HareBirths;
 public double HareDeaths;
 public double LynxDeaths;
 public double LynxBirths;
 public double HareDensity;

 // Stock Variables
 public double Hares;
 public double Lynx;

 /**
  * Writes model variables into given arrays
  */
 public void getScalarPhaseVector(double[] _d, double[] _a){
  _d[0] = Hares;
  _d[1] = Lynx;
 }

 /**
  * Writes given arrays to model variables
  */
 public void putScalarPhaseVector(double[] _d, double[] _a){
  Hares = _d[0];
  Lynx = _d[1];
 }

 public void assignInitialConditions(){
  Hares = _Hares_Initial_Value();
  HareDensity = _HareDensity_Formula();
  Lynx = _Lynx_Initial_Value();
  HareBirths = _HareBirths_Formula();
  HareDeaths = _HareDeaths_Formula();
  LynxDeaths = _LynxDeaths_Formula();
  LynxBirths = _LynxBirths_Formula();
 }

 public void assignInitialConditions( int _variableNameCode ) {
  switch (_variableNameCode) {
    case 69496039:
      Hares = _Hares_Initial_Value();
      break;
    case 842756668:
      HareDensity = _HareDensity_Formula();
      break;
    case 2383927:
      Lynx = _Lynx_Initial_Value();
      break;
    case 2051951552:
      HareBirths = _HareBirths_Formula();
      break;
    case 2105009323:
      HareDeaths = _HareDeaths_Formula();
      break;
    case -2103852138:
      LynxDeaths = _LynxDeaths_Formula();
      break;
    case 2138057387:
      LynxBirths = _LynxBirths_Formula();
      break;
    default:
      break;
   }
 }

 public double _HareBirths_Formula() {
   return
Hares * HareNatality ;
 }

 public double _Hares_Expression() {
   return
HareBirths - HareDeaths;
 }

 public double _Lynx_Expression() {
   return
LynxBirths - LynxDeaths ;
 }

 public double _LynxBirths_Formula() {
   return
Lynx * LynxNatality ;
 }

 public double _Hares_Initial_Value() {
   return 6000 ;
 }

 public double _Lynx_Initial_Value() {
   return 125 ;
 }

 public double _HareDensity_Formula() {
   return Hares / Area ;
 }

 public double _LynxDeaths_Formula() {
   return Lynx * LynxMortality(HareDensity) ;
 }

 public double _HareDeaths_Formula() {
   return HareDensity * Lynx ;
 }

 public void formulasExecute(){
   HareDensity = _HareDensity_Formula();
   HareBirths = _HareBirths_Formula();
   HareDeaths = _HareDeaths_Formula();
   LynxDeaths = _LynxDeaths_Formula();
   LynxBirths = _LynxBirths_Formula();
 }

 public void formulasExecute( int _variableNameCode ) {
   switch (_variableNameCode) {
     case 842756668:
       HareDensity = _HareDensity_Formula();
       break;
```

```
    case 2051951552:
      HareBirths = _HareBirths_Formula();
      break;
    case 2105009323:
      HareDeaths = _HareDeaths_Formula();
      break;
    case -2103852138:
      LynxDeaths = _LynxDeaths_Formula();
      break;
    case 2138057387:
      LynxBirths = _LynxBirths_Formula();
      break;
    default:
      break;
   }
 }

 public void getScalarRightPart( double[] _dr, double[] _ar ) {
  _dr[ 0 ] = _Hares_Expression();
  _dr[ 1 ] = _Lynx_Expression();
 }

 static ActiveObjectIntegrationManager integrationManager_xjal = new
ActiveObjectIntegrationManager( 2, 0, 5 );

 public ActiveObjectIntegrationManager getIntegrationManager(){
  return integrationManager_xjal;
 }

 // Events

 public EventTimeout HareInjectEvent = new EventTimeout(this);
 public EventTimeout LynxInjectEvent = new EventTimeout(this);
 public EventTimeout _plot_autoUpdateEvent_xjal = new
EventTimeout(this);
 public EventTimeout _autoCreatedDS_xjal = new EventTimeout(this);

 @Override
 public String getNameOf( EventTimeout _e ) {
   if( _e == HareInjectEvent ) return "HareInjectEvent";
   if( _e == LynxInjectEvent ) return "LynxInjectEvent";
  if ( _e == _plot_autoUpdateEvent_xjal ) return "plot auto update
event";
  if ( _e == _autoCreatedDS_xjal ) return "Auto-created DataSets auto
update event";
   return super.getNameOf( _e );
 }

 @Override
 public int getModeOf( EventTimeout _e ) {
  if ( _e == HareInjectEvent ) return EVENT_TIMEOUT_MODE_USER;
  if ( _e == LynxInjectEvent ) return EVENT_TIMEOUT_MODE_USER;
  if ( _e == _plot_autoUpdateEvent_xjal ) return
EVENT_TIMEOUT_MODE_CYCLIC;
  if ( _e == _autoCreatedDS_xjal ) return
EVENT_TIMEOUT_MODE_CYCLIC;
  return super.getModeOf( _e );
 }

 @Override
 public double getFirstOccurrenceTime( EventTimeout _e ) {
  if ( _e == HareInjectEvent ) return
time() ;
  if ( _e == LynxInjectEvent ) return
time() ;
  if (
     _e == _plot_autoUpdateEvent_xjal
     || _e == _autoCreatedDS_xjal
   ) return getEngine().getStartTime();
  return super.getFirstOccurrenceTime( _e );
 }

 @Override
 public double evaluateTimeoutOf( EventTimeout _e ) {
  if( _e == HareInjectEvent) return 0 ;
  if( _e == LynxInjectEvent) return 0 ;
  if ( _e == _plot_autoUpdateEvent_xjal ) return 0.5 ;
  if ( _e == _autoCreatedDS_xjal ) return 1 ;
  return super.evaluateTimeoutOf( _e );
 }

 @Override
 public void executeActionOf( EventTimeout _e ) {
  if ( _e == HareInjectEvent ) {

Hares += 1000; ;
    return ;
  }
  if ( _e == LynxInjectEvent ) {

Lynx += 50; ;
    return ;
  }
  if ( _e == _plot_autoUpdateEvent_xjal ) {
   // update data of plot
   _plot_expression0_dataSet_xjal.update();
   _plot_expression1_dataSet_xjal.update();
   return;
  }
  if ( _e == _autoCreatedDS_xjal ) {
   _ds_HareBirths.update();
   _ds_HareDeaths.update();
   _ds_LynxDeaths.update();
   _ds_LynxBirths.update();
   _ds_HareDensity.update();
   _ds_Hares.update();
   _ds_Lynx.update();
  }
  super.executeActionOf( _e );
 }
 // Table Functions

 private final static double[] _LynxMortality_Arguments_xjal =
_LynxMortality_Arguments_xjal();
 private final static double[] _LynxMortality_Arguments_xjal() {
  return new double[] { 0.0, 10.0, 20.0, 30.0, 40.0, 50.0, 60.0, 70.0,
80.0, 90.0, 100.0, };
 }
 private final static double[] _LynxMortality_Values_xjal =
_LynxMortality_Values_xjal();
 private final static double[] _LynxMortality_Values_xjal() {
  return new double[] { 0.5, 0.45, 0.4, 0.35, 0.3, 0.25, 0.2, 0.15, 0.1,
0.05, 0.005, };
 }

 /**
  * LynxMortality Table Function
  */
 public static final TableFunction LynxMortality = new TableFunction(
_LynxMortality_Arguments_xjal, _LynxMortality_Values_xjal,
     TableFunction.INTERPOLATION_SPLINE,                    1,
TableFunction.OUTOFRANGE_NEAREST,
     0.0 );
```

```
 public static final double LynxMortality( double x ) { return
LynxMortality.get( x ); }
 /**
  * Auto-created data set(s) for HareBirths
  */
 public DataSet _ds_HareBirths = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), Model.this.HareBirths );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for HareDeaths
  */
 public DataSet _ds_HareDeaths = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), Model.this.HareDeaths );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for LynxDeaths
  */
 public DataSet _ds_LynxDeaths = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), Model.this.LynxDeaths );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for LynxBirths
  */
 public DataSet _ds_LynxBirths = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), Model.this.LynxBirths );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for HareDensity
  */
 public DataSet _ds_HareDensity = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), Model.this.HareDensity );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Hares
  */
 public DataSet _ds_Hares = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), Model.this.Hares );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Lynx
  */
 public DataSet _ds_Lynx = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), Model.this.Lynx );
   _lastUpdateTime = time();
  }
 };
 public DataSet _plot_expression0_dataSet_xjal = new DataSet( 200 ) {
  double _lastUpdateX = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateX ) { return; }
   double _a =
Hares
;
   add( time(), _a );
   _lastUpdateX = time();
  }
 };
 public DataSet _plot_expression1_dataSet_xjal = new DataSet( 200 ) {
  double _lastUpdateX = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateX ) { return; }
   double _a =
Lynx * 100
;
   add( time(), _a );
   _lastUpdateX = time();
  }
 };
 static final Font _button_Font = new Font("Dialog", 0, 12 );
 static final Font _button1_Font = _button_Font;
 static final Font _text_Font = new Font("SansSerif", 1, 26 );
 static final Font _text1_Font = new Font("SansSerif", 0, 11 );
 static final int slider = 1;
 static final int slider1 = 2;
 static final int slider4 = 3;
 static final int _button = 4;
 static final int _button1 = 5;
 static final int poly = 6;
 static final int poly1 = 7;
 static final int poly2 = 8;
 static final int text = 9;
 static final int line = 10;
 static final int text1 = 11;
 static final int plot = 12;


 /**
  * Top-level presentation group content
  */
```

```
 static final int[] _presentation_content = new int[] {
  slider,
  slider1,
  slider4,
  _button,
  _button1,
  poly,
  poly1,
  poly2,
  text,
  line,
  text1,
  plot,
 };

 /**
  * Top-level presentation group id
  */
 static final int presentation = 0;

 /**
  * Top-level icon group id
  */
 static final int icon = -1;

 static final double[] _poly_pointsDX_xjal() {
  return new double[] { 0, -10, -10,  };
 }
 static final double[] _poly_pointsDY_xjal() {
  return new double[] { 0, 0, -80,  };
 }
 static final double[] _poly1_pointsDX_xjal() {
  return new double[] { 0, 0, 10,  };
 }
 static final double[] _poly1_pointsDY_xjal() {
  return new double[] { 0, -80, -80,  };
 }
 static final double[] _poly2_pointsDX_xjal() {
  return new double[] { 0, 10, 10,  };
 }
 static final double[] _poly2_pointsDY_xjal() {
  return new double[] { 0, 0, -80,  };
 }

 @Override
 public void executeShapeControlAction( int _shape, int index ) {
  switch( _shape ) {
   case _button: {
HareInjectEvent.restart(0);
;}
    break;
   case _button1: {
LynxInjectEvent.restart(0);
;}
    break;
   default:
    super.executeShapeControlAction( _shape, index );
    break;
  }
 }

 @Override
 public void executeShapeControlAction( int _shape, int index, double
value ) {
  switch( _shape ) {
   case slider:
    set_HareNatality( value );
    break;
   case slider1:
    set_LynxNatality( value );
    break;
   case slider4:
    set_Area( value );
    break;
   default:
    super.executeShapeControlAction( _shape, index, value );
    break;
  }
 }

 @Override
 public int getShapeControlValueType( int _shape, int index ) {
  switch( _shape ) {
   case slider: return ShapeControl.TYPE_DOUBLE;
   case slider1: return ShapeControl.TYPE_DOUBLE;
   case slider4: return ShapeControl.TYPE_DOUBLE;
   default: return super.getShapeControlValueType( _shape, index );
  }
 }

 @Override
 public double getShapeControlMinimum( int _shape, int index ) {
  switch( _shape ) {
   case slider: return 0.25 ;
   case slider1: return 0.1 ;
   case slider4: return 20 ;
   default: return super.getShapeControlMinimum( _shape, index );
  }
 }

 @Override
 public double getShapeControlMaximum( int _shape, int index ) {
  switch( _shape ) {
   case slider: return 3 ;
   case slider1: return 0.5 ;
   case slider4: return 500 ;
   default: return super.getShapeControlMaximum( _shape, index );
  }
 }

 @Override
 public double getShapeControlDefaultValueDouble( int _shape, int index
) {
  switch(_shape) {
   case slider: return HareNatality ;
   case slider1: return LynxNatality ;
   case slider4: return Area ;
   default:  return  super.getShapeControlDefaultValueDouble(  _shape,
index );
  }
 }

 @Override
 public int[] getShapeGroupContent( int _shape ){
  switch( _shape ) {
   case presentation: return _presentation_content;
   default: return super.getShapeGroupContent( _shape );
  }
 }

 @Override
 public String getNameOfShape( int _shape ) {
```

```
  switch( _shape ) {
    case poly: return "poly";
    case poly1: return "poly1";
    case poly2: return "poly2";
    case text: return "text";
    case line: return "line";
    case text1: return "text1";
    case slider: return "slider";
    case slider1: return "slider1";
    case slider4: return "slider4";
    case plot: return "plot";
    default: return super.getNameOfShape( _shape );
  }
}

@Override
public int getShapeType( int _shape ) {
  switch( _shape ) {
    case presentation: return SHAPE_GROUP;
    case icon: return SHAPE_GROUP;
    case poly: return SHAPE_POLYLINE;
    case poly1: return SHAPE_POLYLINE;
    case poly2: return SHAPE_POLYLINE;
    case text: return SHAPE_TEXT;
    case line: return SHAPE_LINE;
    case text1: return SHAPE_TEXT;
    case slider: return SHAPE_SLIDER;
    case slider1: return SHAPE_SLIDER;
    case slider4: return SHAPE_SLIDER;
    case plot: return SHAPE_CHART_TIME_PLOT;
    default: return super.getShapeType( _shape );
  }
}

@Override
public double getShapeX( int _shape, int index ) {
  switch( _shape ) {
    case poly: return 190;
    case poly1: return 180;
    case poly2: return 610;
    case text: return 50;
    case line: return 0;
    case text1: return 50;
    case slider: return 150;
    case slider1: return 150;
    case slider4: return 620;
    case plot: return 10;
    default: return super.getShapeX( _shape, index );
  }
}

@Override
public double getShapeY( int _shape, int index ) {
  switch( _shape ) {
    case poly: return 200;
    case poly1: return 350;
    case poly2: return 240;
    case text: return 10;
    case line: return 40;
    case text1: return 50;
    case slider: return 120;
    case slider1: return 270;
    case slider4: return 160;
    case plot: return 420;
    default: return super.getShapeY( _shape, index );
  }
}

@Override
public Color getShapeLineColor( int _shape, int index ) {
  switch( _shape ) {
    // shapes
    case poly: return steelBlue;
    case poly1: return steelBlue;
    case poly2: return steelBlue;
    case line: return steelBlue;
    // controls (text color)
    // charts (border color)
    case plot: return null;
    default: return super.getShapeLineColor( _shape, index );
  }
}

@Override
public Color getShapeFillColor( int _shape, int index ) {
  switch( _shape ) {
    case poly: return null;
    case poly1: return null;
    case poly2: return null;
    case text: return slateGray;
    case text1: return slateGray;
    case slider: return transparent;
    case slider1: return transparent;
    case slider4: return transparent;
    default: return super.getShapeFillColor( _shape, index );
  }
}

@Override
public int getShapeLineStyle( int _shape, int index ) {
  switch( _shape ) {
    case poly: return LINE_STYLE_SOLID;
    case poly1: return LINE_STYLE_SOLID;
    case poly2: return LINE_STYLE_SOLID;
    case line: return LINE_STYLE_SOLID;
    default: return super.getShapeLineStyle( _shape, index );
  }
}

@Override
public double getShapeLineDx( int _shape, int index ) {
  switch( _shape ) {
    case line: return 800;
    default: return super.getShapeLineDx( _shape, index );
  }
}

@Override
public double getShapeLineDy( int _shape, int index ) {
  switch( _shape ) {
    case line: return 0;
    default: return super.getShapeLineDy( _shape, index );
  }
}

@Override
public double getShapeWidth( int _shape, int index ) {
  switch( _shape ) {
    case slider: return 30;
    case slider1: return 30;
```

```
    case slider4: return 30;
    case plot: return 760;
    default: return super.getShapeWidth( _shape, index );
  }
 }

 @Override
 public double getShapeHeight( int _shape, int index ) {
  switch( _shape ) {
    case slider: return 80;
    case slider1: return 80;
    case slider4: return 80;
    case plot: return 170;
    default: return super.getShapeHeight( _shape, index );
  }
 }

 @Override
 public int getShapeNPoints( int _shape, int index ) {
  switch( _shape ) {
    case poly: return 3;
    case poly1: return 3;
    case poly2: return 3;
    default: return super.getShapeNPoints( _shape, index );
  }
 }

 private static final double[] _poly_pointsDX = _poly_pointsDX_xjal();
 private static final double[] _poly_pointsDY = _poly_pointsDY_xjal();
 private static final double[] _poly1_pointsDX = _poly1_pointsDX_xjal();
 private static final double[] _poly1_pointsDY = _poly1_pointsDY_xjal();
 private static final double[] _poly2_pointsDX = _poly2_pointsDX_xjal();
 private static final double[] _poly2_pointsDY = _poly2_pointsDY_xjal();

 @Override
 public double[] getShapePointsDx( int _shape, int index ) {
  int npoints;
  double[] dx;
  switch( _shape ) {
    case poly: return _poly_pointsDX;
    case poly1: return _poly1_pointsDX;
    case poly2: return _poly2_pointsDX;
    default: return super.getShapePointsDx( _shape, index );
  }
 }

 @Override
 public double[] getShapePointsDy( int _shape, int index ) {
  int npoints;
  double[] dy;
  switch( _shape ) {
    case poly: return _poly_pointsDY;
    case poly1: return _poly1_pointsDY;
    case poly2: return _poly2_pointsDY;
    default: return super.getShapePointsDy( _shape, index );
  }
 }



 @Override
 public Object getShapeText( int _shape, int index ) {
  switch( _shape ) {
    case text: return "Predator Prey";
    case text1: return "* Use sliders to change parameters on-the-fly\r\n* Use buttons to inject hares or lynx\r\n* Click on variables or datasets to inspect";
    default: return super.getShapeText( _shape, index );
  }
 }



 @Override
 public Font getShapeFont( int _shape, int index ) {
  switch( _shape ) {
    case text: return _text_Font;
    case text1: return _text1_Font;
    default: return super.getShapeFont( _shape, index );
  }
 }

 @Override
 public int getShapeTextAlignment( int _shape, int index ) {
  switch( _shape ) {
    case text: return ALIGNMENT_LEFT;
    case text1: return ALIGNMENT_LEFT;
    default: return super.getShapeTextAlignment( _shape, index );
  }
 }

 @Override
 public boolean isShapeControlVertical( int _shape, int index ) {
  switch( _shape ) {
    case slider: return true;
    case slider1: return true;
    case slider4: return true;
    default: return super.isShapeControlVertical( _shape, index );
  }
 }



 @Override
 public ArrayList<String> getShapeChartDataTitles( int _shape, int index){
    ArrayList< String > _items;
  switch ( _shape ) {
    case plot:
      _items = new ArrayList< String >(2);
      _items.add( "Number of Hares" );
      _items.add( "Number of Lynx x100" );
      return _items;
    default: return super.getShapeChartDataTitles( _shape, index );
  }
 }

 @Override
 public Chart.Properties getShapeChartProperties( int _shape, int index )
{
   Chart.Properties _prop;
   switch( _shape ) {
    case plot:
      _prop = new Chart.Properties();
      _prop.picOffsetX = 80;
      _prop.picOffsetY = 20;
      _prop.picWidth = 660;
      _prop.picHeight = 100;
      _prop.picBackgroundColor = lavender;
      _prop.picBorderColor = black;
      _prop.gridLineColor = darkGray;
      _prop.gridTextColor = darkGray;
      _prop.gridPositionX = Chart.GRID_DEFAULT;
```

```
      _prop.gridPositionY = Chart.GRID_DEFAULT;
      _prop.legendTextColor = black;
      _prop.legendPos = Chart.SOUTH;
      _prop.legendSize = 30;
      _prop.maximumX = 100 ;
      _prop.scaleTypeY = Chart.SCALE_AUTO;
      return _prop;
    default: return super.getShapeChartProperties( _shape, index );
   }
  }

  @Override
  public ArrayList< DataSet > getShapeChartDataSets( int _shape, int
index ) {
    DataSet _item;
    ArrayList< DataSet > _items;
    switch (_shape) {
     case plot:
      _items = new ArrayList< DataSet >(2);
      _items.add( _plot_expression0_dataSet_xjal );
      _items.add( _plot_expression1_dataSet_xjal );
      return _items;
    default: return super.getShapeChartDataSets( _shape, index );
   }
  }

  @Override
  public ArrayList< Chart2DPlot.Appearance >
getShapeChartDataPlotOptions( int _shape, int index ) {
    ArrayList< Chart2DPlot.Appearance > _items;
    switch (_shape) {
     case plot:
      _items = new ArrayList< Chart2DPlot.Appearance >( 2);
      _items.add( new Chart2DPlot.Appearance( slateGray, true,
Chart.INTERPOLATION_LINEAR, 2, Chart.POINT_NONE ) );
      _items.add( new Chart2DPlot.Appearance( red, true,
Chart.INTERPOLATION_LINEAR, 2, Chart.POINT_NONE ) );
      return _items;
    default: return super.getShapeChartDataPlotOptions( _shape, index
);
   }
  }

  ShapeButton button;
  ShapeButton button1;

  // Static initialization of persistent elements
  {
   button = new ShapeButton(
                                    Model.this, true, 310, 100,
                                    120, 20,
        controlDefault, black,
        _button_Font,
                                    "Inject 1000 Hares"
   ) {

    @Override
    public void action(){
     executeShapeControlAction( _button, 0 );
    }
   };
   button1 = new ShapeButton(
                                    Model.this, true, 310, 360,
                                    120, 20,
        controlDefault, black,
        _button1_Font,
                                    "Inject 50 Lynx"
   ) {

    @Override
    public void action(){
     executeShapeControlAction( _button1, 0 );
    }
   };
  }

  @Override
  public Object getPersistentShape( int _shape ) {
   switch(_shape){

    case _button: return button;
    case _button1: return button1;
    default: return null;
   }
  }

  static final Arc2D.Double arc_PD_1170078444259 = new
Arc2D.Double();
  static { setupPlainDependencyArc( arc_PD_1170078444259, 420,170,
280, 170, 30.76072119832962 );}
  static final Arc2D.Double arc_PD_1170078444263 = new
Arc2D.Double();
  static { setupPlainDependencyArc( arc_PD_1170078444263, 420,310,
550, 170, 34.30296241526213 );}
  static final Arc2D.Double arc_PD_1170078444268 = new
Arc2D.Double();
  static { setupPlainDependencyArc( arc_PD_1170078444268, 420,310,
550, 310, 28.0 );}
  static final Arc2D.Double arc_PD_1170078444274 = new
Arc2D.Double();
  static { setupPlainDependencyArc( arc_PD_1170078444274, 420,310,
280, 310, -29.18282745828124 );}
  static final Arc2D.Double arc_PD_1170078444280 = new
Arc2D.Double();
  static { setupPlainDependencyArc( arc_PD_1170078444280, 490,240,
550, 170, -11.038483914143029 );}
  static final Arc2D.Double arc_PD_1170078444281 = new
Arc2D.Double();
  static { setupPlainDependencyArc( arc_PD_1170078444281, 490,240,
550, 310, 10.0 );}
  static final Arc2D.Double arc_PD_1170078444282 = new
Arc2D.Double();
  static { setupPlainDependencyArc( arc_PD_1170078444282, 420,170,
490, 240, 10.0 );}

  @Override
  public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
    if (!_publicOnly) {
     drawPlainDependencyArc( _panel, _g, arc_PD_1170078444259, null
);
    }
    if (!_publicOnly) {
     drawPlainDependencyArc( _panel, _g, arc_PD_1170078444263, null
);
    }
    if (!_publicOnly) {
     drawPlainDependencyArc( _panel, _g, arc_PD_1170078444268, null
);
    }
    if (!_publicOnly) {
     drawPlainDependencyArc( _panel, _g, arc_PD_1170078444274, null
);
    }
    if (!_publicOnly) {
     drawPlainDependencyArc( _panel, _g, arc_PD_1170078444280, null
);
    }
```

```
  if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1170078444281, null
);
  }
  if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1170078444282, null
);
  }
  if (!_publicOnly) {
    drawPlainDependencyLine( _panel, _g, 590, 240, 510, 240, null );
  }
  if (!_publicOnly) {
    drawPlainDependencyLine( _panel, _g, 209, 196, 261, 177, null );
  }
  if (!_publicOnly) {
    drawPlainDependencyLine( _panel, _g, 209, 274, 262, 301, null );
  }
  if (!_publicOnly) {
    drawFlowStockDependencyLine( _panel, _g, 295, 170, 402, 170, null
);
  }
  if (!_publicOnly) {
    drawFlowStockDependencyLine( _panel, _g, 435, 170, 532, 170, null
);
  }
  if (!_publicOnly) {
    drawFlowStockDependencyLine( _panel, _g, 435, 310, 532, 310, null
);
  }
  if (!_publicOnly) {
    drawFlowStockDependencyLine( _panel, _g, 295, 310, 402, 310, null
);
  }
  if (!_publicOnly) {
    drawEvent( _panel, _g, 420, 140, 10, 0, "HareInjectEvent",
HareInjectEvent );
  }
  if (!_publicOnly) {
    drawEvent( _panel, _g, 420, 340, 10, 0, "LynxInjectEvent",
LynxInjectEvent );
  }
  if (!_publicOnly) {
    drawParameter( _panel, _g, 600, 240, -40, -20, "Area", Area, false,
false );
  }
  if (!_publicOnly) {
    drawParameter( _panel, _g, 200, 200, 15, 0, "HareNatality",
HareNatality, false, false );
  }
  if (!_publicOnly) {
    drawParameter( _panel, _g, 200, 270, 15, -10, "LynxNatality",
LynxNatality, false, false );
  }
  if (!_publicOnly) {
    drawStock( _panel, _g, 420, 170, -30, 15, "Hares", Hares, false );
  }
  if (!_publicOnly) {
    drawStock( _panel, _g, 420, 310, -30, -30, "Lynx", Lynx, false );
  }
  if (!_publicOnly) {
    drawFlow( _panel, _g, 280, 170, -55, -25, "HareBirths", HareBirths,
false, false );
  }
  if (!_publicOnly) {
    drawFlow( _panel, _g, 550, 170, -10, -25, "HareDeaths",
HareDeaths, false, false );
  }
  if (!_publicOnly) {
    drawFlow( _panel, _g, 550, 310, -10, 20, "LynxDeaths", LynxDeaths,
false, false );
  }
  if (!_publicOnly) {
    drawFlow( _panel, _g, 280, 310, -55, 20, "LynxBirths", LynxBirths,
false, false );
  }
  if (!_publicOnly) {
    drawAuxiliaryVariable( _panel, _g, 490, 240, -75, -5, "HareDensity",
HareDensity, false, false, false );
  }
  if (!_publicOnly) {
    drawTableFunction( _panel, _g, 560, 280, 10, 0, "LynxMortality");
  }
 }

 @Override
 public boolean onClickModelAt( Panel panel, double x, double y, int
clickCount, boolean publicOnly ) {
  if( !publicOnly && modelElementContains(x, y, 600, 240) ) {
   panel.addInspect( 600, 240, this, "Area" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 200, 200) ) {
   panel.addInspect( 200, 200, this, "HareNatality" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 200, 270) ) {
   panel.addInspect( 200, 270, this, "LynxNatality" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 280, 170) ) {
   panel.addInspect( 280, 170, this, "HareBirths" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 550, 170) ) {
   panel.addInspect( 550, 170, this, "HareDeaths" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 550, 310) ) {
   panel.addInspect( 550, 310, this, "LynxDeaths" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 280, 310) ) {
   panel.addInspect( 280, 310, this, "LynxBirths" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 490, 240) ) {
   panel.addInspect( 490, 240, this, "HareDensity" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 420, 170) ) {
   panel.addInspect( 420, 170, this, "Hares" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 420, 310) ) {
   panel.addInspect( 420, 310, this, "Lynx" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 420, 140) ) {
   panel.addInspect( 420, 140, this, "HareInjectEvent" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 420, 340) ) {
   panel.addInspect( 420, 340, this, "LynxInjectEvent" );
   return true;
  }
```

```
    if( !publicOnly && modelElementContains(x, y, 560, 280) ) {
      panel.addInspect( 560, 280, this, "LynxMortality" );
      return true;
    }
    return false;
  }

  /**
   * Constructor
   */
  public Model( Engine engine, ActiveObject owner,
ActiveObjectCollection<? extends Model> collection ) {
    super( engine, owner, collection );
    // Registering in Engine continuous part
    getEngine().registerActiveObjectWithEquations( this );
  }

  @Override
  public void create() {
    // Dynamic initialization of persistent elements
                // Port connectors with non-replicated objects
    // Creating replicated embedded objects
    assignInitialConditions();
    onCreate();
  }

  @Override
  public void start() {
    HareInjectEvent.start();
    LynxInjectEvent.start();
    _plot_autoUpdateEvent_xjal.start();
    _autoCreatedDS_xjal.start();
    onStartup();
  }

  public void onDestroy() {
    super.onDestroy();
    HareInjectEvent.onDestroy();
    LynxInjectEvent.onDestroy();
    _plot_autoUpdateEvent_xjal.onDestroy();
    _autoCreatedDS_xjal.onDestroy();
    // Unregistering in Engine continuous part
    getEngine().unregisterActiveObjectWithEquations( this );
  }
}
```

Simulation.java

```
package predator_prey;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import javax.swing.JApplet;

public class Simulation extends ExperimentSimulation<Model> {
  static final Font _button_Font = new Font("Dialog", 0, 11 );
  static final Font _text2_Font = new Font("SansSerif", 1, 10 );
  static final Font _text3_Font = new Font("SansSerif", 0, 9 );
  static final Font _Text_Font = new Font("SansSerif", 0, 11 );
  static final Font _text_Font = new Font("SansSerif", 1, 26 );
  static final int button = 1;
  static final int _text2 = 2;
  static final int _text3 = 3;
  static final int _image = 4;
  static final int _Text = 5;
  static final int text = 6;
  static final int line = 7;


  /**
   * Top-level presentation group content
   */
  static final int[] _presentation_content = new int[] {
    button,
    _text2,
    _text3,
    _image,
    _Text,
    text,
    line,
  };

  /**
   * Top-level presentation group id
   */
  static final int presentation = 0;

  /**
```

```
  * Top-level icon group id
  */
 static final int icon = -1;

 @Override
 public void executeShapeControlAction( int _shape, int index ) {
  switch( _shape ) {
   case button: {
run();

getEngine().getPresentation().setPresentable( getEngine().getRoot() );
;}
    break;
   default:
    super.executeShapeControlAction( _shape, index );
    break;
  }
 }

 @Override
 public int[] getShapeGroupContent( int _shape ){
   switch( _shape ) {
    case presentation: return _presentation_content;
    default: return super.getShapeGroupContent( _shape );
   }
 }

 @Override
 public String getNameOfShape( int _shape ) {
  switch( _shape ) {
   case text: return "text";
   case line: return "line";
   case button: return "button";
   default: return super.getNameOfShape( _shape );
  }
 }

 @Override
 public int getShapeType( int _shape ) {
  switch( _shape ) {
   case presentation: return SHAPE_GROUP;
   case icon: return SHAPE_GROUP;
   case text: return SHAPE_TEXT;
   case line: return SHAPE_LINE;
   case button: return SHAPE_BUTTON;
   default: return super.getShapeType( _shape );
  }
 }

 @Override
 public double getShapeX( int _shape, int index ) {
  switch( _shape ) {
   case text: return 50;
   case line: return 0;
   case button: return 50;
   default: return super.getShapeX( _shape, index );
  }
 }

 @Override
 public double getShapeY( int _shape, int index ) {
  switch( _shape ) {
   case text: return 10;
   case line: return 40;
   case button: return 110;
   default: return super.getShapeY( _shape, index );
  }
 }

 @Override
 public Color getShapeLineColor( int _shape, int index ) {
  switch( _shape ) {
   // shapes
   case line: return steelBlue;
   // controls (text color)
   case button: return black;
   // charts (border color)
   default: return super.getShapeLineColor( _shape, index );
  }
 }

 @Override
 public Color getShapeFillColor( int _shape, int index ) {
  switch( _shape ) {
   case text: return slateGray;
   case button: return controlDefault;
   default: return super.getShapeFillColor( _shape, index );
  }
 }

 @Override
 public int getShapeLineStyle( int _shape, int index ) {
  switch( _shape ) {
   case line: return LINE_STYLE_SOLID;
   default: return super.getShapeLineStyle( _shape, index );
  }
 }

 @Override
 public double getShapeLineDx( int _shape, int index ) {
  switch( _shape ) {
   case line: return 800;
   default: return super.getShapeLineDx( _shape, index );
  }
 }

 @Override
 public double getShapeLineDy( int _shape, int index ) {
  switch( _shape ) {
   case line: return 0;
   default: return super.getShapeLineDy( _shape, index );
  }
 }

 @Override
 public double getShapeWidth( int _shape, int index ) {
  switch( _shape ) {
   case button: return 110;
   default: return super.getShapeWidth( _shape, index );
  }
 }

 @Override
 public double getShapeHeight( int _shape, int index ) {
  switch( _shape ) {
   case button: return 30;
   default: return super.getShapeHeight( _shape, index );
  }
 }
 @Override
```

```
 public Object getShapeText( int _shape, int index ) {
  switch( _shape ) {
   case button: return "Run the model";
   case text: return "Predator Prey";
   default: return super.getShapeText( _shape, index );
  }
 }

 @Override
 public Font getShapeFont( int _shape, int index ) {
  switch( _shape ) {
   case text: return _text_Font;
   case button: return _button_Font;
   default: return super.getShapeFont( _shape, index );
  }
 }

 @Override
 public int getShapeTextAlignment( int _shape, int index ) {
  switch( _shape ) {
   case text: return ALIGNMENT_LEFT;
   default: return super.getShapeTextAlignment( _shape, index );
  }
 }

 @Override
 public boolean isShapeControlEnabled( int _shape, int index ) {
  switch( _shape ) {
   case button: return
getEngine().getState() == Engine.IDLE
;
   default: return super.isShapeControlEnabled( _shape, index );
  }
 }

 ShapeText text2;
 ShapeText text3;
 ShapeImage image;
 ShapeText Text;

 // Static initialization of persistent elements
 {
  text2 = new ShapeText(
    true,750, 40, 0.0,
    slateGray,"System Dynamics Classics",
    _text2_Font, ALIGNMENT_RIGHT
  );
  text3 = new ShapeText(
    true,91, 516, 0.0,
    black,"This AnyLogic™ model is\r\n© 1992-2007 XJ
Technologies\r\nwww.anylogic.com\r\n",
    _text3_Font, ALIGNMENT_LEFT
  );
  image = new ShapeImage(
                  Simulation.this, true, 51, 483, 0.0,
                        116, 40, "/predator_prey/",
                        new String[]{"xjlogo.png",}

  );
  Text = new ShapeText(
    true,50, 60, 0.0,
    slateGray,"The predator prey problem is a simulation that attempts
to predict\r\nthe relationship in populations between a population of lynx
and \r\nhares on isolated area.\r\n",
    _Text_Font, ALIGNMENT_LEFT
  );
 }

 @Override public Object getPersistentShape( int _shape ) {
  switch(_shape){

   case _text2: return text2;
   case _text3: return text3;
   case _image: return image;
   case _Text: return Text;
   default: return null;
  }
 }

 @Override public int getWindowWidth() {
  return 800;
 }

 @Override public int getWindowHeight() {
  return 600;
 }

 /**
  * Applet class to run experiment as java applet
  */
 public static class Applet extends JApplet {

  Simulation ex;

  @Override
  public void init() {
   ex = new Simulation();
   ex.setup( this );
  }

  @Override
  public void destroy() {
   ex.close();
  }
 }

 @Override
 public void initDefaultRandomNumberGenerator(Engine engine) {
  engine.getDefaultRandomGenerator().setSeed( 1 );
 }

 @Override
 public Model createRoot( Engine engine ) {
  // Create the root object
  return new Model( engine, null, null );
 }

 @Override public void setupRootParameters( Model root, boolean
callOnChangeActions ) {
  double Area_xjal = root._Area_DefaultValue_xjal();
  if (callOnChangeActions) {
   root.set_Area( Area_xjal );
  } else {
   root.Area = Area_xjal;
  }
  double HareNatality_xjal = root._HareNatality_DefaultValue_xjal();
  if (callOnChangeActions) {
   root.set_HareNatality( HareNatality_xjal );
  } else {
   root.HareNatality = HareNatality_xjal;
  }
  double LynxNatality_xjal = root._LynxNatality_DefaultValue_xjal();
  if (callOnChangeActions) {
   root.set_LynxNatality( LynxNatality_xjal );
```

```
    } else {
      root.LynxNatality = LynxNatality_xjal;
    }
  }

  /**
   * Engine setup
   */
  @Override public void setupEngine(Engine engine) {
    engine.setATOL( 1.0E-4 );
    engine.setRTOL( 1.0E-4 );
    engine.setTTOL( 1.0E-5 );
    engine.setHTOL( 0.01 );
    engine.setSolverODE( Engine.SOLVER_ODE_EULER );
    engine.setSolverNAE( Engine.SOLVER_NAE_MODIFIED_NEWTON );
    engine.setSolverDAE( Engine.SOLVER_DAE_RK45_NEWTON );
    engine.setVMethods( 16032 );

    engine.setStartTime( 0.0 );
    engine.setTimeUnit( TIME_UNIT_DAY );
    engine.setRealTimeMode( true );
    engine.setRealTimeScale( 4.0 );
  }

  /**
   * Experiment setup
   */
  @Override
  public void setup( JApplet applet ) {
    setName( "Predator Prey : Simulation" );

    // Dynamic initialization of persistent elements
    // Setup presentation
    Presentation _p = new Presentation( this, applet != null ?
Presentation.MODE_APPLET :
        Presentation.MODE_APPLICATION, applet );
    _p.start();

    Panel _panel = _p.getPanel();
    ToolBar _tb = _p.getToolBar();
    StatusBar _sb = _p.getStatusBar();
    _panel.setZoomEnabled( false );
    _panel.setPanningEnabled( false );
    _panel.setFrameManagementBalance( 2.0 );
    _sb.setSectionVisible( StatusBar.DATE, false );
    _sb.setSectionVisible( StatusBar.EPS, false );
    _sb.setSectionVisible( StatusBar.EXPERIMENT, false );
    _sb.setSectionVisible( StatusBar.FPS, false );
    _sb.setSectionVisible( StatusBar.MEMORY, true );
    _sb.setSectionVisible( StatusBar.SECONDS, true );
    _sb.setSectionVisible( StatusBar.SIMULATION, false );
    _sb.setSectionVisible( StatusBar.STATUS, true );
    _sb.setSectionVisible( StatusBar.STEP, false );
    _sb.setSectionVisible( StatusBar.TIME, true );
    _tb.setSectionVisible( ToolBar.ANIMATION, false );
    _tb.setSectionVisible( ToolBar.EXECUTION, true );
    _tb.setSectionVisible( ToolBar.FILE, false );
    _tb.setSectionVisible( ToolBar.NAVIGATION, false );
    _tb.setSectionVisible( ToolBar.TIME_SCALE, true );
    _tb.setSectionVisible( ToolBar.VIEW, false );
  }
}
```

Παράρτημα 4

UML - Διαγράμματα κλάσεων

Hare
+Width: int +CellWidth: int +NumberPerBirth: int +LifeExpectancy: double +Natality: double +MaxPerCell: int +cell: int +HaveBabies: EventRate = new EventRate(this) +statechart: Statechart = Statechart(this, (short)2) +Alive: short=0 +Dead: short=1 +DiedBecauseOfAge: TransitionTimeout = new TransitionTimeout(this) +WasEatenByALynx: TransitionMessage = new TransitionMessage(this) ~_rectangle: int=1 ~_presentation_content: int[*] = int[] { _rectangle } ~presentation: int=0 ~icon: int=-1 ~rectangle: ShapeRectangle ~_DiedBecauseOfAge_pointsX: int[*] = { 250, 250, 263 } ~_DiedBecauseOfAge_pointsY: int[*] = { 80, 120, 126 } ~_WasEatenByALynx_pointsX: int[*] = { 290, 290, 277} ~_WasEatenByALynx_pointsY: int[*] = { 80, 120, 126 }
<<create>>+Hare(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection) +_Width_DefaultValue_xjal(): int +set_Width(Width: int) ~onChange_Width() +_CellWidth_DefaultValue_xjal(): int +set_CellWidth(CellWidth: int) ~onChange_CellWidth() +_NumberPerBirth_DefaultValue_xjal(): int +set_NumberPerBirth(NumberPerBirth: int) ~onChange_NumberPerBirth() +_LifeExpectancy_DefaultValue_xjal(): double +set_LifeExpectancy(LifeExpectancy: double) ~onChange_LifeExpectancy() +_Natality_DefaultValue_xjal(): double +set_ Natality(Natality: double) ~onChange_Natality() +_MaxPerCell_DefaultValue_xjal(): int +set_MaxPerCell(MaxPerCell: int) ~onChange_MaxPerCell() +getNameOf(_e: EventRate): String +evaluateRateOf(_e: EventRate): double +executeActionOf(_e: EventRate) +getNameOf(_s: Statechart): String +executeActionOf(_s: Statechart) +getNameOfState(_state: short): String +enterState(_state: short, _destination: boolean) +exitState(_state: short, _t: Transition, _source: boolean, _statechart: Statechart) +getNameOf(_t: TransitionTimeout): String +getStatechartOf(_t: TransitionTimeout): Statechart +executeActionOf(_t: TransitionTimeout) +evaluateTimeoutOf(_t: TransitionTimeout): double +getNameOf(_t: TransitionMessage): String +getStatechartOf(_t: TransitionMessage): Statechart +executeActionOf(_t: TransitionMessage, _msg: Object) +testMessageOf(_t: TransitionMessage, _msg: Object): boolean +getShapeGroupContent(_shape: int): int

+getShapeType(_shape: int): int +getPersistentShape(_shape: int): Object +drawModelElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean) +onClickModelAt(panel: Panel, x: double, y: double, clickCount: int, publicOnly: boolean): boolean +create() +start() +get_Main(): Main +onChange() +onDestroy()

Lynx
+Width: int +CellWidth: int +cell: int +HaveBabies: EventRate = new EventRate(this) +statechart: Statechart = new Statechart(this, (short)3) +state: short=0 +state3: short=1 +state2: short=2 +branch: short=3 +Dead: short=4 +tran3: TransitionTimeout = new TransitionTimeout(this) +tran4: TransitionTimeout = new TransitionTimeout(this) +Hunt: TransitionTimeout = new TransitionTimeout(this) ~_rectangle: int=1 ~_presentation_content: int[*] = new int[] { rectangle } ~presentation: int=0 ~icon: int=-1 ~rectangle: ShapeRectangle ~_tran3_pointsX: int[*] = { 290, 290, 312 } ~_tran3_pointsY: int[*] = { 220, 250, 250 } ~_tran4_pointsX: int[*] = { 360, 360, 328 } ~_tran4_pointsY: int[*] = { 200, 250, 250 } ~_Hunt_pointsX: int[*] = { 340, 340 } ~_Hunt_pointsY: int[*] = { 130, 161 } ~_NoLuck_pointsX: int[*] = { 329, 310, 310 } ~_NoLuck_pointsY: int[*] = { 170, 170, 130 } ~_Eat_pointsX: int[*] = { 351, 409, 390 } ~_Eat_pointsY: int[*] = { 170, 170, 190 } ~_state3_FillColor: Color = new Color(0xFFF8F184, true)
<<create>>+Lynx(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection) +_Width_DefaultValue_xjal(): int +set_Width(Width: int) ~onChange_Width() +_CellWidth_DefaultValue_xjal(): int +set_CellWidth(CellWidth: int) ~onChange_CellWidth() +getNameOf(_e: EventRate): String +evaluateRateOf(_e: EventRate): double +executeActionOf(_e: EventRate) +getNameOf(_s: Statechart): String +executeActionOf(_s: Statechart) +getNameOfState(_state: short): String +stateContainsState(compstate: short, simpstate: short): boolean +getContainerStateOf(_state: short): short +enterState(_state: short, _destination: boolean) +exitState(_state: short, _t: Transition, _source: boolean, _statechart: Statechart) +getNameOf(_t: TransitionTimeout): String +getStatechartOf(_t: TransitionTimeout): Statechart +executeActionOf(_t: TransitionTimeout) +evaluateTimeoutOf(_t: TransitionTimeout): double +getShapeGroupContent(_shape: int): int +getShapeType(_shape: int): int +getPersistentShape(_shape: int): Object

+drawModelElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean)
+onClickModelAt(panel: Panel, x: double, y: double, clickCount: int, publicOnly: boolean): boolean
+create()
+start()
+get_Main(): Main
+onChange()
+onDestroy()

Main

+HareNatality: double
+HareLifeExpectancy: double
+HareNumberPerBirth: int
+HareMaxPerCell: int
+CellWidth: int
+Width: int
+LynxNatality: double
+LynxHuntingPeriod: double
+LynxNumberPerBirth: int
+LynxHungerDeathThreshold: double
+LynxLifeExpectancy: double
+HaresInitial: int
+LynxInitial: int
+HaresInCell: ArrayList[*]
+auxGoodCells: int[*]
+_plot_autoUpdateEvent_xjal(): EventTimeout = new EventTimeout(this)
+lynx: ActiveObjectArrayList = new ActiveObjectArrayList()
+hares: ActiveObjectArrayList = new ActiveObjectArrayList()
+_plot_expression0_dataSet_xjal: DataSet = new DataSet(500) { double _lastUpdateX = Double.NaN; @Override public void update() { if(time() == _lastUpdateX) { return; } double_a=lynx.size(); add(time(), _a); _lastUpdateX = time(); }}
+_plot_expression1_dataSet_xjal: DataSet = new DataSet(500) { double _lastUpdateX = Double.NaN; @Override public void update() { if(time() == _lastUpdateX) { return; } double_a=lynx.size(); add(time(), _a); _lastUpdateX = time(); }}
~_text_Font: Font = new Font("SansSarif", 1, 12)
~_text1_Font: Font = text_Font
~_text2_Font: Font = text_Font
~_text5_Font: Font = new Font("SansSarif", 1, 11)
~_text6_Font: Font = text5_Font
~_text7_Font: Font = text5_Font
~_text8_Font: Font = text5_Font
~_text9_Font: Font = text5_Font
~_text10_Font: Font = text5_Font
~_text11_Font: Font = text5_Font
~_text12_Font: Font = text5_Font
~_text14_Font: Font = text_Font
~_text16_Font: Font = new Font("SansSarif", 0, 28)
~_text17_Font: Font = text16_Font
~_text18_Font: Font = new Font("SansSarif", 1, 14)
~_text19_Font: Font = new Font("SansSarif", 0, 12)
~_roundRect_FillColor: Color = new Color(0xF4F6F6FA, true)
~_text_Color: Color = new Color(0x0, true)
~_text1_Color: Color = new Color(0x0, true)
~_text2_Color: Color = new Color(0x0, true)
~_text14_Color: Color = new Color(0x0, true)
~slider: int=1
~slider1: int=2
~slider2: int=3
~slider3: int=4
~rect123: int=5
~roundRect: int=6
~text: int=7
~roundRect1: int=8
~text1: int=9
~roundRect2: int=10
~text2: int=11

~rect2: int=12
~text5: int=13
~text6: int=14
~text7: int=15
~text8: int=16
~text9: int=17
~text10: int=18
~text11: int=19
~text12: int=20
~roundRect3: int=21
~text14: int=22
~lynx_presentation: int=23
~hares_presentation: int=24
~text16: int=25
~text17: int=26
~text18: int=27
~line: int=28
~text19: int=29
~poly: int=30
~_plot: int=31
~_presentation_content: int[*] = new int[] { slider, slider1, slider2, slider3, rect123, rect2, text5, text6, text7, text8, text9, text10, text11, text12, lynx_presentation, hares_presentation, text16, text17, text18, line, text19, poly, plot }
~presentation: int=0
~_poly_pointsDX: double[*] = _poly_pointsDX_xjal()
~_poly_pointsDY: double[*] = _poly_pointsDY_xjal()
~poly: TimePlot

<<create>>+Main(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection)
+_HareNatality_DefaultValue_xjal(): double
+set_HareNatality(HareNatality: double)
~onChange_HareNatality()
+_HareLifeExpectancy_DefaultValue_xjal(): double
+set_HareLifeExpectancy(HareLifeExpectancy: double)
~onChange_HareLifeExpectancy()
+_HareNumberPerBirth_DefaultValue_xjal(): int
+set_HareNumberPerBirth(HareNumberPerBirth: int)
~onChange_HareNumberPerBirth()
+_HareMaxPerCell_DefaultValue_xjal(): int
+set_HareMaxPerCell(HareMaxPerCell: int)
~onChange_HareMaxPerCell()
+_CellWidth_DefaultValue_xjal(): int
+set_CellWidth(CellWidth: int)
~onChange_CellWidth()
+_Width_DefaultValue_xjal(): int
+set_Width(Width: int)
~onChange_Width()
+_LynxNatality_DefaultValue_xjal(): double
+set_LynxNatality(LynxNatality: double)
~onChange_LynxNatality()
+_LynxHuntingPeriod_DefaultValue_xjal(): double
+set_LynxHuntingPeriod(LynxHuntingPeriod: double)
~onChange_LynxHuntingPeriod()
+_LynxNumberPerBirth_DefaultValue_xjal(): int
+set_LynxNumberPerBirth(LynxNumberPerBirth: int)
~onChange_LynxNumberPerBirth()
+_LynxHungerDeathThreshold_DefaultValue_xjal(): double
+set_LynxHungerDeathThreshold(HungerDeathThreshold: double)
~onChange_LynxHungerDeathThreshold()
+_LynxLifeExpectancy_DefaultValue_xjal(): double
+set_LynxLifeExpectancy(LynxLifeExpectancy: double)
~onChange_LynxLifeExpectancy()
+_HaresInitial_DefaultValue_xjal(): int
+set_HaresInitial(HaresInitial: int)
~onChange_HaresInitial()
+_LynxInitial_DefaultValue_xjal(): int
+set_LynxInitial(LynxInitial: int)

~onChange_LynxInitial()
+getNameOf(_e: EventTimeout): String
+getModeOf(_e: EventTimeout): int
+getFirstOccurrenceTime(_e: EventTimeout): double
+evaluateTimeoutOf(_e: EventTimeout): double
+executeActionOf(_e: EventTimeout)
+getNameOf(ao: ActiveObject): String
+getNameOf(aolist: ActiveObjectCollection): String
+add_lynx(): Lynx
+add_lynx(Width: int, CellWidth: int): Lynx
+remove_lynx(object: Lynx): boolean
+add_hares(): Hare
+add_hares(Width: int, CellWidth: int, NumberPerBirth: int, LifeExpectancy: double, Natality: double, MaxPerCell: int): Hare
+remove_hares(object: Hare): boolean
-instantiate_lynx_xjal(index: int): Lynx
-setupParameters_lynx_xjal(object: Lynx, index: int)
-create_lynx_xjal(object: Lynx, index: int)
-instantiate_hares_xjal(index: int): Hare
-setupParameters_hares_xjal(object: Hare, index: int)
-create_hares_xjal(object: Hare, index: int)
~NotOverpopulatedCellAround(cell: int): int
~RandomCellAround(cell: int): int
~XGlobal(cellN: int, x: double): double
~YGlobal(cellN: int, x: double): double
~_poly_pointsDX_xjal(): double
~_poly_pointsDY_xjal(): double
+executeShapeControlAction(_shape: int, index: int, value: double)
+getShapeControlValueType(_shape: int, index: int): int
+getShapeControlMinimum(_shape: int, index: int): double
+getShapeControlMaximum(_shape: int, index: int): double
+getShapeControlDefaultValueDouble(_shape: int, index: int): double
+getShapeGroupContent(_shape: int): int
+getNameOfShape(_shape: int): String
+getShapeType(_shape: int): int
+getShapeReplicator(_shape: int): int
+getShapeX(_shape: int, index: int): double
+getShapeY(_shape: int, index: int): double
+getShapeEmbeddedObject(_shape: int): Object
+getShapeLineColor(_shape: int, index: int): Color
+getShapeFillColor(_shape: int, index: int): Color
+getShapeLineStyle(_shape: int, index: int): int
+getShapeLineWidth(_shape: int, index: int): double
+getShapeLineDx(_shape: int, index: int): double
+getShapeLineDy(_shape: int, index: int): double
+getShapeWidth(_shape: int, index: int): double
+getShapeHeight(_shape: int, index: int): double
+getShapeRadiusX(_shape: int, index: int): double
+getShapeRadiusY(_shape: int, index: int): double
+getShapeNPoints(_shape: int, index: int): int
+getShapePointsDx(_shape: int, index: int): double
+getShapePointsDy(_shape: int, index: int): double
+getShapeText(_shape: int, index: int): Object
+getShapeFont(_shape: int, index: int): Font
+getShapeTextAlignment(_shape: int, index: int): int
+isShapeControlVertical(_shape: int, index: int): boolean
+getPersistentShape(_shape: int): Object
+drawModelElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean)
+onClickModelAt(panel: Panel, x: double, y: double, clickCount: int, publicOnly: boolean): boolean
+create()
+start()
+onStartup()
+getEmbeddedObjects(): List
+onDestroy()

Κώδικας Κεφαλαίου 6 – Έργο Predator Prey Agent Based

Hare.java

```
package predator_prey_agent_based;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import java.awt.geom.Arc2D;

public class Hare extends Agent
{
 // Parameters

 public int Width;

 /**
  * Returns default value for parameter <code>Width</code>.
  * <i>This method should not be called by user</i>
  */
 public int _Width_DefaultValue_xjal() {
  return 15;
 }

 public void set_Width(
int Width ) {
  if (Width == this.Width) {
   return;
  }
  this.Width = Width;
  onChange_Width();
  onChange();
 }

 void onChange_Width() {
 }

 public
int CellWidth;

 /**
  * Returns default value for parameter <code>CellWidth</code>.
  * <i>This method should not be called by user</i>
  */
 public
int _CellWidth_DefaultValue_xjal() {
  return
20
;
 }

 public void set_CellWidth(
int CellWidth ) {
  if (CellWidth == this.CellWidth) {
   return;
  }
  this.CellWidth = CellWidth;
  onChange_CellWidth();
  onChange();
 }

 void onChange_CellWidth() {
 }

 public int NumberPerBirth;

 /**
  * Returns default value for parameter
<code>NumberPerBirth</code>.
  * <i>This method should not be called by user</i>
  */
 public
int _NumberPerBirth_DefaultValue_xjal() {
  return 5;
 }
```

```
  public void set_NumberPerBirth(
int  NumberPerBirth ) {
    if (NumberPerBirth == this.NumberPerBirth) {
      return;
    }
    this.NumberPerBirth = NumberPerBirth;
    onChange_NumberPerBirth();
    onChange();
  }

  void onChange_NumberPerBirth() {
  }


  public double  LifeExpectancy;

  /**
   * Returns default value for parameter <code>LifeExpectancy</code>.
   * <i>This method should not be called by user</i>
   */
  public
double  _LifeExpectancy_DefaultValue_xjal() {
    return
3
;
  }

  public void set_LifeExpectancy(
double  LifeExpectancy ) {
    if (LifeExpectancy == this.LifeExpectancy) {
      return;
    }
    this.LifeExpectancy = LifeExpectancy;
    onChange_LifeExpectancy();
    onChange();
  }

  void onChange_LifeExpectancy() {
  }


  public
double  Natality;

  /**
   * Returns default value for parameter <code>Natality</code>.
   * <i>This method should not be called by user</i>
   */
  public
double  _Natality_DefaultValue_xjal() {
    return
4
;
  }

  public void set_Natality(
double  Natality ) {
    if (Natality == this.Natality) {
      return;
    }
    this.Natality = Natality;
    onChange_Natality();
    onChange();
  }

  void onChange_Natality() {
  }


  public int  MaxPerCell;

  /**
   * Returns default value for parameter <code>MaxPerCell</code>.
   * <i>This method should not be called by user</i>
   */
  public
int  _MaxPerCell_DefaultValue_xjal() {
    return
15
;
  }

  public void set_MaxPerCell(
int  MaxPerCell ) {
    if (MaxPerCell == this.MaxPerCell) {
      return;
    }
    this.MaxPerCell = MaxPerCell;
    onChange_MaxPerCell();
    onChange();
  }

  void onChange_MaxPerCell() {
  }

  // Plain Variables

  public
int
 cell;

  // Collection Variables

  // Flow/Auxiliary Variables

  // Stock Variables


  // Events

  public EventRate HaveBabies = new EventRate(this);

  @Override
  public String getNameOf( EventRate _e ) {
    if ( _e == HaveBabies) return "HaveBabies";
    return super.getNameOf( _e );
  }

  @Override
  public double evaluateRateOf( EventRate _e ) {
    if ( _e == HaveBabies) return
Natality
;
    return super.evaluateRateOf( _e );
  }

  @Override
  public void executeActionOf( EventRate _e ) {
    if ( _e == HaveBabies) {
```

```
Main m = get_Main();
for( int i=0; i<NumberPerBirth; i++ ) {
  int newcell = cell;
  if( m.HaresInCell[newcell].size() >= MaxPerCell )
    newcell = m.NotOverpopulatedCellAround( newcell );
  if( newcell != -1 ) {
    Hare baby = m.add_hares();
    baby.cell = newcell;
    m.HaresInCell[newcell].add( baby );
  } else {
    break;
  }
}

;
      return ;
    }
    super.executeActionOf( _e );
  }

  // Statecharts
  public Statechart statechart = new Statechart( this, (short)2 );

  @Override
  public String getNameOf( Statechart _s ) {
    if(_s == this.statechart) return "statechart";
    return super.getNameOf( _s );
  }

  @Override
  public void executeActionOf( Statechart _s ) {
    if( _s == this.statechart ) {
      enterState( Alive, true );
      return;
    }
    super.executeActionOf( _s );
  }

   // States of all statecharts

  public static final short Alive = 0;
  public static final short Dead = 1;


  @Override
  public String getNameOfState( short _state ) {
    switch( _state ) {
      case Alive: return "Alive";
      case Dead: return "Dead";
      default: return super.getNameOfState( _state );
    }
  }

  @Override
  public void enterState( short _state, boolean _destination ) {
    switch( _state ) {
      case Alive: // (Simple state (not composite))
        statechart.setActiveState( Alive );
        DiedBecauseOfAge.start();
        WasEatenByALynx.start();
        return;
      case Dead: // (Final State)
        {
get_Main().HaresInCell[cell].remove( this );
get_Main().remove_hares( this );
;}
        statechart.setActiveState( Dead );
        statechart.onDestroy();
        return;
      default:
        super.enterState( _state, _destination );
        return;
    }
  }

  @Override
  public void exitState( short _state, Transition _t, boolean _source,
Statechart _statechart) {
    switch( _state ) {
      case Alive: // (Simple state (not composite))
        if ( !_source || _t != DiedBecauseOfAge)
DiedBecauseOfAge.cancel();
        if ( !_source || _t != WasEatenByALynx)
WasEatenByALynx.cancel();
        return;
      default:
        super.exitState( _state, _t, _source, _statechart);
        return;
    }
  }


  public TransitionTimeout DiedBecauseOfAge = new TransitionTimeout(
this );

  @Override
  public String getNameOf( TransitionTimeout _t ) {
    if ( _t == DiedBecauseOfAge ) return "DiedBecauseOfAge";
    return super.getNameOf( _t );
  }

  @Override
  public Statechart getStatechartOf( TransitionTimeout _t ) {
    if ( _t == DiedBecauseOfAge ) return statechart;
    return super.getStatechartOf( _t );
  }

  @Override
  public void executeActionOf( TransitionTimeout _t ) {
    if ( _t == DiedBecauseOfAge ) {
      exitState( Alive, _t, true, statechart );
          enterState( Dead, true );
      return;
    }
    super.executeActionOf( _t );
  }
  @Override
  public double evaluateTimeoutOf( TransitionTimeout _t ) {
    if ( _t == DiedBecauseOfAge ) return
LifeExpectancy
;
    return super.evaluateTimeoutOf( _t );
  }



  public TransitionMessage WasEatenByALynx = new TransitionMessage(
this );

  @Override
  public String getNameOf( TransitionMessage _t ) {
```

```
  if ( _t == WasEatenByALynx ) return "WasEatenByALynx";
  return super.getNameOf( _t );
 }

 @Override
 public Statechart getStatechartOf( TransitionMessage _t ) {
  if ( _t == WasEatenByALynx ) return statechart;
  return super.getStatechartOf( _t );
 }

 @Override
 public void executeActionOf( TransitionMessage _t, Object _msg ) {
  if ( _t == WasEatenByALynx ) {
    exitState( Alive, _t, true, statechart );
       enterState( Dead, true );
    return;
  }
  super.executeActionOf( _t, _msg );
 }
 @Override
 public boolean testMessageOf( TransitionMessage _t, Object _msg ) {
  if ( _t == WasEatenByALynx ) {
    Object msg = (Object) _msg;
    return true;
  }
  return super.testMessageOf( _t, _msg );
 }
 static final int _rectangle = 1;


 /**
  * Top-level presentation group content
  */
 static final int[] _presentation_content = new int[] {
  _rectangle,
 };

 /**
  * Top-level presentation group id
  */
 static final int presentation = 0;

 /**
  * Top-level icon group id
  */
 static final int icon = -1;


 @Override
 public int[] getShapeGroupContent( int _shape ){
    switch( _shape ) {
      case presentation: return _presentation_content;
      default: return super.getShapeGroupContent( _shape );
    }
 }


 @Override
 public int getShapeType( int _shape ) {
  switch( _shape ) {
    case presentation: return SHAPE_GROUP;
    case icon: return SHAPE_GROUP;
    default: return super.getShapeType( _shape );
  }
 }

 ShapeRectangle rectangle;

 // Static initialization of persistent elements
 {
  rectangle = new ShapeRectangle(
    true,-1, -1, 0.0,
        null, lime,
                                        2, 2,
                                        1, LINE_STYLE_SOLID
  );
 }

 @Override
 public Object getPersistentShape( int _shape ) {
  switch(_shape){

    case _rectangle: return rectangle;
    default: return null;
  }
 }

 static final int[] _DiedBecauseOfAge_pointsX = {250, 250, 263, };
 static final int[] _DiedBecauseOfAge_pointsY = {80, 120, 126, };

 static final int[] _WasEatenByALynx_pointsX = {290, 290, 277, };
 static final int[] _WasEatenByALynx_pointsY = {80, 120, 126, };


 @Override
 public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
  if (!_publicOnly) {
    drawEvent( _panel, _g, 150, 50, 10, 0, "HaveBabies", HaveBabies );
  }
  if (!_publicOnly) {
    drawFinalState( _panel, _g, 270, 130, -10, 20, "Dead",  Dead,
statechart );
  }
  if (!_publicOnly) {
    drawState( _panel, _g, 240, 60, 60, 20, 10, 10, "Alive", greenYellow,
Alive, statechart );
  }
  if (!_publicOnly) {
    drawTransition(_panel,  _g,  _DiedBecauseOfAge_pointsX,
_DiedBecauseOfAge_pointsY,  150,  100,  "DiedBecauseOfAge",
DiedBecauseOfAge );
  }
  if (!_publicOnly) {
    drawTransition(_panel,  _g,  _WasEatenByALynx_pointsX,
_WasEatenByALynx_pointsY,  290,  100,  "WasEatenByALynx",
WasEatenByALynx );
  }
  if (!_publicOnly) {
    drawStatechartEntryPoint( _panel, _g, 260, 40, 260, 60, 240, 20,
"statechart",  statechart );
  }
  if (!_publicOnly) {
    drawParameter( _panel, _g, 50, 50, 20, 0, "Width", Width, false,
false );
  }
  if (!_publicOnly) {
    drawParameter( _panel, _g, 50, 70, 20, 0, "CellWidth", CellWidth,
false, false );
  }
  if (!_publicOnly) {
    drawParameter( _panel, _g, 50, 120, 20, 0, "NumberPerBirth",
NumberPerBirth, false, false );
  }
  if (!_publicOnly) {
```

```
    drawParameter( _panel, _g, 50, 140, 20, 0, "LifeExpectancy", LifeExpectancy, false, false );
   }
   if (!_publicOnly) {
    drawParameter( _panel, _g, 50, 100, 20, 0, "Natality", Natality, false, false );
   }
   if (!_publicOnly) {
    drawParameter( _panel, _g, 50, 160, 20, 0, "MaxPerCell", MaxPerCell, false, false );
   }
   if (!_publicOnly) {
    drawPlainVariable( _panel, _g, 50, 200, 20, 0, "cell", cell, false );
   }
  }

  @Override
  public boolean onClickModelAt( Panel panel, double x, double y, int clickCount, boolean publicOnly ) {
   if( !publicOnly && modelElementContains(x, y, 50, 50) ) {
    panel.addInspect( 50, 50, this, "Width" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 50, 70) ) {
    panel.addInspect( 50, 70, this, "CellWidth" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 50, 120) ) {
    panel.addInspect( 50, 120, this, "NumberPerBirth" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 50, 140) ) {
    panel.addInspect( 50, 140, this, "LifeExpectancy" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 50, 100) ) {
    panel.addInspect( 50, 100, this, "Natality" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 50, 160) ) {
    panel.addInspect( 50, 160, this, "MaxPerCell" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 50, 200) ) {
    panel.addInspect( 50, 200, this, "cell" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 150, 50) ) {
    panel.addInspect( 150, 50, this, "HaveBabies" );
    return true;
   }
   return false;
  }

  /**
   * Constructor
   */
  public Hare( Engine engine, ActiveObject owner, ActiveObjectCollection<? extends Hare> collection ) {
   super( engine, owner, collection );
  }

  @Override
  public void create() {
   // Assigning initial values for plain variables
   cell =
uniform_discr( 0, Width*Width - 1 );
   // Dynamic initialization of persistent elements
   // Agent properties setup
   if ( getEnvironment() != null && (getEnvironment().getSpaceType() != Environment.SPACE_CONTINUOUS_2D || getEnvironment().getGISMap() != null) ) {
    throw new RuntimeException( "Space type of an agent class 'Hare' doesn't match its environment. See 'Agent' page on the Properties View of 'Hare' in the AnyLogic" );
   }
   setXY(
( ( cell % Width ) * CellWidth ) + uniform( 0, CellWidth )
,
( (int)(cell/Width) * CellWidth ) + uniform( 0, CellWidth )
);
                // Port connectors with non-replicated objects
   // Creating replicated embedded objects
   assignInitialConditions();
   onCreate();
  }

  @Override
  public void start() {
   HaveBabies.start();
   statechart.start();
   onStartup();
  }


// User API ----------------------------------
  public Main get_Main() {
   ActiveObject owner = getOwner();
   if ( owner instanceof Main ) return (Main) owner;
   return null;
  }

  // Reaction on changes -----------------------
  public void onChange() {
   HaveBabies.onChange();
   statechart.onChange();
  }

  public void onDestroy() {
   super.onDestroy();
   HaveBabies.onDestroy();
   statechart.onDestroy();
  }
}
```

Lynx.java

```
package predator_prey_agent_based;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
```

```
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import java.awt.geom.Arc2D;

public class Lynx extends Agent
{
  // Parameters

  public int Width;

  /**
   * Returns default value for parameter <code>Width</code>.
   * <i>This method should not be called by user</i>
   */
  public
int  _Width_DefaultValue_xjal() {
    return 15 ;
  }

  public void set_Width(
int  Width ) {
    if (Width == this.Width) {
      return;
    }
    this.Width = Width;
    onChange_Width();
    onChange();
  }

  void onChange_Width() {
  }


  public
int  CellWidth;

  /**
   * Returns default value for parameter <code>CellWidth</code>.
   * <i>This method should not be called by user</i>
   */
  public
int  _CellWidth_DefaultValue_xjal() {
    return
20
;
  }

  public void set_CellWidth(
int  CellWidth ) {
    if (CellWidth == this.CellWidth) {
      return;
    }
    this.CellWidth = CellWidth;
    onChange_CellWidth();
    onChange();
  }

  void onChange_CellWidth() {
  }

  // Plain Variables

  public
int
 cell;

  // Collection Variables

  // Flow/Auxiliary Variables

  // Stock Variables


  // Events

  public EventRate HaveBabies = new EventRate(this);

  @Override
  public String getNameOf( EventRate _e ) {
    if ( _e == HaveBabies) return "HaveBabies";
    return super.getNameOf( _e );
  }

  @Override
  public double evaluateRateOf( EventRate _e ) {
    if ( _e == HaveBabies) return
get_Main().LynxNatality
;
    return super.evaluateRateOf( _e );
  }

  @Override
  public void executeActionOf( EventRate _e ) {
    if ( _e == HaveBabies) {

Main m = get_Main();
for( int i=0; i<m.LynxNumberPerBirth; i++ ) {
  Lynx baby = m.add_lynx();
  baby.cell = cell;
}

;
```

```
    return ;
  }
  super.executeActionOf( _e );
 }

 // Statecharts
 public Statechart statechart = new Statechart( this, (short)3 );

 @Override
 public String getNameOf( Statechart _s ) {
  if(_s == this.statechart) return "statechart";
  return super.getNameOf( _s );
 }

 @Override
 public void executeActionOf( Statechart _s ) {
  if( _s == this.statechart ) {
    enterState( state, true );
    return;
  }
  super.executeActionOf( _s );
 }

  // States of all statecharts

 public static final short state = 0;
 public static final short state3 = 1;
 public static final short state2 = 2;
 public static final short branch = 3;
 public static final short Dead = 4;


 @Override
 public String getNameOfState( short _state ) {
  switch( _state ) {
    case state: return "state";
    case state3: return "state3";
    case state2: return "state2";
    case branch: return "branch";
    case Dead: return "Dead";
    default: return super.getNameOfState( _state );
  }
 }

 @Override
 public boolean stateContainsState( short compstate, short simpstate ) {
  if (compstate == state && (simpstate == state3)) {
    return true;
  }
  if (compstate == state3 && (simpstate == state2)) {
    return true;
  }
  return super.stateContainsState( compstate, simpstate );
 }

 @Override
 public short getContainerStateOf( short _state ) {
  switch( _state ) {
    case state3: return state;
    case state2: return state3;
    case branch: return state3;
    default: return super.getContainerStateOf( _state );
  }
 }

 @Override
 public void enterState( short _state, boolean _destination ) {
  switch( _state ) {
    case state: // (Composite state)
      tran3.start();
      if ( _destination ) {
        enterState( state3, true );
      }
      return;
    case state3: // (Composite state)
      tran4.start();
      if ( _destination ) {
        enterState( state2, true );
      }
      return;
    case state2: // (Simple state (not composite))
      statechart.setActiveState( state2 );
      Hunt.start();
      return;
    case branch: // (Branch)
      if (
randomTrue( min( 1, get_Main().HaresInCell[cell].size() /
(0.5*(double)get_Main().HareMaxPerCell) ) )
 ) { // Eat
        exitState( state3, null, false, statechart );
        {
Main m = get_Main();
Hare prey = m.HaresInCell[cell].get( uniform_discr( 0,
m.HaresInCell[cell].size()-1 ) );
prey.statechart.receiveMessage( new Object() );
;}
        enterState( state3, true );
        return;
      }
      // NoLuck (default)
        {
cell = get_Main().RandomCellAround( cell );
jumpTo( ( ( cell % Width ) * CellWidth ) + uniform( 0, CellWidth ),
                    ( (int)(cell/Width) * CellWidth ) + uniform( 0,
CellWidth ) );
;}
        enterState( state2, true );
      return;
    case Dead: // (Final State)
      {
get_Main().remove_lynx( this );
;}
      statechart.setActiveState( Dead );
      statechart.onDestroy();
      return;
    default:
      super.enterState( _state, _destination );
      return;
  }
 }

 @Override
 public void exitState( short _state, Transition _t, boolean _source,
Statechart _statechart) {
  switch( _state ) {
    case state: // (Composite state)
      if ( _source ) exitInnerStates(_state, _statechart);
      if ( !_source || _t != tran3 ) tran3.cancel();
      return;
    case state3: // (Composite state)
      if ( _source ) exitInnerStates(_state, _statechart);
      if ( !_source || _t != tran4 ) tran4.cancel();
```

```
      return;
    case state2: // (Simple state (not composite))
      if ( !_source || _t != Hunt) Hunt.cancel();
      return;
    default:
      super.exitState( _state, _t, _source, _statechart);
      return;
  }
 }


 public TransitionTimeout tran3 = new TransitionTimeout( this );
 public TransitionTimeout tran4 = new TransitionTimeout( this );
 public TransitionTimeout Hunt = new TransitionTimeout( this );

 @Override
 public String getNameOf( TransitionTimeout _t ) {
  if ( _t == tran3 ) return "tran3";
  if ( _t == tran4 ) return "tran4";
  if ( _t == Hunt ) return "Hunt";
  return super.getNameOf( _t );
 }

 @Override
 public Statechart getStatechartOf( TransitionTimeout _t ) {
  if ( _t == tran3 ) return statechart;
  if ( _t == tran4 ) return statechart;
  if ( _t == Hunt ) return statechart;
  return super.getStatechartOf( _t );
 }

 @Override
 public void executeActionOf( TransitionTimeout _t ) {
  if ( _t == tran3 ) {
    exitState( state, _t, true, statechart );
        enterState( Dead, true );
    return;
  }
  if ( _t == tran4 ) {
    exitState( state3, _t, true, statechart );
    exitState( state, _t, false, statechart );
        enterState( Dead, true );
    return;
  }
  if ( _t == Hunt ) {
    exitState( state2, _t, true, statechart );
        enterState( branch, true );
    return;
  }
  super.executeActionOf( _t );
 }
 @Override
 public double evaluateTimeoutOf( TransitionTimeout _t ) {
  if ( _t == tran3 ) return
get_Main().LynxLifeExpectancy
;
  if ( _t == tran4 ) return
get_Main().LynxHungerDeathThreshold
;
  if ( _t == Hunt ) return
get_Main().LynxHuntingPeriod
;
  return super.evaluateTimeoutOf( _t );
 }
```

```
static final int _rectangle = 1;


/**
 * Top-level presentation group content
 */
static final int[] _presentation_content = new int[] {
  _rectangle,
};

/**
 * Top-level presentation group id
 */
static final int presentation = 0;

/**
 * Top-level icon group id
 */
static final int icon = -1;

@Override
public int[] getShapeGroupContent( int _shape ){
  switch( _shape ) {
    case presentation: return _presentation_content;
    default: return super.getShapeGroupContent( _shape );
  }
}


@Override
public int getShapeType( int _shape ) {
 switch( _shape ) {
  case presentation: return SHAPE_GROUP;
  case icon: return SHAPE_GROUP;
  default: return super.getShapeType( _shape );
 }
}




ShapeRectangle rectangle;

// Static initialization of persistent elements
{
 rectangle = new ShapeRectangle(
    true,-2, -2, 0.0,
        null, red,
                                        4, 4,
                                        1, LINE_STYLE_SOLID
 );
}

@Override
public Object getPersistentShape( int _shape ) {
 switch(_shape){

  case _rectangle: return rectangle;
  default: return null;
 }
```

```
 }

 static final int[] _tran3_pointsX = {290, 290, 312, };
 static final int[] _tran3_pointsY = {220, 250, 250, };

 static final int[] _tran4_pointsX = {360, 360, 328, };
 static final int[] _tran4_pointsY = {200, 250, 250, };

 static final int[] _Hunt_pointsX = {340, 340, };
 static final int[] _Hunt_pointsY = {130, 161, };

 static final int[] _NoLuck_pointsX = {329, 310, 310, };
 static final int[] _NoLuck_pointsY = {170, 170, 130, };

 static final int[] _Eat_pointsX = {351, 409, 390, };
 static final int[] _Eat_pointsY = {170, 170, 190, };

 static final Color _state3_FillColor = new Color( 0xFFF8F184, true );

 @Override
 public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
  if (!_publicOnly) {
   drawEvent( _panel, _g, 150, 50, 10, 0, "HaveBabies", HaveBabies );
  }
  if (!_publicOnly) {
   drawState( _panel, _g, 260, 50, 160, 170, 20, 10, null, lemonChiffon,
state, statechart );
  }
  if (!_publicOnly) {
   drawFinalState( _panel, _g, 320, 250, -10, 20, null,  Dead, statechart
);
  }
  if (!_publicOnly) {
   drawState( _panel, _g, 280, 70, 110, 130, 10, 10, null,
_state3_FillColor, state3, statechart );
  }
  if (!_publicOnly) {
   drawBranchState( _panel, _g, 340, 170, -20, 20, null,  branch );
  }
  if (!_publicOnly) {
   drawState( _panel, _g, 300, 110, 60, 20, 10, 10, null, gold, state2,
statechart );
  }
  if (!_publicOnly) {
   drawInitialStatePointer( _panel, _g, 320, 90, 320, 110 );
  }
  if (!_publicOnly) {
   drawBranchExit(_panel, _g, _NoLuck_pointsX, _NoLuck_pointsY,
290, 180, "NoLuck" );
  }
  if (!_publicOnly) {
   drawTransition(_panel, _g, _Hunt_pointsX, _Hunt_pointsY, 350, 140,
"Hunt", Hunt );
  }
  if (!_publicOnly) {
   drawTransition(_panel, _g, _tran3_pointsX, _tran3_pointsY, 250,
250, null, tran3 );
  }
  if (!_publicOnly) {
   drawTransition(_panel, _g, _tran4_pointsX, _tran4_pointsY, 370,
250, null, tran4 );
  }
  if (!_publicOnly) {
   drawStatechartEntryPoint( _panel, _g, 300, 30, 300, 50, 310, 30,
"statechart",  statechart );
  }
  if (!_publicOnly) {
   drawInitialStatePointer( _panel, _g, 410, 80, 390, 80 );
  }
  if (!_publicOnly) {
   drawBranchExit(_panel, _g, _Eat_pointsX, _Eat_pointsY, 361, 180,
"Eat" );
  }
  if (!_publicOnly) {
   drawParameter( _panel, _g, 50, 50, 20, 0, "Width", Width, false,
false );
  }
  if (!_publicOnly) {
   drawParameter( _panel, _g, 50, 70, 20, 0, "CellWidth", CellWidth,
false, false );
  }
  if (!_publicOnly) {
   drawPlainVariable( _panel, _g, 50, 200, 20, 0, "cell", cell, false );
  }
 }

 @Override
 public boolean onClickModelAt( Panel panel, double x, double y, int
clickCount, boolean publicOnly ) {
  if( !publicOnly && modelElementContains(x, y, 50, 50) ) {
   panel.addInspect( 50, 50, this, "Width" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 50, 70) ) {
   panel.addInspect( 50, 70, this, "CellWidth" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 50, 200) ) {
   panel.addInspect( 50, 200, this, "cell" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 150, 50) ) {
   panel.addInspect( 150, 50, this, "HaveBabies" );
   return true;
  }
  return false;
 }

 /**
  * Constructor
  */
 public Lynx( Engine engine, ActiveObject owner,
ActiveObjectCollection<? extends Lynx> collection ) {
  super( engine, owner, collection );
 }

 @Override
 public void create() {
  // Assigning initial values for plain variables
  cell =
uniform_discr( 0, get_Main().Width*get_Main().Width-1 )
;
  // Dynamic initialization of persistent elements
  // Agent properties setup
  if ( getEnvironment() != null && (getEnvironment().getSpaceType()
!= Environment.SPACE_CONTINUOUS_2D ||
getEnvironment().getGISMap() != null) ) {
   throw new RuntimeException( "Space type of an agent class 'Lynx'
doesn't match its environment. See 'Agent' page on the Properties View
of 'Lynx' in the AnyLogic" );
  }
  setXY(
( ( cell % Width ) * CellWidth ) + uniform( 0, CellWidth )
,
( (int)(cell/Width) * CellWidth ) + uniform( 0, CellWidth )
```

```
);
                // Port connectors with non-replicated objects
  // Creating replicated embedded objects
  assignInitialConditions();
  onCreate();
 }

 @Override
 public void start() {
  HaveBabies.start();
  statechart.start();
  onStartup();
 }


// User API ------------------------------------
 public Main get_Main() {
  ActiveObject owner = getOwner();
  if ( owner instanceof Main ) return (Main) owner;
  return null;
 }

 // Reaction on changes -------------------------
 public void onChange() {
  HaveBabies.onChange();
  statechart.onChange();
 }

 public void onDestroy() {
  super.onDestroy();
  HaveBabies.onDestroy();
  statechart.onDestroy();
 }
}
```

Main.java

```
package predator_prey_agent_based;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import java.awt.geom.Arc2D;

public class Main extends ActiveObject
{
 // Parameters

 public
double  HareNatality;

 /**
  * Returns default value for parameter <code>HareNatality</code>.
  * <i>This method should not be called by user</i>
  */
 public
double  _HareNatality_DefaultValue_xjal() {
  return 4 ;
 }

 public void set_HareNatality(
double  HareNatality ) {
  if (HareNatality == this.HareNatality) {
   return;
  }
  this.HareNatality = HareNatality;
  onChange_HareNatality();
  onChange();
 }

 void onChange_HareNatality() {
            int index;
  index = 0;
  for ( Hare object : hares ) {
   object.set_Natality(HareNatality);
   index++;
  }
 }


 public double  HareLifeExpectancy;

 /**
  *  Returns  default  value  for  parameter
<code>HareLifeExpectancy</code>.
  * <i>This method should not be called by user</i>
  */
 public
double  _HareLifeExpectancy_DefaultValue_xjal() {
  return 3 ;
```

```
  }

  public void set_HareLifeExpectancy(
double  HareLifeExpectancy ) {
    if (HareLifeExpectancy == this.HareLifeExpectancy) {
      return;
    }
    this.HareLifeExpectancy = HareLifeExpectancy;
    onChange_HareLifeExpectancy();
    onChange();
  }

  void onChange_HareLifeExpectancy() {
                int index;
    index = 0;
    for ( Hare object : hares ) {
      object.set_LifeExpectancy(HareLifeExpectancy);
      index++;
    }
  }


  public int  HareNumberPerBirth;

  /**
   *       Returns       default       value       for       parameter
<code>HareNumberPerBirth</code>.
   * <i>This method should not be called by user</i>
   */
  public
int  _HareNumberPerBirth_DefaultValue_xjal() {
    return 5 ;
  }

  public void set_HareNumberPerBirth(
int  HareNumberPerBirth ) {
    if (HareNumberPerBirth == this.HareNumberPerBirth) {
      return;
    }
    this.HareNumberPerBirth = HareNumberPerBirth;
    onChange_HareNumberPerBirth();
    onChange();
  }

  void onChange_HareNumberPerBirth() {
                int index;
    index = 0;
    for ( Hare object : hares ) {
      object.set_NumberPerBirth(HareNumberPerBirth);
      index++;
    }
  }


  public int  HareMaxPerCell;

  /**
   *       Returns       default       value       for       parameter
<code>HareMaxPerCell</code>.
   * <i>This method should not be called by user</i>
   */
  public
int  _HareMaxPerCell_DefaultValue_xjal() {
    return 15 ;
  }

  public void set_HareMaxPerCell(
int  HareMaxPerCell ) {
    if (HareMaxPerCell == this.HareMaxPerCell) {
      return;
    }
    this.HareMaxPerCell = HareMaxPerCell;
    onChange_HareMaxPerCell();
    onChange();
  }

  void onChange_HareMaxPerCell() {
                int index;
    index = 0;
    for ( Hare object : hares ) {
      object.set_MaxPerCell(HareMaxPerCell);
      index++;
    }
  }


  public int  CellWidth;

  /**
   * Returns default value for parameter <code>CellWidth</code>.
   * <i>This method should not be called by user</i>
   */
  public
int  _CellWidth_DefaultValue_xjal() {
    return 20 ;
  }

  public void set_CellWidth(
int  CellWidth ) {
    if (CellWidth == this.CellWidth) {
      return;
    }
    this.CellWidth = CellWidth;
    onChange_CellWidth();
    onChange();
  }

  void onChange_CellWidth() {
                int index;
    index = 0;
    for ( Hare object : hares ) {
      object.set_CellWidth(CellWidth);
      index++;
    }
  }


  public int  Width;

  /**
   * Returns default value for parameter <code>Width</code>.
   * <i>This method should not be called by user</i>
   */
  public int  _Width_DefaultValue_xjal() {
    return 15 ;
  }

  public void set_Width(
int  Width ) {
    if (Width == this.Width) {
      return;
```

```
  }
  this.Width = Width;
  onChange_Width();
  onChange();
 }

 void onChange_Width() {
            int index;
  index = 0;
  for ( Hare object : hares ) {
   object.set_Width(Width);
   index++;
  }

 }


 public double  LynxNatality;

 /**
  * Returns default value for parameter <code>LynxNatality</code>.
  * <i>This method should not be called by user</i>
  */
 public
double  _LynxNatality_DefaultValue_xjal() {
  return 2 ;
 }

 public void set_LynxNatality(
double  LynxNatality ) {
  if (LynxNatality == this.LynxNatality) {
   return;
  }
  this.LynxNatality = LynxNatality;
  onChange_LynxNatality();
  onChange();
 }

 void onChange_LynxNatality() {
 }

 public double  LynxHuntingPeriod;

 /**
  *      Returns      default      value      for      parameter
<code>LynxHuntingPeriod</code>.
  * <i>This method should not be called by user</i>
  */
 public
double  _LynxHuntingPeriod_DefaultValue_xjal() {
  return 0.01 ;
 }

 public void set_LynxHuntingPeriod(
double  LynxHuntingPeriod ) {
  if (LynxHuntingPeriod == this.LynxHuntingPeriod) {
   return;
  }
  this.LynxHuntingPeriod = LynxHuntingPeriod;
  onChange_LynxHuntingPeriod();
  onChange();
 }

 void onChange_LynxHuntingPeriod() {
 }

 public int  LynxNumberPerBirth;

 /**
  *      Returns      default      value      for      parameter
<code>LynxNumberPerBirth</code>.
  * <i>This method should not be called by user</i>
  */
 public int  _LynxNumberPerBirth_DefaultValue_xjal() {
  return 3 ;
 }

 public void set_LynxNumberPerBirth(
int  LynxNumberPerBirth ) {
  if (LynxNumberPerBirth == this.LynxNumberPerBirth) {
   return;
  }
  this.LynxNumberPerBirth = LynxNumberPerBirth;
  onChange_LynxNumberPerBirth();
  onChange();
 }

 void onChange_LynxNumberPerBirth() {
 }


 public double  LynxHungerDeathThreshold;

 /**
  *      Returns      default      value      for      parameter
<code>LynxHungerDeathThreshold</code>.
  * <i>This method should not be called by user</i>
  */
 public
double  _LynxHungerDeathThreshold_DefaultValue_xjal() {
  return 0.03 ;
 }

 public void set_LynxHungerDeathThreshold(
double  LynxHungerDeathThreshold ) {
  if (LynxHungerDeathThreshold == this.LynxHungerDeathThreshold) {
   return;
  }
  this.LynxHungerDeathThreshold = LynxHungerDeathThreshold;
  onChange_LynxHungerDeathThreshold();
  onChange();
 }

 void onChange_LynxHungerDeathThreshold() {
 }


 public double  LynxLifeExpectancy;

 /**
  *      Returns      default      value      for      parameter
<code>LynxLifeExpectancy</code>.
  * <i>This method should not be called by user</i>
  */
 public
double  _LynxLifeExpectancy_DefaultValue_xjal() {
  return
8
;
 }

 public void set_LynxLifeExpectancy(
```

```
double LynxLifeExpectancy ) {
  if (LynxLifeExpectancy == this.LynxLifeExpectancy) {
    return;
  }
  this.LynxLifeExpectancy = LynxLifeExpectancy;
  onChange_LynxLifeExpectancy();
  onChange();
 }

 void onChange_LynxLifeExpectancy() {
 }


 public
int HaresInitial;

 /**
  * Returns default value for parameter <code>HaresInitial</code>.
  * <i>This method should not be called by user</i>
  */
 public
int _HaresInitial_DefaultValue_xjal() {
  return
1 /*5000*/
;
 }

 public void set_HaresInitial(
int HaresInitial ) {
  if (HaresInitial == this.HaresInitial) {
    return;
  }
  this.HaresInitial = HaresInitial;
  onChange_HaresInitial();
  onChange();
 }

 void onChange_HaresInitial() {
 }


 public
int LynxInitial;

 /**
  * Returns default value for parameter <code>LynxInitial</code>.
  * <i>This method should not be called by user</i>
  */
 public
int _LynxInitial_DefaultValue_xjal() {
  return
1 /*40*/
;
 }

 public void set_LynxInitial(
int LynxInitial ) {
  if (LynxInitial == this.LynxInitial) {
    return;
  }
  this.LynxInitial = LynxInitial;
  onChange_LynxInitial();
  onChange();
 }

 void onChange_LynxInitial() {
 }

 // Plain Variables

 public
ArrayList<Hare>[]
 HaresInCell;
 public
int[]
 auxGoodCells;

 // Collection Variables

 // Flow/Auxiliary Variables

 // Stock Variables


 // Events

 public EventTimeout _plot_autoUpdateEvent_xjal = new
EventTimeout(this);


 @Override
 public String getNameOf( EventTimeout _e ) {
  if ( _e == _plot_autoUpdateEvent_xjal ) return "plot auto update
event";
   return super.getNameOf( _e );
 }

 @Override
 public int getModeOf( EventTimeout _e ) {
  if ( _e == _plot_autoUpdateEvent_xjal ) return
EVENT_TIMEOUT_MODE_CYCLIC;
  return super.getModeOf( _e );
 }

 @Override
 public double getFirstOccurrenceTime( EventTimeout _e ) {
  if (
     _e == _plot_autoUpdateEvent_xjal
   ) return getEngine().getStartTime();
  return super.getFirstOccurrenceTime( _e );
 }

 @Override
 public double evaluateTimeoutOf( EventTimeout _e ) {
  if ( _e == _plot_autoUpdateEvent_xjal ) return
0.02
;
  return super.evaluateTimeoutOf( _e );
 }

 @Override
 public void executeActionOf( EventTimeout _e ) {
  if ( _e == _plot_autoUpdateEvent_xjal ) {
   plot.updateData();
   return;
  }
  super.executeActionOf( _e );
 }
 // Embedded Objects
```

```
public String getNameOf( ActiveObject ao ) {
  return null;
}

public ActiveObjectArrayList<Lynx> lynx = new ActiveObjectArrayList<Lynx>();
public ActiveObjectArrayList<Hare> hares = new ActiveObjectArrayList<Hare>();

public String getNameOf( ActiveObjectCollection<?> aolist ) {
  if( aolist == lynx ) return "lynx";
  if( aolist == hares ) return "hares";
  return null;
}

/**
 * This method creates and adds new embedded object in the replicated embedded object collection lynx<br>
 * @return newly created embedded object
 */
public Lynx add_lynx() {
  int index = lynx.size();
  Lynx object = instantiate_lynx_xjal( index );
  setupParameters_lynx_xjal( object, index );
  create_lynx_xjal( object, index );
  object.start();
  return object;
}

/**
 * This method creates and adds new embedded object in the replicated embedded object collection lynx<br>
 * This method uses given parameter values to setup created embedded object<br>
 * Index of this new embedded object instance can be obtained through calling <code>lynx.size()</code> method <strong>before</strong> this method is called
 * @param Width
 * @param CellWidth
 * @return newly created embedded object
 */
public Lynx add_lynx( int Width, int CellWidth ) {
  int index = lynx.size();
  Lynx object = instantiate_lynx_xjal( index );
  // Setup parameters
  object.Width = Width;
  object.CellWidth = CellWidth;
  // Finish embedded object creation
  create_lynx_xjal( object, index );
  object.start();
  return object;
}

/**
 * This method removes the given embedded object from the replicated embedded object collection lynx<br>
 * The given object is destroyed, but not immediately in common case.
 * @param object the active object - element of replicated embedded object lynx - which should be removed
 * @return <code>true</code> if object was removed successfully, <code>false</code> if it doesn't belong to lynx
 */
public boolean remove_lynx( Lynx object ) {
  if( ! lynx._remove( object ) ){
    return false;
  }
  object.setDestroyed();
  return true;
}
/**
 * This method creates and adds new embedded object in the replicated embedded object collection hares<br>
 * @return newly created embedded object
 */
public Hare add_hares() {
  int index = hares.size();
  Hare object = instantiate_hares_xjal( index );
  setupParameters_hares_xjal( object, index );
  create_hares_xjal( object, index );
  object.start();
  return object;
}

/**
 * This method creates and adds new embedded object in the replicated embedded object collection hares<br>
 * This method uses given parameter values to setup created embedded object<br>
 * Index of this new embedded object instance can be obtained through calling <code>hares.size()</code> method <strong>before</strong> this method is called
 * @param Width
 * @param CellWidth
 * @param NumberPerBirth
 * @param LifeExpectancy
 * @param Natality
 * @param MaxPerCell
 * @return newly created embedded object
 */
public Hare add_hares( int Width, int CellWidth, int NumberPerBirth, double LifeExpectancy, double Natality, int MaxPerCell ) {
  int index = hares.size();
  Hare object = instantiate_hares_xjal( index );
  // Setup parameters
  object.Width = Width;
  object.CellWidth = CellWidth;
  object.NumberPerBirth = NumberPerBirth;
  object.LifeExpectancy = LifeExpectancy;
  object.Natality = Natality;
  object.MaxPerCell = MaxPerCell;
  // Finish embedded object creation
  create_hares_xjal( object, index );
  object.start();
  return object;
}

/**
 * This method removes the given embedded object from the replicated embedded object collection hares<br>
 * The given object is destroyed, but not immediately in common case.
 * @param object the active object - element of replicated embedded object hares - which should be removed
 * @return <code>true</code> if object was removed successfully, <code>false</code> if it doesn't belong to hares
 */
public boolean remove_hares( Hare object ) {
  if( ! hares._remove( object ) ){
    return false;
  }
  object.setDestroyed();
  return true;
}

/**
 * Creates an embedded object instance and adds it to the end of replicated embedded object list<br>
```

```
   * <i>This method should not be called by user</i>
   */
  private Lynx instantiate_lynx_xjal( final int index ) {Lynx object = new
Lynx( getEngine(), this, lynx );

    lynx._add(object);

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_lynx_xjal(Lynx object, final int index ) {
    object.Width = object._Width_DefaultValue_xjal();
    object.CellWidth = object._CellWidth_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_lynx_xjal(Lynx object, final int index ) {
    object.create();

    // Port connections
  }
  /**
   * Creates an embedded object instance and adds it to the end of
replicated embedded object list<br>
   * <i>This method should not be called by user</i>
   */
  private Hare instantiate_hares_xjal( final int index ) {Hare object = new
Hare( getEngine(), this, hares );

    hares._add(object);

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_hares_xjal(Hare object, final int index ) {
    object.Width =
Width
;
    object.CellWidth =
CellWidth
;
    object.NumberPerBirth =
HareNumberPerBirth
;
    object.LifeExpectancy =
HareLifeExpectancy
;
    object.Natality =
HareNatality
;
    object.MaxPerCell =
HareMaxPerCell
;
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_hares_xjal(Hare object, final int index ) {
    object.create();

    // Port connections
  }

  // Functions


int
 NotOverpopulatedCellAround( int cell ) {

int c = cell % Width;
int r = cell / Width;
int newcell;
int ngood = 0;

for( int i=c-1; i<=c+1; i++ )
  for( int j=r-1; j<=r+1; j++ )
    if( 0 <= i && i <= Width-1 && 0 <= j && j <= Width-1 && ( i != c ||
j != r ) ) {
      newcell = i*Width + j;
      if( HaresInCell[newcell].size() < HareMaxPerCell )
        auxGoodCells[ngood++] = newcell;
    }

if( ngood == 0 )
  return -1; //all cells are overpopulated

//otherwise choose random from what is available
return auxGoodCells[ uniform_discr( 0, ngood-1 ) ];


  }


int
 RandomCellAround( int cell ) {

int c = cell % Width;
int r = cell / Width;
int cnew, rnew;
do {
  cnew = c + uniform_discr( -1, 1 );
  rnew = r + uniform_discr( -1, 1 );
} while( cnew < 0 || cnew > Width-1 || rnew < 0 || rnew > Width-1 );
return rnew*Width + cnew;

  }


double
 XGlobal( int celln, double x ) {

return ( ( celln % Width ) * CellWidth ) + x;

  }


double
 YGlobal( int celln, double x ) {
```

```
return ( (int)(celln/Width) * CellWidth ) + x;

  }
  public DataSet _plot_expression0_dataSet_xjal = new DataSet( 500 ) {
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
lynx.size()
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  public DataSet _plot_expression1_dataSet_xjal = new DataSet( 500 ) {
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
hares.size()
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  static final Font _text_Font = new Font("SansSerif", 1, 12 );
  static final Font _text1_Font = _text_Font;
  static final Font _text2_Font = _text_Font;
  static final Font _text5_Font = new Font("SansSerif", 1, 11 );
  static final Font _text6_Font = _text5_Font;
  static final Font _text7_Font = _text5_Font;
  static final Font _text8_Font = _text5_Font;
  static final Font _text9_Font = _text5_Font;
  static final Font _text10_Font = _text5_Font;
  static final Font _text11_Font = _text5_Font;
  static final Font _text12_Font = _text5_Font;
  static final Font _text14_Font = _text_Font;
  static final Font _text16_Font = new Font("SansSerif", 0, 28 );
  static final Font _text17_Font = _text16_Font;
  static final Font _text18_Font = new Font("SansSerif", 1, 14 );
  static final Font _text19_Font = new Font("SansSerif", 0, 12 );
  static final Color _roundRect_FillColor = new Color( 0xF4F6FA, true );
  static final Color _text_Color = new Color( 0x0, true );
  static final Color _text1_Color = new Color( 0x0, true );
  static final Color _text2_Color = new Color( 0x0, true );
  static final Color _text14_Color = new Color( 0x0, true );
  static final int slider = 1;
  static final int slider1 = 2;
  static final int slider2 = 3;
  static final int slider3 = 4;
  static final int rect123 = 5;
  static final int roundRect = 6;
  static final int text = 7;
  static final int roundRect1 = 8;
  static final int text1 = 9;
  static final int roundRect2 = 10;
  static final int text2 = 11;
  static final int rect2 = 12;
  static final int text5 = 13;
  static final int text6 = 14;
  static final int text7 = 15;
  static final int text8 = 16;
  static final int text9 = 17;
  static final int text10 = 18;
  static final int text11 = 19;
  static final int text12 = 20;
  static final int roundRect3 = 21;
  static final int text14 = 22;
  static final int lynx_presentation = 23;
  static final int hares_presentation = 24;
  static final int text16 = 25;
  static final int text17 = 26;
  static final int text18 = 27;
  static final int line = 28;
  static final int text19 = 29;
  static final int poly = 30;
  static final int _plot = 31;


  /**
   * Top-level presentation group content
   */
  static final int[] _presentation_content = new int[] {
    slider,
    slider1,
    slider2,
    slider3,
    rect123,
    rect2,
    text5,
    text6,
    text7,
    text8,
    text9,
    text10,
    text11,
    text12,
    lynx_presentation,
    hares_presentation,
    text16,
    text17,
    text18,
    line,
    text19,
    poly,
    _plot,
  };

  /**
   * Top-level presentation group id
   */
  static final int presentation = 0;

  /**
   * Top-level icon group id
   */
  static final int icon = -1;

  static final double[] _poly_pointsDX_xjal() {
    return new double[] { 0, -20, -20, 340, 340, 180,  };
  }
  static final double[] _poly_pointsDY_xjal() {
    return new double[] { 0, 0, 310, 310, 0, 0,  };
  }

  @Override
  public void executeShapeControlAction( int _shape, int index, double
value ) {
    switch( _shape ) {
```

```
    case slider:
      set_HareNatality( value );
      break;
    case slider1:
      set_HareNumberPerBirth( (int) value );
      break;
    case slider2:
      set_LynxNatality( value );
      break;
    case slider3:
      set_LynxNumberPerBirth( (int) value );
      break;
    default:
      super.executeShapeControlAction( _shape, index, value );
      break;
  }
 }

 @Override
 public int getShapeControlValueType( int _shape, int index ) {
  switch( _shape ) {
    case slider: return ShapeControl.TYPE_DOUBLE;
    case slider1: return ShapeControl.TYPE_INT;
    case slider2: return ShapeControl.TYPE_DOUBLE;
    case slider3: return ShapeControl.TYPE_INT;
    default: return super.getShapeControlValueType( _shape, index );
  }
 }

 @Override
 public double getShapeControlMinimum( int _shape, int index ) {
  switch( _shape ) {
    case slider: return
1
;
    case slider1: return
2
;
    case slider2: return
0.5
;
    case slider3: return
1
;
    default: return super.getShapeControlMinimum( _shape, index );
  }
 }

 @Override
 public double getShapeControlMaximum( int _shape, int index ) {
  switch( _shape ) {
    case slider: return
6
;
    case slider1: return
9
;
    case slider2: return
3
;
    case slider3: return
5
;
    default: return super.getShapeControlMaximum( _shape, index );
  }
 }

 @Override
 public double getShapeControlDefaultValueDouble( int _shape, int index
) {
   switch(_shape) {
     case slider: return
HareNatality
;
     case slider1: return
HareNumberPerBirth
;
     case slider2: return
LynxNatality
;
     case slider3: return
LynxNumberPerBirth
;
     default: return super.getShapeControlDefaultValueDouble( _shape,
index );
   }
 }

 @Override
 public int[] getShapeGroupContent( int _shape ){
   switch( _shape ) {
     case presentation: return _presentation_content;
     default: return super.getShapeGroupContent( _shape );
   }
 }

 @Override
 public String getNameOfShape( int _shape ) {
  switch( _shape ) {
    case rect123: return "rect123";
    case roundRect: return "roundRect";
    case text: return "text";
    case roundRect1: return "roundRect1";
    case text1: return "text1";
    case roundRect2: return "roundRect2";
    case text2: return "text2";
    case rect2: return "rect2";
    case text5: return "text5";
    case text6: return "text6";
    case text7: return "text7";
    case text8: return "text8";
    case text9: return "text9";
    case text10: return "text10";
    case text11: return "text11";
    case text12: return "text12";
    case roundRect3: return "roundRect3";
    case text14: return "text14";
    case lynx_presentation: return "lynx_presentation";
    case hares_presentation: return "hares_presentation";
    case text16: return "text16";
    case text17: return "text17";
    case text18: return "text18";
    case line: return "line";
    case text19: return "text19";
    case poly: return "poly";
    case slider: return "slider";
    case slider1: return "slider1";
    case slider2: return "slider2";
    case slider3: return "slider3";
    default: return super.getNameOfShape( _shape );
```

```
  }
 }

 @Override
 public int getShapeType( int _shape ) {
  switch( _shape ) {
   case presentation: return SHAPE_GROUP;
   case icon: return SHAPE_GROUP;
   case rect123: return SHAPE_RECTANGLE;
   case roundRect: return SHAPE_ROUNDED_RECTANGLE;
   case text: return SHAPE_TEXT;
   case roundRect1: return SHAPE_ROUNDED_RECTANGLE;
   case text1: return SHAPE_TEXT;
   case roundRect2: return SHAPE_ROUNDED_RECTANGLE;
   case text2: return SHAPE_TEXT;
   case rect2: return SHAPE_RECTANGLE;
   case text5: return SHAPE_TEXT;
   case text6: return SHAPE_TEXT;
   case text7: return SHAPE_TEXT;
   case text8: return SHAPE_TEXT;
   case text9: return SHAPE_TEXT;
   case text10: return SHAPE_TEXT;
   case text11: return SHAPE_TEXT;
   case text12: return SHAPE_TEXT;
   case roundRect3: return SHAPE_ROUNDED_RECTANGLE;
   case text14: return SHAPE_TEXT;
   case lynx_presentation: return SHAPE_EMBEDDED_OBJECT;
   case hares_presentation: return SHAPE_EMBEDDED_OBJECT;
   case text16: return SHAPE_TEXT;
   case text17: return SHAPE_TEXT;
   case text18: return SHAPE_TEXT;
   case line: return SHAPE_LINE;
   case text19: return SHAPE_TEXT;
   case poly: return SHAPE_POLYLINE;
   case slider: return SHAPE_SLIDER;
   case slider1: return SHAPE_SLIDER;
   case slider2: return SHAPE_SLIDER;
   case slider3: return SHAPE_SLIDER;
   default: return super.getShapeType( _shape );
  }
 }

 @Override
 public int getShapeReplication( int _shape ) {
  switch( _shape ) {
   case lynx_presentation: return
lynx.size()
;
   case hares_presentation: return
hares.size()
;
   default: return super.getShapeReplication( _shape );
  }
 }

 @Override
 public double getShapeX( int _shape, int index ) {
  switch( _shape ) {
   case rect123: return 0;
   case roundRect: return -420;
   case text: return -410;
   case roundRect1: return -220;
   case text1: return -210;
   case roundRect2: return -220;
   case text2: return -210;
   case rect2: return 440;
   case text5: return 70;
   case text6: return 210;
   case text7: return 70;
   case text8: return 210;
   case text9: return 70;
   case text10: return 210;
   case text11: return 70;
   case text12: return 210;
   case roundRect3: return -420;
   case text14: return -410;
   case lynx_presentation: return 445;
   case hares_presentation: return 445;
   case text16: return 50;
   case text17: return 230;
   case text18: return 780;
   case line: return 50;
   case text19: return 80;
   case poly: return 70;
   case slider: return 240;
   case slider1: return 240;
   case slider2: return 240;
   case slider3: return 240;
   default: return super.getShapeX( _shape, index );
  }
 }

 @Override
 public double getShapeY( int _shape, int index ) {
  switch( _shape ) {
   case rect123: return 0;
   case roundRect: return 30;
   case text: return 10;
   case roundRect1: return 30;
   case text1: return 10;
   case roundRect2: return 181;
   case text2: return 160;
   case rect2: return 100;
   case text5: return 130;
   case text6: return 130;
   case text7: return 160;
   case text8: return 160;
   case text9: return 280;
   case text10: return 280;
   case text11: return 310;
   case text12: return 310;
   case roundRect3: return 370;
   case text14: return 350;
   case lynx_presentation: return 105;
   case hares_presentation: return 105;
   case text16: return 30;
   case text17: return 30;
   case text18: return 10;
   case line: return 60;
   case text19: return 90;
   case poly: return 100;
   case slider: return 130;
   case slider1: return 160;
   case slider2: return 280;
   case slider3: return 310;
   default: return super.getShapeY( _shape, index );
  }
 }

 @Override
```

```
public Object getShapeEmbeddedObject( int _shape ) {
  switch( _shape ) {
    case lynx_presentation: return lynx;
    case hares_presentation: return hares;
    default: return super.getShapeEmbeddedObject( _shape );
  }
}

@Override
public Color getShapeLineColor( int _shape, int index ) {
  switch( _shape ) {
    // shapes
    case rect123: return null;
    case roundRect: return steelBlue;
    case roundRect1: return steelBlue;
    case roundRect2: return steelBlue;
    case rect2: return null;
    case roundRect3: return steelBlue;
    // controls (text color)
    // charts (border color)
    default: return super.getShapeLineColor( _shape, index );
  }
}

@Override
public Color getShapeFillColor( int _shape, int index ) {
  switch( _shape ) {
    case rect123: return paleGoldenRod;
    case roundRect: return _roundRect_FillColor;
    case text: return _text_Color;
    case roundRect1: return null;
    case text1: return _text1_Color;
    case roundRect2: return null;
    case text2: return _text2_Color;
    case rect2: return black;
    case text5: return green;
    case text6: return black;
    case text7: return green;
    case text8: return black;
    case text9: return red;
    case text10: return black;
    case text11: return red;
    case text12: return black;
    case roundRect3: return null;
    case text14: return _text14_Color;
    case text16: return gray;
    case text17: return red;
    case text18: return green;
    case text19: return black;
    case poly: return null;
    case slider: return transparent;
    case slider1: return transparent;
    case slider2: return transparent;
    case slider3: return transparent;
    default: return super.getShapeFillColor( _shape, index );
  }
}

@Override
public int getShapeLineStyle( int _shape, int index ) {
  switch( _shape ) {
    case rect123: return LINE_STYLE_SOLID;
    case roundRect: return LINE_STYLE_SOLID;
    case roundRect1: return LINE_STYLE_SOLID;
    case roundRect2: return LINE_STYLE_SOLID;
    case rect2: return LINE_STYLE_SOLID;
    case roundRect3: return LINE_STYLE_SOLID;
    case line: return LINE_STYLE_SOLID;
    case poly: return LINE_STYLE_SOLID;
    default: return super.getShapeLineStyle( _shape, index );
  }
}

@Override
public double getShapeLineWidth( int _shape, int index ) {
  switch( _shape ) {
    case line: return 0;
    case poly: return 0;
    default: return super.getShapeLineWidth( _shape, index );
  }
}

@Override
public double getShapeLineDx( int _shape, int index ) {
  switch( _shape ) {
    case line: return 700;
    default: return super.getShapeLineDx( _shape, index );
  }
}

@Override
public double getShapeLineDy( int _shape, int index ) {
  switch( _shape ) {
    case line: return 0;
    default: return super.getShapeLineDy( _shape, index );
  }
}

@Override
public double getShapeWidth( int _shape, int index ) {
  switch( _shape ) {
    case rect123: return 800;
    case roundRect: return 190;
    case roundRect1: return 190;
    case roundRect2: return 190;
    case rect2: return 310;
    case roundRect3: return 390;
    case slider: return 160;
    case slider1: return 160;
    case slider2: return 160;
    case slider3: return 160;
    default: return super.getShapeWidth( _shape, index );
  }
}

@Override
public double getShapeHeight( int _shape, int index ) {
  switch( _shape ) {
    case rect123: return 600;
    case roundRect: return 310;
    case roundRect1: return 120;
    case roundRect2: return 159;
    case rect2: return 310;
    case roundRect3: return 40;
    case slider: return 20;
    case slider1: return 20;
    case slider2: return 20;
    case slider3: return 20;
    default: return super.getShapeHeight( _shape, index );
  }
```

```
 }

 @Override
 public double getShapeRadiusX( int _shape, int index ) {
  switch( _shape ) {
    case roundRect: return 10;
    case roundRect1: return 10;
    case roundRect2: return 10;
    case roundRect3: return 10;
    default: return super.getShapeRadiusX( _shape, index );
  }
 }

 @Override
 public double getShapeRadiusY( int _shape, int index ) {
  switch( _shape ) {
    case roundRect: return 10;
    case roundRect1: return 10;
    case roundRect2: return 10;
    case roundRect3: return 10;
    default: return super.getShapeRadiusY( _shape, index );
  }
 }

 @Override
 public int getShapeNPoints( int _shape, int index ) {
  switch( _shape ) {
    case poly: return 6;
    default: return super.getShapeNPoints( _shape, index );
  }
 }

 private static final double[] _poly_pointsDX = _poly_pointsDX_xjal();
 private static final double[] _poly_pointsDY = _poly_pointsDY_xjal();

 @Override
 public double[] getShapePointsDx( int _shape, int index ) {
  int npoints;
  double[] dx;
  switch( _shape ) {
    case poly: return _poly_pointsDX;
    default: return super.getShapePointsDx( _shape, index );
  }
 }

 @Override
 public double[] getShapePointsDy( int _shape, int index ) {
  int npoints;
  double[] dy;
  switch( _shape ) {
    case poly: return _poly_pointsDY;
    default: return super.getShapePointsDy( _shape, index );
  }
 }



 @Override
 public Object getShapeText( int _shape, int index ) {
  switch( _shape ) {
    case text: return "Parameters";
    case text1: return "Variables";
    case text2: return "Functions";
    case text5: return "Hare Births per Year:";
    case text6: return
format( HareNatality )
;
    case text7: return "Hare Babies per Birth:";
    case text8: return
format( HareNumberPerBirth )
;
    case text9: return "Lynx Births per Year:";
    case text10: return
format( LynxNatality )
;
    case text11: return "Lynx Babies per Birth:";
    case text12: return
format( LynxNumberPerBirth )
;
    case text14: return "Embedded Objects";
    case text16: return "Predator Prey";
    case text17: return "Agent Based Model";
    case text18: return "[simulation]";
    case text19: return "Change parameters on-the-fly";
    default: return super.getShapeText( _shape, index );
   }
  }

  @Override
  public Font getShapeFont( int _shape, int index ) {
   switch( _shape ) {
    case text: return _text_Font;
    case text1: return _text1_Font;
    case text2: return _text2_Font;
    case text5: return _text5_Font;
    case text6: return _text6_Font;
    case text7: return _text7_Font;
    case text8: return _text8_Font;
    case text9: return _text9_Font;
    case text10: return _text10_Font;
    case text11: return _text11_Font;
    case text12: return _text12_Font;
    case text14: return _text14_Font;
    case text16: return _text16_Font;
    case text17: return _text17_Font;
    case text18: return _text18_Font;
    case text19: return _text19_Font;
    default: return super.getShapeFont( _shape, index );
   }
  }

  @Override
  public int getShapeTextAlignment( int _shape, int index ) {
   switch( _shape ) {
    case text: return ALIGNMENT_LEFT;
    case text1: return ALIGNMENT_LEFT;
    case text2: return ALIGNMENT_LEFT;
    case text5: return ALIGNMENT_LEFT;
    case text6: return ALIGNMENT_LEFT;
    case text7: return ALIGNMENT_LEFT;
    case text8: return ALIGNMENT_LEFT;
    case text9: return ALIGNMENT_LEFT;
    case text10: return ALIGNMENT_LEFT;
    case text11: return ALIGNMENT_LEFT;
    case text12: return ALIGNMENT_LEFT;
    case text14: return ALIGNMENT_LEFT;
    case text16: return ALIGNMENT_LEFT;
    case text17: return ALIGNMENT_LEFT;
    case text18: return ALIGNMENT_RIGHT;
```

```
    case text19: return ALIGNMENT_LEFT;
    default: return super.getShapeTextAlignment( _shape, index );
   }
  }

  @Override
  public boolean isShapeControlVertical( int _shape, int index ) {
   switch( _shape ) {
    case slider: return false;
    case slider1: return false;
    case slider2: return false;
    case slider3: return false;
    default: return super.isShapeControlVertical( _shape, index );
   }
  }


  TimePlot plot;

  @Override
  public Object getPersistentShape( int _shape ) {
   switch(_shape){

    case _plot: return plot;
    default: return null;
   }
  }


  @Override
  public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
    if (!_publicOnly) {
     drawParameter( _panel, _g, -400, 150, 20, 0, "HareNatality",
HareNatality, false, false );
    }
    if (!_publicOnly) {
     drawParameter( _panel, _g, -400, 190, 20, 0, "HareLifeExpectancy",
HareLifeExpectancy, false, false );
    }
    if (!_publicOnly) {
     drawParameter( _panel, _g, -400, 170, 20, 0,
"HareNumberPerBirth", HareNumberPerBirth, false, false );
    }
    if (!_publicOnly) {
     drawParameter( _panel, _g, -400, 210, 20, 0, "HareMaxPerCell",
HareMaxPerCell, false, false );
    }
    if (!_publicOnly) {
     drawParameter( _panel, _g, -400, 120, 20, 0, "CellWidth", CellWidth,
false, false );
    }
    if (!_publicOnly) {
     drawParameter( _panel, _g, -400, 100, 20, 0, "Width", Width, false,
false );
    }
    if (!_publicOnly) {
     drawParameter( _panel, _g, -400, 240, 20, 0, "LynxNatality",
LynxNatality, false, false );
    }
    if (!_publicOnly) {
     drawParameter( _panel, _g, -400, 300, 20, 0, "LynxHuntingPeriod",
LynxHuntingPeriod, false, false );
    }
    if (!_publicOnly) {
     drawParameter( _panel, _g, -400, 260, 20, 0, "LynxNumberPerBirth",
LynxNumberPerBirth, false, false );
    }
    if (!_publicOnly) {
     drawParameter( _panel, _g, -400, 320, 20, 0,
"LynxHungerDeathThreshold", LynxHungerDeathThreshold, false, false );
    }
    if (!_publicOnly) {
     drawParameter( _panel, _g, -400, 280, 20, 0, "LynxLifeExpectancy",
LynxLifeExpectancy, false, false );
    }
    if (!_publicOnly) {
     drawParameter( _panel, _g, -400, 50, 20, 0, "HaresInitial",
HaresInitial, false, false );
    }
    if (!_publicOnly) {
     drawParameter( _panel, _g, -400, 70, 20, 0, "LynxInitial",
LynxInitial, false, false );
    }
    if (!_publicOnly) {
     drawPlainVariable( _panel, _g, -200, 50, 20, 0, "HaresInCell",
HaresInCell, false );
    }
    if (!_publicOnly) {
     drawPlainVariable( _panel, _g, -200, 80, 20, 0, "auxGoodCells",
auxGoodCells, false );
    }
    if (!_publicOnly) {
     drawFunction( _panel, _g, -200, 210, 20, 0,
"NotOverpopulatedCellAround");
    }
    if (!_publicOnly) {
     drawFunction( _panel, _g, -200, 230, 20, 0, "RandomCellAround");
    }
    if (!_publicOnly) {
     drawFunction( _panel, _g, -200, 260, 20, 0, "XGlobal");
    }
    if (!_publicOnly) {
     drawFunction( _panel, _g, -200, 280, 20, 0, "YGlobal");
    }
    // Embedded object "lynx"
    if (!_publicOnly) {
     drawEmbeddedObjectModelDefault( _panel, _g, -400 , 390 , 15, 0,
"lynx", this.lynx );
    }
    // Embedded object "hares"
    if (!_publicOnly) {
     drawEmbeddedObjectModelDefault( _panel, _g, -200 , 390 , 15, 0,
"hares", this.hares );
    }
   }


   @Override
   public boolean onClickModelAt( Panel panel, double x, double y, int
clickCount, boolean publicOnly ) {
    if( !publicOnly && modelElementContains(x, y, -400, 150) ) {
     panel.addInspect( -400, 150, this, "HareNatality" );
     return true;
    }
    if( !publicOnly && modelElementContains(x, y, -400, 190) ) {
     panel.addInspect( -400, 190, this, "HareLifeExpectancy" );
     return true;
    }
    if( !publicOnly && modelElementContains(x, y, -400, 170) ) {
     panel.addInspect( -400, 170, this, "HareNumberPerBirth" );
     return true;
    }
    if( !publicOnly && modelElementContains(x, y, -400, 210) ) {
     panel.addInspect( -400, 210, this, "HareMaxPerCell" );
     return true;
    }
```

```
  if( !publicOnly && modelElementContains(x, y, -400, 120) ) {
   panel.addInspect( -400, 120, this, "CellWidth" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -400, 100) ) {
   panel.addInspect( -400, 100, this, "Width" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -400, 240) ) {
   panel.addInspect( -400, 240, this, "LynxNatality" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -400, 300) ) {
   panel.addInspect( -400, 300, this, "LynxHuntingPeriod" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -400, 260) ) {
   panel.addInspect( -400, 260, this, "LynxNumberPerBirth" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -400, 320) ) {
   panel.addInspect( -400, 320, this, "LynxHungerDeathThreshold" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -400, 280) ) {
   panel.addInspect( -400, 280, this, "LynxLifeExpectancy" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -400, 50) ) {
   panel.addInspect( -400, 50, this, "HaresInitial" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -400, 70) ) {
   panel.addInspect( -400, 70, this, "LynxInitial" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -200, 50) ) {
   panel.addInspect( -200, 50, this, "HaresInCell" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -200, 80) ) {
   panel.addInspect( -200, 80, this, "auxGoodCells" );
   return true;
  }
  if ( !lynx.isEmpty() && modelElementContains(x, y, -400, 390) ) {
   if ( clickCount == 2 ) {
    panel.browseEmbeddedObject( -400, 390, this, "lynx" );
   } else {
    panel.addInspect( -400, 390, this, "lynx" );
   }
   return true;
  }
  if ( !hares.isEmpty() && modelElementContains(x, y, -200, 390) ) {
   if ( clickCount == 2 ) {
    panel.browseEmbeddedObject( -200, 390, this, "hares" );
   } else {
    panel.addInspect( -200, 390, this, "hares" );
   }
   return true;
  }
  return false;
 }

 /**
  * Constructor
  */
 public Main( Engine engine, ActiveObject owner,
ActiveObjectCollection<? extends Main> collection ) {
  super( engine, owner, collection );
 }

 @Override
 public void create() {
  // Creating embedded object instances
  for ( int i = 0; i <
LynxInitial
; i++ ) {
   instantiate_lynx_xjal( i );
  }
  for ( int i = 0; i <
HaresInitial
; i++ ) {
   instantiate_hares_xjal( i );
  }
  // Assigning initial values for plain variables
  HaresInCell =
new ArrayList[ Width * Width ]
;
  auxGoodCells =
new int[8]
;
  // Dynamic initialization of persistent elements
  {
  DataSet _item;
  List<DataSet> _items = new ArrayList<DataSet>( 2 );
  _items.add( _plot_expression0_dataSet_xjal );
  _items.add( _plot_expression1_dataSet_xjal );
  List<String> _titles = new ArrayList<String>( 2 );
  _titles.add( "Lynx" );
  _titles.add( "Hares" );
  List<Chart2DPlot.Appearance> _appearances = new
ArrayList<Chart2DPlot.Appearance>( 2 );
  _appearances.add( new Chart2DPlot.Appearance( red, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
  _appearances.add( new Chart2DPlot.Appearance( forestGreen, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
  plot = new TimePlot(
                    Main.this, true, 10, 430,
                        770, 150,
       null, null,
       40, 20,
                        700, 80, new Color(
0xFFF1FCDE, true ), black, green,
       30, Chart.SOUTH,

10
       , null, Chart.SCALE_AUTO,
       0, 0, Chart.GRID_DEFAULT, Chart.GRID_DEFAULT,
       gray, green, _items, _titles, _appearances
  );
  }
          // Port connectors with non-replicated objects
  // Creating replicated embedded objects
  for ( int i = 0; i < lynx.size(); i++ ) {
   setupParameters_lynx_xjal( lynx.get(i), i );
   create_lynx_xjal( lynx.get(i), i );
  }
  for ( int i = 0; i < hares.size(); i++ ) {
   setupParameters_hares_xjal( hares.get(i), i );
   create_hares_xjal( hares.get(i), i );
  }
  assignInitialConditions();
```

```
    onCreate();
  }

  @Override
  public void start() {
    _plot_autoUpdateEvent_xjal.start();
    for (ActiveObject embeddedObject : lynx){
      embeddedObject.start();
    }
    for (ActiveObject embeddedObject : hares){
      embeddedObject.start();
    }
    onStartup();
  }

  public void onStartup() {
    super.onStartup();

for( int i=0; i<HaresInCell.length; i++ )
  HaresInCell[i] = new ArrayList();
for( Hare h : hares )
  HaresInCell[h.cell].add( h );

  }

  public List<Object> getEmbeddedObjects() {
    LinkedList<Object> list = new LinkedList<Object>();
    list.add( lynx );
    list.add( hares );
    return list;
  }

  public void onDestroy() {
    super.onDestroy();
    _plot_autoUpdateEvent_xjal.onDestroy();
    for (ActiveObject embeddedObject : lynx) {
      embeddedObject.onDestroy();
    }
    for (ActiveObject embeddedObject : hares) {
      embeddedObject.onDestroy();
    }
  }


}
```

Simulation.java

```
package predator_prey_agent_based;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import javax.swing.JApplet;

public class Simulation extends ExperimentSimulation<Main> {
  // Plain Variables
  public int  HaresInitial = 5000 ;
  public int  LynxInitial = 40 ;

  static final Font _button_Font = new Font("Dialog", 0, 12 );
  static final Font _text_Font = new Font("SansSerif", 0, 28 );
  static final Font _text1_Font = _text_Font;
  static final Font _text2_Font = new Font("SansSerif", 1, 14 );
  static final Font _text5_Font = new Font("SansSerif", 0, 12 );
  static final Font _text7_Font = new Font("SansSerif", 1, 12 );
  static final Font _text8_Font = _text7_Font;
  static final Font _text9_Font = _text7_Font;
  static final Font _text10_Font = _text7_Font;
  static final Font _text11_Font = new Font("SansSerif", 0, 10 );
  static final Font _text12_Font = _text7_Font;
  static final Font _text13_Font = _text7_Font;
  static final Font _text14_Font = _text5_Font;
  static final Font _text15_Font = _text5_Font;
  static final int slider = 1;
  static final int slider1 = 2;
  static final int button = 3;
  static final int rect = 4;
  static final int poly1 = 5;
  static final int text = 6;
  static final int text1 = 7;
  static final int text2 = 8;
  static final int text5 = 9;
  static final int text7 = 10;
  static final int text8 = 11;
  static final int text9 = 12;
  static final int text10 = 13;
```

```
 static final int image = 14;
 static final int text11 = 15;
 static final int text12 = 16;
 static final int poly = 17;
 static final int text13 = 18;
 static final int text14 = 19;
 static final int line = 20;
 static final int text15 = 21;


 /**
  * Top-level presentation group content
  */
 static final int[] _presentation_content = new int[] {
  slider,
  slider1,
  button,
  rect,
  poly1,
  text,
  text1,
  text2,
  text5,
  text7,
  text8,
  text9,
  text10,
  image,
  text11,
  text12,
  poly,
  text13,
  text14,
  line,
  text15,
 };

 /**
  * Top-level presentation group id
  */
 static final int presentation = 0;

 /**
  * Top-level icon group id
  */
 static final int icon = -1;

 static final double[] _poly1_pointsDX_xjal() {
  return new double[] { 0, -15, -15, 360, 360, 145,  };
 }
 static final double[] _poly1_pointsDY_xjal() {
  return new double[] { 0, 0, 90, 90, 0, 0,  };
 }
 static final double[] _poly_pointsDX_xjal() {
  return new double[] { 0, -15, -15, 360, 360, 285,  };
 }
 static final double[] _poly_pointsDY_xjal() {
  return new double[] { 0, 0, 90, 90, 0, 0,  };
 }

 @Override
 public void executeShapeControlAction( int _shape, int index ) {
  switch( _shape ) {
    case button: {
run();
;}
      break;
    default:
      super.executeShapeControlAction( _shape, index );
      break;
  }
 }

 @Override
 public void executeShapeControlAction( int _shape, int index, double
value ) {
  switch( _shape ) {
    case slider:
      HaresInitial = (int) value;
      break;
    case slider1:
      LynxInitial = (int) value;
      break;
    default:
      super.executeShapeControlAction( _shape, index, value );
      break;
  }
 }

 @Override
 public int getShapeControlValueType( int _shape, int index ) {
  switch( _shape ) {
    case slider: return ShapeControl.TYPE_INT;
    case slider1: return ShapeControl.TYPE_INT;
    default: return super.getShapeControlValueType( _shape, index );
  }
 }

 @Override
 public double getShapeControlMinimum( int _shape, int index ) {
  switch( _shape ) {
    case slider: return
1
;
    case slider1: return
1
;
    default: return super.getShapeControlMinimum( _shape, index );
  }
 }

 @Override
 public double getShapeControlMaximum( int _shape, int index ) {
  switch( _shape ) {
    case slider: return
10000
;
    case slider1: return
100
;
    default: return super.getShapeControlMaximum( _shape, index );
  }
 }

 @Override
 public double getShapeControlDefaultValueDouble( int _shape, int index
) {
  switch(_shape) {
    case slider: return
HaresInitial
;
```

```
    case slider1: return
LynxInitial
;
    default:  return  super.getShapeControlDefaultValueDouble(  _shape,
index );
   }
 }

 @Override
 public int[] getShapeGroupContent( int _shape ){
    switch( _shape ) {
      case presentation: return _presentation_content;
      default: return super.getShapeGroupContent( _shape );
    }
 }


 @Override
 public String getNameOfShape( int _shape ) {
  switch( _shape ) {
    case rect: return "rect";
    case poly1: return "poly1";
    case text: return "text";
    case text1: return "text1";
    case text2: return "text2";
    case text5: return "text5";
    case text7: return "text7";
    case text8: return "text8";
    case text9: return "text9";
    case text10: return "text10";
    case image: return "image";
    case text11: return "text11";
    case text12: return "text12";
    case poly: return "poly";
    case text13: return "text13";
    case text14: return "text14";
    case line: return "line";
    case text15: return "text15";
    case slider: return "slider";
    case slider1: return "slider1";
    case button: return "button";
    default: return super.getNameOfShape( _shape );
  }
 }

 @Override
 public int getShapeType( int _shape ) {
  switch( _shape ) {
    case presentation: return SHAPE_GROUP;
    case icon: return SHAPE_GROUP;
    case rect: return SHAPE_RECTANGLE;
    case poly1: return SHAPE_POLYLINE;
    case text: return SHAPE_TEXT;
    case text1: return SHAPE_TEXT;
    case text2: return SHAPE_TEXT;
    case text5: return SHAPE_TEXT;
    case text7: return SHAPE_TEXT;
    case text8: return SHAPE_TEXT;
    case text9: return SHAPE_TEXT;
    case text10: return SHAPE_TEXT;
    case image: return SHAPE_IMAGE;
    case text11: return SHAPE_TEXT;
    case text12: return SHAPE_TEXT;
    case poly: return SHAPE_POLYLINE;
    case text13: return SHAPE_TEXT;
    case text14: return SHAPE_TEXT;
    case line: return SHAPE_LINE;
    case text15: return SHAPE_TEXT;
    case slider: return SHAPE_SLIDER;
    case slider1: return SHAPE_SLIDER;
    case button: return SHAPE_BUTTON;
    default: return super.getShapeType( _shape );
  }
 }

 @Override
 public double getShapeX( int _shape, int index ) {
  switch( _shape ) {
    case rect: return 0;
    case poly1: return 65;
    case text: return 50;
    case text1: return 230;
    case text2: return 780;
    case text5: return 120;
    case text7: return 80;
    case text8: return 130;
    case text9: return 80;
    case text10: return 130;
    case image: return 50;
    case text11: return 90;
    case text12: return 70;
    case poly: return 65;
    case text13: return 70;
    case text14: return 120;
    case line: return 50;
    case text15: return 50;
    case slider: return 180;
    case slider1: return 180;
    case button: return 80;
    default: return super.getShapeX( _shape, index );
  }
 }

 @Override
 public double getShapeY( int _shape, int index ) {
  switch( _shape ) {
    case rect: return 0;
    case poly1: return 270;
    case text: return 30;
    case text1: return 30;
    case text2: return 10;
    case text5: return 130;
    case text7: return 165;
    case text8: return 165;
    case text9: return 195;
    case text10: return 195;
    case image: return 530;
    case text11: return 570;
    case text12: return 130;
    case poly: return 140;
    case text13: return 260;
    case text14: return 260;
    case line: return 60;
    case text15: return 80;
    case slider: return 165;
    case slider1: return 195;
    case button: return 290;
    default: return super.getShapeY( _shape, index );
  }
 }
```

```
@Override
public Color getShapeLineColor( int _shape, int index ) {
  switch( _shape ) {
    // shapes
    case rect: return null;
    // controls (text color)
    case button: return black;
    // charts (border color)
    default: return super.getShapeLineColor( _shape, index );
  }
}

@Override
public Color getShapeFillColor( int _shape, int index ) {
  switch( _shape ) {
    case rect: return paleGoldenRod;
    case poly1: return null;
    case text: return gray;
    case text1: return red;
    case text2: return green;
    case text5: return black;
    case text7: return green;
    case text8: return gray;
    case text9: return red;
    case text10: return gray;
    case text11: return gray;
    case text12: return black;
    case poly: return null;
    case text13: return black;
    case text14: return black;
    case text15: return black;
    case slider: return transparent;
    case slider1: return transparent;
    case button: return controlDefault;
    default: return super.getShapeFillColor( _shape, index );
  }
}

@Override
public int getShapeLineStyle( int _shape, int index ) {
  switch( _shape ) {
    case rect: return LINE_STYLE_SOLID;
    case poly1: return LINE_STYLE_SOLID;
    case poly: return LINE_STYLE_SOLID;
    case line: return LINE_STYLE_SOLID;
    default: return super.getShapeLineStyle( _shape, index );
  }
}

@Override
public double getShapeLineWidth( int _shape, int index ) {
  switch( _shape ) {
    case poly1: return 0;
    case poly: return 0;
    case line: return 0;
    default: return super.getShapeLineWidth( _shape, index );
  }
}

@Override
public double getShapeLineDx( int _shape, int index ) {
  switch( _shape ) {
    case line: return 700;
    default: return super.getShapeLineDx( _shape, index );
  }
}

@Override
public double getShapeLineDy( int _shape, int index ) {
  switch( _shape ) {
    case line: return 0;
    default: return super.getShapeLineDy( _shape, index );
  }
}

@Override
public double getShapeWidth( int _shape, int index ) {
  switch( _shape ) {
    case rect: return 800;
    case image: return 116;
    case slider: return 220;
    case slider1: return 220;
    case button: return 60;
    default: return super.getShapeWidth( _shape, index );
  }
}

@Override
public double getShapeHeight( int _shape, int index ) {
  switch( _shape ) {
    case rect: return 600;
    case image: return 40;
    case slider: return 20;
    case slider1: return 20;
    case button: return 30;
    default: return super.getShapeHeight( _shape, index );
  }
}

@Override
public int getShapeNPoints( int _shape, int index ) {
  switch( _shape ) {
    case poly1: return 6;
    case poly: return 6;
    default: return super.getShapeNPoints( _shape, index );
  }
}

private static final double[] _poly1_pointsDX = _poly1_pointsDX_xjal();
private static final double[] _poly1_pointsDY = _poly1_pointsDY_xjal();
private static final double[] _poly_pointsDX = _poly_pointsDX_xjal();
private static final double[] _poly_pointsDY = _poly_pointsDY_xjal();

@Override
public double[] getShapePointsDx( int _shape, int index ) {
  int npoints;
  double[] dx;
  switch( _shape ) {
    case poly1: return _poly1_pointsDX;
    case poly: return _poly_pointsDX;
    default: return super.getShapePointsDx( _shape, index );
  }
}

@Override
public double[] getShapePointsDy( int _shape, int index ) {
  int npoints;
  double[] dy;
  switch( _shape ) {
    case poly1: return _poly1_pointsDY;
```

```
    case poly: return _poly_pointsDY;
    default: return super.getShapePointsDy( _shape, index );
   }
  }

  // Images files
  static final String[] _image_filenames = {
    "xjlogo.png",
  };

  @Override
  public String getShapePackagePrefix( int shape ) {
   switch( shape ) {
    case image: return "/predator_prey_agent_based/";
    default: return super.getShapePackagePrefix( shape );
   }
  }

  @Override
  public String[] getShapeImageFileNames( int shape, int index ) {
   switch( shape ) {
    case image: return _image_filenames;
    default: return super.getShapeImageFileNames( shape, index );
   }
  }


  @Override
  public Object getShapeText( int _shape, int index ) {
   switch( _shape ) {
    case button: return "Run";
    case text: return "Predator Prey";
    case text1: return "Agent Based Model";
    case text2: return "[experiment setup]";
    case text5: return "Set the initial number of hares and lynx";
    case text7: return "Hares:";
    case text8: return
format( HaresInitial )
;
    case text9: return "Lynx:";
    case text10: return
format( LynxInitial )
;
    case text11: return "AnyLogic and this model is (c) XJ Technologies,
www.anylogic.com. All rights reserved.";
    case text12: return "Step 1.";
    case text13: return "Step 2.";
    case text14: return "Run the model";
    case text15: return "The predator prey problem is a simulation that
attempts to predict the relationship in populations between a
population\r\nof lynx and hares isolated on an island. Enjoy this old
classic example with the help of the new Agent Based technology!";
    default: return super.getShapeText( _shape, index );
   }
  }


  @Override
  public Font getShapeFont( int _shape, int index ) {
   switch( _shape ) {
    case text: return _text_Font;
    case text1: return _text1_Font;
    case text2: return _text2_Font;
    case text5: return _text5_Font;
    case text7: return _text7_Font;
    case text8: return _text8_Font;
    case text9: return _text9_Font;
    case text10: return _text10_Font;
    case text11: return _text11_Font;
    case text12: return _text12_Font;
    case text13: return _text13_Font;
    case text14: return _text14_Font;
    case text15: return _text15_Font;
    case button: return _button_Font;
    default: return super.getShapeFont( _shape, index );
   }
  }

  @Override
  public int getShapeTextAlignment( int _shape, int index ) {
   switch( _shape ) {
    case text: return ALIGNMENT_LEFT;
    case text1: return ALIGNMENT_LEFT;
    case text2: return ALIGNMENT_RIGHT;
    case text5: return ALIGNMENT_LEFT;
    case text7: return ALIGNMENT_LEFT;
    case text8: return ALIGNMENT_LEFT;
    case text9: return ALIGNMENT_LEFT;
    case text10: return ALIGNMENT_LEFT;
    case text11: return ALIGNMENT_LEFT;
    case text12: return ALIGNMENT_LEFT;
    case text13: return ALIGNMENT_LEFT;
    case text14: return ALIGNMENT_LEFT;
    case text15: return ALIGNMENT_LEFT;
    default: return super.getShapeTextAlignment( _shape, index );
   }
  }

  @Override
  public boolean isShapeControlEnabled( int _shape, int index ) {
   switch( _shape ) {
    case slider: return
getEngine().getState() == Engine.IDLE
;
    case slider1: return
getEngine().getState() == Engine.IDLE
;
    case button: return
getEngine().getState() == Engine.IDLE
;
    default: return super.isShapeControlEnabled( _shape, index );
   }
  }

  @Override
  public boolean isShapeControlVertical( int _shape, int index ) {
   switch( _shape ) {
    case slider: return false;
    case slider1: return false;
    default: return super.isShapeControlVertical( _shape, index );
   }
  }



  @Override
  public int getWindowWidth() {
   return 800;
  }

  @Override
```

```
 public int getWindowHeight() {
  return 600;
 }

 /**
  * Applet class to run experiment as java applet
  */
 public static class Applet extends JApplet {

  Simulation ex;

  @Override
  public void init() {
   ex = new Simulation();
   ex.setup( this );
  }

  @Override
  public void destroy() {
   ex.close();
  }

 }

 @Override
 public void initDefaultRandomNumberGenerator(Engine engine) {
  engine.getDefaultRandomGenerator().setSeed( 1 );
 }

 @Override
 public Main createRoot( Engine engine ) {
  // Create the root object
  return new Main( engine, null, null );
 }

 @Override
 public void setupRootParameters( Main root, boolean
callOnChangeActions ) {
   double HareNatality_xjal = root._HareNatality_DefaultValue_xjal();
   if (callOnChangeActions) {
    root.set_HareNatality( HareNatality_xjal );
   } else {
    root.HareNatality = HareNatality_xjal;
   }
   double HareLifeExpectancy_xjal =
root._HareLifeExpectancy_DefaultValue_xjal();
   if (callOnChangeActions) {
    root.set_HareLifeExpectancy( HareLifeExpectancy_xjal );
   } else {
    root.HareLifeExpectancy = HareLifeExpectancy_xjal;
   }
   int HareNumberPerBirth_xjal =
root._HareNumberPerBirth_DefaultValue_xjal();
   if (callOnChangeActions) {
    root.set_HareNumberPerBirth( HareNumberPerBirth_xjal );
   } else {
    root.HareNumberPerBirth = HareNumberPerBirth_xjal;
   }
   int HareMaxPerCell_xjal = root._HareMaxPerCell_DefaultValue_xjal();
   if (callOnChangeActions) {
    root.set_HareMaxPerCell( HareMaxPerCell_xjal );
   } else {
    root.HareMaxPerCell = HareMaxPerCell_xjal;
   }
   int CellWidth_xjal = root._CellWidth_DefaultValue_xjal();
   if (callOnChangeActions) {
    root.set_CellWidth( CellWidth_xjal );
   } else {
    root.CellWidth = CellWidth_xjal;
   }
   int Width_xjal = root._Width_DefaultValue_xjal();
   if (callOnChangeActions) {
    root.set_Width( Width_xjal );
   } else {
    root.Width = Width_xjal;
   }
   double LynxNatality_xjal = root._LynxNatality_DefaultValue_xjal();
   if (callOnChangeActions) {
    root.set_LynxNatality( LynxNatality_xjal );
   } else {
    root.LynxNatality = LynxNatality_xjal;
   }
   double LynxHuntingPeriod_xjal =
root._LynxHuntingPeriod_DefaultValue_xjal();
   if (callOnChangeActions) {
    root.set_LynxHuntingPeriod( LynxHuntingPeriod_xjal );
   } else {
    root.LynxHuntingPeriod = LynxHuntingPeriod_xjal;
   }
   int LynxNumberPerBirth_xjal =
root._LynxNumberPerBirth_DefaultValue_xjal();
   if (callOnChangeActions) {
    root.set_LynxNumberPerBirth( LynxNumberPerBirth_xjal );
   } else {
    root.LynxNumberPerBirth = LynxNumberPerBirth_xjal;
   }
   double LynxHungerDeathThreshold_xjal =
root._LynxHungerDeathThreshold_DefaultValue_xjal();
   if (callOnChangeActions) {
    root.set_LynxHungerDeathThreshold(
LynxHungerDeathThreshold_xjal );
   } else {
    root.LynxHungerDeathThreshold = LynxHungerDeathThreshold_xjal;
   }
   double LynxLifeExpectancy_xjal =
root._LynxLifeExpectancy_DefaultValue_xjal();
   if (callOnChangeActions) {
    root.set_LynxLifeExpectancy( LynxLifeExpectancy_xjal );
   } else {
    root.LynxLifeExpectancy = LynxLifeExpectancy_xjal;
   }
   int HaresInitial_xjal = root._HaresInitial_DefaultValue_xjal();
   if (callOnChangeActions) {
    root.set_HaresInitial( HaresInitial_xjal );
   } else {
    root.HaresInitial = HaresInitial_xjal;
   }
   int LynxInitial_xjal = root._LynxInitial_DefaultValue_xjal();
   if (callOnChangeActions) {
    root.set_LynxInitial( LynxInitial_xjal );
   } else {
    root.LynxInitial = LynxInitial_xjal;
   }
 }

 @Override
 public void onBeforeSimulationRun( Main root ) {
  // Before simulation run code

getEngine().getPresentation().setPresentable( root );
root.HaresInitial = HaresInitial;
root.LynxInitial = LynxInitial;
 }
```

```
 /**
  * Engine setup
  */
 @Override
 public void setupEngine(Engine engine) {
  engine.setATOL( 1.0E-5 );
  engine.setRTOL( 1.0E-5 );
  engine.setTTOL( 1.0E-5 );
  engine.setHTOL( 0.01 );
  engine.setSolverODE( Engine.SOLVER_ODE_EULER );
  engine.setSolverNAE( Engine.SOLVER_NAE_MODIFIED_NEWTON );
  engine.setSolverDAE( Engine.SOLVER_DAE_RK45_NEWTON );
  engine.setVMethods( 16032 );

  engine.setStartTime( 0.0 );
  engine.setTimeUnit( TIME_UNIT_DAY );
  engine.setRealTimeMode( false );
 }

 /**
  * Experiment setup
  */
 @Override
 public void setup( JApplet applet ) {
  setName( "" );

  // Dynamic initialization of persistent elements
  // Setup presentation
  Presentation _p = new Presentation( this, applet != null ?
Presentation.MODE_APPLET :
      Presentation.MODE_APPLICATION, applet );
  _p.start();

  Panel _panel = _p.getPanel();
  ToolBar _tb = _p.getToolBar();
  StatusBar _sb = _p.getStatusBar();

  _panel.setZoomEnabled( false );
  _panel.setPanningEnabled( false );
  _panel.setFrameManagementAdaptive( false );
  _panel.setFrameRate( 5.0 );

  _sb.setSectionVisible( StatusBar.DATE, false );
  _sb.setSectionVisible( StatusBar.EPS, true );
  _sb.setSectionVisible( StatusBar.EXPERIMENT, false );
  _sb.setSectionVisible( StatusBar.FPS, false );
  _sb.setSectionVisible( StatusBar.MEMORY, true );
  _sb.setSectionVisible( StatusBar.SECONDS, true );
  _sb.setSectionVisible( StatusBar.SIMULATION, false );
  _sb.setSectionVisible( StatusBar.STATUS, true );
  _sb.setSectionVisible( StatusBar.STEP, true );
  _sb.setSectionVisible( StatusBar.TIME, true );
  _tb.setSectionVisible( ToolBar.ANIMATION, false );
  _tb.setSectionVisible( ToolBar.EXECUTION, true );
  _tb.setSectionVisible( ToolBar.FILE, false );
  _tb.setSectionVisible( ToolBar.NAVIGATION, true );
  _tb.setSectionVisible( ToolBar.TIME_SCALE, true );
  _tb.setSectionVisible( ToolBar.VIEW, false );
 }

}
```

Παράρτημα 5

UML - Διαγράμματα κλάσεων

IDUser
+Experience: double +UsesSafeSyringe: boolean +XHome: double +YHome: double +InvitedUsers: java.util.LinkedList = new java.util.LinkedList() +DrugUsageBehavior: Statechart = new Statechart(this, (short)2) +HealthState: Statechart = new Statechart(this, (short)1) +Idle: short=0 +InjectingActivity: short=1 +Infected: short=2 +Susceptible: short=3 +branch: short=4 +Dead: short=5 +Finished: TransitionTimeout = new TransitionTimeout(this) +Organizes: TransitionTimeout = new TransitionTimeout(this) +Death: TransitionTimeout = new TransitionTimeout(this) +Invitation: TransitionTimeout = new TransitionTimeout(this) +Infection: TransitionTimeout = new TransitionTimeout(this) ~oval: ShapeOval ~oval1: ShapeOval ~presentation: ShapeGroup ~icon: ~Oval: ShapeGroup ~_Invitation_pointsX: int[*] = {270, 270} ~_Invitation_pointsY: int[*] = {130, 170} ~_Finished_pointsX: int[*] = {350, 400, 400, 350} ~_Finished_pointsY: int[*] = {190, 190, 110, 110} ~_Organizes_pointsX: int[*] = {270, 270} ~_Organizes_pointsY: int[*] = {130, 170} ~_Infection_pointsX: int[*] = {600, 600} ~_Infection_pointsY: int[*] = {130, 170} ~_Death_pointsX: int[*] = {650, 692} ~_Death_pointsY: int[*] = {190, 190} ~_transition1_pointsX: int[*] = {500, 500, 550} ~_transition1_pointsY: int[*] = {160, 190, 190} ~_transition2_pointsX: int[*] = {500, 500, 550} ~_transition2_pointsY: int[*] = {140, 110, 110}
<<create>>+IDUser(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection) +_Experience_DefaultValue_xjal(): double +set_Experience(Experience: double) ~onChange_Experience() +_UsesSafeSyringe_DefaultValue_xjal(): boolean +set_UsesSafeSyringe(UsesSafeSyringe: boolean) ~onChange_UsesSafeSyringe () +getNameOf(_s: Statechart): String +executeActionOf(_s: Statechart) +getNameOfState(_state: short): String +enterState(_state: short, _destination: boolean) +exitState(_state: short, _t: Transition, _source: boolean, _statechart: Statechart) +getNameOf(_t: TransitionTimeout): String +getStatechartOf(_t: TransitionTimeout): Statechart +testGuardOf(_t: TransitionTimeout): boolean +executeActionOf(_t: TransitionTimeout) +evaluateTimeoutOf(_t: TransitionTimeout): double +getNameOf(_t: TransitionRate): String +getStatechartOf(_t: TransitionRate): Statechart +executeActionOf(_t: TransitionRate) +evaluateTimeoutOf(_t: TransitionRate): double

+getNameOf(_t: TransitionMessage): String
+getStatechartOf(_t: TransitionMessage): Statechart
+executeActionOf(_t: TransitionMessage, _msg: Object)
+testMessageOf(_t: TransitionMessage, _msg: Object): boolean
-oval_SetDynamicParams(shape: ShapeOval)
-oval1_SetDynamicParams(shape: ShapeOval)
+getPersistentShape(_shape: int): Object
+drawModelElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean)
+onClickModelAt(panel: Panel, x: double, y: double, clickCount: int, publicOnly: boolean): boolean
+onReceive(msg: Object, sender: Agent)
+create()
+start()
+get_Main(): Main
+onChange()
+onDestroy()

Main

+InitialNumberOfUsers: double
+SafeSyringeUsersFraction: double
+PercentInitiallyInfected: double
+MeanHIVLifeDuration: double
+FractionInvited: double
+InfectionProbabilityUnsafe: double
+InfectionProbabilitySafe: double
+ExperienceDistribution: CustomDistribution
+UpdateHistograms: EventTimeout = new EventTimeout(this)
+_chart_autoUpdateEvent_xjal: EventTimeout = new EventTimeout(this)
-_ExperiencePDF_Arguments_xjal: double[*] = ExperiencePDF_Arguments_xjal()
-_ExperiencePDF_Values_xjal: double[*] = ExperiencePDF_Arguments_xjal()
+ExperiencePDF: TableFunction = new TableFunction(_ExperiencePDF_Arguments_xjal, _ExperiencePDF_Arguments_xjal, TableFunction.INTERPOLATION_LINEAR, TableFunction.OUTOFRANGE_ERROR, 0, 0)
+NInfectedDS: DataSet = new DataSet(365) { @Override public void update() { add(time(), _NInfectedDS_YValue()); }}
+NSusceptibleDS: DataSet = new DataSet(365) { @Override public void update() { add(time(), _NSusceptibleDS_YValue()); }}
+SusceptibleH: HistogramSimpleData = new HistogramSimpleData(5, 0, 4, false, false, 0.1, 0.1)
+InfectedH: HistogramSimpleData = new HistogramSimpleData(5, 0, 4, false, false, 0.1, 0.1)
~_text_Font: Font = new Font("Arial", 1, 28)
~_text1_Font: Font = new Font("Arial", 1, 11)
~_text2_Font: Font = new Font("Arial", 1, 11)
~_rectangle: int=1
~_citymap: int=2
~_citybounds: int=3
~IDUsers_presentation: int=4
~_text: int=5
~_text1: int=6
~_line: int=7
~_line1: int=8
~_text2: int=9
~_chart: int=10
~_chart1: int=11
~_chart2: int=12
~_presentation: int=0
~_icon: int=-1
~chart: TimeStackChart
~chart1: Histogram
~chart2: Histogram
~rectangle: ShapeRectangle
~citymap: ShapeImage
~citybounds: ShapePolyLine
~text: ShapeText
~text1: ShapeText
~text2: ShapeText
~line: ShapeLine
~line1: ShapeLine

~presentation: ShapeGroup
~icon: ShapeGroup
+environment: Environment = new Environment(this)

<<create>>+Main(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection)
+_InitialNumberOfUsers_DefaultValue_xjal(): double
+set_InitialNumberOfUsers(InitialNumberOfUsers: double)
~onChange_InitialNumberOfUsers()
+_SafeSyringeUsersFraction_DefaultValue_xjal(): double
+set_SafeSyringeUsersFraction(SafeSyringeUsersFraction: double)
~onChange_SafeSyringeUsersFraction()
+_PercentInitiallyInfected_DefaultValue_xjal(): double
+set_PercentInitiallyInfected(PercentInitiallyInfected: double)
~onChange_PercentInitiallyInfected()
+_MeanHIVLifeDuration_DefaultValue_xjal(): double
+set_MeanHIVLifeDuration(MeanHIVLifeDuration: double)
~onChange_MeanHIVLifeDuration()
+_FractionInvited_DefaultValue_xjal(): double
+set_FractionInvited(FractionInvited: double)
~onChange_FractionInvited()
+_InfectionProbabilityUnsafe_DefaultValue_xjal(): double
+set_InfectionProbabilityUnsafe(InfectionProbabilityUnsafe: double)
~onChange_InfectionProbabilityUnsafe()
+_InfectionProbabilitySafe_DefaultValue_xjal(): double
+set_InfectionProbabilitySafe(InfectionProbabilitySafe: double)
~onChange_InfectionProbabilitySafe()
+getNameOf(_e: EventTimeout): String
+getModeOf(_e: EventTimeout): int
+getFirstOccurenceTime(_e: EventTimeout): double
+evaluateTimeout(_e: EventTimeout): double
+executeActionOf(_e: EventTimeout)
+getNameOf(ao: ActiveObject): String
+getNameOf(aolist: ActiveObjectCollection): String
+add_IDUsers(): IDUser
+add_IDUsers(Experience: double, UsesSafeSyringe: boolean): IDUser
+remove_IDUsers(object: IDUser): boolean
-instantiate_IDUsers_xjal(index: int): IDUser
-setupParameters_IDUsers_xjal(object: IDUser, index: int)
-create_IDUsers_xjal(object: IDUser, index: int)
-_IDUsers_NInfected_xjal(): int
~setHomeLocation(user: IDUser)
-_ExperiencePDF_Arguments_xjal(): double
-_ExperiencePDF_Values_xjal(): double
+ExperiencePDF(x: double): double
-_NInfectedDS_YValue(): double
-_NSusceptibleDS_YValue(): double
+getNameOfShape(_shape: int): String
+getShapeType(_shape: int): int
+getShapeReplication(_shape: int): int
+getShapeX(_shape: int, index: int): double
+getShapeY(_shape: int, index: int): double
+getShapeEmbeddedObject(_shape: int): Object
-_citybounds_SetDynamicParams(shape: ShapePolyLine)
+getPersistentShape(_shape: int): Object
+drawModelElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean)
+onClickModelAt(panel: Panel, x: double, y: double, clickCount: int, publicOnly: boolean): boolean
+create()
+start()
+onStartup()
+get_EmbeddedObjects(): List
+onDestroy()

Simulation
~_button_Font: Font = Font("Dialog", 0, 11) ~_text1_Font: Font = new Font("SansSerif", 0, 9) ~_text2_Font: Font = new Font("SansSerif", 0, 9) ~_text3_Font: Font = new Font("SansSerif", 0, 10) ~_text4_Font: Font = new Font("SansSerif", 0, 10) ~_text5_Font: Font = new Font("SansSerif", 0, 10) ~_text6_Font: Font = new Font("SansSerif", 0, 10) ~_text7_Font: Font = new Font("SansSerif", 0, 10) ~_text8_Font: Font = new Font("SansSerif", 0, 10) ~_text_Font: Font = new Font("Arial", 1, 28) ~_text9_Font: Font = new Font("Arial", 1, 11) ~_text10_Font: Font = new Font("Arial", 1, 12) ~_text11_Font: Font = new Font("Arial", 1, 12) ~_text12_Font: Font = new Font("Arial", 1, 12) ~_button: int=1 ~_slider: int=2 ~_rectangle: int=3 ~_text: int=4 ~_image: int=5 ~_text1: int=6 ~_image1: int=7 ~_text2: int=8 ~_text3: int=9 ~_text4: int=10 ~_text5: int=11 ~_text6: int=12 ~_text7: int=13 ~_text8: int=14 ~_line: int=15 ~_text9: int=16 ~_text10: int=17 ~_text11: int=18 ~_text12: int=19 ~_presentation: int=0 ~_icon: int=-1 ~button: ShapeButton ~slider: ShapeSlider ~rectangle: ShapeRectangle ~text: ShapeText ~image: ShapeImage ~text1: ShapeText ~image1: ShapeImage ~text2: ShapeText ~text3: ShapeText ~text4: ShapeText ~text5: ShapeText ~text6: ShapeText ~text7: ShapeText ~text8: ShapeText ~line: ShapeLine ~text9: ShapeText ~text10: ShapeText ~text11: ShapeText ~text12: ShapeText ~presentation: ShapeGroup ~icon: ShapeGroup
+executeShapeControlAction(_shape: int, index: int) +getShapeControlMinimum(_shape: int, index: int): double +getShapeControlMaximum(_shape: int, index: int): double +getShapeControlDefaultValueDouble(_shape: int, index: int): double -_slider_SetDynamicParams(shape: ShapeSlider) -_text12_SetDynamicParams(shape: ShapeText)

+getPersistentShape(_shape: int): Object
+getWindowWidth(): int
+getWindowHeight(): int
+initDefaultRandomNumberGenerator(engine: Engine)
+createRoot(engine: Engine): Main
+setupRootParameters(root: Main, callOnChangeActions: boolean)
+setupEngine(engine: Engine)
+setup(applet: JApplet)

Κώδικας Κεφαλαίου 7 – Έργο HIV

Main.java

```
package hiv_diffusion_and_syringe_usage;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import java.awt.geom.Arc2D;

public class Main extends ActiveObject
{
 // Parameters

 public double  InitialNumberOfUsers;

 /**
  *  Returns default value for parameter
<code>InitialNumberOfUsers</code>.
  * <i>This method should not be called by user</i>
  */
 public double  _InitialNumberOfUsers_DefaultValue_xjal() {
  return 500 ;
 }

 public void set_InitialNumberOfUsers(
double  InitialNumberOfUsers ) {
  if (InitialNumberOfUsers == this.InitialNumberOfUsers) {
   return;
  }
  this.InitialNumberOfUsers = InitialNumberOfUsers;
  onChange_InitialNumberOfUsers();
  onChange();
 }

 void onChange_InitialNumberOfUsers() {
 }


 public double  SafeSyringeUsersFraction;

 /**
  *  Returns default value for parameter
<code>SafeSyringeUsersFraction</code>.
  * <i>This method should not be called by user</i>
  */
 public
double  _SafeSyringeUsersFraction_DefaultValue_xjal() {
  return 0.2 ;
 }

 public void set_SafeSyringeUsersFraction(
double  SafeSyringeUsersFraction ) {
  if (SafeSyringeUsersFraction == this.SafeSyringeUsersFraction) {
   return;
  }
  this.SafeSyringeUsersFraction = SafeSyringeUsersFraction;
  onChange_SafeSyringeUsersFraction();
  onChange();
 }

 void onChange_SafeSyringeUsersFraction() {
        int index;
  index = 0;
  for ( IDUser object : IDUsers ) {
   object.set_UsesSafeSyringe(randomTrue(  SafeSyringeUsersFraction
));
   index++;
  }

 }

 public double  PercentInitiallyInfected;

 /**
  *  Returns default value for parameter
<code>PercentInitiallyInfected</code>.
  * <i>This method should not be called by user</i>
```

```
  */
  public
double  _PercentInitiallyInfected_DefaultValue_xjal() {
    return 0.1 ;
  }

  public void set_PercentInitiallyInfected(
double  PercentInitiallyInfected ) {
    if (PercentInitiallyInfected == this.PercentInitiallyInfected) {
      return;
    }
    this.PercentInitiallyInfected = PercentInitiallyInfected;
    onChange_PercentInitiallyInfected();
    onChange();
  }

  void onChange_PercentInitiallyInfected() {
  }


  public
double  MeanHIVLifeDuration;

  /**
   *  Returns  default  value  for  parameter
<code>MeanHIVLifeDuration</code>.
   * <i>This method should not be called by user</i>
   */
  public
double  _MeanHIVLifeDuration_DefaultValue_xjal() {
    return
2 * 365
;
  }

  public void set_MeanHIVLifeDuration(
double  MeanHIVLifeDuration ) {
    if (MeanHIVLifeDuration == this.MeanHIVLifeDuration) {
      return;
    }
    this.MeanHIVLifeDuration = MeanHIVLifeDuration;
    onChange_MeanHIVLifeDuration();
    onChange();
  }

  void onChange_MeanHIVLifeDuration() {
  }


  public
double  FractionInvited;

  /**
   * Returns default value for parameter <code>FractionInvited</code>.
   * <i>This method should not be called by user</i>
   */
  public
double  _FractionInvited_DefaultValue_xjal() {
    return
0.5
;
  }

  public void set_FractionInvited(
double  FractionInvited ) {
    if (FractionInvited == this.FractionInvited) {
      return;
    }
    this.FractionInvited = FractionInvited;
    onChange_FractionInvited();
    onChange();
  }

  void onChange_FractionInvited() {
  }


  public
double  InfectionProbabilityUnsafe;

  /**
   *  Returns  default  value  for  parameter
<code>InfectionProbabilityUnsafe</code>.
   * <i>This method should not be called by user</i>
   */
  public
double  _InfectionProbabilityUnsafe_DefaultValue_xjal() {
    return
0.008
;
  }

  public void set_InfectionProbabilityUnsafe(
double  InfectionProbabilityUnsafe ) {
    if (InfectionProbabilityUnsafe == this.InfectionProbabilityUnsafe) {
      return;
    }
    this.InfectionProbabilityUnsafe = InfectionProbabilityUnsafe;
    onChange_InfectionProbabilityUnsafe();
    onChange();
  }

  void onChange_InfectionProbabilityUnsafe() {
  }


  public
double  InfectionProbabilitySafe;

  /**
   *  Returns  default  value  for  parameter
<code>InfectionProbabilitySafe</code>.
   * <i>This method should not be called by user</i>
   */
  public
double  _InfectionProbabilitySafe_DefaultValue_xjal() {
    return
0.00008
;
  }

  public void set_InfectionProbabilitySafe(
double  InfectionProbabilitySafe ) {
    if (InfectionProbabilitySafe == this.InfectionProbabilitySafe) {
      return;
    }
    this.InfectionProbabilitySafe = InfectionProbabilitySafe;
    onChange_InfectionProbabilitySafe();
    onChange();
  }

  void onChange_InfectionProbabilitySafe() {
```

```
  }

  // Plain Variables

  public
CustomDistribution
 ExperienceDistribution;

  // Collection Variables

  // Flow/Auxiliary Variables

  // Stock Variables


  // Events

  public EventTimeout UpadateHistograms = new EventTimeout(this);
  public     EventTimeout     _chart_autoUpdateEvent_xjal     =     new
EventTimeout(this);


  @Override
  public String getNameOf( EventTimeout _e ) {
     if( _e == UpadateHistograms ) return "UpadateHistograms";
    if ( _e == _chart_autoUpdateEvent_xjal ) return "chart auto update
event";
     return super.getNameOf( _e );
  }

  @Override
  public int getModeOf( EventTimeout _e ) {
    if      (      _e      ==      UpadateHistograms      )      return
EVENT_TIMEOUT_MODE_CYCLIC;
    if     (     _e     ==     _chart_autoUpdateEvent_xjal     )     return
EVENT_TIMEOUT_MODE_CYCLIC;
    return super.getModeOf( _e );
  }

  @Override
  public double getFirstOccurrenceTime( EventTimeout _e ) {
    if ( _e == UpadateHistograms ) return
0
;
    if (
       _e == _chart_autoUpdateEvent_xjal
     ) return getEngine().getStartTime();
    return super.getFirstOccurrenceTime( _e );
  }

  @Override
  public double evaluateTimeoutOf( EventTimeout _e ) {
    if( _e == UpadateHistograms) return
3
;
    if ( _e == _chart_autoUpdateEvent_xjal ) return
3 * day()
;
    return super.evaluateTimeoutOf( _e );
  }

  @Override
  public void executeActionOf( EventTimeout _e ) {
    if ( _e == UpadateHistograms ) {

//clear histograms
SusceptibleH.reset();
InfectedH.reset();
//fill out accoding to the current state
for( IDUser user : IDUsers ) {
        if( user.HealthState.isStateActive( IDUser.Susceptible ) )
                SusceptibleH.add( user.Experience );
        else
                InfectedH.add( user.Experience );
}
;
     return ;
    }
    if ( _e == _chart_autoUpdateEvent_xjal ) {
      chart.updateData();
      return;
    }
    super.executeActionOf( _e );
  }
  // Embedded Objects


  public String getNameOf( ActiveObject ao ) {
    return null;
  }

  public class _IDUsers_Class extends ActiveObjectArrayList<IDUser> {

    public int NInfected() {
      return _IDUsers_NInfected_xjal();
    }
  }

  public _IDUsers_Class IDUsers = new _IDUsers_Class();

  public String getNameOf( ActiveObjectCollection<?> aolist ) {
    if( aolist == IDUsers ) return "IDUsers";
    return null;
  }


  /**
   * This method creates and adds new embedded object in the
replicated embedded object collection IDUsers<br>
   * @return newly created embedded object
   */
  public IDUser add_IDUsers() {
    int index = IDUsers.size();
    IDUser object = instantiate_IDUsers_xjal( index );
    setupParameters_IDUsers_xjal( object, index );
    create_IDUsers_xjal( object, index );
    object.start();
    return object;
  }


  /**
   * This method creates and adds new embedded object in the
replicated embedded object collection IDUsers<br>
   * This method uses given parameter values to setup created
embedded object<br>
   * Index of this new embedded object instance can be obtained
through calling <code>IDUsers.size()</code> method
<strong>before</strong> this method is called
   * @param Experience
   * @param UsesSafeSyringe
   * @return newly created embedded object
   */
  public IDUser add_IDUsers( double Experience, boolean
UsesSafeSyringe ) {
    int index = IDUsers.size();
```

```
    IDUser object = instantiate_IDUsers_xjal( index );
    // Setup parameters
    object.Experience = Experience;
    object.UsesSafeSyringe = UsesSafeSyringe;
    // Finish embedded object creation
    create_IDUsers_xjal( object, index );
    object.start();
    return object;
  }

  /**
   * This method removes the given embedded object from the replicated embedded object collection IDUsers<br>
   * The given object is destroyed, but not immediately in common case.
   * @param object the active object - element of replicated embedded object IDUsers - which should be removed
   * @return <code>true</code> if object was removed successfully, <code>false</code> if it doesn't belong to IDUsers
   */
  public boolean remove_IDUsers( IDUser object ) {
    if( !IDUsers._remove( object ) ){
      return false;
    }
    object.setDestroyed();
    return true;
  }

  /**
   * Creates an embedded object instance and adds it to the end of replicated embedded object list<br>
   * <i>This method should not be called by user</i>
   */
  private IDUser instantiate_IDUsers_xjal( final int index ) {IDUser object = new IDUser( getEngine(), this, IDUsers );

    IDUsers._add(object);

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_IDUsers_xjal(IDUser object, final int index ) {
    object.Experience =
ExperienceDistribution.get()
;
    object.UsesSafeSyringe =
randomTrue( SafeSyringeUsersFraction )
;
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_IDUsers_xjal(IDUser object, final int index ) {
    object.setEnvironment(
environment
);
    object.create();

    // Port connections
  }

  /**
   * <i>This method should not be called by user</i>
   */
  private int _IDUsers_NInfected_xjal() {
    int _value = 0;
    for ( IDUser item : IDUsers ) {
      boolean _t =
item.HealthState.isStateActive( IDUser.Infected )
;
      if ( _t ) {
        _value++;
      }
    }
    return _value;
  }
  // Functions


void
 setHomeLocation( IDUser user ) {

double x;
double y;
do {
                x = uniform( citymap.getX(), citymap.getX() +
citymap.getWidth() );
                y = uniform( citymap.getY(), citymap.getY() +
citymap.getHeight() );
} while( !citybounds.contains( x, y ) );
user.setXY( x, y );
  }
  // Table Functions

  private final static double[] _ExperiencePDF_Arguments_xjal =
_ExperiencePDF_Arguments_xjal();
  private final static double[] _ExperiencePDF_Arguments_xjal() {
    return new double[] { 0.0, 1.0, 2.0, 3.0, 4.0, 5.0,  };
  }
  private final static double[] _ExperiencePDF_Values_xjal =
_ExperiencePDF_Values_xjal();
  private final static double[] _ExperiencePDF_Values_xjal() {
    return new double[] { 0, 50, 40, 20, 10, 5,  };
  }

  /**
   * ExperiencePDF Table Function
   */
  public static final TableFunction ExperiencePDF = new TableFunction(
_ExperiencePDF_Arguments_xjal, _ExperiencePDF_Values_xjal,
      TableFunction.INTERPOLATION_LINEAR,                    1,
TableFunction.OUTOFRANGE_ERROR,
      0.0 );

  public static final double ExperiencePDF( double x ) { return
ExperiencePDF.get( x ); }
  // Analysis Data Elements
  public DataSet _chart_expression0_dataSet_xjal = new DataSet( 365 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
IDUsers.NInfected()
;
      add( time(), _a );
      _lastUpdateX = time();
    }
```

```
  };
  public DataSet _chart_expression1_dataSet_xjal = new DataSet( 365 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
IDUsers.size() - IDUsers.NInfected()
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  public HistogramSimpleData SusceptibleH = new HistogramSimpleData(
5,
0
,
4
, false, false, 0.1, 0.1 );

  public HistogramSimpleData InfectedH = new HistogramSimpleData( 5,
0
,
4
, false, false, 0.1, 0.1 );

  static final Font _text_Font = new Font("Arial", 1, 28 );
  static final Font _text1_Font = new Font("Arial", 1, 11 );
  static final Font _text2_Font = _text1_Font;
  static final Font _text3_Font = new Font("SansSerif", 0, 10 );
  static final Font _text4_Font = _text3_Font;
  static final Font _text5_Font = _text3_Font;
  static final int _rectangle = 1;
  static final int _citymap = 2;
  static final int _citybounds = 3;
  static final int IDUsers_presentation = 4;
  static final int _text = 5;
  static final int _text1 = 6;
  static final int _line = 7;
  static final int _line1 = 8;
  static final int _text2 = 9;
  static final int _oval1 = 10;
  static final int _oval = 11;
  static final int _text3 = 12;
  static final int _oval2 = 13;
  static final int _text4 = 14;
  static final int _oval3 = 15;
  static final int _oval4 = 16;
  static final int _oval5 = 17;
  static final int _text5 = 18;
  static final int _chart = 19;
  static final int _chart1 = 20;
  static final int _chart2 = 21;

  /**
   * Top-level presentation group id
   */
  static final int _presentation = 0;

  /**
   * Top-level icon group id
   */
  static final int _icon = -1;

  static final double[] _citybounds_pointsDX_xjal() {
    return new double[] { 0, 12, 27, -5, -14, -30, -58, -73, -78, -105, -
119, -189, -193, -174, -172, -181, -186, -216, -218, -202, -210, -186, -
166, -176, -168, -172, -163, -143, -130, -120, -63, -70, -60, -52, -41, -
46, -33, 1, 9, 66, 100, 130, 146, 153, 234, 237, 226, 178, 182, 168, 120,
99, 117, 109, 83, 83, 71, 61, 53, 86, 84, 101, 103, 126, 192, 216, 221,
237, 234, 180, 133, 142, 139, 115, 108, 61, 49, 63, 61, 64, 85, 119, 139,
150, 124, 76, 50,  };
  }
  static final double[] _citybounds_pointsDY_xjal() {
    return new double[] { 0, -22, -134, -139, -136, -150, -148, -84, -47, -
1, 6, -4, -34, -45, -94, -94, -76, -83, -94, -98, -110, -126, -177, -216, -
264, -290, -310, -311, -320, -370, -398, -410, -420, -410, -386, -381, -
362, -366, -357, -378, -414, -431, -434, -406, -422, -355, -321, -320, -
298, -284, -289, -305, -329, -336, -333, -322, -316, -320, -285, -269, -
260, -252, -271, -278, -187, -188, -167, -156, -102, -118, -134, -174, -
197, -206, -186, -192, -140, -125, -97, -76, -98, -139, -116, -98, -61, -
14, 10,  };
  }


  @Override
  public String getNameOfShape( int _shape ) {
    switch( _shape ) {
      case IDUsers_presentation: return "IDUsers_presentation";
      default: return super.getNameOfShape( _shape );
    }
  }

  @Override
  public int getShapeType( int _shape ) {
    switch( _shape ) {
      case IDUsers_presentation: return SHAPE_EMBEDDED_OBJECT;
      default: return super.getShapeType( _shape );
    }
  }

  @Override
  public int getShapeReplication( int _shape ) {
    switch( _shape ) {
      case IDUsers_presentation: return
IDUsers.size()
;
      default: return super.getShapeReplication( _shape );
    }
  }

  @Override
  public double getShapeX( int _shape, int index ) {
    switch( _shape ) {
      case IDUsers_presentation: return 0;
      default: return super.getShapeX( _shape, index );
    }
  }

  @Override
  public double getShapeY( int _shape, int index ) {
    switch( _shape ) {
      case IDUsers_presentation: return 0;
      default: return super.getShapeY( _shape, index );
    }
  }

  @Override
  public Object getShapeEmbeddedObject( int _shape ) {
    switch( _shape ) {
      case IDUsers_presentation: return IDUsers;
      default: return super.getShapeEmbeddedObject( _shape );
    }
  }
```

```
  TimeStackChart chart;
  Histogram chart1;
  Histogram chart2;
  ShapeRectangle rectangle;
  ShapeImage citymap;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _citybounds_SetDynamicParams( ShapePolyLine shape ) {
    boolean _visible;
    _visible =
false
;
    shape.setVisible( _visible );
                if ( _visible ) {
                }
  }

  ShapePolyLine citybounds;
  ShapeText text;
  ShapeText text1;
  ShapeLine line;
  ShapeLine line1;
  ShapeText text2;
  ShapeOval oval1;
  ShapeOval oval;
  ShapeText text3;
  ShapeOval oval2;
  ShapeText text4;
  ShapeOval oval3;
  ShapeOval oval4;
  ShapeOval oval5;
  ShapeText text5;

  // Static initialization of persistent elements
  {
    rectangle = new ShapeRectangle(
      true,0, 0, 0.0,
          null, black,
                                        900, 600,
                                        1, LINE_STYLE_SOLID
    );
    citymap = new ShapeImage(
                            Main.this, true, 38, 60, 0.0,
                                        462,                    450,
"/hiv_diffusion_and_syringe_usage/",
                                        new
String[]{"urupinsknight.png",}
    );
    citybounds = new ShapePolyLine(
                            true,260, 500,
          gold, null,
          87, _citybounds_pointsDX_xjal(), _citybounds_pointsDY_xjal(),
          true, 2, LINE_STYLE_SOLID
    ) {

      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
        _citybounds_SetDynamicParams( this );
        super.draw( panel, graphics, xform, publicOnly );
      }
    };
    text = new ShapeText(
      true,40, 10, 0.0,
      dodgerBlue,"HIV Diffusion and Syringe Usage",
      _text_Font, ALIGNMENT_LEFT
  );
  text1 = new ShapeText(
    true,560, 60, 0.0,
    cornflowerBlue,"Number of users (by HIV status)",
    _text1_Font, ALIGNMENT_LEFT
  );
  line = new ShapeLine(
                          true,560, 75,
    cornflowerBlue,
                          300,          0,
                          1, LINE_STYLE_SOLID
  );
  line1 = new ShapeLine(
                          true,560, 265,
    cornflowerBlue,
                          300,          0,
                          1, LINE_STYLE_SOLID
  );
  text2 = new ShapeText(
    true,560, 250, 0.0,
    cornflowerBlue,"Distribution of Experience (by HIV status)",
    _text2_Font, ALIGNMENT_LEFT
  );
  oval1 = new ShapeOval(
    true,297, 548, 0.0,
    gold, null,
              3, 3,
              2, LINE_STYLE_SOLID
  );
  oval = new ShapeOval(
    true,183, 548, 0.0,
    null, dodgerBlue,
              2, 2,
              1, LINE_STYLE_SOLID
  );
  text3 = new ShapeText(
    true,190, 541, 0.0,
    cornflowerBlue,"HIV-",
    _text3_Font, ALIGNMENT_LEFT
  );
  oval2 = new ShapeOval(
    true,237, 548, 0.0,
    null, red,
              2, 2,
              1, LINE_STYLE_SOLID
  );
  text4 = new ShapeText(
    true,245, 541, 0.0,
    cornflowerBlue,"HIV+",
    _text4_Font, ALIGNMENT_LEFT
  );
  oval3 = new ShapeOval(
    true,287, 548, 0.0,
    null, dodgerBlue,
              2, 2,
              1, LINE_STYLE_SOLID
  );
  oval4 = new ShapeOval(
    true,287, 548, 0.0,
    gold, null,
              3, 3,
              2, LINE_STYLE_SOLID
  );
```

```
  oval5 = new ShapeOval(
    true,297, 548, 0.0,
    null, red,
              2, 2,
              1, LINE_STYLE_SOLID
  );
  text5 = new ShapeText(
     true,305, 541, 0.0,
     cornflowerBlue,"Party organizer",
     _text5_Font, ALIGNMENT_LEFT
  );
 }
 ShapeGroup presentation;
 ShapeGroup icon;

 @Override
 public Object getPersistentShape( int _shape ) {
  switch(_shape){
    case _presentation: return presentation;
    case _icon: return icon;

    case _chart: return chart;
    case _chart1: return chart1;
    case _chart2: return chart2;
    case _rectangle: return rectangle;
    case _citymap: return citymap;
    case _citybounds: return citybounds;
    case _text: return text;
    case _text1: return text1;
    case _line: return line;
    case _line1: return line1;
    case _text2: return text2;
    case _oval1: return oval1;
    case _oval: return oval;
    case _text3: return text3;
    case _oval2: return oval2;
    case _text4: return text4;
    case _oval3: return oval3;
    case _oval4: return oval4;
    case _oval5: return oval5;
    case _text5: return text5;
    default: return null;
  }
 }


 @Override
 public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
   if (!_publicOnly) {
     drawEvent( _panel, _g, -200, 250, 10, 0, "UpadateHistograms",
UpadateHistograms );
   }
   if (!_publicOnly) {
     drawParameter( _panel, _g, -600, 50, 10, 0, "InitialNumberOfUsers",
InitialNumberOfUsers, false, false );
   }
   if (!_publicOnly) {
     drawParameter( _panel, _g, -600, 100, 10, 0,
"SafeSyringeUsersFraction", SafeSyringeUsersFraction, false, false );
   }
   if (!_publicOnly) {
     drawParameter( _panel, _g, -600, 150, 10, 0,
"PercentInitiallyInfected", PercentInitiallyInfected, false, false );
   }
   if (!_publicOnly) {
     drawParameter( _panel, _g, -600, 200, 10, 0,
"MeanHIVLifeDuration", MeanHIVLifeDuration, false, false );
   }
   if (!_publicOnly) {
     drawParameter( _panel, _g, -600, 250, 10, 0, "FractionInvited",
FractionInvited, false, false );
   }
   if (!_publicOnly) {
     drawParameter( _panel, _g, -600, 300, 10, 0,
"InfectionProbabilityUnsafe", InfectionProbabilityUnsafe, false, false );
   }
   if (!_publicOnly) {
     drawParameter( _panel, _g, -600, 350, 10, 0,
"InfectionProbabilitySafe", InfectionProbabilitySafe, false, false );
   }
   if (!_publicOnly) {
     drawPlainVariable( _panel, _g, -600, 450, 10, 0,
"ExperienceDistribution", ExperienceDistribution, false );
   }
   if (!_publicOnly) {
     drawFunction( _panel, _g, -400, 50, 10, 0, "setHomeLocation");
   }
   if (!_publicOnly) {
     drawTableFunction( _panel, _g, -600, 400, 10, 0, "ExperiencePDF");
   }
   if (!_publicOnly) {
     drawDataset( _panel, _g, -200, 150, 10, 0, "SusceptibleH",
SusceptibleH );
   }
   if (!_publicOnly) {
     drawDataset( _panel, _g, -200, 200, 10, 0, "InfectedH", InfectedH );
   }
   // Embedded object "IDUsers"
   if (!_publicOnly) {
     drawEmbeddedObjectModelDefault( _panel, _g, -200 , 100 , 10, 0,
"IDUsers", this.IDUsers );
   }
   if (!_publicOnly) {
     drawEnvironment( _panel, _g, -200, 50, 20, 0, "environment",
environment );
   }
  }


  @Override
  public boolean onClickModelAt( Panel panel, double x, double y, int
clickCount, boolean publicOnly ) {
   if( !publicOnly && modelElementContains(x, y, -600, 50) ) {
     panel.addInspect( -600, 50, this, "InitialNumberOfUsers" );
     return true;
   }
   if( !publicOnly && modelElementContains(x, y, -600, 100) ) {
     panel.addInspect( -600, 100, this, "SafeSyringeUsersFraction" );
     return true;
   }
   if( !publicOnly && modelElementContains(x, y, -600, 150) ) {
     panel.addInspect( -600, 150, this, "PercentInitiallyInfected" );
     return true;
   }
   if( !publicOnly && modelElementContains(x, y, -600, 200) ) {
     panel.addInspect( -600, 200, this, "MeanHIVLifeDuration" );
     return true;
   }
   if( !publicOnly && modelElementContains(x, y, -600, 250) ) {
     panel.addInspect( -600, 250, this, "FractionInvited" );
     return true;
   }
   if( !publicOnly && modelElementContains(x, y, -600, 300) ) {
     panel.addInspect( -600, 300, this, "InfectionProbabilityUnsafe" );
```

```
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, -600, 350) ) {
    panel.addInspect( -600, 350, this, "InfectionProbabilitySafe" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, -600, 450) ) {
    panel.addInspect( -600, 450, this, "ExperienceDistribution" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, -200, 250) ) {
    panel.addInspect( -200, 250, this, "UpadateHistograms" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, -600, 400) ) {
    panel.addInspect( -600, 400, this, "ExperiencePDF" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, -200, 150) ) {
    panel.addInspect( -200, 150, this, "SusceptibleH" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, -200, 200) ) {
    panel.addInspect( -200, 200, this, "InfectedH" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, -200, 50) ) {
    panel.addInspect( -200, 50, this, "environment" );
    return true;
   }
   if ( !IDUsers.isEmpty() && modelElementContains(x, y, -200, 100) ) {
    if ( clickCount == 2 ) {
     panel.browseEmbeddedObject( -200, 100, this, "IDUsers" );
    } else {
     panel.addInspect( -200, 100, this, "IDUsers" );
    }
    return true;
   }
   return false;
  }

  // Environments
  public final Environment environment = new Environment( this );


  /**
   * Constructor
   */
  public Main( Engine engine, ActiveObject owner,
ActiveObjectCollection<? extends Main> collection ) {
   super( engine, owner, collection );
  }

  @Override
  public void create() {
   // Creating embedded object instances
   for ( int i = 0; i <
InitialNumberOfUsers
; i++ ) {
    instantiate_IDUsers_xjal( i );
   }
   // Assigning initial values for plain variables
   ExperienceDistribution =
new CustomDistribution( ExperiencePDF, getDefaultRandomGenerator() )
;
   // Dynamic initialization of persistent elements
   {
   DataSet _item;
   List<DataSet> _items = new ArrayList<DataSet>( 2 );
   _items.add( _chart_expression0_dataSet_xjal );
   _items.add( _chart_expression1_dataSet_xjal );
   List<String> _titles = new ArrayList<String>( 2 );
   _titles.add( "Infected users" );
   _titles.add( "Susceptible users" );
   List<Color> _colors = new ArrayList<Color>( 2 );
   _colors.add( tomato );
   _colors.add( gold );
   chart = new TimeStackChart(
                    Main.this, true, 520, 70,
                              360, 170,
        null, null,
        40, 20,
                              300, 100, gray, cornflowerBlue,
cornflowerBlue,
        30, Chart.SOUTH,

3 * 365 * day() , Chart.DEFAULT_DATE_PATTERN, Chart.SCALE_FIXED
        ,
InitialNumberOfUsers
        , Chart.GRID_DEFAULT, Chart.GRID_DEFAULT,
        gray, cornflowerBlue, _items, _titles, _colors
   );
   }
   {
   HistogramData _item;
   List<HistogramData> _items = new ArrayList<HistogramData>( 1 );
   _item =
SusceptibleH
;
   _items.add( _item );
   List<String> _titles = new ArrayList<String>( 1 );
   _titles.add( "Experience (Susceptible)" );
   List<Histogram.Appearance> _appearances = new
ArrayList<Histogram.Appearance>( 1 );
   _appearances.add( new Histogram.Appearance( darkKhaki, orange,
gold, yellowGreen, 1, yellowGreen ) );
   chart1 = new Histogram(
                    Main.this, true, 690, 260,
                              190, 150,
        null, null,
        40, 20,
                              130, 80, gray, cornflowerBlue,
cornflowerBlue,
        30, Chart.SOUTH,
        Chart.GRID_DEFAULT, Chart.GRID_NONE,
        darkGray, cornflowerBlue,
        true, false, false, 0.8,
        _items, _titles, _appearances
   );
   }
   {
   HistogramData _item;
   List<HistogramData> _items = new ArrayList<HistogramData>( 1 );
   _item =
InfectedH
;
   _items.add( _item );
   List<String> _titles = new ArrayList<String>( 1 );
   _titles.add( "Experience (Infected)" );
   List<Histogram.Appearance> _appearances = new
ArrayList<Histogram.Appearance>( 1 );
   _appearances.add( new Histogram.Appearance( violetRed,
mediumSeaGreen, tomato, slateGray, 1, slateGray ) );
```

```
  chart2 = new Histogram(
                                Main.this, true, 520, 260,
                                        190, 150,
        null, null,
        40, 20,
                                        130, 80, gray, cornflowerBlue,
cornflowerBlue,
        30, Chart.SOUTH,
        Chart.GRID_DEFAULT, Chart.GRID_NONE,
        darkGray, cornflowerBlue,
        true, false, false, 0.8,
        _items, _titles, _appearances
  );
  }
  presentation = new ShapeGroup( Main.this, true, 0, 0, 0, rectangle,
citymap, citybounds, IDUsers_presentation, text, text1, line, line1, text2,
oval1, oval, text3, oval2, text4, oval3, oval4, oval5, text5, chart, chart1,
chart2 );
  icon = new ShapeGroup( Main.this, true, 0, 0, 0
);
  // Environments setup
  environment.disableSteps();
  environment.setSpaceContinuous(
500 ,
500 );
  environment.setNetworkAllInRange(
50 );
  environment.setLayoutType( Environment.LAYOUT_USER_DEFINED );
            // Port connectors with non-replicated objects
  // Creating replicated embedded objects
  for ( int i = 0; i < IDUsers.size(); i++ ) {
    setupParameters_IDUsers_xjal( IDUsers.get(i), i );
    create_IDUsers_xjal( IDUsers.get(i), i );
  }
  assignInitialConditions();
  onCreate();
 }

 @Override
 public void start() {
  UpadateHistograms.start();
  _chart_autoUpdateEvent_xjal.start();
  for (ActiveObject embeddedObject : IDUsers){
   embeddedObject.start();
  }
  onStartup();
 }

 public void onStartup() {
  super.onStartup();

environment.applyNetwork();
 }

 public List<Object> getEmbeddedObjects() {
  LinkedList<Object> list = new LinkedList<Object>();
  list.add( IDUsers );
  return list;
 }

 public void onDestroy() {
  super.onDestroy();
  UpadateHistograms.onDestroy();
  environment.onDestroy();
  _chart_autoUpdateEvent_xjal.onDestroy();
  for (ActiveObject embeddedObject : IDUsers) {
   embeddedObject.onDestroy();
  }
 }
}
```

IDUser.java

```
package hiv_diffusion_and_syringe_usage;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import java.awt.geom.Arc2D;

public class IDUser extends Agent
{
 // Parameters

 public double  Experience;

 /**
  * Returns default value for parameter <code>Experience</code>.
  * <i>This method should not be called by user</i>
  */
```

```
  public
double  _Experience_DefaultValue_xjal() {
    return 0.0;
  }

  public void set_Experience(
double  Experience ) {
    if (Experience == this.Experience) {
      return;
    }
    this.Experience = Experience;
    onChange_Experience();
    onChange();
  }

  void onChange_Experience() {
  }



  public
boolean  UsesSafeSyringe;

  /**
   *  Returns  default  value  for  parameter
<code>UsesSafeSyringe</code>.
   * <i>This method should not be called by user</i>
   */
  public
boolean  _UsesSafeSyringe_DefaultValue_xjal() {
    return false;
  }

  public void set_UsesSafeSyringe(
boolean  UsesSafeSyringe ) {
    if (UsesSafeSyringe == this.UsesSafeSyringe) {
      return;
    }
    this.UsesSafeSyringe = UsesSafeSyringe;
    onChange_UsesSafeSyringe();
    onChange();
  }

  void onChange_UsesSafeSyringe() {
  }

  // Plain Variables

  public
IDUser
 Organizer;
  public
double
 XHome;
  public
double
 YHome;

  // Collection Variables
  public
java.util.LinkedList <
IDUser > InvitedUsers = new java.util.LinkedList<IDUser>();

  // Flow/Auxiliary Variables

  // Stock Variables

  // Statecharts
  public Statechart DrugUsageBehavior = new Statechart( this, (short)2
);
  public Statechart HealthState = new Statechart( this, (short)1 );

  @Override
  public String getNameOf( Statechart _s ) {
    if(_s == this.DrugUsageBehavior) return "DrugUsageBehavior";
    if(_s == this.HealthState) return "HealthState";
    return super.getNameOf( _s );
  }

  @Override
  public void executeActionOf( Statechart _s ) {
    if( _s == this.DrugUsageBehavior ) {
      enterState( Idle, true );
      return;
    }
    if( _s == this.HealthState ) {
      enterState( branch, true );
      return;
    }
    super.executeActionOf( _s );
  }

  // States of all statecharts

  public static final short Idle = 0;
  public static final short InjectingActivity = 1;

  public static final short Infected = 2;
  public static final short Susceptible = 3;
  public static final short branch = 4;
  public static final short Dead = 5;

  @Override
  public String getNameOfState( short _state ) {
    switch( _state ) {
      case Idle: return "Idle";
      case InjectingActivity: return "InjectingActivity";
      case Infected: return "Infected";
      case Susceptible: return "Susceptible";
      case branch: return "branch";
      case Dead: return "Dead";
      default: return super.getNameOfState( _state );
    }
  }

  @Override
  public void enterState( short _state, boolean _destination ) {
    switch( _state ) {
      case Idle: // (Simple state (not composite))
        DrugUsageBehavior.setActiveState( Idle );
        Invitation.start();
        Organizes.start();
        return;
      case InjectingActivity: // (Simple state (not composite))
        DrugUsageBehavior.setActiveState( InjectingActivity );
        Finished.start();
        return;
      case Infected: // (Simple state (not composite))
        HealthState.setActiveState( Infected );
        Death.start();
        return;
      case Susceptible: // (Simple state (not composite))
        HealthState.setActiveState( Susceptible );
```

```
      Infection.start();
      return;
    case branch: // (Branch)
      if (
randomTrue( get_Main().PercentInitiallyInfected )
 ) { // transition1
        enterState( Infected, true );
        return;
      }
      // transition2 (default)
        enterState( Susceptible, true );
      return;
    case Dead: // (Final State)
      {
//delete self from the model
get_Main().remove_IDUsers( this );
;}
      HealthState.setActiveState( Dead );
      HealthState.onDestroy();
      return;
    default:
      super.enterState( _state, _destination );
      return;
  }
 }


 @Override
 public void exitState( short _state, Transition _t, boolean _source,
Statechart _statechart) {
  switch( _state ) {
    case Idle: // (Simple state (not composite))
      if ( !_source || _t != Invitation) Invitation.cancel();
      if ( !_source || _t != Organizes) Organizes.cancel();
      return;
    case InjectingActivity: // (Simple state (not composite))
      if ( !_source || _t != Finished) Finished.cancel();
      {
//the injecting activity is controlled by organizer
if( Organizer == this ) {

            //detect if anyone has safe syringe -
            //this affects the infection probability
            //we assume that if yes it will be used

            double                  infectionprob                  =
get_Main().InfectionProbabilityUnsafe;

            for( IDUser buddy : InvitedUsers ) {
                  if( buddy.UsesSafeSyringe ) {
                        infectionprob                  =
get_Main().InfectionProbabilitySafe;
                        break;
                  }
            }

            //the injection
            //is done in the order of InvitedUsers collection,
            //i.e. experineced users get injected first

            boolean syringeinfected = false;

            for( IDUser buddy : InvitedUsers ) {
                  if(     syringeinfected     &&     randomTrue(
infectionprob ) )
                        send( "Infection", buddy );
                  if( buddy.HealthState.isStateActive( Infected )
)
                        syringeinfected = true;
            }
}
;}
      return;
    case Infected: // (Simple state (not composite))
      if ( !_source || _t != Death) Death.cancel();
      return;
    case Susceptible: // (Simple state (not composite))
      if ( !_source || _t != Infection) Infection.cancel();
      return;
    default:
      super.exitState( _state, _t, _source, _statechart);
      return;
  }
 }


 public TransitionTimeout Finished = new TransitionTimeout( this );
 public TransitionTimeout Organizes = new TransitionTimeout( this );

 @Override
 public String getNameOf( TransitionTimeout _t ) {
  if ( _t == Finished ) return "Finished";
  if ( _t == Organizes ) return "Organizes";
  return super.getNameOf( _t );
 }

 @Override
 public Statechart getStatechartOf( TransitionTimeout _t ) {
  if ( _t == Finished ) return DrugUsageBehavior;
  if ( _t == Organizes ) return DrugUsageBehavior;
  return super.getStatechartOf( _t );
 }

 @Override
 public boolean testGuardOf( TransitionTimeout _t ) {
  if ( _t == Organizes ) return
getConnections() != null //no buddies - no parties
;
  return super.testGuardOf(  _t );
 }

 @Override
 public void executeActionOf( TransitionTimeout _t ) {
  if ( _t == Finished ) {
   exitState( InjectingActivity, _t, true, DrugUsageBehavior );
   {
//go home if this is not my place
if( Organizer != this )
            moveTo( XHome, YHome );
//clear injecting activity info
Organizer = null;
InvitedUsers.clear();
;}
      enterState( Idle, true );
    return;
  }
  if ( _t == Organizes ) {
   exitState( Idle, _t, true, DrugUsageBehavior );
   {
//this is the injecting activity organizer
Organizer = this;
//add self to the invited user list
InvitedUsers.add( this );
```

```
//iterate through all knwn drug users
for( Agent a : getConnections() ) {
                IDUser buddy = (IDUser)a;
                //invite people with a certain probability
                if ( randomFalse( get_Main().FractionInvited ) )
                                continue;
                send( "Invitation", buddy );
                //those who confirm participation will be added
                //to the InvitedUsers collection
}
;}
        enterState( InjectingActivity, true );
     return;
    }
    super.executeActionOf( _t );
  }
  @Override
  public double evaluateTimeoutOf( TransitionTimeout _t ) {
    if ( _t == Finished ) return
5 * hour()
;
    if ( _t == Organizes ) return
uniform_discr( 1, 3 )*day() - 5*hour()
;
    return super.evaluateTimeoutOf( _t );
  }


  public TransitionRate Death = new TransitionRate( this );

  @Override
  public String getNameOf( TransitionRate _t ) {
    if ( _t == Death ) return "Death";
    return super.getNameOf( _t );
  }

  @Override
  public Statechart getStatechartOf( TransitionRate _t ) {
    if ( _t == Death ) return HealthState;
    return super.getStatechartOf( _t );
  }

  @Override
  public void executeActionOf( TransitionRate _t ) {
    if ( _t == Death ) {
      exitState( Infected, _t, true, HealthState );
          enterState( Dead, true );
      return;
    }
    super.executeActionOf( _t );
  }
  @Override
  public double evaluateRateOf( TransitionRate _t ) {
    if ( _t == Death ) return
1 / get_Main().MeanHIVLifeDuration
;
    return super.evaluateRateOf( _t );
  }

  public TransitionMessage Invitation = new TransitionMessage( this );
  public TransitionMessage Infection = new TransitionMessage( this );

  @Override
  public String getNameOf( TransitionMessage _t ) {
    if ( _t == Invitation ) return "Invitation";
    if ( _t == Infection ) return "Infection";
    return super.getNameOf( _t );
  }

  @Override
  public Statechart getStatechartOf( TransitionMessage _t ) {
    if ( _t == Invitation ) return DrugUsageBehavior;
    if ( _t == Infection ) return HealthState;
    return super.getStatechartOf( _t );
  }

  @Override
  public void executeActionOf( TransitionMessage _t, Object _msg ) {
    if ( _t == Invitation ) {
      exitState( Idle, _t, true, DrugUsageBehavior );
      {
        String msg = (String) _msg;
//confirm participation
send( "Confirm", Organizer );
//move to the injecting activity organizer's place (for animation only)
moveTo( Organizer.getX(), Organizer.getY() );
;}
          enterState( InjectingActivity, true );
      return;
    }
    if ( _t == Infection ) {
      exitState( Susceptible, _t, true, HealthState );
          enterState( Infected, true );
      return;
    }
    super.executeActionOf( _t, _msg );
  }
  @Override
  public boolean testMessageOf( TransitionMessage _t, Object _msg ) {
    if ( _t == Invitation ) {
      if ( !(_msg instanceof String) )
        return false;
      String msg = (String) _msg;
      Object _g =
"Invitation"
;
      return msg.equals( _g );
    }
    if ( _t == Infection ) {
      if ( !(_msg instanceof String) )
        return false;
      String msg = (String) _msg;
      Object _g =
"Infection"
;
      return msg.equals( _g );
    }
    return super.testMessageOf( _t, _msg );
  }
  static final int _oval = 1;
  static final int _oval1 = 2;

  /**
   * Top-level presentation group id
   */
  static final int _presentation = 0;

  /**
   * Top-level icon group id
   */
```

```
  static final int _icon = -1;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _oval_SetDynamicParams( ShapeOval shape ) {
    boolean _visible;
    shape.setFillColor(
HealthState.isStateActive( Infected ) ? red : dodgerBlue
);
  }

  ShapeOval oval;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _oval1_SetDynamicParams( ShapeOval shape ) {
    boolean _visible;
    _visible =
Organizer == this
;
    shape.setVisible( _visible );
                if ( _visible ) {
                }
  }

  ShapeOval oval1;

  // Static initialization of persistent elements
  {
    oval = new ShapeOval(
      true,0, 0, 0.0,
      null, dodgerBlue,
                  2, 2,
                  1, LINE_STYLE_SOLID
    ) {

      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
        _oval_SetDynamicParams( this );
        super.draw( panel, graphics, xform, publicOnly );
      }
    };
    oval1 = new ShapeOval(
      true,0, 0, 0.0,
      gold, null,
                  3, 3,
                  2, LINE_STYLE_SOLID
    ) {

      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
        _oval1_SetDynamicParams( this );
        super.draw( panel, graphics, xform, publicOnly );
      }
    };
  }
  ShapeGroup presentation;
  ShapeGroup icon;

  @Override
  public Object getPersistentShape( int _shape ) {
    switch(_shape){
      case _presentation: return presentation;
      case _icon: return icon;

      case _oval: return oval;
      case _oval1: return oval1;
      default: return null;
    }
  }

  static final int[] _Invitation_pointsX = {270, 270, };
  static final int[] _Invitation_pointsY = {130, 170, };

  static final int[] _Finished_pointsX = {350, 400, 400, 350, };
  static final int[] _Finished_pointsY = {190, 190, 110, 110, };

  static final int[] _Organizes_pointsX = {330, 330, };
  static final int[] _Organizes_pointsY = {130, 170, };

  static final int[] _transition1_pointsX = {500, 500, 550, };
  static final int[] _transition1_pointsY = {160, 190, 190, };

  static final int[] _Death_pointsX = {650, 692, };
  static final int[] _Death_pointsY = {190, 190, };

  static final int[] _transition2_pointsX = {500, 500, 550, };
  static final int[] _transition2_pointsY = {140, 110, 110, };

  static final int[] _Infection_pointsX = {600, 600, };
  static final int[] _Infection_pointsY = {130, 170, };


  @Override
  public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
    if (!_publicOnly) {
      drawState( _panel, _g, 250, 100, 100, 30, 10, 10, "Idle",
greenYellow, Idle, DrugUsageBehavior );
    }
    if (!_publicOnly) {
      drawState( _panel, _g, 250, 170, 100, 30, 10, 10, "InjectingActivity",
GOLD, InjectingActivity, DrugUsageBehavior );
    }
    if (!_publicOnly) {
      drawStatechartEntryPoint( _panel, _g, 300, 70, 300, 100, 310, 70,
"DrugUsageBehavior", DrugUsageBehavior );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _Invitation_pointsX, _Invitation_pointsY,
210, 140, "Invitation", Invitation );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _Organizes_pointsX, _Organizes_pointsY,
330, 140, "Organizes", Organizes );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _Finished_pointsX, _Finished_pointsY,
350, 200, "Finished", Finished );
    }
    if (!_publicOnly) {
      drawState( _panel, _g, 550, 100, 100, 30, 10, 10, "Susceptible",
GOLD, Susceptible, HealthState );
    }
    if (!_publicOnly) {
      drawState( _panel, _g, 550, 170, 100, 30, 10, 10, "Infected",
tomato, Infected, HealthState );
    }
    if (!_publicOnly) {
```

```
    drawFinalState( _panel, _g, 700, 190, 10, 0, "Dead", Dead, HealthState );
   }
   if (!_publicOnly) {
    drawBranchState( _panel, _g, 500, 150, 10, 0, null, branch );
   }
   if (!_publicOnly) {
    drawStatechartEntryPoint( _panel, _g, 460, 150, 488, 150, 430, 130, "HealthState", HealthState );
   }
   if (!_publicOnly) {
    drawBranchExit(_panel, _g, _transition1_pointsX, _transition1_pointsY, 510, 160, null );
   }
   if (!_publicOnly) {
    drawBranchExit(_panel, _g, _transition2_pointsX, _transition2_pointsY, 510, 140, null );
   }
   if (!_publicOnly) {
    drawTransition(_panel, _g, _Infection_pointsX, _Infection_pointsY, 600, 140, "Infection", Infection );
   }
   if (!_publicOnly) {
    drawTransition(_panel, _g, _Death_pointsX, _Death_pointsY, 650, 200, "Death", Death );
   }
   if (!_publicOnly) {
    drawParameter( _panel, _g, 50, 50, 10, 0, "Experience", Experience, false, false );
   }
   if (!_publicOnly) {
    drawParameter( _panel, _g, 50, 100, 10, 0, "UsesSafeSyringe", UsesSafeSyringe, false, false );
   }
   if (!_publicOnly) {
    drawPlainVariable( _panel, _g, 50, 250, 10, 0, "Organizer", Organizer, false );
   }
   if (!_publicOnly) {
    drawPlainVariable( _panel, _g, 50, 150, 10, 0, "XHome", XHome, false );
   }
   if (!_publicOnly) {
    drawPlainVariable( _panel, _g, 50, 200, 10, 0, "YHome", YHome, false );
   }
   if (!_publicOnly) {
    drawCollection( _panel, _g, 50, 300, 10, 0, "InvitedUsers", InvitedUsers );
   }
  }

  @Override
  public boolean onClickModelAt( Panel panel, double x, double y, int clickCount, boolean publicOnly ) {
   if( !publicOnly && modelElementContains(x, y, 50, 50) ) {
    panel.addInspect( 50, 50, this, "Experience" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 50, 100) ) {
    panel.addInspect( 50, 100, this, "UsesSafeSyringe" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 50, 250) ) {
    panel.addInspect( 50, 250, this, "Organizer" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 50, 150) ) {
    panel.addInspect( 50, 150, this, "XHome" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 50, 200) ) {
    panel.addInspect( 50, 200, this, "YHome" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 50, 300) ) {
    panel.addInspect( 50, 300, this, "InvitedUsers" );
    return true;
   }
   return false;
  }

  @Override
  public void onReceive( Object msg, Agent sender ) {

IDUser buddy = (IDUser)sender;
if( msg.equals( "Invitation") ) {
        //invitation should be forwarded to the
        //DrugUsageBehavior statechart
        DrugUsageBehavior.receiveMessage( msg );
        //and the sender should be remembered as Organizer
        Organizer = buddy;
} else if( msg.equals( "Confirm" ) ) {
        //buddy confirms participation
        //add him to the InvitedUsers list
        //according to the experience
        ListIterator<IDUser> it = InvitedUsers.listIterator();
        while( it.hasNext() ) {
                IDUser u = it.next();
                if( u.Experience < buddy.Experience ) {
                        it.previous();
                        break;
                }
        }
        it.add( buddy );
} else if( msg.equals( "Infection") ) {
        //Infection should be forwarded to the
        //HealthState statechart
        HealthState.receiveMessage( msg );
}
  }

  /**
   * Constructor
   */
  public IDUser( Engine engine, ActiveObject owner, ActiveObjectCollection<? extends IDUser> collection ) {
    super( engine, owner, collection );
  }

  @Override
  public void create() {
    // Dynamic initialization of persistent elements
    presentation = new ShapeGroup( IDUser.this, true, 0, 0, 0, oval, oval1 ) {
      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform xform, boolean publicOnly ) {
        setX( getShapeX( 0, 0 ) );
        setY( getShapeY( 0, 0 ) );
        setRotation( getShapeRotation( 0, 0 ) );
        setVisible( isShapeVisible( 0 ) );
        super.draw( panel, graphics, xform, publicOnly );
      }
    };
```

```
    icon = new ShapeGroup( IDUser.this, true, 0, 0, 0       );
    // Agent properties setup
    if ( getEnvironment() != null && (getEnvironment().getSpaceType()
!=                  Environment.SPACE_CONTINUOUS_2D                  ||
getEnvironment().getGISMap() != null) ) {
      throw new RuntimeException( "Space type of an agent class 'IDUser'
doesn't match its environment. See 'Agent' page on the Properties View
of 'IDUser' in the AnyLogic" );
    }
    setVelocity(
500
);
                // Port connectors with non-replicated objects
    // Creating replicated embedded objects
    assignInitialConditions();
    onCreate();
  }

  @Override
  public void start() {
    DrugUsageBehavior.start();
    HealthState.start();
    onStartup();
  }

  public void onStartup() {
    super.onStartup();

//setup home location (within the city bounds that are defined in Main)
get_Main().setHomeLocation( this );
//remember home location - we will return here after every injecting
activity elsewhere
XHome = getX();
YHome = getY();
  }

// User API ------------------------------------
  public Main get_Main() {
    ActiveObject owner = getOwner();
    if ( owner instanceof Main ) return (Main) owner;
    return null;
  }

  // Reaction on changes -------------------------
  public void onChange() {
    DrugUsageBehavior.onChange();
    HealthState.onChange();
  }

  public void onDestroy() {
    super.onDestroy();
    DrugUsageBehavior.onDestroy();
    HealthState.onDestroy();
  }
}
```

Simulation.java

```
package hiv_diffusion_and_syringe_usage;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import javax.swing.JApplet;

public class Simulation extends ExperimentSimulation<Main> {
  static final Font _button_Font = new Font("Dialog", 0, 11 );
  static final Font _text_Font = new Font("Arial", 1, 28 );
  static final Font _text1_Font = new Font("SansSerif", 0, 9 );
  static final Font _text2_Font = _text1_Font;
  static final Font _text3_Font = new Font("SansSerif", 0, 10 );
  static final Font _text4_Font = _text3_Font;
  static final Font _text5_Font = _text3_Font;
  static final Font _text6_Font = _text3_Font;
  static final Font _text7_Font = _text3_Font;
  static final Font _text8_Font = _text3_Font;
  static final Font _text9_Font = new Font("Arial", 1, 11 );
  static final Font _text10_Font = new Font("Arial", 1, 12 );
  static final Font _text11_Font = _text10_Font;
  static final Font _text12_Font = _text10_Font;
  static final int _button = 1;
  static final int _slider = 2;
  static final int _rectangle = 3;
  static final int _text = 4;
  static final int _image = 5;
  static final int _text1 = 6;
  static final int _image1 = 7;
  static final int _text2 = 8;
  static final int _text3 = 9;
  static final int _text4 = 10;
```

```
  static final int _text5 = 11;
  static final int _text6 = 12;
  static final int _text7 = 13;
  static final int _text8 = 14;
  static final int _line = 15;
  static final int _text9 = 16;
  static final int _text10 = 17;
  static final int _text11 = 18;
  static final int _text12 = 19;

  /**
   * Top-level presentation group id
   */
  static final int _presentation = 0;

  /**
   * Top-level icon group id
   */
  static final int _icon = -1;



  @Override
  public void executeShapeControlAction( int _shape, int index ) {
    switch( _shape ) {
      case _button: {
if ( getState() == IDLE )
                run();
getEngine().getPresentation().setPresentable( getEngine().getRoot() );
;}
        break;
      default:
        super.executeShapeControlAction( _shape, index );
        break;
    }
  }

  @Override
  public double getShapeControlMinimum( int _shape, int index ) {
    switch( _shape ) {
      case _slider: return
0
;
      default: return super.getShapeControlMinimum( _shape, index );
    }
  }

  @Override
  public double getShapeControlMaximum( int _shape, int index ) {
    switch( _shape ) {
      case _slider: return
1
;
      default: return super.getShapeControlMaximum( _shape, index );
    }
  }

  @Override
  public double getShapeControlDefaultValueDouble( int _shape, int index
) {
    switch(_shape) {
      case _slider: return 0.1;
      default: return super.getShapeControlDefaultValueDouble( _shape,
index );
    }
  }

  ShapeButton button;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _slider_SetDynamicParams( ShapeSlider shape ) {
    boolean _visible;
    shape.setRange( getShapeControlMinimum( _slider ),
getShapeControlMaximum( _slider ) );
  }

  ShapeSlider slider;
  ShapeRectangle rectangle;
  ShapeText text;
  ShapeImage image;
  ShapeText text1;
  ShapeImage image1;
  ShapeText text2;
  ShapeText text3;
  ShapeText text4;
  ShapeText text5;
  ShapeText text6;
  ShapeText text7;
  ShapeText text8;
  ShapeLine line;
  ShapeText text9;
  ShapeText text10;
  ShapeText text11;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _text12_SetDynamicParams( ShapeText shape ) {
    boolean _visible;
    shape.setX(
slider.getX() + slider.getWidth() * slider.getValue()
);
    shape.setText(
(int)( 100 * slider.getValue() ) + "%"
);
  }

  ShapeText text12;

  // Static initialization of persistent elements
  {
    button = new ShapeButton(
                                        Simulation.this, true, 40, 440,
                                        100, 30,
        controlDefault, controlDefault,
        _button_Font,
                                        "Run the model"
    ) {

      @Override
      public void action(){
        executeShapeControlAction( _button, 0 );
      }
    };
    slider = new ShapeSlider(
                                Simulation.this, true,445, 430,
                                        390, 30,
        transparent,
        false, getShapeControlMinimum( _slider ),
getShapeControlMaximum( _slider ), ShapeControl.TYPE_DOUBLE
    ) {
```

```
    @Override
    public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
      _slider_SetDynamicParams( this );
      super.draw( panel, graphics, xform, publicOnly );
    }

    @Override
    public void setValueToDefault() {
                        setValue( limit( getMin(),
getShapeControlDefaultValueDouble( _slider, 0 ), getMax() ) );
    }
  };
  rectangle = new ShapeRectangle(
    true,0, 0, 0.0,
      null, black,
                                    900, 600,
                                    1, LINE_STYLE_SOLID
  );
  text = new ShapeText(
    true,40, 10, 0.0,
    dodgerBlue,"HIV Diffusion and Syringe Usage",
    _text_Font, ALIGNMENT_LEFT
  );
  image = new ShapeImage(
                    Simulation.this, true, 330, 530, 0.0,
                                    96,                 40,
"/hiv_diffusion_and_syringe_usage/",
                                    new String[]{"rtilogo.png",}

  );
  text1 = new ShapeText(
    true,450, 540, 0.0,
    lightBlue,"The problem definition by \r\nResearch Triangle
Institute\r\nwww.rti.org\r\n",
    _text1_Font, ALIGNMENT_LEFT
  );
  image1 = new ShapeImage(
                    Simulation.this, true, 50, 530, 0.0,
                                    116,                40,
"/hiv_diffusion_and_syringe_usage/",
                                    new String[]{"xjlogo.png",}

  );
  text2 = new ShapeText(
    true,180, 540, 0.0,
    lightBlue,"This AnyLogic™ Model is\r\n© 1992-2007 XJ
Technologies\r\nwww.anylogic.com\r\n\r\n",
    _text2_Font, ALIGNMENT_LEFT
  );
  text3 = new ShapeText(
    true,40, 60, 0.0,
    lightBlue,"At the beginning of the HIV epidemic in the US it was
noticed that\r\nheroin users who have been injecting for a longer time
have smaller prevalence\r\nof HIV than drug users with less history of
injecting. The answer was unexpectedly\r\nsimple. Those who used
longer were higher on the social hierarchy of users\r\nbecause they
knew more dealers and were more likely to find heroin. Thus
they\r\nwere more likely to inject before the others and thus an infected
syringe would\r\nbe less likely to reach them. Such effect is difficult to
model using conventional\r\nmethods, but Agent-Based modeling allows
one to incorporate such\r\nmodification easily.",
    _text3_Font, ALIGNMENT_LEFT
  );
  text4 = new ShapeText(
    true,40, 180, 0.0,
    lightBlue,"We distinguish between the Susceptible and Infected
states of the drug user,\r\ntherefore we use a statechart with two states.
A certain percentage of drug users\r\nare initially HIV infected, they start
in Infected state. Otherwise the drug user\r\ninitializes in the Susceptible
state and will transition to Infected in the event of\r\nusing an infected
syringe during an injecting activity with a certain probability.
The\r\ncorresponding transition Infection is triggered by the message
“Infection” coming\r\nfrom the organizer. We also model the reduced life
duration of an infected\r\nuser by adding a transition from Infected to a
final state Dead triggered by a\r\ntimeout, which is a draw from the
distribution of life duration of an HIV infected\r\nperson, and the action
of the transition is deleting this user from the model.",
    _text4_Font, ALIGNMENT_LEFT
  );
  text5 = new ShapeText(
    true,40, 320, 0.0,
    lightBlue,"The central part of the drug user model is the injecting
activity behavior. Each\r\nuser has a number of buddies who periodically
get together at somebody’s place\r\nand use drugs. The number of
buddies is defined by the network. We use a\r\nstatechart with two
states – Idle and InjectingActivity – to model the drug usage\r\nbehavior.
There are two ways to get to the InjectingActivity state: organize it or
be\r\ninvited. We assume that anybody can organize an injecting activity
and invite\r\nothers. The organizer invites his buddies with a certain
probability FractionInvited.\r\nIf the invited drug user is already with a
different company the invitation is ignored.",
    _text5_Font, ALIGNMENT_LEFT
  );
  text6 = new ShapeText(
    true,450, 60, 0.0,
    lightBlue,"The duration of the injecting activity itself is not
interesting to us, but we set it to 5\r\nhours deterministically for
animation purposes. We will assume that if at least one\r\nuser uses a
safe syringe, that syringe will be used by everyone. Therefore we\r\nfirst
check if anyone has a safe syringe and, if yes, use
InfectionProbabilitySafe for\r\ninfection, which is obviously lower than
InfectionProbability for a regular (unsafe)\r\nsyringe. The actual injection
sequence is modeled in the exit action of the state\r\nInjectingActivity of
the organizer. The organizer iterates through the list of\r\nparticipants
(sorted by the Experience, most experienced users at the
beginning),\r\nstarting with a clean syringe. If a user is already infected,
the syringe is marked as\r\ninfected and the subsequent users will get
infected with the corresponding probability\r\n(in the model terms the
message “Infection” is sent to the user).",
    _text6_Font, ALIGNMENT_LEFT
  );
  text7 = new ShapeText(
    true,450, 210, 0.0,
    lightBlue,"Experience is a parameter that indicates how long the
user has been injecting and\r\ndefines his rank in the social hierarchy of
users. The initial value of Experience for\r\na particular user can be set
up e.g. as a draw from the distribution of experience\r\nthroughout the
drug user population (which in this case is the required input data\r\nfor
the model), or deterministically. Note that Experience is the (only) source
of\r\nagent heterogeneity in this model.",
    _text7_Font, ALIGNMENT_LEFT
  );
  text8 = new ShapeText(
    true,450, 300, 0.0,
    lightBlue,"The following outputs are collected: \r\n• The dynamics
of the drug user population (we do not add new users during\r\n the
simulation).\r\n• The dynamics of the fraction of HIV infected drug
users.\r\n• The distribution of experience among the infected and not
infected users at\r\n the end of the simulation.\r\n",
    _text8_Font, ALIGNMENT_LEFT
  );
  line = new ShapeLine(
                    true,450, 405,
    cornflowerBlue,
                    380,      0,
                    1, LINE_STYLE_SOLID
  );
  text9 = new ShapeText(
    true,450, 390, 0.0,
    cornflowerBlue,"Fraction of drug users who use safe syringes",
    _text9_Font, ALIGNMENT_LEFT
  );
  text10 = new ShapeText(
    true,450, 420, 0.0,
    cornflowerBlue,"0",
    _text10_Font, ALIGNMENT_LEFT
  );
```

```
    text11 = new ShapeText(
        true,830, 420, 0.0,
        cornflowerBlue,"100%",
        _text11_Font, ALIGNMENT_RIGHT
    );
    text12 = new ShapeText(
        true,650, 460, 0.0,
        cornflowerBlue,"10%",
        _text12_Font, ALIGNMENT_CENTER
    ) {

      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
        _text12_SetDynamicParams( this );
        super.draw( panel, graphics, xform, publicOnly );
      }
    };
  }
  ShapeGroup presentation;
  ShapeGroup icon;

  @Override
  public Object getPersistentShape( int _shape ) {
    switch(_shape){
      case _presentation: return presentation;
      case _icon: return icon;

      case _button: return button;
      case _slider: return slider;
      case _rectangle: return rectangle;
      case _text: return text;
      case _image: return image;
      case _text1: return text1;
      case _image1: return image1;
      case _text2: return text2;
      case _text3: return text3;
      case _text4: return text4;
      case _text5: return text5;
      case _text6: return text6;
      case _text7: return text7;
      case _text8: return text8;
      case _line: return line;
      case _text9: return text9;
      case _text10: return text10;
      case _text11: return text11;
      case _text12: return text12;
      default: return null;
    }
  }


  @Override
  public int getWindowWidth() {
    return 900;
  }

  @Override
  public int getWindowHeight() {
    return 600;
  }

  /**
   * Applet class to run experiment as java applet
   */
  public static class Applet extends JApplet {
    Simulation ex;

    @Override
    public void init() {
      ex = new Simulation();
      ex.setup( this );
    }

    @Override
    public void destroy() {
      ex.close();
    }

  }

  @Override
  public void initDefaultRandomNumberGenerator(Engine engine) {
    engine.setDefaultRandomGenerator( new java.util.Random() );
  }

  @Override
  public Main createRoot( Engine engine ) {
    // Create the root object
    return new Main( engine, null, null );
  }

  @Override
  public void setupRootParameters( Main root, boolean
callOnChangeActions ) {
    double InitialNumberOfUsers_xjal =
root._InitialNumberOfUsers_DefaultValue_xjal();
    if (callOnChangeActions) {
      root.set_InitialNumberOfUsers( InitialNumberOfUsers_xjal );
    } else {
      root.InitialNumberOfUsers = InitialNumberOfUsers_xjal;
    }
    double SafeSyringeUsersFraction_xjal =
slider.getValue()
;
    if (callOnChangeActions) {
      root.set_SafeSyringeUsersFraction( SafeSyringeUsersFraction_xjal );
    } else {
      root.SafeSyringeUsersFraction = SafeSyringeUsersFraction_xjal;
    }
    double PercentInitiallyInfected_xjal =
root._PercentInitiallyInfected_DefaultValue_xjal();
    if (callOnChangeActions) {
      root.set_PercentInitiallyInfected( PercentInitiallyInfected_xjal );
    } else {
      root.PercentInitiallyInfected = PercentInitiallyInfected_xjal;
    }
    double MeanHIVLifeDuration_xjal =
root._MeanHIVLifeDuration_DefaultValue_xjal();
    if (callOnChangeActions) {
      root.set_MeanHIVLifeDuration( MeanHIVLifeDuration_xjal );
    } else {
      root.MeanHIVLifeDuration = MeanHIVLifeDuration_xjal;
    }
    double FractionInvited_xjal =
root._FractionInvited_DefaultValue_xjal();
    if (callOnChangeActions) {
      root.set_FractionInvited( FractionInvited_xjal );
    } else {
      root.FractionInvited = FractionInvited_xjal;
    }
    double InfectionProbabilityUnsafe_xjal =
root._InfectionProbabilityUnsafe_DefaultValue_xjal();
```

```
    if (callOnChangeActions) {
      root.set_InfectionProbabilityUnsafe(    InfectionProbabilityUnsafe_xjal
);
    } else {
      root.InfectionProbabilityUnsafe = InfectionProbabilityUnsafe_xjal;
    }
    double                    InfectionProbabilitySafe_xjal                    =
root._InfectionProbabilitySafe_DefaultValue_xjal();
    if (callOnChangeActions) {
      root.set_InfectionProbabilitySafe( InfectionProbabilitySafe_xjal );
    } else {
      root.InfectionProbabilitySafe = InfectionProbabilitySafe_xjal;
    }
  }


  /**
   * Engine setup
   */
  @Override
  public void setupEngine(Engine engine) {
    engine.setATOL( 1.0E-5 );
    engine.setRTOL( 1.0E-5 );
    engine.setTTOL( 1.0E-5 );
    engine.setHTOL( 0.0010 );
    engine.setSolverODE( Engine.SOLVER_ODE_EULER );
    engine.setSolverNAE( Engine.SOLVER_NAE_MODIFIED_NEWTON );
    engine.setSolverDAE( Engine.SOLVER_DAE_RK45_NEWTON );
    engine.setVMethods( 16032 );

    engine.setStartTime( 0.0 );
    engine.setTimeUnit( TIME_UNIT_DAY );
    engine.setStartDate( toDate( 2008, JANUARY, 1, 7, 56, 46 ) );
    engine.setStopDate( toDate( 2010, DECEMBER, 31, 7, 56, 46 ) );
    engine.setRealTimeMode( true );
    engine.setRealTimeScale( 4.0 );
  }

  /**
   * Experiment setup
   */
  @Override
  public void setup( JApplet applet ) {
    setName( "HIV Diffusion and Syringe Usage : Simulation" );

    // Dynamic initialization of persistent elements
    presentation = new ShapeGroup( Simulation.this, true, 0, 0, 0, button,
slider, rectangle, text, image, text1, image1, text2, text3, text4, text5,
text6, text7, text8, line, text9, text10, text11, text12 );
    icon    =    new    ShapeGroup(    Simulation.this,    true,    0,    0,    0
);
    slider.setValueToDefault();
    // Setup presentation
    Presentation    _p    =    new    Presentation(    this,    applet    !=    null    ?
Presentation.MODE_APPLET :
        Presentation.MODE_APPLICATION, applet );
    _p.start();

    Panel _panel = _p.getPanel();
    ToolBar _tb = _p.getToolBar();
    StatusBar _sb = _p.getStatusBar();

    _panel.setZoomEnabled( false );
    _panel.setPanningEnabled( false );
    _panel.setFrameManagementBalance( 2.0 );

    _sb.setSectionVisible( StatusBar.DATE, false );
    _sb.setSectionVisible( StatusBar.EPS, false );
    _sb.setSectionVisible( StatusBar.EXPERIMENT, false );
    _sb.setSectionVisible( StatusBar.FPS, false );
    _sb.setSectionVisible( StatusBar.MEMORY, true );
    _sb.setSectionVisible( StatusBar.SECONDS, true );
    _sb.setSectionVisible( StatusBar.SIMULATION, true );
    _sb.setSectionVisible( StatusBar.STATUS, true );
    _sb.setSectionVisible( StatusBar.STEP, false );
    _sb.setSectionVisible( StatusBar.TIME, true );
    _tb.setSectionVisible( ToolBar.ANIMATION, false );
    _tb.setSectionVisible( ToolBar.EXECUTION, true );
    _tb.setSectionVisible( ToolBar.FILE, false );
    _tb.setSectionVisible( ToolBar.NAVIGATION, true );
    _tb.setSectionVisible( ToolBar.TIME_SCALE, true );
    _tb.setSectionVisible( ToolBar.VIEW, false );
  }

}
```

Παράρτημα 6

UML - Διαγράμματα κλάσεων

Main
+_histLOS_autoUpdateEvent_xjal: EventTimeout = new EventTimeout(this) +_chart_autoUpdateEvent_xjal: EventTimeout = new EventTimeout(this) +network: Network +arrive: Source +networkEnter: NetworkEnter +gotoRegistration: NetworkMoveTo +queueAtReg: Queue +registration: Delay +gotoWaitingRoom1: NetworkMoveTo +gotoWaitingRoom2: NetworkMoveTo +gotoExit: NetworkMoveTo +networkExit: NetworkExit +leave: Sink +triageRoom: NetworkResourcePool +nurse: NetworkResourcePool +seizeTriageRoom: NetworkSeize +gotoTiageRoom: NetworkMoveTo +triage: Delay +releaseTriageRoom: NetworkRelease +releaseNurse: NetworkRelease +seizeECRoom: : NetworkSeize +ECRoom: : NetworkResourcePool +gotoECRoom: : NetworkMoveTo +releaseECRoom: : NetworkRelease +PA: : NetworkResourcePool +technician: : NetworkResourcePool +XRay: : NetworkResourcePool +releasePA: : NetworkRelease +USound: : NetworkResourcePool +selectProcess: SelectOutput +clock: Clock +callNurse: NetworkSeize +callPA: NetworkSeize +histLOS: HistogramSmartData = new HistogramSmartData(20, 0.1, false, false, 0.1, 0.1) ~_radio_Font: Font = new Font("Dialog", 0, 11) ~_radio1_Font: Font = new Font("Dialog", 0, 11) ~_radio2_Font: Font = new Font("Dialog", 0, 11) ~_radio3_Font: Font = new Font("Dialog", 0, 11) ~_text_Font: Font = new Font("SansSerif", 0, 10) ~_text1_Font: Font = new Font("SansSerif", 0, 10) ~_text2_Font: Font = new Font("SansSerif", 0, 10) ~_text7_Font: Font = new Font("SansSerif", 0, 10) ~_text8_Font: Font = new Font("SansSerif", 0, 10) ~_text9_Font: Font = new Font("SansSerif", 0, 10) ~_text15_Font: Font = new Font("SansSerif", 0, 18) ~_text16_Font: Font = new Font("SansSerif", 0, 18) ~_text17_Font: Font = new Font("SansSerif", 0, 18) ~_text18_Font: Font = new Font("SansSerif", 0, 18) ~_text19_Font: Font = new Font("SansSerif", 1, 12) ~_text20_Font: Font = new Font("SansSerif", 1, 12) ~_text21_Font: Font = new Font("SansSerif", 0, 12) ~_text22_Font: Font = new Font("SansSerif", 0, 12) ~_text24_Font: Font = new Font("SansSerif", 1, 12) ~_text25_Font: Font = new Font("SansSerif", 1, 12) ~_text26_Font: Font = new Font("SansSerif", 0, 12) ~_text27_Font: Font = new Font("SansSerif", 0, 12) ~_text28_Font: Font = new Font("SansSerif", 0, 12)

```
~_text29_Font: Font = new Font("SansSerif", 0, 12)
~_text30_Font: Font = new Font("SansSerif", 0, 10)
~_text31_Font: Font = new Font("SansSerif", 0, 10)
~_text32_Font: Font = new Font("SansSerif", 1, 12)
~_text6_Font: Font = new Font("SansSerif", 0, 10)
~_text10_Font: Font = new Font("SansSerif", 0, 10)
~_text11_Font: Font = new Font("SansSerif", 0, 10)
~_text12_Font: Font = new Font("SansSerif", 0, 10)
~_text14_Font: Font = new Font("SansSerif", 0, 10)
~_text3_Font: Font = new Font("SansSerif", 0, 10)
~_text5_Font: Font = new Font("SansSerif", 0, 10)
~_rectXRayFloor_FillColor: Color: new Color(0xFF7F1DEE, true)
~_arc_FillColor: Color=yellow
~_arc1_FillColor: Color=yellow
~_arc2_FillColor: Color=yellow
~_shapeNurse_FillColor: Color = new Color(0xFFE5F6FA, true)
~_shapePA_FillColor: Color = new Color(0xFFC4FE87, true)
~_text19_Color: Color = new Color(0xFF454545, true)
~_text20_Color: Color = new Color(0xFF454545, true)
~_text24_Color: Color = new Color(0xFF454545, true)
~_text25_Color: Color = new Color(0xFF454545, true)
~_text32_Color: Color = new Color(0xFF454545, true)
~_oval6_LineColor: Color = new Color(0xFF7F1DEE, true)
~_radio: int=1
~_radio1: int=2
~_radio2: int=3
~_radio3: int=4
~_rectXRayFloor: int=5
~_arc: int=6
~_arc1: int=7
~_arc2: int=8
~_oval6: int=9
~_group1: int=10
~_rectangle41: int=11
~_rectangle38: int=12
~_rectangle37: int=13
~_rectangle36: int=14
~_rectangle35: int=15
~_rectangle33: int=16
~_polyline18: int=17
~_polyline4: int=18
~_polyline5: int=19
~_polyline6: int=20
~_line11: int=21
~_line12: int=22
~_line14: int=23
~_line15: int=24
~_line16: int=25
~_line17: int=26
~_line18: int=27
~_line19: int=28
~_line20: int=29
~_line21: int=30
~_line22: int=31
~_line23: int=32
~_line24: int=33
~_line26: int=34
~_polyline7: int=35
~_polyline8: int=36
~_polyline9: int=37
~_line27: int=38
~_line29: int=39
~_line30: int=40
~_line31: int=41
~_polyline10: int=42
```

~_polyline11: int=43
~_rectangle25: int=44
~_line33: int=45
~_line34: int=46
~_rectangle26: int=47
~_text: int=48
~_text1: int=49
~_text2: int=50
~_text7: int=51
~_text8: int=52
~_text9: int=53
~_line35: int=54
~_polyline12: int=55
~_polyline13: int=56
~_line36: int=57
~_polyline14: int=58
~_polyline15: int=59
~_line37: int=60
~_line38: int=61
~_polyQueueAtReg: int=62
~_polyTriageRooms: int=63
~_polyline16: int=64
~_polyECRooms: int=65
~_shapePatient: int=66
~_shapeNurse: int=67
~_shapePA: int=68
~_shapeTech: int=69
~_polyline17: int=70
~_line41: int=71
~_line42: int=72
~_curve: int=73
~_line43: int=74
~_line44: int=75
~_line45: int=76
~_curve1: int=77
~_curve2: int=78
~_curve3: int=79
~_curve4: int=80
~_curve5: int=81
~_curve6: int=82
~_curve7: int=83
~_NurseHomePath: int=84
~_PAHomePath: int=85
~_TechHomePath: int=86
~_line46: int=87
~_line49: int=88
~_line48: int=89
~_rectangle31: int=90
~_rectangle30: int=91
~_waitingRoom: int=92
~_rectangle3: int=93
~_rectangle4: int=94
~_rectangle6: int=95
~_rectangle7: int=96
~_rectangle8: int=97
~_rectangle9: int=98
~_rectangle10: int=99
~_xRayRoom: int=100
~_USoundStorage: int=101
~_techHomeRoom: int=102
~_nurseHomeRoom: int=103
~_doctorHomeRoom: int=104
~_entryDoor: int=105
~_exitDoor: int=106
~_rectangle11: int=107

```
~_rectangle12: int=108
~_rectangle13: int=109
~_rectangle15: int=110
~_rectangle16: int=111
~_rectangle17: int=112
~_rectangle18: int=113
~_rectangle19: int=114
~_rectangle20: int=115
~_rectangle21: int=116
~_rectangle22: int=117
~_rectangle23: int=118
~_rectangle24: int=119
~_polyline: int=120
~_line: int=121
~_line1: int=122
~_line4: int=123
~_line5: int=124
~_polyline1: int=125
~_line6: int=126
~_line7: int=127
~_line8: int=128
~_line9: int=129
~_polyline2: int=130
~_polyline3: int=131
~_line10: int=132
~_line32: int=133
~_rectangle27: int=134
~_line39: int=135
~_rectangle29: int=136
~_line40: int=137
~_registrationDesk: int=138
~_rectangle: int=139
~_line47: int=140
~_group: int=141
~_oval1: int=142
~_oval: int=143
~_shapeBusy: int=144
~_oval2: int=145
~_oval3: int=146
~_shapeIdle: int=147
~_XRayHomePath: int=148
~_USoundHomePath: int=149
~_line50: int=150
~_oval5: int=151
~_rectangle32: int=152
~_polyline19: int=153
~_oval4: int=154
~_shapeUSound: int=155
~_text15: int=156
~_text16: int=157
~_line51: int=158
~_line52: int=159
~_text17: int=160
~_rectangle34: int=161
~_text18: int=162
~_line53: int=163
~_line54: int=164
~_text19: int=165
~_text20: int=166
~_polyline20: int=167
~_polyline21: int=168
~_text21: int=169
~_text22: int=170
~_line55: int=171
~_line56: int=172
```

~_rectangle39: int=173
~_text24: int=174
~_polyline22: int=175
~_rectangle40: int=176
~_text25: int=177
~_polyline23: int=178
~_text26: int=179
~_text27: int=180
~_text28: int=181
~_text29: int=182
~_text30: int=183
~_text31: int=184
~_rectangle42: int=185
~_rectangle43: int=186
~_text32: int=187
~_polyline24: int=188
~_rectangle1: int=189
~_text6: int=190
~_text10: int=191
~_text11: int=192
~_text12: int=193
~_text14: int=194
~_text3: int=195
~_text5: int=196
~_chart: int=197
~_chart1: int=198
~_presentation: int=0
~_icon: int=-1
~radio: ShapeRadioButtonGroup
~radio1: ShapeRadioButtonGroup
~radio2: ShapeRadioButtonGroup
~radio3: ShapeRadioButtonGroup
~chart: BarChart
~chart1: Histogram
~rectXRayFloor: ShapeRectangle
~arc: ShapeArc
~arc1: ShapeArc
~arc2: ShapeArc
~oval: ShapeOval
~oval1: ShapeOval
~oval2: ShapeOval
~oval3: ShapeOval
~oval4: ShapeOval
~oval5: ShapeOval
~oval6: ShapeOval
~group: ShapeGroup
~group1: ShapeGroup
~shapeBusy: ShapeGroup
~shapeIdle: ShapeGroup
~shapeUSound: ShapeGroup
~rectangle: ShapeRectangle
~rectangle1: ShapeRectangle
~rectangle3: ShapeRectangle
~rectangle4: ShapeRectangle
~rectangle6: ShapeRectangle
~rectangle7: ShapeRectangle
~rectangle8: ShapeRectangle
~rectangle9: ShapeRectangle
~rectangle10: ShapeRectangle
~rectangle11: ShapeRectangle
~rectangle12: ShapeRectangle
~rectangle13: ShapeRectangle
~rectangle15: ShapeRectangle
~rectangle16: ShapeRectangle
~rectangle17: ShapeRectangle

~rectangle18: ShapeRectangle
~rectangle19: ShapeRectangle
~rectangle20: ShapeRectangle
~rectangle21: ShapeRectangle
~rectangle22: ShapeRectangle
~rectangle23: ShapeRectangle
~rectangle24: ShapeRectangle
~rectangle25: ShapeRectangle
~rectangle26: ShapeRectangle
~rectangle27: ShapeRectangle
~rectangle29: ShapeRectangle
~rectangle30: ShapeRectangle
~rectangle31: ShapeRectangle
~rectangle32: ShapeRectangle
~rectangle33: ShapeRectangle
~rectangle34: ShapeRectangle
~rectangle35: ShapeRectangle
~rectangle36: ShapeRectangle
~rectangle37: ShapeRectangle
~rectangle38: ShapeRectangle
~rectangle39: ShapeRectangle
~rectangle40: ShapeRectangle
~rectangle41: ShapeRectangle
~rectangle42: ShapeRectangle
~rectangle43: ShapeRectangle
~polyline: ShapePolyLine
~polyline1: ShapePolyLine
~polyline2: ShapePolyLine
~polyline3: ShapePolyLine
~polyline4: ShapePolyLine
~polyline5: ShapePolyLine
~polyline6: ShapePolyLine
~polyline7: ShapePolyLine
~polyline8: ShapePolyLine
~polyline9: ShapePolyLine
~polyline10: ShapePolyLine
~polyline11: ShapePolyLine
~polyline12: ShapePolyLine
~polyline13: ShapePolyLine
~polyline14: ShapePolyLine
~polyline15: ShapePolyLine
~polyline16: ShapePolyLine
~polyline17: ShapePolyLine
~polyline18: ShapePolyLine
~polyline19: ShapePolyLine
~polyline20: ShapePolyLine
~polyline21: ShapePolyLine
~polyline22: ShapePolyLine
~polyline23: ShapePolyLine
~polyline24: ShapePolyLine
~line: ShapeLine
~line1: ShapeLine
~line4: ShapeLine
~line5: ShapeLine
~line6: ShapeLine
~line7: ShapeLine
~line8: ShapeLine
~line9: ShapeLine
~line10: ShapeLine
~line11: ShapeLine
~line12: ShapeLine
~line14: ShapeLine
~line15: ShapeLine
~line16: ShapeLine
~line17: ShapeLine

~line18: ShapeLine
~line19: ShapeLine
~line20: ShapeLine
~line21: ShapeLine
~line22: ShapeLine
~line23: ShapeLine
~line24: ShapeLine
~line26: ShapeLine
~line27: ShapeLine
~line29: ShapeLine
~line30: ShapeLine
~line31: ShapeLine
~line32: ShapeLine
~line33: ShapeLine
~line34: ShapeLine
~line35: ShapeLine
~line36: ShapeLine
~line37: ShapeLine
~line38: ShapeLine
~line39: ShapeLine
~line40: ShapeLine
~line41: ShapeLine
~line42: ShapeLine
~line43: ShapeLine
~line44: ShapeLine
~line45: ShapeLine
~line46: ShapeLine
~line47: ShapeLine
~line48: ShapeLine
~line49: ShapeLine
~line50: ShapeLine
~line51: ShapeLine
~line52: ShapeLine
~line53: ShapeLine
~line54: ShapeLine
~line55: ShapeLine
~line56: ShapeLine
~text: ShapeText
~text1: ShapeText
~text2: ShapeText
~text3: ShapeText
~text5: ShapeText
~text6: ShapeText
~text7: ShapeText
~text8: ShapeText
~text9: ShapeText
~text10: ShapeText
~text11: ShapeText
~text12: ShapeText
~text14: ShapeText
~text15: ShapeText
~text16: ShapeText
~text17: ShapeText
~text18: ShapeText
~text19: ShapeText
~text20: ShapeText
~text21: ShapeText
~text22: ShapeText
~text24: ShapeText
~text25: ShapeText
~text26: ShapeText
~text27: ShapeText
~text28: ShapeText
~text29: ShapeText
~text30: ShapeText

~text31: ShapeText
~text32: ShapeText
~polyQueueAtReg: ShapePolyLine
~polyTriageRooms: ShapePolyLine
~polyECRooms: ShapePolyLine
~shapeNurse: ShapePolyLine
~shapePA: ShapePolyLine
~shapeTech: ShapePolyLine
~NurseHomePath: ShapePolyLine
~PAHomePath: ShapePolyLine
~TechHomePath: ShapePolyLine
~XRayHomePath: ShapePolyLine
~USoundHomePath: ShapePolyLine
~shapePatient: ShapeCurve
~curve: ShapeCurve
~curve1: ShapeCurve
~curve2: ShapeCurve
~curve3: ShapeCurve
~curve4: ShapeCurve
~curve5: ShapeCurve
~curve6: ShapeCurve
~curve7: ShapeCurve
~waitingRoom: ShapeRectangle
~xRayRoom: ShapeRectangle
~USoundStorage: ShapeRectangle
~techHomeRoom: ShapeRectangle
~nurseHomeRoom: ShapeRectangle
~doctorHomeRoom: ShapeRectangle
~entryDoor: ShapeRectangle
~exitDoor: ShapeRectangle
~registrationDesk: ShapeRectangle
~presentation: ShapeGroup
~icon: ShapeGroup
~_connector_pointsX: int[*] = {280, 330}
~_connector_pointsY: int[*] = {1140, 1140}
~_connector1_pointsX: int[*] = {380, 430}
~_connector1_pointsY: int[*] = {1140, 1140}
~_connector2_pointsX: int[*] = {480, 530}
~_connector2_pointsY: int[*] = {1140, 1140}
~_connector3_pointsX: int[*] = {580, 630}
~_connector3_pointsY: int[*] = {1140, 1140}
~_connector4_pointsX: int[*] = {680, 730}
~_connector4_pointsY: int[*] = {1140, 1140}
~_connector5_pointsX: int[*] = {660, 710}
~_connector5_pointsY: int[*] = {1460, 1460}
~_connector6_pointsX: int[*] = {760, 810}
~_connector6_pointsY: int[*] = {1460, 1460}
~_connector8_pointsX: int[*] = {70, 70}
~_connector8_pointsY: int[*] = {1150, 1200}
~_connector9_pointsX: int[*] = {70, 70, 30, 30, 70, 70}
~_connector9_pointsY: int[*] = {1150, 1180, 1180, 1250, 1250, 1270}
~_connector10_pointsX: int[*] = {780, 800, 800, 300, 300, 330}
~_connector10_pointsY: int[*] = {1140, 1140, 1190, 1190, 1240, 1240}
~_connector11_pointsX: int[*] = {380, 430}
~_connector11_pointsY: int[*] = {1240, 1240}
~_connector12_pointsX: int[*] = {480, 530}
~_connector12_pointsY: int[*] = {1240, 1240}
~_connector13_pointsX: int[*] = {580, 640}
~_connector13_pointsY: int[*] = {1240, 1240}
~_connector14_pointsX: int[*] = {690, 730}
~_connector14_pointsY: int[*] = {1240, 1240}
~_connector15_pointsX: int[*] = {770, 800, 800, 300, 300, 330}
~_connector15_pointsY: int[*] = {1240, 1240, 1290, 1290, 1340, 1340}
~_connector16_pointsX: int[*] = {380, 430}
~_connector16_pointsY: int[*] = {1340, 1340}

~_connector17_pointsX: int[*] = {70, 70, 140, 140}
~_connector17_pointsY: int[*] = {1150, 1180, 1180, 1200}
~_connector18_pointsX: int[*] = {480, 530}
~_connector18_pointsY: int[*] = {1340, 1340}
~_connector19_pointsX: int[*] = {580, 630}
~_connector19_pointsY: int[*] = {1340, 1340}
~_connector20_pointsX: int[*] = {550, 610}
~_connector20_pointsY: int[*] = {1460, 1460}
~_connector22_pointsX: int[*] = {70, 70, 30, 30, 140, 140}
~_connector22_pointsY: int[*] = {1150, 1180, 1180, 1250, 1250, 1270}
~_connector23_pointsX: int[*] = {70, 70, 30, 30, 70, 70}
~_connector23_pointsY: int[*] = {1150, 1180, 1180, 1320, 1320, 1340}
~_connector24_pointsX: int[*] = {670, 730}
~_connector24_pointsY: int[*] = {1340, 1340}
~_connector25_pointsX: int[*] = {780, 800, 800, 240, 240, 270}
~_connector25_pointsY: int[*] = {1340, 1340, 1390, 1390, 1460, 1460}
~_connector27_pointsX: int[*] = {70, 70, 30, 30, 140, 140}
~_connector27_pointsY: int[*] = {1150, 1180, 1180, 1320, 1320, 1340}
~_connector32_pointsX: int[*] = {450, 510}
~_connector32_pointsY: int[*] = {1460, 1460}
~_connector34_pointsX: int[*] = {70, 70, 30, 30, 70, 70}
~_connector34_pointsY: int[*] = {1150, 1180, 1180, 1390, 1390, 1410}
~_connector35_pointsX: int[*] = {300, 320, 330}
~_connector35_pointsY: int[*] = {1450, 1450, 1440}
~_connector36_pointsX: int[*] = {300, 320, 330}
~_connector36_pointsY: int[*] = {1470, 1470, 1490}
~_connector37_pointsX: int[*] = {370, 410}
~_connector37_pointsY: int[*] = {1440, 1460}
~_connector38_pointsX: int[*] = {370, 410}
~_connector38_pointsY: int[*] = {1490, 1460}

<<create>>+Main(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection)
+getNameOf(_e: EventTimeout): String
+getModeOf(_e: EventTimeout): int
+getFirstOccurrenceTime(_e: EventTimeout): double
+evaluateTimeoutOf(_e: EventTimeout): double
+executeActionOf(_e: EventTimeout)
+getNameOf(ao: ActiveObject): String
+getNameOf(aolist: ActiveObjectCollection): String
-instantiate_network_xjal(): Network
-setupParameters_network_xjal(object: Network)
-create_network_xjal(object: Network)
-instantiate_arrive_xjal(): Network
-setupParameters_arrive_xjal(object: Network)
-create_arrive_xjal(object: Network)
-instantiate_networkEnter_xjal(): NetworkEnter
-setupParameters_networkEnter_xjal(object: NetworkEnter)
-create_networkEnter_xjal(object: NetworkEnter)
-instantiate_gotoRegistration_xjal(): NetworkMoveTo
-setupParameters_gotoRegistration_xjal(object: NetworkMoveTo)
-create_gotoRegistration_xjal(object: NetworkMoveTo)
-instantiate_queueAtReg_xjal(): Queue
-setupParameters_queueAtReg_xjal(object: Queue)
-create_queueAtReg_xjal(object: Queue)
-instantiate_registration_xjal(): Delay
-setupParameters_registration_xjal(object: Delay)
-create_registration_xjal(object: Delay)
-instantiate_triage_xjal(): Delay
-setupParameters_triage_xjal(object: Delay)
-create_triage_xjal(object: Delay)
-instantiate_gotoWaitingRoom1_xjal(): NetworkMoveTo
-setupParameters_gotoWaitingRoom1_xjal(object: NetworkMoveTo)
-create_gotoWaitingRoom1_xjal(object: NetworkMoveTo)
-instantiate_gotoWaitingRoom2_xjal(): NetworkMoveTo
-setupParameters_gotoWaitingRoom2_xjal(object: NetworkMoveTo)
-create_gotoWaitingRoom2_xjal(object: NetworkMoveTo)

-instantiate_gotoExit_xjal(): NetworkMoveTo
-setupParameters_gotoExit_xjal(object: NetworkMoveTo)
-create_gotoExit_xjal(object: NetworkMoveTo)
-instantiate_networkExit_xjal(): NetworkExit
-setupParameters_networkExit_xjal(object: NetworkExit)
-create_networkExit_xjal(object: NetworkExit)
-instantiate_leave_xjal(): Sink
-setupParameters_leave_xjal(object: Sink)
-create_leave_xjal(object: Sink)
-instantiate_triageRoom_xjal(): NetworkResourcePool
-setupParameters_triageRoom_xjal(object: NetworkResourcePool)
-create_triageRoom_xjal(object: NetworkResourcePool)
-instantiate_nurse_xjal(): NetworkResourcePool
-setupParameters_nurse_xjal(object: NetworkResourcePool)
-create_nurse_xjal(object: NetworkResourcePool)
-instantiate_ECRoom_xjal(): NetworkResourcePool
-setupParameters_ECRoom_xjal(object: NetworkResourcePool)
-create_ECRoom_xjal(object: NetworkResourcePool)
-instantiate_PA_xjal(): NetworkResourcePool
-setupParameters_PA_xjal(object: NetworkResourcePool)
-create_PA_xjal(object: NetworkResourcePool)
-instantiate_technician_xjal(): NetworkResourcePool
-setupParameters_technician_xjal(object: NetworkResourcePool)
-create_technician_xjal(object: NetworkResourcePool)
-instantiate_XRay_xjal(): NetworkResourcePool
-setupParameters_XRay_xjal(object: NetworkResourcePool)
-create_XRay_xjal(object: NetworkResourcePool)
-instantiate_USound_xjal(): NetworkResourcePool
-setupParameters_USound_xjal(object: NetworkResourcePool)
-create_USound_xjal(object: NetworkResourcePool)
-instantiate_seizeTriageRoom_xjal(): NetworkSeize
-setupParameters_seizeTriageRoom_xjal(object: NetworkSeize
-create_seizeTriageRoom_xjal(object: NetworkSeize)
-instantiate_seizeECRoom_xjal(): NetworkSeize
-setupParameters_seizeECRoom_xjal(object: NetworkSeize
-create_seizeECRoom_xjal(object: NetworkSeize)
-instantiate_gotoTriageRoom_xjal(): NetworkMoveTo
-setupParameters_gotoTriageRoom_xjal(object: NetworkMoveTo)
-create_gotoTriageRoom_xjal(object: NetworkMoveTo)
-instantiate_gotoECRoom_xjal(): NetworkMoveTo
-setupParameters_gotoECRoom_xjal(object: NetworkMoveTo)
-create_gotoECRoom_xjal(object: NetworkMoveTo)
-instantiate_releaseTriageRoom_xjal(): NetworkRelease
-setupParameters_releaseTriageRoom_xjal(object: NetworkRelease)
-create_releaseTriageRoom_xjal(object: NetworkRelease)
-instantiate_releaseNurse_xjal(): NetworkRelease
-setupParameters_releaseNurse_xjal(object: NetworkRelease)
-create_releaseNurse_xjal(object: NetworkRelease)
-instantiate_releaseECRoom_xjal(): NetworkRelease
-setupParameters_releaseECRoom_xjal(object: NetworkRelease)
-create_releaseECRoom_xjal(object: NetworkRelease)
-instantiate_releasePA_xjal(): NetworkRelease
-setupParameters_releasePA_xjal(object: NetworkRelease)
-create_releasePA_xjal(object: NetworkRelease)
-instantiate_xRayProcess_xjal(): XRayProcess
-setupParameters_xRayProcess_xjal(object: XRayProcess)
-create_xRayProcess_xjal(object: XRayProcess)
-instantiate_uSoundProcess_xjal(): USoundProcess
-setupParameters_uSoundProcess_xjal(object: USoundProcess)
-create_uSoundProcess_xjal(object: USoundProcess)
-instantiate_selectProcess_xjal(): SelectOutput
-setupParameters_selectProcess_xjal(object: SelectOutput)
-create_selectProcess_xjal(object: SelectOutput)
-instantiate_clock_xjal(): Clock
-setupParameters_clock_xjal(object: Clock)

-create_clock_xjal(object: Clock)
-instantiate_callNurse_xjal(): NetworkSeize
-setupParameters_callNurse_xjal(object: NetworkSeize
-create_callNurse_xjal(object: NetworkSeize)
-instantiate_callPA_xjal(): NetworkSeize
-setupParameters_callPA_xjal(object: NetworkSeize
-create_callPA_xjal(object: NetworkSeize)
+_arrive_newEntity_xjal(): Entity
+_arrive_entityShape_xjal(entity: Patient): Shape
+networkEnter_entryNode_xjal(entity: Entity): ShapeRectangle
+networkEnter_speed_xjal(entity: Entity): double
+_gotoRegistration_destinationNode_xjal(entity: Entity): ShapeRectangle
+_gotoWaitingRoom1_destinationNode_xjal(entity: Entity): ShapeRectangle
+_gotoWaitingRoom2_destinationNode_xjal(entity: Entity): ShapeRectangle
+_gotoExit_destinationNode_xjal(entity: Entity): ShapeRectangle
+_leave_onEnter_xjal(entity: Patient)
+_triageRoom_onNewUnit_xjal(unit: ResourceUnit)
+_triageRoom_idleUnitShape_xjal(unit: ResourceUnit): Shape
+_triageRoom_busyUnitShape_xjal(unit: ResourceUnit): Shape
+_nurse_idleUnitShape_xjal(unit: ResourceUnit): Shape
+_triageRoom_onSeize_xjal(unit: ResourceUnit, entity: Entity)
+_triageRoom_onRelease_xjal(unit: ResourceUnit, entity: Entity)
+_seizeTriageRoom_resources_xjal(entity: Entity): NetworkResourcePool
+_gotoTriageRoom_destinationResource_xjal(): NetworkResourcePool
+_releaseTriageRoom_resources_xjal(entity: Entity): NetworkResourcePool
+_releaseNurse_resources_xjal(entity: Entity): NetworkResourcePool
+_triage_delayTime_xjal(entity: Entity): double
+_seizeECRoom_resources_xjal(entity: Entity): NetworkResourcePool
+_gotoECRoom_destinationResource_xjal(): NetworkResourcePool
+_releaseECRoom_resources_xjal(entity: Entity): NetworkResourcePool
+_releasePA_resources_xjal(entity: Entity): NetworkResourcePool
+_callNurse_resources_xjal(entity: Entity): NetworkResourcePool
+_callPA_resources_xjal(entity: Entity): NetworkResourcePool
+_ECRoom_onNewUnit_xjal(unit: ResourceUnit)
+_ECRoom_onSeize_xjal(unit: ResourceUnit, entity: Entity)
+_ECRoom_onRelease_xjal(unit: ResourceUnit, entity: Entity)
+_ECRoom_idleUnitShape_xjal(unit: ResourceUnit): Shape
+_ECRoom_busyUnitShape_xjal(unit: ResourceUnit): Shape
+_PA_idleUnitShape_xjal(unit: ResourceUnit): Shape
+_technician_idleUnitShape_xjal(unit: ResourceUnit): Shape
+_XRay_idleUnitShape_xjal(unit: ResourceUnit): Shape
+_XRay_busyUnitShape_xjal(unit: ResourceUnit): Shape
+_USound_idleUnitShape_xjal(unit: ResourceUnit): Shape
~resetStats()
+onShapeClick(_shape: int, index: int, clickx: double, clicky: double): boolean
+executeShapeControlAction(_shape: int, index: int, value: int)
+getShapeControlDefaultValueInt(_shape: int, index: int): int
-_chart_DataItem0Value(): double
-_chart_DataItem1Value(): double
-_chart_DataItem2Value(): double
-_chart_DataItem3Value(): double
-_chart_DataItem4Value(): double
-_chart_DataItem5Value(): double
-_chart_DataItem6Value(): double
-_rectXRayFloor_SetDynamicParams(shape: ShapeRectangle)
-_group1_SetDynamicParams(shape: ShapeGroup)
-_polyQueueAtReg_SetDynamicParams(shape: ShapePolyLine)
-_text30_SetDynamicParams(shape: ShapeText)
-_text31_SetDynamicParams(shape: ShapeText)
+getPersistentShape(_shape: int): Object
+drawModelElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean)
+onClickModelAt(panel: Panel, x: double, y: double, clickCount: int, publicOnly: boolean): boolean
+create()
+start()
+getEmbeddedObjects(): List

+onDestroy()

XRayProcess
+ed: Main +seizeXRay: NetworkSeize +doExamination: Delay +gotoXRay: NetworkMoveTo +doXRay: Delay +gotoECRoom: NetworkMoveTo +releaseTechXRay: NetworkRelease +callTech: NetworkSeize ~_text_Font: Font = new Font("SansSerif", 1, 18) ~_text1_Font: Font = new Font("SansSerif", 0, 18) ~_text2_Font: Font = new Font("SansSerif", 0, 18) ~_line: int=1 ~_oval: int=2 ~_text: int=3 ~_text1: int=4 ~_line1: int=5 ~_text2: int=6 ~_line2: int=7 ~_rectangle: int=8 ~_presentation: int=0 ~_icon: int=-1 ~line: ShapeLine ~line1: ShapeLine ~line2: ShapeLine ~oval: ShapeOval ~text: ShapeText ~text1: ShapeText ~text2: ShapeText ~rectangle: ShapeRectangle ~presentation: ShapeGroup ~icon: ShapeGroup ~_connector_pointsX: int[*] = {200, 250} ~_connector_pointsY: int[*] = {200, 200} ~_connector1_pointsX: int[*] = {590, 650} ~_connector1_pointsY: int[*] = {200, 200} ~_connector2_pointsX: int[*] = {400, 450} ~_connector2_pointsY: int[*] = {200, 200} ~_connector3_pointsX: int[*] = {100, 150} ~_connector3_pointsY: int[*] = {200, 200} ~_connector4_pointsX: int[*] = {300, 350} ~_connector4_pointsY: int[*] = {200, 200} ~_connector5_pointsX: int[*] = {500, 550} ~_connector5_pointsY: int[*] = {200, 200} +in: Port = new Port(this) +out: Port = new Port(this)
<<create>>+XRayProcess(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection +getNameOf(ao: ActiveObject): String +getNameOf(object: ActiveObjectCollection): String -instantiate_seizeXRay_xjal(): NetworkSeize -setupParameters_seizeXRay_xjal(object: NetworkSeize) -create_seizeXRay_xjal(object: NetworkSeize) -instantiate_doExamination_xjal(): Delay -setupParameters_doExamination_xjal(object: Delay) -create_doExamination_xjal(object: Delay) -instantiate_gotoXRay_xjal(): NetworkMoveTo -setupParameters_gotoXRay_xjal(object: NetworkMoveTo) -create_gotoXRay_xjal(object: NetworkMoveTo) -instantiate_doXRay_xjal(): Delay -setupParameters_doXRay_xjal(object: Delay) -create_doXRay_xjal(object: Delay) -instantiate_gotoECRoom_xjal(): NetworkMoveTo

-setupParameters_gotoECRoom_xjal(object: NetworkMoveTo)
-create_gotoECRoom_xjal(object: NetworkMoveTo)
-instantiate_releaseTechXRay_xjal(): NetworkRelease
-setupParameters_releaseTechXRay_xjal(object: NetworkRelease)
-create_releaseTechXRay_xjal(object: NetworkRelease)
-instantiate_callTech_xjal(): NetworkSeize
-setupParameters_callTech_xjal(object: NetworkSeize)
-create_callTech_xjal(object: NetworkSeize)
+_seizeXRay_resources_xjal(entity: Entity): NetworkResourcePool
+_doExamination_delayTime_xjal(entity: Entity): double
+_gotoXRay_destinationResource_xjal(entity: Entity): NetworkResourcePool
+_doXRay_delayTime_xjal(entity: Entity): double
+_gotoECRoom_destinationResource_xjal(entity: Entity): NetworkResourcePool
+_releaseTechXRay_resources_xjal(entity: Entity): NetworkResourcePool
+_callTech_resources_xjal(entity: Entity): NetworkResourcePool
+onShapeClick(_shape: int, index: int, clickx: double, clicky: double): boolean
+getPersistentShape(_shape: int): Object
+drawModelElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean)
+onClickModelAt(panel: Panel, x: double, y: double, clickCount: intt, publicOnly: boolean): boolean
+getNameOf(_p: Port): String
+create()
+start()
+get_Main(): Main
+getEmbeddedObjects(): List
+onDestroy()

USoundProcess

+ed: Main
+doUSound: Delay
+seizeTechUSound: NetworkSeize
+sendTechToUS: NetworkSendTo
+bringUSound: NetworkSendTo
+returnUSound: NetworkSendTo
+releaseTechUSound: NetworkRelease
~_text_Font: Font = new Font("SansSerif", 1, 18)
~_text1_Font: Font = new Font("SansSerif", 0, 18)
~_text2_Font: Font = new Font("SansSerif", 0, 18)
~_line: int=1
~_oval: int=2
~_text: int=3
~_rectangle: int=4
~_line1: int=5
~_text1: int=6
~_line2: int=7
~_text2: int=8
~_presentation: int=0
~_icon: int=-1
~line: ShapeLine
~line1: ShapeLine
~line2: ShapeLine
~oval: ShapeOval
~text: ShapeText
~text1: ShapeText
~text2: ShapeText
~rectangle: ShapeRectangle
~presentation: ShapeGroup
~icon: ShapeGroup
~_connector_pointsX: int[*] = {400, 450}
~_connector_pointsY: int[*] = {200, 200}
~_connector3_pointsX: int[*] = {300, 350}
~_connector3_pointsY: int[*] = {200, 200}
~_connector8_pointsX: int[*] = {100, 150}
~_connector8_pointsY: int[*] = {200, 200}

~_connector9_pointsX: int[*] = {200, 250}
~_connector9_pointsY: int[*] = {200, 200}
~_connector10_pointsX: int[*] = {500, 550}
~_connector10_pointsY: int[*] = {200, 200}
+in: Port = new Port(this)
+out: Port = new Port(this)

<<create>>+USoundProcess(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection
+getNameOf(ao: ActiveObject): String
+getNameOf(object: ActiveObjectCollection): String
-instantiate_doUSound_xjal(): Delay
-setupParameters_doUSound_xjal(object: Delay)
-create_doUSound_xjal(object: Delay)
-instantiate_seizeTechUSound_xjal(): NetworkSeize
-setupParameters_seizeTechUSound_xjal(object: NetworkSeize)
-create_seizeTechUSound_xjal(object: NetworkSeize)
-instantiate_sendTechtoUS_xjal(): NetworkSendTo
-setupParameters_sendTechtoUS_xjal(object: NetworkSendTo)
-create_sendTechtoUS_xjal(object: NetworkSendTo)
-instantiate_bringUSound_xjal(): NetworkSendTo
-setupParameters_bringUSound_xjal(object: NetworkSendTo)
-create_bringUSound_xjal(object: NetworkSendTo)
-instantiate_returnUSound_xjal(): NetworkSendTo
-setupParameters_returnUSound_xjal(object: NetworkSendTo)
-create_returnUSound_xjal(object: NetworkSendTo)
-instantiate_releaseUSound_xjal(): NetworkRelease
-setupParameters_releaseUSound_xjal(object: NetworkRelease)
-create_releaseUSound_xjal(object: NetworkRelease)
+_doUSound_delayTime_xjal(entity: Entity): double
+_seizeTechUSound_resources_xjal(entity: Entity): NetworkResourcePool
+_sendTechToUS_resources_xjal(entity: Entity): NetworkResourcePool
+_sendTechToUS_destinationResource_xjal(entity: Entity): NetworkResourcePool
+_bringUSound_resources_xjal(entity: Entity): NetworkResourcePool
+_returnUSound_resources_xjal(entity: Entity): NetworkResourcePool
+_returnUSound_destinationResource_xjal(entity: Entity): NetworkResourcePool
+_releaseTechUSound_resources_xjal(entity: Entity): NetworkResourcePool
+onShapeClick(_shape: int, index: int, clickx: double, clicky: double): boolean
+getPersistentShape(_shape: int): Object
+drawModelElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean)
+onClickModelAt(panel: Panel, x: double, y: double, clickCount: intt, publicOnly: boolean): boolean
+getNameOf(_p: Port): String
+create()
+start()
+get_Main(): Main
+getEmbeddedObjects(): List
+onDestroy()

Patient

~timeAdmitted: double
-serialVersionUID: long=-7006651055925022762L

<<create>>+Patient()
<<create>>+Patient(timeAdmitted: double)
+toString(): String

Simulation

~_button_Font: Font = new Font("Dialog", 0, 11)
~_text_Font: Font = new Font("SansSerif", 1, 28)
~_text1_Font: Font = new Font("SansSerif", 0, 9)
~_text2_Font: Font = new Font("SansSerif", 1, 11)
~_text1_Color: Color = new Color(0xFF333333, true)
~_text2_Color: Color = new Color(0xFF454545, true)
~_button: int=1

~_text: int=2 ~_image: int=3 ~_text1: int=4 ~_line: int=5 ~_text2: int=6 ~_presentation: int=0 ~_icon: int=-1 ~button: ShapeButton ~text: ShapeText ~text1: ShapeText ~text2: ShapeText ~image: ShapeImage ~line: ShapeLine ~presentation: ShapeGroup ~icon: ShapeGroup
+executeShapeControlAction(_shape: int, index: int) -_button_SetDynamicParams(shape: ShapeButton) +getPersistentShape(_shape: int): Object +getWindowWidth(): int +getWindowHeight(): int +initDefaultRandomNumberGenerator(engine: Engine) +createRoot(engine: Engine): Main +setupRootParameters(root: Main, callOnChangeActions: boolean) +setupEngine(engine: Engine) +setup(applet: JApplet)

Κώδικας Κεφαλαίου 8 – Έργο Emergency Department

Main.java

```
package emergency_department;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import com.xj.anylogic.libraries.enterprise.*;
import java.awt.geom.Arc2D;

public class Main extends ActiveObject
{
  // Events

  public EventTimeout _histLOS_autoUpdateEvent_xjal = new
EventTimeout(this);
  public EventTimeout _chart_autoUpdateEvent_xjal = new
EventTimeout(this);

  @Override
  public String getNameOf( EventTimeout _e ) {
    if ( _e == _histLOS_autoUpdateEvent_xjal ) return "histLOS auto
update event";
    if ( _e == _chart_autoUpdateEvent_xjal ) return "chart auto update
event";
     return super.getNameOf( _e );
  }

  @Override
  public int getModeOf( EventTimeout _e ) {
    if ( _e == _histLOS_autoUpdateEvent_xjal ) return
EVENT_TIMEOUT_MODE_CYCLIC;
    if ( _e == _chart_autoUpdateEvent_xjal ) return
EVENT_TIMEOUT_MODE_CYCLIC;
    return super.getModeOf( _e );
  }

  @Override
  public double getFirstOccurrenceTime( EventTimeout _e ) {
    if (
       _e == _histLOS_autoUpdateEvent_xjal
       || _e == _chart_autoUpdateEvent_xjal
      ) return getEngine().getStartTime();
    return super.getFirstOccurrenceTime( _e );
  }

  @Override
  public double evaluateTimeoutOf( EventTimeout _e ) {
    if ( _e == _histLOS_autoUpdateEvent_xjal ) return
1
;
    if ( _e == _chart_autoUpdateEvent_xjal ) return
1
;
    return super.evaluateTimeoutOf( _e );
  }

  @Override
  public void executeActionOf( EventTimeout _e ) {
    if ( _e == _histLOS_autoUpdateEvent_xjal ) {
      histLOS.update();
      return;
    }
    if ( _e == _chart_autoUpdateEvent_xjal ) {
      chart.updateData();
      return;
    }
    super.executeActionOf( _e );
  }
  // Embedded Objects

  public Network network;
  public Source<
Patient > arrive;
  public NetworkEnter<
Entity > networkEnter;
```

```
  public NetworkMoveTo<
Entity > gotoRegistration;
  public Queue<
Entity > queueAtReg;
  public Delay<
Entity > registration;
  public NetworkMoveTo<
Entity > gotoWaitingRoom1;
  public NetworkMoveTo<
Entity > gotoExit;
  public NetworkExit<
Entity > networkExit;
  public Sink<
Patient > leave;
  public NetworkResourcePool<
ResourceUnit > triageRoom;
  public NetworkResourcePool<
ResourceUnit > nurse;
  public NetworkSeize<
Entity > seizeTriageRoom;
  public NetworkMoveTo<
Entity > gotoTriageRoom;
  public Delay<
Entity > triage;
  public NetworkMoveTo<
Entity > gotoWaitingRoom2;
  public NetworkRelease<
Entity > releaseTriageRoom;
  public NetworkRelease<
Entity > releaseNurse;
  public NetworkSeize<
Entity > seizeECRoom;
  public NetworkResourcePool<
ResourceUnit > ECRoom;
  public NetworkMoveTo<
Entity > gotoECRoom;
  public NetworkRelease<
Entity > releaseECRoom;
  public NetworkResourcePool<
ResourceUnit > PA;
  public NetworkResourcePool<
ResourceUnit > technician;
  public NetworkResourcePool<
ResourceUnit > XRay;
  public NetworkRelease<
Entity > releasePA;
  public XRayProcess xRayProcess;
  public NetworkResourcePool<
ResourceUnit > USound;
  public USoundProcess uSoundProcess;
  public SelectOutput<
Entity > selectProcess;
  public Clock clock;
  public NetworkSeize<
Entity > callNurse;
  public NetworkSeize<
Entity > callPA;

  public String getNameOf( ActiveObject ao ) {
    if ( ao == network ) return "network";
    if ( ao == arrive ) return "arrive";
    if ( ao == networkEnter ) return "networkEnter";
    if ( ao == gotoRegistration ) return "gotoRegistration";
    if ( ao == queueAtReg ) return "queueAtReg";
    if ( ao == registration ) return "registration";
    if ( ao == gotoWaitingRoom1 ) return "gotoWaitingRoom1";
    if ( ao == gotoExit ) return "gotoExit";
    if ( ao == networkExit ) return "networkExit";
    if ( ao == leave ) return "leave";
    if ( ao == triageRoom ) return "triageRoom";
    if ( ao == nurse ) return "nurse";
    if ( ao == seizeTriageRoom ) return "seizeTriageRoom";
    if ( ao == gotoTriageRoom ) return "gotoTriageRoom";
    if ( ao == triage ) return "triage";
    if ( ao == gotoWaitingRoom2 ) return "gotoWaitingRoom2";
    if ( ao == releaseTriageRoom ) return "releaseTriageRoom";
    if ( ao == releaseNurse ) return "releaseNurse";
    if ( ao == seizeECRoom ) return "seizeECRoom";
    if ( ao == ECRoom ) return "ECRoom";
    if ( ao == gotoECRoom ) return "gotoECRoom";
    if ( ao == releaseECRoom ) return "releaseECRoom";
    if ( ao == PA ) return "PA";
    if ( ao == technician ) return "technician";
    if ( ao == XRay ) return "XRay";
    if ( ao == releasePA ) return "releasePA";
    if ( ao == xRayProcess ) return "xRayProcess";
    if ( ao == USound ) return "USound";
    if ( ao == uSoundProcess ) return "uSoundProcess";
    if ( ao == selectProcess ) return "selectProcess";
    if ( ao == clock ) return "clock";
    if ( ao == callNurse ) return "callNurse";
    if ( ao == callPA ) return "callPA";
    return null;
  }


  public String getNameOf( ActiveObjectCollection<?> aolist ) {
    return null;
  }


  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private Network instantiate_network_xjal() {Network object = new
Network( getEngine(), this, null ) {
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_network_xjal(Network object ) {
    object.networkGroup =
group
;
    object.hideNetwork = object._hideNetwork_DefaultValue_xjal();
    object.staticDrawMode                                          =
object._staticDrawMode_DefaultValue_xjal();
    object.enablePriorities = object._enablePriorities_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_network_xjal(Network object ) {
```

```
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private Source<
Patient > instantiate_arrive_xjal() {Source<
Patient > object = new Source<
Patient >( getEngine(), this, null ) {
      @Override
      public Entity newEntity( ) {
        return _arrive_newEntity_xjal( );
      }
      @Override
      public Shape entityShape( Patient entity ) {
        return _arrive_entityShape_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_arrive_xjal(Source<
Patient > object ) {
    object.arrivalType = object._arrivalType_DefaultValue_xjal();
    object.rate =
0.14
;
    object.rateTable = object._rateTable_DefaultValue_xjal();
    object.arrivalTable = object._arrivalTable_DefaultValue_xjal();
    object.limitArrivals = object._limitArrivals_DefaultValue_xjal();
    object.maxArrivals = object._maxArrivals_DefaultValue_xjal();
    object.uniqueShape = object._uniqueShape_DefaultValue_xjal();
    object.enableRotation =
false
;
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_arrive_xjal(Source<
Patient > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkEnter<
Entity > instantiate_networkEnter_xjal() {NetworkEnter<
Entity > object = new NetworkEnter<
Entity >( getEngine(), this, null ) {
      @Override
      public ShapeRectangle entryNode( Entity entity ) {
        return _networkEnter_entryNode_xjal( entity );
      }
      @Override
      public double speed( Entity entity ) {
        return _networkEnter_speed_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_networkEnter_xjal(NetworkEnter<
Entity > object ) {
    object.network =
network
;
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_networkEnter_xjal(NetworkEnter<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkMoveTo<
Entity > instantiate_gotoRegistration_xjal() {NetworkMoveTo<
Entity > object = new NetworkMoveTo<
Entity >( getEngine(), this, null ) {
      @Override
      public ShapeRectangle destinationNode( Entity entity ) {
        return _gotoRegistration_destinationNode_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_gotoRegistration_xjal(NetworkMoveTo<
Entity > object ) {
    object.destinationIsNode                                    =
object._destinationIsNode_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_gotoRegistration_xjal(NetworkMoveTo<
Entity > object ) {
    object.create();

  }
  /**
```

```
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private Queue<
Entity > instantiate_queueAtReg_xjal() {Queue<
Entity > object = new Queue<
Entity >( getEngine(), this, null ) {
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_queueAtReg_xjal(Queue<
Entity > object ) {
    object.capacity =
20
;
    object.maximumCapacity =
object._maximumCapacity_DefaultValue_xjal();
    object.enableTimeout = object._enableTimeout_DefaultValue_xjal();
    object.enablePreemption =
object._enablePreemption_DefaultValue_xjal();
    object.animationGuide =
polyQueueAtReg
;
    object.animationType = object._animationType_DefaultValue_xjal();
    object.animationForward =
object._animationForward_DefaultValue_xjal();
    object.enableStats = object._enableStats_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_queueAtReg_xjal(Queue<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private Delay<
Entity > instantiate_registration_xjal() {Delay<
Entity > object = new Delay<
Entity >( getEngine(), this, null ) {
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_registration_xjal(Delay<
Entity > object ) {
    object.delayTimeDefinedByPath =
object._delayTimeDefinedByPath_DefaultValue_xjal();
    object.capacity = object._capacity_DefaultValue_xjal();
    object.maximumCapacity =
object._maximumCapacity_DefaultValue_xjal();
    object.animationGuide = object._animationGuide_DefaultValue_xjal();
    object.animationType = object._animationType_DefaultValue_xjal();
    object.animationForward =
object._animationForward_DefaultValue_xjal();
    object.enableStats = object._enableStats_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_registration_xjal(Delay<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkMoveTo<
Entity > instantiate_gotoWaitingRoom1_xjal() {NetworkMoveTo<
Entity > object = new NetworkMoveTo<
Entity >( getEngine(), this, null ) {
      @Override
      public ShapeRectangle destinationNode( Entity entity ) {
        return _gotoWaitingRoom1_destinationNode_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void
setupParameters_gotoWaitingRoom1_xjal(NetworkMoveTo<
Entity > object ) {
    object.destinationIsNode =
object._destinationIsNode_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_gotoWaitingRoom1_xjal(NetworkMoveTo<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkMoveTo<
Entity > instantiate_gotoExit_xjal() {NetworkMoveTo<
Entity > object = new NetworkMoveTo<
Entity >( getEngine(), this, null ) {
      @Override
      public ShapeRectangle destinationNode( Entity entity ) {
        return _gotoExit_destinationNode_xjal( entity );
      }
```

```
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_gotoExit_xjal(NetworkMoveTo<
Entity > object ) {
    object.destinationIsNode                                    =
object._destinationIsNode_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_gotoExit_xjal(NetworkMoveTo<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkExit<
Entity > instantiate_networkExit_xjal() {NetworkExit<
Entity > object = new NetworkExit<
Entity >( getEngine(), this, null ) {
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_networkExit_xjal(NetworkExit<
Entity > object ) {
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_networkExit_xjal(NetworkExit<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private Sink<
Patient > instantiate_leave_xjal() {Sink<
Patient > object = new Sink<
Patient >( getEngine(), this, null ) {
      @Override
      public void onEnter( Patient entity ) {
        _leave_onEnter_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_leave_xjal(Sink<
Patient > object ) {
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_leave_xjal(Sink<
Patient > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkResourcePool<
ResourceUnit > instantiate_triageRoom_xjal() {NetworkResourcePool<
ResourceUnit > object = new NetworkResourcePool<
ResourceUnit >( getEngine(), this, null ) {
      @Override
      public void onNewUnit( ResourceUnit unit ) {
        _triageRoom_onNewUnit_xjal( unit );
      }
      @Override
      public void onSeize( ResourceUnit unit, Entity entity ) {
        _triageRoom_onSeize_xjal( unit, entity );
      }
      @Override
      public void onRelease( ResourceUnit unit, Entity entity ) {
        _triageRoom_onRelease_xjal( unit, entity );
      }
      @Override
      public Shape idleUnitShape( ResourceUnit unit ) {
        return _triageRoom_idleUnitShape_xjal( unit );
      }
      @Override
      public Shape busyUnitShape( ResourceUnit unit ) {
        return _triageRoom_busyUnitShape_xjal( unit );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_triageRoom_xjal(NetworkResourcePool<
ResourceUnit > object ) {
    object.type = object._type_DefaultValue_xjal();
    object.capacityDefinitionType =
NetworkResourcePool.CAPACITY_HOME_SHAPE
;
```

```
    object.capacity = object._capacity_DefaultValue_xjal();
    object.capacityTable = object._capacityTable_DefaultValue_xjal();
    object.speed = object._speed_DefaultValue_xjal();
    object.uniqueShape = object._uniqueShape_DefaultValue_xjal();
    object.enableRotation = object._enableRotation_DefaultValue_xjal();
    object.homeShapeType =
NetworkResourcePool.HOME_PATH
;
    object.homeNode = object._homeNode_DefaultValue_xjal();
    object.homePath =
polyTriageRooms
;
    object.enableStats =
true
;
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_triageRoom_xjal(NetworkResourcePool<
ResourceUnit > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkResourcePool<
ResourceUnit > instantiate_nurse_xjal() {NetworkResourcePool<
ResourceUnit > object = new NetworkResourcePool<
ResourceUnit >( getEngine(), this, null ) {
      @Override
      public Shape idleUnitShape( ResourceUnit unit ) {
        return _nurse_idleUnitShape_xjal( unit );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_nurse_xjal(NetworkResourcePool<
ResourceUnit > object ) {
    object.type =
Network.RESOURCE_MOVING
;
    object.capacityDefinitionType                                    =
object._capacityDefinitionType_DefaultValue_xjal();
    object.capacity =
5
;
    object.capacityTable = object._capacityTable_DefaultValue_xjal();
    object.speed =
uniform( 2000, 2500 )
;
    object.uniqueShape = object._uniqueShape_DefaultValue_xjal();
    object.enableRotation =
false
;
    object.homeShapeType =
NetworkResourcePool.HOME_PATH
;
    object.homeNode =
nurseHomeRoom
;
    object.homePath =
NurseHomePath
;
    object.enableStats =
true
;
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_nurse_xjal(NetworkResourcePool<
ResourceUnit > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkSeize<
Entity > instantiate_seizeTriageRoom_xjal() {NetworkSeize<
Entity > object = new NetworkSeize<
Entity >( getEngine(), this, null ) {
      @Override
      public NetworkResourcePool[] resources( Entity entity ) {
        return _seizeTriageRoom_resources_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_seizeTriageRoom_xjal(NetworkSeize<
Entity > object ) {
    object.capacity = object._capacity_DefaultValue_xjal();
    object.maximumCapacity =
true
;
    object.enableTimeout = object._enableTimeout_DefaultValue_xjal();
    object.enablePreemption                                          =
object._enablePreemption_DefaultValue_xjal();
    object.sendResources = object._sendResources_DefaultValue_xjal();
    object.destinationType                                           =
object._destinationType_DefaultValue_xjal();
    object.attachResources                                           =
object._attachResources_DefaultValue_xjal();
    object.animationGuide = object._animationGuide_DefaultValue_xjal();
    object.animationType = object._animationType_DefaultValue_xjal();
    object.animationForward                                          =
object._animationForward_DefaultValue_xjal();
    object.enableStats = object._enableStats_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
```

```
  */
  private void create_seizeTriageRoom_xjal(NetworkSeize<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkMoveTo<
Entity > instantiate_gotoTriageRoom_xjal() {NetworkMoveTo<
Entity > object = new NetworkMoveTo<
Entity >( getEngine(), this, null ) {
      @Override
      public NetworkResourcePool destinationResource(  ) {
        return _gotoTriageRoom_destinationResource_xjal(  );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_gotoTriageRoom_xjal(NetworkMoveTo<
Entity > object ) {
    object.destinationIsNode =
false
;
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_gotoTriageRoom_xjal(NetworkMoveTo<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private Delay<
Entity > instantiate_triage_xjal() {Delay<
Entity > object = new Delay<
Entity >( getEngine(), this, null ) {
      @Override
      public double delayTime( Entity entity ) {
        return _triage_delayTime_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_triage_xjal(Delay<
Entity > object ) {
    object.delayTimeDefinedByPath                                =
object._delayTimeDefinedByPath_DefaultValue_xjal();
    object.capacity = object._capacity_DefaultValue_xjal();
    object.maximumCapacity =
true
;
    object.animationGuide = object._animationGuide_DefaultValue_xjal();
    object.animationType = object._animationType_DefaultValue_xjal();
    object.animationForward                                      =
object._animationForward_DefaultValue_xjal();
    object.enableStats = object._enableStats_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_triage_xjal(Delay<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkMoveTo<
Entity > instantiate_gotoWaitingRoom2_xjal() {NetworkMoveTo<
Entity > object = new NetworkMoveTo<
Entity >( getEngine(), this, null ) {
      @Override
      public ShapeRectangle destinationNode( Entity entity ) {
        return _gotoWaitingRoom2_destinationNode_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private                                                        void
setupParameters_gotoWaitingRoom2_xjal(NetworkMoveTo<
Entity > object ) {
    object.destinationIsNode                                     =
object._destinationIsNode_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_gotoWaitingRoom2_xjal(NetworkMoveTo<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkRelease<
Entity > instantiate_releaseTriageRoom_xjal() {NetworkRelease<
Entity > object = new NetworkRelease<
```

```
Entity >( getEngine(), this, null ) {
      @Override
      public NetworkResourcePool[] resources( Entity entity ) {
        return _releaseTriageRoom_resources_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private                                                    void
setupParameters_releaseTriageRoom_xjal(NetworkRelease<
Entity > object ) {
    object.releaseAll = object._releaseAll_DefaultValue_xjal();
    object.movingGoHome                                      =
object._movingGoHome_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_releaseTriageRoom_xjal(NetworkRelease<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkRelease<
Entity > instantiate_releaseNurse_xjal() {NetworkRelease<
Entity > object = new NetworkRelease<
Entity >( getEngine(), this, null ) {
      @Override
      public NetworkResourcePool[] resources( Entity entity ) {
        return _releaseNurse_resources_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_releaseNurse_xjal(NetworkRelease<
Entity > object ) {
    object.releaseAll = object._releaseAll_DefaultValue_xjal();
    object.movingGoHome                                      =
object._movingGoHome_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_releaseNurse_xjal(NetworkRelease<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkSeize<
Entity > instantiate_seizeECRoom_xjal() {NetworkSeize<
Entity > object = new NetworkSeize<
Entity >( getEngine(), this, null ) {
      @Override
      public NetworkResourcePool[] resources( Entity entity ) {
        return _seizeECRoom_resources_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_seizeECRoom_xjal(NetworkSeize<
Entity > object ) {
    object.capacity = object._capacity_DefaultValue_xjal();
    object.maximumCapacity =
true
;
    object.enableTimeout = object._enableTimeout_DefaultValue_xjal();
    object.enablePreemption                                  =
object._enablePreemption_DefaultValue_xjal();
    object.sendResources = object._sendResources_DefaultValue_xjal();
    object.destinationType                                   =
object._destinationType_DefaultValue_xjal();
    object.attachResources                                   =
object._attachResources_DefaultValue_xjal();
    object.animationGuide = object._animationGuide_DefaultValue_xjal();
    object.animationType = object._animationType_DefaultValue_xjal();
    object.animationForward                                  =
object._animationForward_DefaultValue_xjal();
    object.enableStats = object._enableStats_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_seizeECRoom_xjal(NetworkSeize<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkResourcePool<
ResourceUnit > instantiate_ECRoom_xjal() {NetworkResourcePool<
ResourceUnit > object = new NetworkResourcePool<
ResourceUnit >( getEngine(), this, null ) {
      @Override
      public void onNewUnit( ResourceUnit unit ) {
        _ECRoom_onNewUnit_xjal( unit );
      }
      @Override
      public void onSeize( ResourceUnit unit, Entity entity ) {
```

```
      _ECRoom_onSeize_xjal( unit, entity );
    }
    @Override
    public void onRelease( ResourceUnit unit, Entity entity ) {
      _ECRoom_onRelease_xjal( unit, entity );
    }
    @Override
    public Shape idleUnitShape( ResourceUnit unit ) {
      return _ECRoom_idleUnitShape_xjal( unit );
    }
    @Override
    public Shape busyUnitShape( ResourceUnit unit ) {
      return _ECRoom_busyUnitShape_xjal( unit );
    }
   };

   return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_ECRoom_xjal(NetworkResourcePool<
ResourceUnit > object ) {
    object.type = object._type_DefaultValue_xjal();
    object.capacityDefinitionType =
NetworkResourcePool.CAPACITY_HOME_SHAPE
;
    object.capacity = object._capacity_DefaultValue_xjal();
    object.capacityTable = object._capacityTable_DefaultValue_xjal();
    object.speed = object._speed_DefaultValue_xjal();
    object.uniqueShape = object._uniqueShape_DefaultValue_xjal();
    object.enableRotation = object._enableRotation_DefaultValue_xjal();
    object.homeShapeType =
NetworkResourcePool.HOME_PATH
;
    object.homeNode = object._homeNode_DefaultValue_xjal();
    object.homePath =
polyECRooms
;
    object.enableStats =
true
;
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_ECRoom_xjal(NetworkResourcePool<
ResourceUnit > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkMoveTo<
Entity > instantiate_gotoECRoom_xjal() {NetworkMoveTo<
Entity > object = new NetworkMoveTo<
Entity >( getEngine(), this, null ) {
      @Override
      public NetworkResourcePool destinationResource( ) {
        return _gotoECRoom_destinationResource_xjal( );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_gotoECRoom_xjal(NetworkMoveTo<
Entity > object ) {
    object.destinationIsNode =
false
;
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_gotoECRoom_xjal(NetworkMoveTo<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkRelease<
Entity > instantiate_releaseECRoom_xjal() {NetworkRelease<
Entity > object = new NetworkRelease<
Entity >( getEngine(), this, null ) {
      @Override
      public NetworkResourcePool[] resources( Entity entity ) {
        return _releaseECRoom_resources_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_releaseECRoom_xjal(NetworkRelease<
Entity > object ) {
    object.releaseAll = object._releaseAll_DefaultValue_xjal();
    object.movingGoHome                                          =
object._movingGoHome_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_releaseECRoom_xjal(NetworkRelease<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
```

```
   * <i>This method should not be called by user</i>
   */
  private NetworkResourcePool<
ResourceUnit > instantiate_PA_xjal() {NetworkResourcePool<
ResourceUnit > object = new NetworkResourcePool<
ResourceUnit >( getEngine(), this, null ) {
      @Override
      public Shape idleUnitShape( ResourceUnit unit ) {
        return _PA_idleUnitShape_xjal( unit );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_PA_xjal(NetworkResourcePool<
ResourceUnit > object ) {
    object.type =
Network.RESOURCE_MOVING
;
    object.capacityDefinitionType                                    =
object._capacityDefinitionType_DefaultValue_xjal();
    object.capacity =
5
;
    object.capacityTable = object._capacityTable_DefaultValue_xjal();
    object.speed =
uniform( 2000, 2500 )
;
    object.uniqueShape = object._uniqueShape_DefaultValue_xjal();
    object.enableRotation =
false
;
    object.homeShapeType =
NetworkResourcePool.HOME_PATH
;
    object.homeNode =
nurseHomeRoom
;
    object.homePath =
PAHomePath
;
    object.enableStats =
true
;
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_PA_xjal(NetworkResourcePool<
ResourceUnit > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkResourcePool<
ResourceUnit > instantiate_technician_xjal() {NetworkResourcePool<
ResourceUnit > object = new NetworkResourcePool<
ResourceUnit >( getEngine(), this, null ) {
      @Override
      public Shape idleUnitShape( ResourceUnit unit ) {
        return _technician_idleUnitShape_xjal( unit );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_technician_xjal(NetworkResourcePool<
ResourceUnit > object ) {
    object.type =
Network.RESOURCE_MOVING
;
    object.capacityDefinitionType                                    =
object._capacityDefinitionType_DefaultValue_xjal();
    object.capacity =
3
;
    object.capacityTable = object._capacityTable_DefaultValue_xjal();
    object.speed =
uniform( 2000, 2500 )
;
    object.uniqueShape = object._uniqueShape_DefaultValue_xjal();
    object.enableRotation =
false
;
    object.homeShapeType =
NetworkResourcePool.HOME_PATH
;
    object.homeNode =
techHomeRoom
;
    object.homePath =
TechHomePath
;
    object.enableStats =
true
;
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_technician_xjal(NetworkResourcePool<
ResourceUnit > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkResourcePool<
ResourceUnit > instantiate_XRay_xjal() {NetworkResourcePool<
ResourceUnit > object = new NetworkResourcePool<
ResourceUnit >( getEngine(), this, null ) {
      @Override
      public Shape idleUnitShape( ResourceUnit unit ) {
```

```
      return _XRay_idleUnitShape_xjal( unit );
    }
    @Override
    public Shape busyUnitShape( ResourceUnit unit ) {
     return _XRay_busyUnitShape_xjal( unit );
    }
   };

   return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_XRay_xjal(NetworkResourcePool<
ResourceUnit > object ) {
    object.type = object._type_DefaultValue_xjal();
    object.capacityDefinitionType                                    =
object._capacityDefinitionType_DefaultValue_xjal();
    object.capacity = object._capacity_DefaultValue_xjal();
    object.capacityTable = object._capacityTable_DefaultValue_xjal();
    object.speed = object._speed_DefaultValue_xjal();
    object.uniqueShape = object._uniqueShape_DefaultValue_xjal();
    object.enableRotation = object._enableRotation_DefaultValue_xjal();
    object.homeShapeType =
NetworkResourcePool.HOME_PATH
;
    object.homeNode =
xRayRoom
;
    object.homePath =
XRayHomePath
;
    object.enableStats =
true
;
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_XRay_xjal(NetworkResourcePool<
ResourceUnit > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkRelease<
Entity > instantiate_releasePA_xjal() {NetworkRelease<
Entity > object = new NetworkRelease<
Entity >( getEngine(), this, null ) {
     @Override
     public NetworkResourcePool[] resources( Entity entity ) {
      return _releasePA_resources_xjal( entity );
     }
    };

   return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_releasePA_xjal(NetworkRelease<
Entity > object ) {
    object.releaseAll = object._releaseAll_DefaultValue_xjal();
    object.movingGoHome                                              =
object._movingGoHome_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_releasePA_xjal(NetworkRelease<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private XRayProcess instantiate_xRayProcess_xjal() {XRayProcess
object = new XRayProcess( getEngine(), this, null );

   return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_xRayProcess_xjal(XRayProcess object ) {
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_xRayProcess_xjal(XRayProcess object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkResourcePool<
ResourceUnit > instantiate_USound_xjal() {NetworkResourcePool<
ResourceUnit > object = new NetworkResourcePool<
ResourceUnit >( getEngine(), this, null ) {
     @Override
     public Shape idleUnitShape( ResourceUnit unit ) {
      return _USound_idleUnitShape_xjal( unit );
     }
    };

   return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_USound_xjal(NetworkResourcePool<
```

```
ResourceUnit > object ) {
    object.type = Network.RESOURCE_PORTABLE ;
    object.capacityDefinitionType =
object._capacityDefinitionType_DefaultValue_xjal();
    object.capacity = 2 ;
    object.capacityTable = object._capacityTable_DefaultValue_xjal();
    object.speed = object._speed_DefaultValue_xjal();
    object.uniqueShape = object._uniqueShape_DefaultValue_xjal();
    object.enableRotation = false ;
    object.homeShapeType = NetworkResourcePool.HOME_PATH ;
    object.homeNode = object._homeNode_DefaultValue_xjal();
    object.homePath =
USoundHomePath
;
    object.enableStats =
true
;
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_USound_xjal(NetworkResourcePool<
ResourceUnit > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private USoundProcess instantiate_uSoundProcess_xjal()
{USoundProcess object = new USoundProcess( getEngine(), this, null );

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_uSoundProcess_xjal(USoundProcess
object ) {
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_uSoundProcess_xjal(USoundProcess object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private SelectOutput<
Entity > instantiate_selectProcess_xjal() {SelectOutput<
Entity > object = new SelectOutput<
Entity >( getEngine(), this, null ) {
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_selectProcess_xjal(SelectOutput<
Entity > object ) {
    object.conditionIsProbabilistic =
object._conditionIsProbabilistic_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_selectProcess_xjal(SelectOutput<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private Clock instantiate_clock_xjal() {Clock object = new Clock(
getEngine(), this, null );

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_clock_xjal(Clock object ) {
    object.backgroundColor =
object._backgroundColor_DefaultValue_xjal();
    object.borderColor = object._borderColor_DefaultValue_xjal();
    object.hourMarkColor = object._hourMarkColor_DefaultValue_xjal();
    object.textColor = object._textColor_DefaultValue_xjal();
    object.handsColor = object._handsColor_DefaultValue_xjal();
    object.secondHandColor =
object._secondHandColor_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_clock_xjal(Clock object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkSeize<
Entity > instantiate_callNurse_xjal() {NetworkSeize<
Entity > object = new NetworkSeize<
Entity >( getEngine(), this, null ) {
      @Override
      public NetworkResourcePool[] resources( Entity entity ) {
        return _callNurse_resources_xjal( entity );
      }
    };
```

```
    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_callNurse_xjal(NetworkSeize<
Entity > object ) {
    object.capacity = object._capacity_DefaultValue_xjal();
    object.maximumCapacity =
true
;
    object.enableTimeout = object._enableTimeout_DefaultValue_xjal();
    object.enablePreemption =
object._enablePreemption_DefaultValue_xjal();
    object.sendResources =
true
;
    object.destinationType =
NetworkSendTo.DEST_ENTITY
;
    object.attachResources =
true
;
    object.animationGuide = object._animationGuide_DefaultValue_xjal();
    object.animationType = object._animationType_DefaultValue_xjal();
    object.animationForward =
object._animationForward_DefaultValue_xjal();
    object.enableStats = object._enableStats_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_callNurse_xjal(NetworkSeize<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkSeize<
Entity > instantiate_callPA_xjal() {NetworkSeize<
Entity > object = new NetworkSeize<
Entity >( getEngine(), this, null ) {
      @Override
      public NetworkResourcePool[] resources( Entity entity ) {
        return _callPA_resources_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_callPA_xjal(NetworkSeize<
Entity > object ) {
    object.capacity = object._capacity_DefaultValue_xjal();
    object.maximumCapacity =
true
;
    object.enableTimeout = object._enableTimeout_DefaultValue_xjal();
    object.enablePreemption =
object._enablePreemption_DefaultValue_xjal();
    object.sendResources =
true
;
    object.destinationType =
NetworkSendTo.DEST_ENTITY
;
    object.attachResources =
true
;
    object.animationGuide = object._animationGuide_DefaultValue_xjal();
    object.animationType = object._animationType_DefaultValue_xjal();
    object.animationForward =
object._animationForward_DefaultValue_xjal();
    object.enableStats = object._enableStats_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_callPA_xjal(NetworkSeize<
Entity > object ) {
    object.create();

  }

  public Entity _arrive_newEntity_xjal( ) {
    return
new Patient( time() )
;
  }
  public Shape _arrive_entityShape_xjal( Patient entity ) {
    return
shapePatient
;
  }
  public ShapeRectangle _networkEnter_entryNode_xjal( Entity entity ) {
    return
entryDoor
;
  }
  public double _networkEnter_speed_xjal( Entity entity ) {
    return
uniform( 1500, 2000 )
;
  }
  public ShapeRectangle _gotoRegistration_destinationNode_xjal( Entity
entity ) {
    return
registrationDesk
;
  }
  public ShapeRectangle _gotoWaitingRoom1_destinationNode_xjal(
Entity entity ) {
    return
waitingRoom
;
  }
  public ShapeRectangle _gotoExit_destinationNode_xjal( Entity entity ) {
    return
exitDoor
;
```

```
  }
  public void _leave_onEnter_xjal( Patient entity ) {

histLOS.add( time() - entity.timeAdmitted );
;
  }
  public void _triageRoom_onNewUnit_xjal( ResourceUnit unit ) {

unit.getNetworkHomeLocation().setVisible( true );
unit.getNetworkHomeLocation().setLineColor( null );
;
  }
  public void _triageRoom_onSeize_xjal( ResourceUnit unit, Entity entity )
{

unit.getNetworkHomeLocation().setFillColor( paleGreen );
;
  }
  public void _triageRoom_onRelease_xjal( ResourceUnit unit, Entity
entity ) {

unit.getNetworkHomeLocation().setFillColor( null );
;
  }
  public Shape _triageRoom_idleUnitShape_xjal( ResourceUnit unit ) {
    return
shapeIdle
;
  }
  public Shape _triageRoom_busyUnitShape_xjal( ResourceUnit unit ) {
    return
shapeBusy
;
  }
  public Shape _nurse_idleUnitShape_xjal( ResourceUnit unit ) {
    return
shapeNurse
;
  }
  public NetworkResourcePool[] _seizeTriageRoom_resources_xjal( Entity
entity ) {
    return new NetworkResourcePool[]
{ triageRoom }
;
  }
  public                                              NetworkResourcePool
_gotoTriageRoom_destinationResource_xjal( ) {
    return
triageRoom
;
  }
  public double _triage_delayTime_xjal( Entity entity ) {
    return
triangular( 5, 8, 15 )
;
  }
  public ShapeRectangle _gotoWaitingRoom2_destinationNode_xjal(
Entity entity ) {
    return
waitingRoom
;
  }
  public NetworkResourcePool[] _releaseTriageRoom_resources_xjal(
Entity entity ) {
    return new NetworkResourcePool[]
{ triageRoom }
;
  }
  public NetworkResourcePool[] _releaseNurse_resources_xjal( Entity
entity ) {
    return new NetworkResourcePool[]
{ nurse }
;
  }
  public NetworkResourcePool[] _seizeECRoom_resources_xjal( Entity
entity ) {
    return new NetworkResourcePool[]
{ ECRoom }
;
  }
  public void _ECRoom_onNewUnit_xjal( ResourceUnit unit ) {

unit.getNetworkHomeLocation().setVisible( true );
unit.getNetworkHomeLocation().setLineColor( null );
;
  }
  public void _ECRoom_onSeize_xjal( ResourceUnit unit, Entity entity ) {

unit.getNetworkHomeLocation().setFillColor( khaki );
;
  }
  public void _ECRoom_onRelease_xjal( ResourceUnit unit, Entity entity )
{

unit.getNetworkHomeLocation().setFillColor( null );
;
  }
  public Shape _ECRoom_idleUnitShape_xjal( ResourceUnit unit ) {
    return
shapeIdle
;
  }
  public Shape _ECRoom_busyUnitShape_xjal( ResourceUnit unit ) {
    return
shapeBusy
;
  }
  public NetworkResourcePool _gotoECRoom_destinationResource_xjal(
) {
    return
ECRoom
;
  }
  public NetworkResourcePool[] _releaseECRoom_resources_xjal( Entity
entity ) {
    return new NetworkResourcePool[]
{ ECRoom }
;
  }
  public Shape _PA_idleUnitShape_xjal( ResourceUnit unit ) {
    return
shapePA
;
  }
  public Shape _technician_idleUnitShape_xjal( ResourceUnit unit ) {
    return
shapeTech
;
  }
  public Shape _XRay_idleUnitShape_xjal( ResourceUnit unit ) {
    return
shapeIdle
;
  }
```

```
  public Shape _XRay_busyUnitShape_xjal( ResourceUnit unit ) {
    return
shapeBusy
;
  }
  public NetworkResourcePool[] _releasePA_resources_xjal( Entity entity
) {
    return new NetworkResourcePool[]
{ PA }
;
  }
  public Shape _USound_idleUnitShape_xjal( ResourceUnit unit ) {
    return
shapeUSound
;
  }
  public NetworkResourcePool[] _callNurse_resources_xjal( Entity entity )
{
    return new NetworkResourcePool[]
{ nurse }
;
  }
  public NetworkResourcePool[] _callPA_resources_xjal( Entity entity ) {
    return new NetworkResourcePool[]
{ PA }
;
  }
  // Functions


void resetStats( ) {

nurse.resetStats();
PA.resetStats();
technician.resetStats();
USound.resetStats();
XRay.resetStats();
triageRoom.resetStats();
ECRoom.resetStats();
histLOS.reset();
  }
  // Analysis Data Elements
  public HistogramSmartData histLOS = new HistogramSmartData( 20,
0.1
, false, false, 0.1, 0.1 );


  // View areas
  public ViewArea ModelLogic = new ViewArea( this, "ModelLogic", 0,
1000, ViewArea.TOP_LEFT, ViewArea.NONE, 1.0, 100, 100 );
  public ViewArea AnimationAndOutput = new ViewArea( this,
"AnimationAndOutput", 0, 0, ViewArea.TOP_LEFT, ViewArea.NONE, 1.0,
100, 100 );


  static final Font _radio_Font = new Font("Dialog", 0, 11 );
  static final Font _radio1_Font = _radio_Font;
  static final Font _radio2_Font = _radio_Font;
  static final Font _radio3_Font = _radio_Font;
  static final Font _text_Font = new Font("SansSerif", 0, 10 );
  static final Font _text1_Font = _text_Font;
  static final Font _text2_Font = _text_Font;
  static final Font _text7_Font = _text_Font;
  static final Font _text8_Font = _text_Font;
  static final Font _text9_Font = _text_Font;
  static final Font _text15_Font = new Font("SansSerif", 0, 18 );
  static final Font _text16_Font = _text15_Font;
  static final Font _text17_Font = _text15_Font;
  static final Font _text18_Font = _text15_Font;
  static final Font _text19_Font = new Font("SansSerif", 1, 12 );
  static final Font _text20_Font = _text19_Font;
  static final Font _text21_Font = new Font("SansSerif", 0, 12 );
  static final Font _text22_Font = _text21_Font;
  static final Font _text24_Font = _text19_Font;
  static final Font _text25_Font = _text19_Font;
  static final Font _text26_Font = _text21_Font;
  static final Font _text27_Font = _text21_Font;
  static final Font _text28_Font = _text21_Font;
  static final Font _text29_Font = _text21_Font;
  static final Font _text30_Font = _text_Font;
  static final Font _text31_Font = _text_Font;
  static final Font _text32_Font = _text19_Font;
  static final Font _text6_Font = _text_Font;
  static final Font _text10_Font = _text_Font;
  static final Font _text11_Font = _text_Font;
  static final Font _text12_Font = _text_Font;
  static final Font _text14_Font = _text_Font;
  static final Font _text3_Font = _text_Font;
  static final Font _text5_Font = _text_Font;
  static final Font _text13_Font = new Font("SansSerif", 1, 24 );
  static final Font _text23_Font = _text13_Font;
  static final Color _rectXRayFloor_FillColor = new Color( 0xFF7F1DEE,
true );
  static final Color _arc_FillColor = yellow;
  static final Color _arc2_FillColor = yellow;
  static final Color _arc1_FillColor = yellow;
  static final Color _shapeNurse_FillColor = new Color( 0xFFE5F6FA, true
);
  static final Color _shapePA_FillColor = new Color( 0xFFC4FE87, true );
  static final Color _text19_Color = new Color( 0xFF454545, true );
  static final Color _text20_Color = new Color( 0xFF454545, true );
  static final Color _text24_Color = new Color( 0xFF454545, true );
  static final Color _text25_Color = new Color( 0xFF454545, true );
  static final Color _text32_Color = new Color( 0xFF454545, true );
  static final Color _oval6_LineColor = new Color( 0xFF7F1DEE, true );
  static final int _radio = 1;
  static final int _radio1 = 2;
  static final int _radio2 = 3;
  static final int _radio3 = 4;
  static final int _rectXRayFloor = 5;
  static final int _arc = 6;
  static final int _arc2 = 7;
  static final int _arc1 = 8;
  static final int _oval6 = 9;
  static final int _group1 = 10;
  static final int _rectangle41 = 11;
  static final int _rectangle38 = 12;
  static final int _rectangle37 = 13;
  static final int _rectangle36 = 14;
  static final int _rectangle35 = 15;
  static final int _rectangle33 = 16;
  static final int _polyline18 = 17;
  static final int _polyline4 = 18;
  static final int _polyline5 = 19;
  static final int _polyline6 = 20;
  static final int _line11 = 21;
  static final int _line12 = 22;
  static final int _line14 = 23;
  static final int _line15 = 24;
  static final int _line16 = 25;
  static final int _line17 = 26;
  static final int _line18 = 27;
  static final int _line19 = 28;
  static final int _line20 = 29;
  static final int _line21 = 30;
```

```
static final int _line22 = 31;
static final int _line23 = 32;
static final int _line24 = 33;
static final int _line26 = 34;
static final int _polyline7 = 35;
static final int _polyline8 = 36;
static final int _polyline9 = 37;
static final int _line27 = 38;
static final int _line29 = 39;
static final int _line30 = 40;
static final int _line31 = 41;
static final int _polyline10 = 42;
static final int _polyline11 = 43;
static final int _rectangle25 = 44;
static final int _line33 = 45;
static final int _line34 = 46;
static final int _rectangle26 = 47;
static final int _text = 48;
static final int _text1 = 49;
static final int _text2 = 50;
static final int _text7 = 51;
static final int _text8 = 52;
static final int _text9 = 53;
static final int _line35 = 54;
static final int _polyline12 = 55;
static final int _polyline13 = 56;
static final int _line36 = 57;
static final int _polyline14 = 58;
static final int _polyline15 = 59;
static final int _line37 = 60;
static final int _line38 = 61;
static final int _polyQueueAtReg = 62;
static final int _polyTriageRooms = 63;
static final int _polyline16 = 64;
static final int _polyECRooms = 65;
static final int _shapePatient = 66;
static final int _shapeNurse = 67;
static final int _shapePA = 68;
static final int _shapeTech = 69;
static final int _polyline17 = 70;
static final int _line41 = 71;
static final int _line42 = 72;
static final int _curve = 73;
static final int _line43 = 74;
static final int _line44 = 75;
static final int _line45 = 76;
static final int _curve1 = 77;
static final int _curve2 = 78;
static final int _curve3 = 79;
static final int _curve4 = 80;
static final int _curve5 = 81;
static final int _curve6 = 82;
static final int _curve7 = 83;
static final int _NurseHomePath = 84;
static final int _PAHomePath = 85;
static final int _TechHomePath = 86;
static final int _line46 = 87;
static final int _line49 = 88;
static final int _line48 = 89;
static final int _rectangle31 = 90;
static final int _rectangle30 = 91;
static final int _waitingRoom = 92;
static final int _rectangle3 = 93;
static final int _rectangle4 = 94;
static final int _rectangle6 = 95;
static final int _rectangle7 = 96;
static final int _rectangle8 = 97;
static final int _rectangle9 = 98;
static final int _rectangle10 = 99;
static final int _xRayRoom = 100;
static final int _USoundStorage = 101;
static final int _techHomeRoom = 102;
static final int _nurseHomeRoom = 103;
static final int _doctorHomeRoom = 104;
static final int _entryDoor = 105;
static final int _exitDoor = 106;
static final int _rectangle11 = 107;
static final int _rectangle12 = 108;
static final int _rectangle13 = 109;
static final int _rectangle15 = 110;
static final int _rectangle16 = 111;
static final int _rectangle17 = 112;
static final int _rectangle18 = 113;
static final int _rectangle19 = 114;
static final int _rectangle20 = 115;
static final int _rectangle21 = 116;
static final int _rectangle22 = 117;
static final int _rectangle23 = 118;
static final int _rectangle24 = 119;
static final int _polyline = 120;
static final int _line = 121;
static final int _line1 = 122;
static final int _line4 = 123;
static final int _line5 = 124;
static final int _polyline1 = 125;
static final int _line6 = 126;
static final int _line7 = 127;
static final int _line8 = 128;
static final int _line9 = 129;
static final int _polyline2 = 130;
static final int _polyline3 = 131;
static final int _line10 = 132;
static final int _line32 = 133;
static final int _rectangle27 = 134;
static final int _line39 = 135;
static final int _rectangle29 = 136;
static final int _line40 = 137;
static final int _registrationDesk = 138;
static final int _rectangle = 139;
static final int _line47 = 140;
static final int _group = 141;
static final int _oval1 = 142;
static final int _oval = 143;
static final int _shapeBusy = 144;
static final int _oval2 = 145;
static final int _oval3 = 146;
static final int _shapeIdle = 147;
static final int _XRayHomePath = 148;
static final int _USoundHomePath = 149;
static final int _line50 = 150;
static final int _oval5 = 151;
static final int _rectangle32 = 152;
static final int _polyline19 = 153;
static final int _oval4 = 154;
static final int _shapeUSound = 155;
static final int _text15 = 156;
static final int _text16 = 157;
static final int _line51 = 158;
static final int _line52 = 159;
static final int _text17 = 160;
```

```
  static final int _rectangle34 = 161;
  static final int _text18 = 162;
  static final int _line53 = 163;
  static final int _line54 = 164;
  static final int _text19 = 165;
  static final int _text20 = 166;
  static final int _polyline20 = 167;
  static final int _polyline21 = 168;
  static final int _text21 = 169;
  static final int _text22 = 170;
  static final int _line55 = 171;
  static final int _line56 = 172;
  static final int _rectangle39 = 173;
  static final int _text24 = 174;
  static final int _polyline22 = 175;
  static final int _rectangle40 = 176;
  static final int _text25 = 177;
  static final int _polyline23 = 178;
  static final int _text26 = 179;
  static final int _text27 = 180;
  static final int _text28 = 181;
  static final int _text29 = 182;
  static final int _text30 = 183;
  static final int _text31 = 184;
  static final int _rectangle42 = 185;
  static final int _rectangle43 = 186;
  static final int _text32 = 187;
  static final int _polyline24 = 188;
  static final int _text6 = 189;
  static final int _text10 = 190;
  static final int _text11 = 191;
  static final int _text12 = 192;
  static final int _text14 = 193;
  static final int _text3 = 194;
  static final int _text5 = 195;
  static final int _text13 = 196;
  static final int _text23 = 197;
  static final int _chart = 198;
  static final int _chart1 = 199;

  /**
   * Top-level presentation group id
   */
  static final int _presentation = 0;

  /**
   * Top-level icon group id
   */
  static final int _icon = -1;

  static final double[] _polyline18_pointsDX_xjal() {
    return new double[] { 0, 40, 40, 150, 150, 250, 250, 290, 290, 350,
350, 495, 495, 390, 390, 395, 380, 375, 40, 40, 145, 145, 0,  };
  }
  static final double[] _polyline18_pointsDY_xjal() {
    return new double[] { 0, 0, 200, 200, 150, 150, 50, 50, 100, 100, 0,
0, 40, 40, 225, 230, 245, 240, 240, 280, 280, 340, 340,  };
  }
  static final double[] _polyline4_pointsDX_xjal() {
    return new double[] { 0, 0, -150, -150, 460,  };
  }
  static final double[] _polyline4_pointsDY_xjal() {
    return new double[] { 0, -10, -10, -360, -360,  };
  }
  static final double[] _polyline5_pointsDX_xjal() {
    return new double[] { 0, -10, -10, -40, -400, -400,  };
  }
  static final double[] _polyline5_pointsDY_xjal() {
    return new double[] { 0, 0, 270, 300, 300, 310,  };
  }
  static final double[] _polyline6_pointsDX_xjal() {
    return new double[] { 0, 0, -20,  };
  }
  static final double[] _polyline6_pointsDY_xjal() {
    return new double[] { 0, 200, 200,  };
  }
  static final double[] _polyline7_pointsDX_xjal() {
    return new double[] { 0, 0, 10,  };
  }
  static final double[] _polyline7_pointsDY_xjal() {
    return new double[] { 0, 100, 100,  };
  }
  static final double[] _polyline8_pointsDX_xjal() {
    return new double[] { 0, 0, -10,  };
  }
  static final double[] _polyline8_pointsDY_xjal() {
    return new double[] { 0, 100, 100,  };
  }
  static final double[] _polyline9_pointsDX_xjal() {
    return new double[] { 0, 0, 140, 140,  };
  }
  static final double[] _polyline9_pointsDY_xjal() {
    return new double[] { 0, -30, -30, 70,  };
  }
  static final double[] _polyline10_pointsDX_xjal() {
    return new double[] { 0, 0, -20,  };
  }
  static final double[] _polyline10_pointsDY_xjal() {
    return new double[] { 0, 100, 100,  };
  }
  static final double[] _polyline11_pointsDX_xjal() {
    return new double[] { 0, -100, -100,  };
  }
  static final double[] _polyline11_pointsDY_xjal() {
    return new double[] { 0, 0, 60,  };
  }
  static final double[] _polyline12_pointsDX_xjal() {
    return new double[] { 0, 5, 10,  };
  }
  static final double[] _polyline12_pointsDY_xjal() {
    return new double[] { 0, -10, 0,  };
  }
  static final double[] _polyline13_pointsDX_xjal() {
    return new double[] { 0, 10, 0,  };
  }
  static final double[] _polyline13_pointsDY_xjal() {
    return new double[] { 0, 5, 10,  };
  }
  static final double[] _polyline14_pointsDX_xjal() {
    return new double[] { 0, -90, -100, -100,  };
  }
  static final double[] _polyline14_pointsDY_xjal() {
    return new double[] { 0, 0, -10, -50,  };
  }
  static final double[] _polyline15_pointsDX_xjal() {
    return new double[] { 0, 0, -10, -50,  };
  }
  static final double[] _polyline15_pointsDY_xjal() {
    return new double[] { 0, -90, -100, -100,  };
  }
  static final double[] _polyQueueAtReg_pointsDX_xjal() {
```

```
  return new double[] { 0, 30, 30, 0, 0, -10,  };
 }
 static final double[] _polyQueueAtReg_pointsDY_xjal() {
  return new double[] { 0, 0, -30, -30, -50, -50,  };
 }
 static final double[] _polyTriageRooms_pointsDX_xjal() {
  return new double[] { 0, 0,  };
 }
 static final double[] _polyTriageRooms_pointsDY_xjal() {
  return new double[] { 0, 50,  };
 }
 static final double[] _polyline16_pointsDX_xjal() {
  return new double[] { 0, 0, 40, 40,  };
 }
 static final double[] _polyline16_pointsDY_xjal() {
  return new double[] { 0, -10, -10, 0,  };
 }
 static final double[] _polyECRooms_pointsDX_xjal() {
  return new double[] { 0, 90, 210, 210, 210,  };
 }
 static final double[] _polyECRooms_pointsDY_xjal() {
  return new double[] { 0, 0, -10, -100, -200,  };
 }
 static final double[] _shapePatient_pointsDX_xjal() {
  return new double[] { 0, -1, 2, 5, 4, 3, 6, 3, 4, 0, 1, -2, 1,  };
 }
 static final double[] _shapePatient_pointsDY_xjal() {
  return new double[] { 0, 4, 1, 4, 0, -3, -3, -5, -7, -7, -5, -3, -3,  };
 }
 static final double[] _shapeNurse_pointsDX_xjal() {
  return new double[] { 0, 2, 1, 3, 3, 5, 5, 7, 6, 8, 6, 8, 5, 6, 2, 3, 0, 2,
};
 }
 static final double[] _shapeNurse_pointsDY_xjal() {
  return new double[] { 0, 0, 2, 2, 0, 0, 2, 2, 0, 0, -5, -5, -7, -10, -10, -
7, -5, -5,  };
 }
 static final double[] _shapePA_pointsDX_xjal() {
  return new double[] { 0, 2, 1, 3, 3, 5, 5, 7, 6, 8, 6, 8, 5, 6, 2, 3, 0, 2,
};
 }
 static final double[] _shapePA_pointsDY_xjal() {
  return new double[] { 0, 0, 2, 2, 0, 0, 2, 2, 0, 0, -5, -5, -7, -10, -10, -
7, -5, -5,  };
 }
 static final double[] _shapeTech_pointsDX_xjal() {
  return new double[] { 0, 1, 0, 1, 2, 3, 4, 3, 4, 4, 6, 6, 3, 3, 1, 1, -2, -
2, 0,  };
 }
 static final double[] _shapeTech_pointsDY_xjal() {
  return new double[] { 0, 0, 5, 5, 3, 5, 5, 0, 0, -3, -3, -4, -4, -7, -7, -4,
-4, -3, -3,  };
 }
 static final double[] _polyline17_pointsDX_xjal() {
  return new double[] { 0, 0, 20,  };
 }
 static final double[] _polyline17_pointsDY_xjal() {
  return new double[] { 0, 110, 110,  };
 }
 static final double[] _curve_pointsDX_xjal() {
  return new double[] { 0, -3, -8, -12, -8, -3, 0,  };
 }
 static final double[] _curve_pointsDY_xjal() {
  return new double[] { 0, 0, -1, 2, 5, 4, 4,  };
 }
 static final double[] _curve1_pointsDX_xjal() {
  return new double[] { 0, -3, -8, -12, -8, -3, 0,  };
 }
 static final double[] _curve1_pointsDY_xjal() {
  return new double[] { 0, 0, -1, 2, 5, 4, 4,  };
 }
 static final double[] _curve2_pointsDX_xjal() {
  return new double[] { 0, -3, -8, -12, -8, -3, 0,  };
 }
 static final double[] _curve2_pointsDY_xjal() {
  return new double[] { 0, 0, -1, 2, 5, 4, 4,  };
 }
 static final double[] _curve3_pointsDX_xjal() {
  return new double[] { 0, -3, -8, -12, -8, -3, 0,  };
 }
 static final double[] _curve3_pointsDY_xjal() {
  return new double[] { 0, 0, -1, 2, 5, 4, 4,  };
 }
 static final double[] _curve4_pointsDX_xjal() {
  return new double[] { 0, 3, 8, 12, 8, 3, 0,  };
 }
 static final double[] _curve4_pointsDY_xjal() {
  return new double[] { 0, 0, 1, -2, -5, -4, -4,  };
 }
 static final double[] _curve5_pointsDX_xjal() {
  return new double[] { 0, 3, 8, 12, 8, 3, 0,  };
 }
 static final double[] _curve5_pointsDY_xjal() {
  return new double[] { 0, 0, 1, -2, -5, -4, -4,  };
 }
 static final double[] _curve6_pointsDX_xjal() {
  return new double[] { 0, 3, 8, 12, 8, 3, 0,  };
 }
 static final double[] _curve6_pointsDY_xjal() {
  return new double[] { 0, 0, 1, -2, -5, -4, -4,  };
 }
 static final double[] _curve7_pointsDX_xjal() {
  return new double[] { 0, 3, 8, 12, 8, 3, 0,  };
 }
 static final double[] _curve7_pointsDY_xjal() {
  return new double[] { 0, 0, 1, -2, -5, -4, -4,  };
 }
 static final double[] _NurseHomePath_pointsDX_xjal() {
  return new double[] { 0, 15, 30, 45, 60, 60, 45, 30, 15, 0,  };
 }
 static final double[] _NurseHomePath_pointsDY_xjal() {
  return new double[] { 0, 0, 0, 0, 0, 20, 20, 20, 20, 20,  };
 }
 static final double[] _PAHomePath_pointsDX_xjal() {
  return new double[] { 0, 15, 30, 45, 60, 60, 45, 30, 15, 0,  };
 }
 static final double[] _PAHomePath_pointsDY_xjal() {
  return new double[] { 0, 0, 0, 0, 0, 20, 20, 20, 20, 20,  };
 }
 static final double[] _TechHomePath_pointsDX_xjal() {
  return new double[] { 0, 15, 30, 45, 60, 76, 91, 105,  };
 }
 static final double[] _TechHomePath_pointsDY_xjal() {
  return new double[] { 0, 0, 0, 0, 0, 0, 0, 0,  };
 }
 static final double[] _polyline_pointsDX_xjal() {
  return new double[] { 0, 0, -40, -40, -40, -40, -40, -40, -90,  };
 }
 static final double[] _polyline_pointsDY_xjal() {
  return new double[] { 0, -30, -30, -120, -170, -220, -270, -320, -320,
};
 }
 static final double[] _polyline1_pointsDX_xjal() {
```

```
  return new double[] { 0, 200, 300, 330, 340, 350, 350, 350, 350, 380,
470,  };
 }
 static final double[] _polyline1_pointsDY_xjal() {
  return new double[] { 0, 0, 0, 0, -10, -20, -50, -110, -180, -210, -210,
};
 }
 static final double[] _polyline2_pointsDX_xjal() {
  return new double[] { 0, 0, 20, 40, 40, 40, 40,  };
 }
 static final double[] _polyline2_pointsDY_xjal() {
  return new double[] { 0, -60, -60, -80, -110, -160, -200,  };
 }
 static final double[] _polyline3_pointsDX_xjal() {
  return new double[] { 0, 50, 50,  };
 }
 static final double[] _polyline3_pointsDY_xjal() {
  return new double[] { 0, 0, -40,  };
 }
 static final double[] _XRayHomePath_pointsDX_xjal() {
  return new double[] { 0, -10,  };
 }
 static final double[] _XRayHomePath_pointsDY_xjal() {
  return new double[] { 0, 10,  };
 }
 static final double[] _USoundHomePath_pointsDX_xjal() {
  return new double[] { 0, 10, 0, 10, 0, 10, 0,  };
 }
 static final double[] _USoundHomePath_pointsDY_xjal() {
  return new double[] { 0, 10, 20, 30, 40, 50, 60,  };
 }
 static final double[] _polyline19_pointsDX_xjal() {
  return new double[] { 0, 3, 2,  };
 }
 static final double[] _polyline19_pointsDY_xjal() {
  return new double[] { 0, 1, 4,  };
 }
 static final double[] _polyline20_pointsDX_xjal() {
  return new double[] { 0, 0, 150, 150,  };
 }
 static final double[] _polyline20_pointsDY_xjal() {
  return new double[] { 0, -20, -20, 0,  };
 }
 static final double[] _polyline21_pointsDX_xjal() {
  return new double[] { 0, 0, 690, 690,  };
 }
 static final double[] _polyline21_pointsDY_xjal() {
  return new double[] { 0, -20, -20, 0,  };
 }
 static final double[] _polyline22_pointsDX_xjal() {
  return new double[] { 0, 0, 630, 630,  };
 }
 static final double[] _polyline22_pointsDY_xjal() {
  return new double[] { 0, -20, -20, 0,  };
 }
 static final double[] _polyline23_pointsDX_xjal() {
  return new double[] { 0, 0, 200, 200,  };
 }
 static final double[] _polyline23_pointsDY_xjal() {
  return new double[] { 0, -20, -20, 0,  };
 }
 static final double[] _polyline24_pointsDX_xjal() {
  return new double[] { 0, 0, 210, 210,  };
 }
 static final double[] _polyline24_pointsDY_xjal() {
  return new double[] { 0, -20, -20, 0,  };
 }

 @Override
 public boolean onShapeClick( int _shape, int index, double clickx,
double clicky ){
   switch( _shape ){
     case _rectangle41:
       if (true) {

getEngine().setRealTimeMode( false );
       }
       break;
     case _rectangle38:
       if (true) {

getPresentation().setPresentable( xRayProcess );
       }
       break;
     case _rectangle37:
       if (true) {

getPresentation().setPresentable( uSoundProcess );
       }
       break;
     case _rectangle33:
       if (true) {

ModelLogic.navigateTo();
       }
       break;
     case _rectangle34:
       if (true) {

AnimationAndOutput.navigateTo();
       }
       break;
     case _rectangle42:
       if (true) {

getEngine().setRealTimeMode( true );
       }
       break;
     default: return super.onShapeClick( _shape, index, clickx, clicky );
               }
               return false;
 }

 @Override
 public void executeShapeControlAction( int _shape, int index, int value )
{
   switch( _shape ) {
     case _radio: {
nurse.set_capacity( value + 1 );
resetStats();
;}
       break;
     case _radio1: {
PA.set_capacity( value + 1 );
resetStats();
;}
       break;
     case _radio2: {
technician.set_capacity( value + 1 );
resetStats();
;}
       break;
```

```
    case _radio3: {
USound.set_capacity( value + 1 );
resetStats();
;}
      break;
    default:
     super.executeShapeControlAction( _shape, index, value );
     break;
   }
  }

  @Override
  public int getShapeControlDefaultValueInt( int _shape, int index ) {
   switch(_shape) {
    case _radio: return
nurse.capacity - 1
;
    case _radio1: return
PA.capacity - 1
;
    case _radio2: return
technician.capacity - 1
;
    case _radio3: return
USound.capacity - 1
;
    default: return super.getShapeControlDefaultValueInt( _shape, index
);
   }
  }


  /**
   * <i>This method should not be called by user</i>
   */
  private double _chart_DataItem0Value() {
   return
triageRoom.statsUtilization.mean()
;
  }

  /**
   * <i>This method should not be called by user</i>
   */
  private double _chart_DataItem1Value() {
   return
ECRoom.statsUtilization.mean()
;
  }

  /**
   * <i>This method should not be called by user</i>
   */
  private double _chart_DataItem2Value() {
   return
XRay.statsUtilization.mean()
;
  }

  /**
   * <i>This method should not be called by user</i>
   */
  private double _chart_DataItem3Value() {
   return
USound.statsUtilization.mean()
;
  }

  /**
   * <i>This method should not be called by user</i>
   */
  private double _chart_DataItem4Value() {
   return
nurse.statsUtilization.mean()
;
  }

  /**
   * <i>This method should not be called by user</i>
   */
  private double _chart_DataItem5Value() {
   return
PA.statsUtilization.mean()
;
  }

  /**
   * <i>This method should not be called by user</i>
   */
  private double _chart_DataItem6Value() {
   return
technician.statsUtilization.mean()
;
  }
  ShapeRadioButtonGroup radio;
  ShapeRadioButtonGroup radio1;
  ShapeRadioButtonGroup radio2;
  ShapeRadioButtonGroup radio3;
  BarChart chart;
  Histogram chart1;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _rectXRayFloor_SetDynamicParams( ShapeRectangle shape
) {
   boolean _visible;
   _visible =
xRayProcess.doXRay.size() > 0
;
   shape.setVisible( _visible );
        if ( _visible ) {
        }
  }

  ShapeRectangle rectXRayFloor;
  ShapeArc arc;
  ShapeArc arc2;
  ShapeArc arc1;
  ShapeOval oval6;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _group1_SetDynamicParams( ShapeGroup shape ) {
   boolean _visible;
   _visible =
xRayProcess.doXRay.size() > 0
;
   shape.setVisible( _visible );
        if ( _visible ) {
```

```
            }
        }

        ShapeGroup group1;
        ShapeRectangle rectangle41;
        ShapeRectangle rectangle38;
        ShapeRectangle rectangle37;
        ShapeRectangle rectangle36;
        ShapeRectangle rectangle35;
        ShapeRectangle rectangle33;
        ShapePolyLine polyline18;
        ShapePolyLine polyline4;
        ShapePolyLine polyline5;
        ShapePolyLine polyline6;
        ShapeLine line11;
        ShapeLine line12;
        ShapeLine line14;
        ShapeLine line15;
        ShapeLine line16;
        ShapeLine line17;
        ShapeLine line18;
        ShapeLine line19;
        ShapeLine line20;
        ShapeLine line21;
        ShapeLine line22;
        ShapeLine line23;
        ShapeLine line24;
        ShapeLine line26;
        ShapePolyLine polyline7;
        ShapePolyLine polyline8;
        ShapePolyLine polyline9;
        ShapeLine line27;
        ShapeLine line29;
        ShapeLine line30;
        ShapeLine line31;
        ShapePolyLine polyline10;
        ShapePolyLine polyline11;
        ShapeRectangle rectangle25;
        ShapeLine line33;
        ShapeLine line34;
        ShapeRectangle rectangle26;
        ShapeText text;
        ShapeText text1;
        ShapeText text2;
        ShapeText text7;
        ShapeText text8;
        ShapeText text9;
        ShapeLine line35;
        ShapePolyLine polyline12;
        ShapePolyLine polyline13;
        ShapeLine line36;
        ShapePolyLine polyline14;
        ShapePolyLine polyline15;
        ShapeLine line37;
        ShapeLine line38;

        /**
         * <i>This method should not be called by user</i>
         */
        private void _polyQueueAtReg_SetDynamicParams( ShapePolyLine
shape ) {
            boolean _visible;
            _visible =
false
;
            shape.setVisible( _visible );
            if ( _visible ) {
            }
        }

        ShapePolyLine polyQueueAtReg;
        ShapePolyLine polyTriageRooms;
        ShapePolyLine polyline16;
        ShapePolyLine polyECRooms;
        ShapeCurve shapePatient;
        ShapePolyLine shapeNurse;
        ShapePolyLine shapePA;
        ShapePolyLine shapeTech;
        ShapePolyLine polyline17;
        ShapeLine line41;
        ShapeLine line42;
        ShapeCurve curve;
        ShapeLine line43;
        ShapeLine line44;
        ShapeLine line45;
        ShapeCurve curve1;
        ShapeCurve curve2;
        ShapeCurve curve3;
        ShapeCurve curve4;
        ShapeCurve curve5;
        ShapeCurve curve6;
        ShapeCurve curve7;
        ShapePolyLine NurseHomePath;
        ShapePolyLine PAHomePath;
        ShapePolyLine TechHomePath;
        ShapeLine line46;
        ShapeLine line49;
        ShapeLine line48;
        ShapeRectangle rectangle31;
        ShapeRectangle rectangle30;
        ShapeRectangle waitingRoom;
        ShapeRectangle rectangle3;
        ShapeRectangle rectangle4;
        ShapeRectangle rectangle6;
        ShapeRectangle rectangle7;
        ShapeRectangle rectangle8;
        ShapeRectangle rectangle9;
        ShapeRectangle rectangle10;
        ShapeRectangle xRayRoom;
        ShapeRectangle USoundStorage;
        ShapeRectangle techHomeRoom;
        ShapeRectangle nurseHomeRoom;
        ShapeRectangle doctorHomeRoom;
        ShapeRectangle entryDoor;
        ShapeRectangle exitDoor;
        ShapeRectangle rectangle11;
        ShapeRectangle rectangle12;
        ShapeRectangle rectangle13;
        ShapeRectangle rectangle15;
        ShapeRectangle rectangle16;
        ShapeRectangle rectangle17;
        ShapeRectangle rectangle18;
        ShapeRectangle rectangle19;
        ShapeRectangle rectangle20;
        ShapeRectangle rectangle21;
        ShapeRectangle rectangle22;
        ShapeRectangle rectangle23;
        ShapeRectangle rectangle24;
        ShapePolyLine polyline;
        ShapeLine line;
        ShapeLine line1;
```

```
  ShapeLine line4;
  ShapeLine line5;
  ShapePolyLine polyline1;
  ShapeLine line6;
  ShapeLine line7;
  ShapeLine line8;
  ShapeLine line9;
  ShapePolyLine polyline2;
  ShapePolyLine polyline3;
  ShapeLine line10;
  ShapeLine line32;
  ShapeRectangle rectangle27;
  ShapeLine line39;
  ShapeRectangle rectangle29;
  ShapeLine line40;
  ShapeRectangle registrationDesk;
  ShapeRectangle rectangle;
  ShapeLine line47;
  ShapeGroup group;
  ShapeOval oval1;
  ShapeOval oval;
  ShapeGroup shapeBusy;
  ShapeOval oval2;
  ShapeOval oval3;
  ShapeGroup shapeIdle;
  ShapePolyLine XRayHomePath;
  ShapePolyLine USoundHomePath;
  ShapeLine line50;
  ShapeOval oval5;
  ShapeRectangle rectangle32;
  ShapePolyLine polyline19;
  ShapeOval oval4;
  ShapeGroup shapeUSound;
  ShapeText text15;
  ShapeText text16;
  ShapeLine line51;
  ShapeLine line52;
  ShapeText text17;
  ShapeRectangle rectangle34;
  ShapeText text18;
  ShapeLine line53;
  ShapeLine line54;
  ShapeText text19;
  ShapeText text20;
  ShapePolyLine polyline20;
  ShapePolyLine polyline21;
  ShapeText text21;
  ShapeText text22;
  ShapeLine line55;
  ShapeLine line56;
  ShapeRectangle rectangle39;
  ShapeText text24;
  ShapePolyLine polyline22;
  ShapeRectangle rectangle40;
  ShapeText text25;
  ShapePolyLine polyline23;
  ShapeText text26;
  ShapeText text27;
  ShapeText text28;
  ShapeText text29;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _text30_SetDynamicParams( ShapeText shape ) {
    boolean _visible;
    shape.setColor(
getEngine().getRealTimeMode() ? black : lightGrey
);
  }

  ShapeText text30;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _text31_SetDynamicParams( ShapeText shape ) {
    boolean _visible;
    shape.setColor(
getEngine().getRealTimeMode() ? lightGrey : black
);
  }

  ShapeText text31;
  ShapeRectangle rectangle42;
  ShapeRectangle rectangle43;
  ShapeText text32;
  ShapePolyLine polyline24;
  ShapeText text6;
  ShapeText text10;
  ShapeText text11;
  ShapeText text12;
  ShapeText text14;
  ShapeText text3;
  ShapeText text5;
  ShapeText text13;
  ShapeText text23;

  // Static initialization of persistent elements
  {
    radio = new ShapeRadioButtonGroup(
                                  Main.this, true,25, 210,
                                        190, 20,
          transparent, black,
          _radio_Font, false,
          new String[]{"1", "2", "3", "4", "5", "6", }
    ) {

      @Override
      public void action(){
        executeShapeControlAction( _radio, 0, value );
      }

      @Override
      public void setValueToDefault() {
                              setValue(     getShapeControlDefaultValueInt(
_radio, 0 ) );
      }
    };
    radio1 = new ShapeRadioButtonGroup(
                                  Main.this, true,25, 270,
                                        190, 20,
          transparent, black,
          _radio1_Font, false,
          new String[]{"1", "2", "3", "4", "5", "6", }
    ) {

      @Override
      public void action(){
        executeShapeControlAction( _radio1, 0, value );
      }
```

```
    @Override
    public void setValueToDefault() {
                        setValue(    getShapeControlDefaultValueInt(
_radio1, 0 ) );
    }
  };
  radio2 = new ShapeRadioButtonGroup(
                          Main.this, true,25, 330,
                                  190, 20,
      transparent, black,
      _radio2_Font, false,
      new String[]{"1", "2", "3", "4", "5", "6", }
  ) {

    @Override
    public void action(){
     executeShapeControlAction( _radio2, 0, value );
    }

    @Override
    public void setValueToDefault() {
                        setValue(    getShapeControlDefaultValueInt(
_radio2, 0 ) );
    }
  };
  radio3 = new ShapeRadioButtonGroup(
                          Main.this, true,25, 390,
                                  190, 20,
      transparent, black,
      _radio3_Font, false,
      new String[]{"1", "2", "3", "4", "5", "6", }
  ) {

    @Override
    public void action(){
     executeShapeControlAction( _radio3, 0, value );
    }

    @Override
    public void setValueToDefault() {
                        setValue(    getShapeControlDefaultValueInt(
_radio3, 0 ) );
    }
  };
  rectXRayFloor = new ShapeRectangle(
    true,0, 0, 0.0,
      null, _rectXRayFloor_FillColor,
                                  90, 90,
                                  1, LINE_STYLE_SOLID
  ) {

    @Override
    public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
      _rectXRayFloor_SetDynamicParams( this );
     super.draw( panel, graphics, xform, publicOnly );
    }
  };
  arc = new ShapeArc(
   true,25, 30, 0.0,
      null, _arc_FillColor,
                                  10, 10,
                                  0.5235987755982988,
1.0471975511965976,
                                  1, LINE_STYLE_SOLID
  );
  arc2 = new ShapeArc(
   true,25, 30, 0.0,
      null, _arc2_FillColor,
                                  10, 10,
                                  -2.6179938779914944,    -
1.0471975511965976,
                                  1, LINE_STYLE_SOLID
  );
  arc1 = new ShapeArc(
   true,25, 30, 0.0,
      null, _arc1_FillColor,
                                  10, 10,
                                  -0.5235987755982988,    -
1.0471975511965976,
                                  1, LINE_STYLE_SOLID
  );
  oval6 = new ShapeOval(
    true,25, 30, 0.0,
    _oval6_LineColor, yellow,
              3, 3,
              2, LINE_STYLE_SOLID
  );
  rectangle41 = new ShapeRectangle(
    true,180, 60, 0.0,
      steelBlue, null,
                                  40, 20,
                                  1, LINE_STYLE_SOLID
  ) {

    @Override
    public boolean onClick( double clickx, double clicky ) {
     return onShapeClick( _rectangle41, 0, clickx, clicky );
    }
  };
  rectangle38 = new ShapeRectangle(
    true,305, 1495, 0.0,
      null, null,
                                  90, 25,
                                  1, LINE_STYLE_SOLID
  ) {

    @Override
    public boolean onClick( double clickx, double clicky ) {
     return onShapeClick( _rectangle38, 0, clickx, clicky );
    }
  };
  rectangle37 = new ShapeRectangle(
    true,290, 1400, 0.0,
      null, null,
                                  125, 25,
                                  1, LINE_STYLE_SOLID
  ) {

    @Override
    public boolean onClick( double clickx, double clicky ) {
     return onShapeClick( _rectangle37, 0, clickx, clicky );
    }
  };
  rectangle36 = new ShapeRectangle(
    true,190, 1070, 0.0,
      null, ghostWhite,
                                  690, 10,
                                  1, LINE_STYLE_SOLID
  );
  rectangle35 = new ShapeRectangle(
    true,20, 1070, 0.0,
```

```
        null, ghostWhite,
                                    150, 10,
                                    1, LINE_STYLE_SOLID
  );
  rectangle33 = new ShapeRectangle(
    true,755, 10, 0.0,
        null, null,
                                    105, 30,
                                    1, LINE_STYLE_SOLID
  ) {

    @Override
    public boolean onClick( double clickx, double clicky ) {
      return onShapeClick( _rectangle33, 0, clickx, clicky );
    }
  };
  polyline18 = new ShapePolyLine(
                            true,345, 65,
        null, lavender,
        23, _polyline18_pointsDX_xjal(), _polyline18_pointsDY_xjal(),
        true, 1, LINE_STYLE_SOLID
  );
  polyline4 = new ShapePolyLine(
                            true,390, 420,
        cornflowerBlue, null,
        5, _polyline4_pointsDX_xjal(), _polyline4_pointsDY_xjal(),
        false, 4, LINE_STYLE_SOLID
  );
  polyline5 = new ShapePolyLine(
                            true,850, 110,
        cornflowerBlue, null,
        6, _polyline5_pointsDX_xjal(), _polyline5_pointsDY_xjal(),
        false, 4, LINE_STYLE_SOLID
  );
  polyline6 = new ShapePolyLine(
                            true,490, 60,
        cornflowerBlue, null,
        3, _polyline6_pointsDX_xjal(), _polyline6_pointsDY_xjal(),
        false, 2, LINE_STYLE_SOLID
  );
  line11 = new ShapeLine(
                            true,390, 60,
     cornflowerBlue,
                            0,          20,
                            2, LINE_STYLE_SOLID
  );
  line12 = new ShapeLine(
                            true,390, 100,
     cornflowerBlue,
                            0,          30,
                            2, LINE_STYLE_SOLID
  );
  line14 = new ShapeLine(
                            true,440, 160,
     cornflowerBlue,
                            0,          100,
                            2, LINE_STYLE_SOLID
  );
  line15 = new ShapeLine(
                            true,490, 210,
     cornflowerBlue,
                            70,         0,
                            2, LINE_STYLE_SOLID
  );
  line16 = new ShapeLine(
                            true,390, 160,
     cornflowerBlue,
                            100,        0,
                            2, LINE_STYLE_SOLID
  );
  line17 = new ShapeLine(
                            true,390, 110,
     cornflowerBlue,
                            210,        0,
                            2, LINE_STYLE_SOLID
  );
  line18 = new ShapeLine(
                            true,340, 60,
     cornflowerBlue,
                            0,          20,
                            2, LINE_STYLE_SOLID
  );
  line19 = new ShapeLine(
                            true,340, 100,
     cornflowerBlue,
                            0,          80,
                            2, LINE_STYLE_SOLID
  );
  line20 = new ShapeLine(
                            true,340, 250,
     cornflowerBlue,
                            0,          120,
                            2, LINE_STYLE_SOLID
  );
  line21 = new ShapeLine(
                            true,240, 210,
     cornflowerBlue,
                            100,        0,
                            2, LINE_STYLE_SOLID
  );
  line22 = new ShapeLine(
                            true,240, 140,
     cornflowerBlue,
                            100,        0,
                            2, LINE_STYLE_SOLID
  );
  line23 = new ShapeLine(
                            true,340, 390,
     cornflowerBlue,
                            0,          20,
                            2, LINE_STYLE_SOLID
  );
  line24 = new ShapeLine(
                            true,740, 210,
     cornflowerBlue,
                            100,        0,
                            2, LINE_STYLE_SOLID
  );
  line26 = new ShapeLine(
                            true,620, 110,
     cornflowerBlue,
                            20,         0,
                            2, LINE_STYLE_SOLID
  );
  polyline7 = new ShapePolyLine(
                            true,640, 60,
        cornflowerBlue, null,
        3, _polyline7_pointsDX_xjal(), _polyline7_pointsDY_xjal(),
        false, 2, LINE_STYLE_SOLID
  );
```

```
polyline8 = new ShapePolyLine(
                        true,590, 110,
    cornflowerBlue, null,
    3, _polyline8_pointsDX_xjal(), _polyline8_pointsDY_xjal(),
    false, 2, LINE_STYLE_SOLID
);
polyline9 = new ShapePolyLine(
                        true,390, 340,
    cornflowerBlue, null,
    4, _polyline9_pointsDX_xjal(), _polyline9_pointsDY_xjal(),
    false, 2, LINE_STYLE_SOLID
);
line27 = new ShapeLine(
                        true,630, 310,
  cornflowerBlue,
                        0,          100,
                        2, LINE_STYLE_SOLID
);
line29 = new ShapeLine(
                        true,740, 190,
  cornflowerBlue,
                        0,          40,
                        2, LINE_STYLE_SOLID
);
line30 = new ShapeLine(
                        true,580, 310,
  cornflowerBlue,
                        80,         0,
                        2, LINE_STYLE_SOLID
);
line31 = new ShapeLine(
                        true,530, 310,
  cornflowerBlue,
                        30,         0,
                        2, LINE_STYLE_SOLID
);
polyline10 = new ShapePolyLine(
                        true,690, 60,
    cornflowerBlue, null,
    3, _polyline10_pointsDX_xjal(), _polyline10_pointsDY_xjal(),
    false, 2, LINE_STYLE_SOLID
);
polyline11 = new ShapePolyLine(
                        true,840, 110,
    cornflowerBlue, null,
    3, _polyline11_pointsDX_xjal(), _polyline11_pointsDY_xjal(),
    false, 2, LINE_STYLE_SOLID
);
rectangle25 = new ShapeRectangle(
  true,630, 210, 0.0,
    cornflowerBlue, null,
                                60, 50,
                                2, LINE_STYLE_SOLID
);
line33 = new ShapeLine(
                        true,630, 210,
  cornflowerBlue,
                        60,         50,
                        1, LINE_STYLE_SOLID
);
line34 = new ShapeLine(
                        true,630, 260,
  cornflowerBlue,
                        60,         -50,
                        1, LINE_STYLE_SOLID
);
rectangle26 = new ShapeRectangle(
  true,530, 250, 0.0,
    cornflowerBlue, white,
                                10, 10,
                                1, LINE_STYLE_SOLID
);
text = new ShapeText(
  true,250, 215, 0.0,
  royalBlue,"WAITING ROOM",
  _text_Font, ALIGNMENT_LEFT
);
text1 = new ShapeText(
  true,250, 145, 0.0,
  royalBlue,"NURSES",
  _text1_Font, ALIGNMENT_LEFT
);
text2 = new ShapeText(
  true,250, 65, 0.0,
  royalBlue,"PAs",
  _text2_Font, ALIGNMENT_LEFT
);
text7 = new ShapeText(
  true,495, 65, 0.0,
  royalBlue,"TECHNICIANS",
  _text7_Font, ALIGNMENT_LEFT
);
text8 = new ShapeText(
  true,495, 115, 0.0,
  royalBlue,"X-RAY",
  _text8_Font, ALIGNMENT_LEFT
);
text9 = new ShapeText(
  true,645, 65, 0.0,
  royalBlue,"U SND",
  _text9_Font, ALIGNMENT_LEFT
);
line35 = new ShapeLine(
                        true,420, 435,
  cornflowerBlue,
                        0,          -40,
                        2, LINE_STYLE_SOLID
);
polyline12 = new ShapePolyLine(
                        true,415, 400,
    null, cornflowerBlue,
    3, _polyline12_pointsDX_xjal(), _polyline12_pointsDY_xjal(),
    false, 1, LINE_STYLE_SOLID
);
polyline13 = new ShapePolyLine(
                        true,855, 80,
    null, cornflowerBlue,
    3, _polyline13_pointsDX_xjal(), _polyline13_pointsDY_xjal(),
    false, 1, LINE_STYLE_SOLID
);
line36 = new ShapeLine(
                        true,820, 85,
  cornflowerBlue,
                        40,         0,
                        2, LINE_STYLE_SOLID
);
polyline14 = new ShapePolyLine(
                        true,840, 300,
    cornflowerBlue, null,
    4, _polyline14_pointsDX_xjal(), _polyline14_pointsDY_xjal(),
```

```
        false, 2, LINE_STYLE_SOLID
    );
    polyline15 = new ShapePolyLine(
                            true,730, 410,
        cornflowerBlue, null,
        4, _polyline15_pointsDX_xjal(), _polyline15_pointsDY_xjal(),
        false, 2, LINE_STYLE_SOLID
    );
    line37 = new ShapeLine(
                            true,725, 315,
      cornflowerBlue,
                            5,              -5,
                            2, LINE_STYLE_SOLID
    );
    line38 = new ShapeLine(
                            true,740, 300,
      cornflowerBlue,
                            5,              -5,
                            2, LINE_STYLE_SOLID
    );
    polyQueueAtReg = new ShapePolyLine(
                            true,440, 400,
        red, null,
        6,                          _polyQueueAtReg_pointsDX_xjal(),
_polyQueueAtReg_pointsDY_xjal(),
        false, 1, LINE_STYLE_SOLID
    ) {

      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
        _polyQueueAtReg_SetDynamicParams( this );
        super.draw( panel, graphics, xform, publicOnly );
      }
    };
    polyTriageRooms = new ShapePolyLine(
                            true,480, 100,
        red, null,
        2,                          _polyTriageRooms_pointsDX_xjal(),
_polyTriageRooms_pointsDY_xjal(),
        false, 2, LINE_STYLE_SOLID
    );
    polyline16 = new ShapePolyLine(
                            true,400, 330,
        cornflowerBlue, lavender,
        4, _polyline16_pointsDX_xjal(), _polyline16_pointsDY_xjal(),
        false, 2, LINE_STYLE_SOLID
    );
    polyECRooms = new ShapePolyLine(
                            true,620, 320,
        red, null,
        5,                          _polyECRooms_pointsDX_xjal(),
_polyECRooms_pointsDY_xjal(),
        false, 2, LINE_STYLE_SOLID
    );
    shapePatient = new ShapeCurve(
      true,50, -20,
      chocolate, gold,
      13, false,
          new double[] { 0, -1, 2, 5, 4, 3, 6, 3, 4, 0, 1, -2, 1,  },
         new double[] { 0, 4, 1, 4, 0, -3, -3, -5, -7, -7, -5, -3, -3,  },
              true, 1, LINE_STYLE_SOLID
    );
    shapeNurse = new ShapePolyLine(
                            true,67, -18,
        steelBlue, _shapeNurse_FillColor,
        18, _shapeNurse_pointsDX_xjal(), _shapeNurse_pointsDY_xjal(),
        true, 1, LINE_STYLE_SOLID
    );
    shapePA = new ShapePolyLine(
                            true,87, -18,
        limeGreen, _shapePA_FillColor,
        18, _shapePA_pointsDX_xjal(), _shapePA_pointsDY_xjal(),
        true, 1, LINE_STYLE_SOLID
    );
    shapeTech = new ShapePolyLine(
                            true,110, -21,
        royalBlue, lightSkyBlue,
        19, _shapeTech_pointsDX_xjal(), _shapeTech_pointsDY_xjal(),
        true, 1, LINE_STYLE_SOLID
    );
    polyline17 = new ShapePolyLine(
                            true,390, 150,
        cornflowerBlue, null,
        3, _polyline17_pointsDX_xjal(), _polyline17_pointsDY_xjal(),
        false, 2, LINE_STYLE_SOLID
    );
    line41 = new ShapeLine(
                            true,430, 260,
      cornflowerBlue,
                            20,             0,
                            2, LINE_STYLE_SOLID
    );
    line42 = new ShapeLine(
                            true,410, 240,
      cornflowerBlue,
                            60,             0,
                            2, LINE_STYLE_SOLID
    );
    curve = new ShapeCurve(
     true,437, 228,
     cornflowerBlue, lavender,
     7, false,
          new double[] { 0, -3, -8, -12, -8, -3, 0,  },
         new double[] { 0, 0, -1, 2, 5, 4, 4,  },
              false, 1, LINE_STYLE_SOLID
    );
    line43 = new ShapeLine(
                            true,420, 220,
      cornflowerBlue,
                            40,             0,
                            2, LINE_STYLE_SOLID
    );
    line44 = new ShapeLine(
                            true,420, 200,
      cornflowerBlue,
                            40,             0,
                            2, LINE_STYLE_SOLID
    );
    line45 = new ShapeLine(
                            true,420, 180,
      cornflowerBlue,
                            40,             0,
                            2, LINE_STYLE_SOLID
    );
    curve1 = new ShapeCurve(
     true,437, 208,
     cornflowerBlue, lavender,
     7, false,
          new double[] { 0, -3, -8, -12, -8, -3, 0,  },
         new double[] { 0, 0, -1, 2, 5, 4, 4,  },
              false, 1, LINE_STYLE_SOLID
```

```
  );
  curve2 = new ShapeCurve(
   true,437, 188,
   cornflowerBlue, lavender,
   7, false,
        new double[] { 0, -3, -8, -12, -8, -3, 0,  },
       new double[] { 0, 0, -1, 2, 5, 4, 4,  },
            false, 1, LINE_STYLE_SOLID
  );
  curve3 = new ShapeCurve(
   true,437, 168,
   cornflowerBlue, lavender,
   7, false,
        new double[] { 0, -3, -8, -12, -8, -3, 0,  },
       new double[] { 0, 0, -1, 2, 5, 4, 4,  },
            false, 1, LINE_STYLE_SOLID
  );
  curve4 = new ShapeCurve(
   true,443, 172,
   cornflowerBlue, lavender,
   7, false,
        new double[] { 0, 3, 8, 12, 8, 3, 0,  },
       new double[] { 0, 0, 1, -2, -5, -4, -4,  },
            false, 1, LINE_STYLE_SOLID
  );
  curve5 = new ShapeCurve(
   true,443, 192,
   cornflowerBlue, lavender,
   7, false,
        new double[] { 0, 3, 8, 12, 8, 3, 0,  },
       new double[] { 0, 0, 1, -2, -5, -4, -4,  },
            false, 1, LINE_STYLE_SOLID
  );
  curve6 = new ShapeCurve(
   true,443, 212,
   cornflowerBlue, lavender,
   7, false,
        new double[] { 0, 3, 8, 12, 8, 3, 0,  },
       new double[] { 0, 0, 1, -2, -5, -4, -4,  },
            false, 1, LINE_STYLE_SOLID
  );
  curve7 = new ShapeCurve(
   true,443, 232,
   cornflowerBlue, lavender,
   7, false,
        new double[] { 0, 3, 8, 12, 8, 3, 0,  },
       new double[] { 0, 0, 1, -2, -5, -4, -4,  },
            false, 1, LINE_STYLE_SOLID
  );
  NurseHomePath = new ShapePolyLine(
                       true,255, 170,
       red, null,
       10,                          _NurseHomePath_pointsDX_xjal(),
_NurseHomePath_pointsDY_xjal(),
       false, 2, LINE_STYLE_SOLID
  );
  PAHomePath = new ShapePolyLine(
                       true,255, 95,
       red, null,
       10,                          _PAHomePath_pointsDX_xjal(),
_PAHomePath_pointsDY_xjal(),
       false, 2, LINE_STYLE_SOLID
  );
  TechHomePath = new ShapePolyLine(
                       true,515, 90,
       red, null,
       8,                           _TechHomePath_pointsDX_xjal(),
_TechHomePath_pointsDY_xjal(),
       false, 2, LINE_STYLE_SOLID
  );
  line46 = new ShapeLine(
                       true,340, 200,
    cornflowerBlue,
                       0,          30,
                       2, LINE_STYLE_SOLID
  );
  line49 = new ShapeLine(
                       true,562, 133,
    indigo,
                       0,          10,
                       3, LINE_STYLE_SOLID
  );
  line48 = new ShapeLine(
                       true,550, 143,
    indigo,
                       25,         0,
                       2, LINE_STYLE_SOLID
  );
  rectangle31 = new ShapeRectangle(
    true,558, 132, 0.0,
       indigo, blueViolet,
                                   7, 3,
                                   1, LINE_STYLE_SOLID
  );
  rectangle30 = new ShapeRectangle(
    true,555, 143, 0.0,
       indigo, blueViolet,
                                   15, 7,
                                   1, LINE_STYLE_SOLID
  );
  waitingRoom = new ShapeRectangle(
    true,50, 170, 0.0,
       black, null,
                                   80, 180,
                                   1, LINE_STYLE_SOLID
  );
  rectangle3 = new ShapeRectangle(
    true,200, 70, 0.0,
       black, null,
                                   80, 30,
                                   1, LINE_STYLE_SOLID
  );
  rectangle4 = new ShapeRectangle(
    true,200, 20, 0.0,
       black, null,
                                   80, 30,
                                   1, LINE_STYLE_SOLID
  );
  rectangle6 = new ShapeRectangle(
    true,340, 270, 0.0,
       black, null,
                                   80, 80,
                                   1, LINE_STYLE_SOLID
  );
  rectangle7 = new ShapeRectangle(
    true,440, 270, 0.0,
       black, null,
                                   80, 80,
                                   1, LINE_STYLE_SOLID
  );
  rectangle8 = new ShapeRectangle(
    true,540, 260, 0.0,
```

```
    black, null,
                                90, 70,
                                1, LINE_STYLE_SOLID
  );
  rectangle9 = new ShapeRectangle(
    true,550, 170, 0.0,
    black, null,
                                80, 70,
                                1, LINE_STYLE_SOLID
  );
  rectangle10 = new ShapeRectangle(
    true,550, 70, 0.0,
    black, null,
                                80, 80,
                                1, LINE_STYLE_SOLID
  );
  xRayRoom = new ShapeRectangle(
    true,300, 70, 0.0,
    black, null,
                                80, 80,
                                1, LINE_STYLE_SOLID
  );
  USoundStorage = new ShapeRectangle(
    true,450, 20, 0.0,
    black, null,
                                30, 80,
                                1, LINE_STYLE_SOLID
  );
  techHomeRoom = new ShapeRectangle(
    true,300, 20, 0.0,
    black, null,
                                130, 30,
                                1, LINE_STYLE_SOLID
  );
  nurseHomeRoom = new ShapeRectangle(
    true,50, 100, 0.0,
    black, null,
                                80, 50,
                                1, LINE_STYLE_SOLID
  );
  doctorHomeRoom = new ShapeRectangle(
    true,50, 20, 0.0,
    black, null,
                                80, 60,
                                1, LINE_STYLE_SOLID
  );
  entryDoor = new ShapeRectangle(
    true,200, 360, 0.0,
    black, null,
                                40, 10,
                                1, LINE_STYLE_SOLID
  );
  exitDoor = new ShapeRectangle(
    true,640, 20, 0.0,
    black, null,
                                10, 30,
                                1, LINE_STYLE_SOLID
  );
  rectangle11 = new ShapeRectangle(
    true,160, 30, 0.0,
    black, null,
                                10, 10,
                                1, LINE_STYLE_SOLID
  );
  rectangle12 = new ShapeRectangle(
    true,160, 80, 0.0,
    black, null,
                                10, 10,
                                1, LINE_STYLE_SOLID
  );
  rectangle13 = new ShapeRectangle(
    true,160, 130, 0.0,
    black, null,
                                10, 10,
                                1, LINE_STYLE_SOLID
  );
  rectangle15 = new ShapeRectangle(
    true,160, 230, 0.0,
    black, null,
                                10, 10,
                                1, LINE_STYLE_SOLID
  );
  rectangle16 = new ShapeRectangle(
    true,360, 240, 0.0,
    black, null,
                                10, 10,
                                1, LINE_STYLE_SOLID
  );
  rectangle17 = new ShapeRectangle(
    true,460, 240, 0.0,
    black, null,
                                10, 10,
                                1, LINE_STYLE_SOLID
  );
  rectangle18 = new ShapeRectangle(
    true,520, 190, 0.0,
    black, null,
                                10, 10,
                                1, LINE_STYLE_SOLID
  );
  rectangle19 = new ShapeRectangle(
    true,520, 130, 0.0,
    black, null,
                                10, 10,
                                1, LINE_STYLE_SOLID
  );
  rectangle20 = new ShapeRectangle(
    true,460, 120, 0.0,
    black, null,
                                10, 10,
                                1, LINE_STYLE_SOLID
  );
  rectangle21 = new ShapeRectangle(
    true,410, 70, 0.0,
    black, null,
                                10, 10,
                                1, LINE_STYLE_SOLID
  );
  rectangle22 = new ShapeRectangle(
    true,410, 120, 0.0,
    black, null,
                                10, 10,
                                1, LINE_STYLE_SOLID
  );
  rectangle23 = new ShapeRectangle(
    true,360, 170, 0.0,
    black, null,
                                10, 10,
                                1, LINE_STYLE_SOLID
  );
```

```
rectangle24 = new ShapeRectangle(
  true,160, 320, 0.0,
      black, null,
                                  10, 10,
                                  1, LINE_STYLE_SOLID
);
polyline = new ShapePolyLine(
                      true,210, 360,
      black, null,
      9, _polyline_pointsDX_xjal(), _polyline_pointsDY_xjal(),
      false, 1, LINE_STYLE_SOLID
);
line = new ShapeLine(
                      true,170, 40,
   black,
                      40,          0,
                      1, LINE_STYLE_SOLID
);
line1 = new ShapeLine(
                      true,170, 90,
   black,
                      40,          0,
                      1, LINE_STYLE_SOLID
);
line4 = new ShapeLine(
                      true,120, 330,
   black,
                      50,          0,
                      1, LINE_STYLE_SOLID
);
line5 = new ShapeLine(
                      true,120, 140,
   black,
                      50,          0,
                      1, LINE_STYLE_SOLID
);
polyline1 = new ShapePolyLine(
                      true,170, 240,
      black, null,
      11, _polyline1_pointsDX_xjal(), _polyline1_pointsDY_xjal(),
      false, 1, LINE_STYLE_SOLID
);
line6 = new ShapeLine(
                      true,370, 240,
   black,
                      0,           40,
                      1, LINE_STYLE_SOLID
);
line7 = new ShapeLine(
                      true,470, 240,
   black,
                      0,           40,
                      1, LINE_STYLE_SOLID
);
line8 = new ShapeLine(
                      true,520, 190,
   black,
                      40,          0,
                      1, LINE_STYLE_SOLID
);
line9 = new ShapeLine(
                      true,520, 130,
   black,
                      40,          0,
                      1, LINE_STYLE_SOLID
);
polyline2 = new ShapePolyLine(
                      true,370, 240,
      black, null,
      7, _polyline2_pointsDX_xjal(), _polyline2_pointsDY_xjal(),
      false, 1, LINE_STYLE_SOLID
);
polyline3 = new ShapePolyLine(
                      true,410, 130,
      black, null,
      3, _polyline3_pointsDX_xjal(), _polyline3_pointsDY_xjal(),
      false, 1, LINE_STYLE_SOLID
);
line10 = new ShapeLine(
                      true,460, 130,
   black,
                      60,          0,
                      1, LINE_STYLE_SOLID
);
line32 = new ShapeLine(
                      true,370, 180,
   black,
                      0,           -40,
                      1, LINE_STYLE_SOLID
);
rectangle27 = new ShapeRectangle(
  true,505, 225, 0.0,
      black, null,
                                  10, 10,
                                  1, LINE_STYLE_SOLID
);
line39 = new ShapeLine(
                      true,510, 230,
   black,
                      40,          40,
                      1, LINE_STYLE_SOLID
);
rectangle29 = new ShapeRectangle(
  true,200, 320, 0.0,
      black, white,
                                  10, 10,
                                  1, LINE_STYLE_SOLID
);
line40 = new ShapeLine(
                      true,210, 290,
   black,
                      0,           40,
                      1, LINE_STYLE_SOLID
);
registrationDesk = new ShapeRectangle(
  true,210, 280, 0.0,
      black, white,
                                  20, 10,
                                  1, LINE_STYLE_SOLID
);
rectangle = new ShapeRectangle(
  true,160, 180, 0.0,
      black, null,
                                  10, 10,
                                  1, LINE_STYLE_SOLID
);
line47 = new ShapeLine(
                      true,120, 190,
   black,
                      50,          0,
```

```
                    1, LINE_STYLE_SOLID
  );
  oval1 = new ShapeOval(
    true,0, 0, 0.0,
    null, lightSalmon,
          6, 6,
          1, LINE_STYLE_SOLID
  );
  oval = new ShapeOval(
    true,0, 0, 0.0,
    null, red,
          4, 4,
          1, LINE_STYLE_SOLID
  );
  oval2 = new ShapeOval(
    true,0, 0, 0.0,
    null, greenYellow,
          6, 6,
          1, LINE_STYLE_SOLID
  );
  oval3 = new ShapeOval(
    true,0, 0, 0.0,
    null, limeGreen,
          4, 4,
          1, LINE_STYLE_SOLID
  );
  XRayHomePath = new ShapePolyLine(
                    true,580, 120,
      red, null,
      2,                    _XRayHomePath_pointsDX_xjal(),
_XRayHomePath_pointsDY_xjal(),
      false, 2, LINE_STYLE_SOLID
  );
  USoundHomePath = new ShapePolyLine(
                    true,660, 80,
      red, null,
      7,                    _USoundHomePath_pointsDX_xjal(),
_USoundHomePath_pointsDY_xjal(),
      false, 2, LINE_STYLE_SOLID
  );
  line50 = new ShapeLine(
                    true,2, -6,
    blue,
                    -2,     4,
                    2, LINE_STYLE_SOLID
  );
  oval5 = new ShapeOval(
    true,3, 6, 0.0,
    null, blue,
          2, 2,
          1, LINE_STYLE_SOLID
  );
  rectangle32 = new ShapeRectangle(
    true,-4, 0, 0.0,
      blue, blue,
                    9, 3,
                    1, LINE_STYLE_SOLID
  );
  polyline19 = new ShapePolyLine(
                    true,0, -4,
      blue, null,
      3, _polyline19_pointsDX_xjal(), _polyline19_pointsDY_xjal(),
      false, 1, LINE_STYLE_SOLID
  );
  oval4 = new ShapeOval(
    true,-2, 6, 0.0,
    null, blue,
          2, 2,
          1, LINE_STYLE_SOLID
);
text15 = new ShapeText(
  true,590, 15, 0.0,
  black,"Animation & Output",
  _text15_Font, ALIGNMENT_LEFT
);
text16 = new ShapeText(
  true,760, 15, 0.0,
  royalBlue,"Model Logic",
  _text16_Font, ALIGNMENT_LEFT
);
line51 = new ShapeLine(
                    true,750, 10,
  black,
                    0,      30,
                    1, LINE_STYLE_SOLID
);
line52 = new ShapeLine(
                    true,760, 34,
  royalBlue,
                    95,     0,
                    1, LINE_STYLE_SOLID
);
text17 = new ShapeText(
  true,590, 1015, 0.0,
  royalBlue,"Animation & Output",
  _text17_Font, ALIGNMENT_LEFT
);
rectangle34 = new ShapeRectangle(
  true,585, 1010, 0.0,
    null, null,
                    160, 30,
                    1, LINE_STYLE_SOLID
) {

 @Override
 public boolean onClick( double clickx, double clicky ) {
  return onShapeClick( _rectangle34, 0, clickx, clicky );
 }
};
text18 = new ShapeText(
  true,760, 1015, 0.0,
  black,"Model Logic",
  _text18_Font, ALIGNMENT_LEFT
);
line53 = new ShapeLine(
                    true,750, 1010,
  black,
                    0,      30,
                    1, LINE_STYLE_SOLID
);
line54 = new ShapeLine(
                    true,590, 1034,
  royalBlue,
                    155,    0,
                    1, LINE_STYLE_SOLID
);
text19 = new ShapeText(
  true,30, 1075, 0.0,
  _text19_Color,"Resources",
  _text19_Font, ALIGNMENT_LEFT
);
```

```
text20 = new ShapeText(
    true,200, 1075, 0.0,
    _text20_Color,"Process flowchart",
    _text20_Font, ALIGNMENT_LEFT
);
polyline20 = new ShapePolyLine(
                        true,20, 1090,
      slateGray, null,
      4, _polyline20_pointsDX_xjal(), _polyline20_pointsDY_xjal(),
      false, 1, LINE_STYLE_SOLID
);
polyline21 = new ShapePolyLine(
                        true,190, 1090,
      slateGray, null,
      4, _polyline21_pointsDX_xjal(), _polyline21_pointsDY_xjal(),
      false, 1, LINE_STYLE_SOLID
);
text21 = new ShapeText(
    true,295, 1405, 0.0,
    blue,"Ultra Sound Process",
    _text21_Font, ALIGNMENT_LEFT
);
text22 = new ShapeText(
    true,310, 1500, 0.0,
    blue,"X-Ray Process",
    _text22_Font, ALIGNMENT_LEFT
);
line55 = new ShapeLine(
                        true,295, 1418,
    blue,
                        115,        0,
                        1, LINE_STYLE_SOLID
);
line56 = new ShapeLine(
                        true,310, 1513,
    blue,
                        80,         0,
                        1, LINE_STYLE_SOLID
);
rectangle39 = new ShapeRectangle(
  true,20, 450, 0.0,
      null, ghostWhite,
                                630, 10,
                                1, LINE_STYLE_SOLID
);
text24 = new ShapeText(
    true,30, 455, 0.0,
    _text24_Color,"Resource utilization",
    _text24_Font, ALIGNMENT_LEFT
);
polyline22 = new ShapePolyLine(
                        true,20, 470,
      slateGray, null,
      4, _polyline22_pointsDX_xjal(), _polyline22_pointsDY_xjal(),
      false, 1, LINE_STYLE_SOLID
);
rectangle40 = new ShapeRectangle(
  true,20, 160, 0.0,
      null, ghostWhite,
                                201, 10,
                                1, LINE_STYLE_SOLID
);
text25 = new ShapeText(
    true,30, 165, 0.0,
    _text25_Color,"Parameters",
    _text25_Font, ALIGNMENT_LEFT
);
polyline23 = new ShapePolyLine(
                        true,20, 180,
      slateGray, null,
      4, _polyline23_pointsDX_xjal(), _polyline23_pointsDY_xjal(),
      false, 1, LINE_STYLE_SOLID
);
text26 = new ShapeText(
    true,30, 190, 0.0,
    navy,"Nurses",
    _text26_Font, ALIGNMENT_LEFT
);
text27 = new ShapeText(
    true,30, 250, 0.0,
    navy,"PAs",
    _text27_Font, ALIGNMENT_LEFT
);
text28 = new ShapeText(
    true,30, 310, 0.0,
    navy,"Technicians",
    _text28_Font, ALIGNMENT_LEFT
);
text29 = new ShapeText(
    true,30, 370, 0.0,
    navy,"U-Sound devices",
    _text29_Font, ALIGNMENT_LEFT
);
text30 = new ShapeText(
    true,200, 65, 0.0,
    black,"FAST",
    _text30_Font, ALIGNMENT_CENTER
) {

  @Override
  public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
    _text30_SetDynamicParams( this );
    super.draw( panel, graphics, xform, publicOnly );
  }
};
text31 = new ShapeText(
    true,40, 65, 0.0,
    black,"SLOW",
    _text31_Font, ALIGNMENT_CENTER
) {

  @Override
  public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
    _text31_SetDynamicParams( this );
    super.draw( panel, graphics, xform, publicOnly );
  }
};
rectangle42 = new ShapeRectangle(
  true,20, 60, 0.0,
      steelBlue, null,
                                40, 20,
                                1, LINE_STYLE_SOLID
) {

  @Override
  public boolean onClick( double clickx, double clicky ) {
    return onShapeClick( _rectangle42, 0, clickx, clicky );
  }
};
```

```
  rectangle43 = new ShapeRectangle(
    true,670, 450, 0.0,
        null, ghostWhite,
                                    210, 10,
                                    1, LINE_STYLE_SOLID
  );
  text32 = new ShapeText(
     true,680, 455, 0.0,
     _text32_Color,"Length of Stay",
     _text32_Font, ALIGNMENT_LEFT
  );
  polyline24 = new ShapePolyLine(
                          true,670, 470,
        slateGray, null,
        4, _polyline24_pointsDX_xjal(), _polyline24_pointsDY_xjal(),
        false, 1, LINE_STYLE_SOLID
  );
  text6 = new ShapeText(
     true,540, 390, 0.0,
     royalBlue,"EC ROOM 1",
     _text6_Font, ALIGNMENT_LEFT
  );
  text10 = new ShapeText(
     true,640, 390, 0.0,
     royalBlue,"EC ROOM 2",
     _text10_Font, ALIGNMENT_LEFT
  );
  text11 = new ShapeText(
     true,780, 280, 0.0,
     royalBlue,"EC ROOM 4",
     _text11_Font, ALIGNMENT_LEFT
  );
  text12 = new ShapeText(
     true,780, 190, 0.0,
     royalBlue,"EC ROOM 5",
     _text12_Font, ALIGNMENT_LEFT
  );
  text14 = new ShapeText(
     true,740, 390, 0.0,
     royalBlue,"EC ROOM 3",
     _text14_Font, ALIGNMENT_LEFT
  );
  text3 = new ShapeText(
     true,395, 65, 0.0,
     royalBlue,"TRIAGE ROOM 1",
     _text3_Font, ALIGNMENT_LEFT
  );
  text5 = new ShapeText(
     true,395, 115, 0.0,
     royalBlue,"TRIAGE ROOM 2",
     _text5_Font, ALIGNMENT_LEFT
  );
  text13 = new ShapeText(
     true,20, 10, 0.0,
     navy,"Emergency Department",
     _text13_Font, ALIGNMENT_LEFT
  );
  text23 = new ShapeText(
     true,20, 1010, 0.0,
     navy,"Emergency Department",
     _text23_Font, ALIGNMENT_LEFT
  );
 }
 ShapeGroup presentation;
 ShapeGroup icon;

 @Override
 public Object getPersistentShape( int _shape ) {
  switch(_shape){
    case _presentation: return presentation;
    case _icon: return icon;

    case _radio: return radio;
    case _radio1: return radio1;
    case _radio2: return radio2;
    case _radio3: return radio3;
    case _chart: return chart;
    case _chart1: return chart1;
    case _rectXRayFloor: return rectXRayFloor;
    case _arc: return arc;
    case _arc2: return arc2;
    case _arc1: return arc1;
    case _oval6: return oval6;
    case _group1: return group1;
    case _rectangle41: return rectangle41;
    case _rectangle38: return rectangle38;
    case _rectangle37: return rectangle37;
    case _rectangle36: return rectangle36;
    case _rectangle35: return rectangle35;
    case _rectangle33: return rectangle33;
    case _polyline18: return polyline18;
    case _polyline4: return polyline4;
    case _polyline5: return polyline5;
    case _polyline6: return polyline6;
    case _line11: return line11;
    case _line12: return line12;
    case _line14: return line14;
    case _line15: return line15;
    case _line16: return line16;
    case _line17: return line17;
    case _line18: return line18;
    case _line19: return line19;
    case _line20: return line20;
    case _line21: return line21;
    case _line22: return line22;
    case _line23: return line23;
    case _line24: return line24;
    case _line26: return line26;
    case _polyline7: return polyline7;
    case _polyline8: return polyline8;
    case _polyline9: return polyline9;
    case _line27: return line27;
    case _line29: return line29;
    case _line30: return line30;
    case _line31: return line31;
    case _polyline10: return polyline10;
    case _polyline11: return polyline11;
    case _rectangle25: return rectangle25;
    case _line33: return line33;
    case _line34: return line34;
    case _rectangle26: return rectangle26;
    case _text: return text;
    case _text1: return text1;
    case _text2: return text2;
    case _text7: return text7;
    case _text8: return text8;
    case _text9: return text9;
    case _line35: return line35;
    case _polyline12: return polyline12;
    case _polyline13: return polyline13;
```

```
case _line36: return line36;
case _polyline14: return polyline14;
case _polyline15: return polyline15;
case _line37: return line37;
case _line38: return line38;
case _polyQueueAtReg: return polyQueueAtReg;
case _polyTriageRooms: return polyTriageRooms;
case _polyline16: return polyline16;
case _polyECRooms: return polyECRooms;
case _shapePatient: return shapePatient;
case _shapeNurse: return shapeNurse;
case _shapePA: return shapePA;
case _shapeTech: return shapeTech;
case _polyline17: return polyline17;
case _line41: return line41;
case _line42: return line42;
case _curve: return curve;
case _line43: return line43;
case _line44: return line44;
case _line45: return line45;
case _curve1: return curve1;
case _curve2: return curve2;
case _curve3: return curve3;
case _curve4: return curve4;
case _curve5: return curve5;
case _curve6: return curve6;
case _curve7: return curve7;
case _NurseHomePath: return NurseHomePath;
case _PAHomePath: return PAHomePath;
case _TechHomePath: return TechHomePath;
case _line46: return line46;
case _line49: return line49;
case _line48: return line48;
case _rectangle31: return rectangle31;
case _rectangle30: return rectangle30;
case _waitingRoom: return waitingRoom;
case _rectangle3: return rectangle3;
case _rectangle4: return rectangle4;
case _rectangle6: return rectangle6;
case _rectangle7: return rectangle7;
case _rectangle8: return rectangle8;
case _rectangle9: return rectangle9;
case _rectangle10: return rectangle10;
case _xRayRoom: return xRayRoom;
case _USoundStorage: return USoundStorage;
case _techHomeRoom: return techHomeRoom;
case _nurseHomeRoom: return nurseHomeRoom;
case _doctorHomeRoom: return doctorHomeRoom;
case _entryDoor: return entryDoor;
case _exitDoor: return exitDoor;
case _rectangle11: return rectangle11;
case _rectangle12: return rectangle12;
case _rectangle13: return rectangle13;
case _rectangle15: return rectangle15;
case _rectangle16: return rectangle16;
case _rectangle17: return rectangle17;
case _rectangle18: return rectangle18;
case _rectangle19: return rectangle19;
case _rectangle20: return rectangle20;
case _rectangle21: return rectangle21;
case _rectangle22: return rectangle22;
case _rectangle23: return rectangle23;
case _rectangle24: return rectangle24;
case _polyline: return polyline;
case _line: return line;
case _line1: return line1;
case _line4: return line4;
case _line5: return line5;
case _polyline1: return polyline1;
case _line6: return line6;
case _line7: return line7;
case _line8: return line8;
case _line9: return line9;
case _polyline2: return polyline2;
case _polyline3: return polyline3;
case _line10: return line10;
case _line32: return line32;
case _rectangle27: return rectangle27;
case _line39: return line39;
case _rectangle29: return rectangle29;
case _line40: return line40;
case _registrationDesk: return registrationDesk;
case _rectangle: return rectangle;
case _line47: return line47;
case _group: return group;
case _oval1: return oval1;
case _oval: return oval;
case _shapeBusy: return shapeBusy;
case _oval2: return oval2;
case _oval3: return oval3;
case _shapeIdle: return shapeIdle;
case _XRayHomePath: return XRayHomePath;
case _USoundHomePath: return USoundHomePath;
case _line50: return line50;
case _oval5: return oval5;
case _rectangle32: return rectangle32;
case _polyline19: return polyline19;
case _oval4: return oval4;
case _shapeUSound: return shapeUSound;
case _text15: return text15;
case _text16: return text16;
case _line51: return line51;
case _line52: return line52;
case _text17: return text17;
case _rectangle34: return rectangle34;
case _text18: return text18;
case _line53: return line53;
case _line54: return line54;
case _text19: return text19;
case _text20: return text20;
case _polyline20: return polyline20;
case _polyline21: return polyline21;
case _text21: return text21;
case _text22: return text22;
case _line55: return line55;
case _line56: return line56;
case _rectangle39: return rectangle39;
case _text24: return text24;
case _polyline22: return polyline22;
case _rectangle40: return rectangle40;
case _text25: return text25;
case _polyline23: return polyline23;
case _text26: return text26;
case _text27: return text27;
case _text28: return text28;
case _text29: return text29;
case _text30: return text30;
case _text31: return text31;
case _rectangle42: return rectangle42;
case _rectangle43: return rectangle43;
```

```
    case _text32: return text32;
    case _polyline24: return polyline24;
    case _text6: return text6;
    case _text10: return text10;
    case _text11: return text11;
    case _text12: return text12;
    case _text14: return text14;
    case _text3: return text3;
    case _text5: return text5;
    case _text13: return text13;
    case _text23: return text23;
    default: return null;
  }
}

static final int[] _connector_pointsX = {
  280,  330 };
static final int[] _connector_pointsY = {
  1140,1140 };

static final int[] _connector1_pointsX = {
  380,  430 };
static final int[] _connector1_pointsY = {
  1140,1140 };

static final int[] _connector2_pointsX = {
  480,  530 };
static final int[] _connector2_pointsY = {
  1140,1140 };

static final int[] _connector3_pointsX = {
  580,  630 };
static final int[] _connector3_pointsY = {
  1140,1140 };

static final int[] _connector4_pointsX = {
  680,  730 };
static final int[] _connector4_pointsY = {
  1140,1140 };

static final int[] _connector5_pointsX = {
  660,  710 };
static final int[] _connector5_pointsY = {
  1460,1460 };

static final int[] _connector6_pointsX = {
  760,  810 };
static final int[] _connector6_pointsY = {
  1460,1460 };

static final int[] _connector8_pointsX = {
  70,  70 };
static final int[] _connector8_pointsY = {
  1150,1200 };

static final int[] _connector9_pointsX = {
  70,  70, 30, 30, 70, 70 };
static final int[] _connector9_pointsY = {
  1150,1180, 1180, 1250, 1250, 1270 };

static final int[] _connector10_pointsX = {
  780,  800, 800, 300, 300, 330 };
static final int[] _connector10_pointsY = {
  1140,1140, 1190, 1190, 1240, 1240 };

static final int[] _connector11_pointsX = {
  380,  430 };
static final int[] _connector11_pointsY = {
  1240,1240 };

static final int[] _connector12_pointsX = {
  480,  530 };
static final int[] _connector12_pointsY = {
  1240,1240 };

static final int[] _connector13_pointsX = {
  580,  640 };
static final int[] _connector13_pointsY = {
  1240,1240 };

static final int[] _connector14_pointsX = {
  690,  730 };
static final int[] _connector14_pointsY = {
  1240,1240 };

static final int[] _connector15_pointsX = {
  770,  800, 800, 300, 300, 330 };
static final int[] _connector15_pointsY = {
  1240,1240, 1290, 1290, 1340, 1340 };

static final int[] _connector16_pointsX = {
  380,  430 };
static final int[] _connector16_pointsY = {
  1340,1340 };

static final int[] _connector17_pointsX = {
  70,  70, 140, 140 };
static final int[] _connector17_pointsY = {
  1150,1180, 1180, 1200 };

static final int[] _connector18_pointsX = {
  480,  530 };
static final int[] _connector18_pointsY = {
  1340,1340 };

static final int[] _connector19_pointsX = {
  580,  630 };
static final int[] _connector19_pointsY = {
  1340,1340 };

static final int[] _connector20_pointsX = {
  550,  610 };
static final int[] _connector20_pointsY = {
  1460,1460 };

static final int[] _connector22_pointsX = {
  70,  70, 30, 30, 140, 140 };
static final int[] _connector22_pointsY = {
  1150,1180, 1180, 1250, 1250, 1270 };

static final int[] _connector23_pointsX = {
  70,  70, 30, 30, 70, 70 };
static final int[] _connector23_pointsY = {
  1150,1180, 1180, 1320, 1320, 1340 };

static final int[] _connector24_pointsX = {
  670,  730 };
static final int[] _connector24_pointsY = {
  1340,1340 };
```

```
  static final int[] _connector25_pointsX = {
    780,  800, 800, 240, 240, 270 };
  static final int[] _connector25_pointsY = {
    1340,1340, 1390, 1390, 1460, 1460 };

  static final int[] _connector27_pointsX = {
    70,  70, 30, 30, 140, 140 };
  static final int[] _connector27_pointsY = {
    1150,1180, 1180, 1320, 1320, 1340 };

  static final int[] _connector32_pointsX = {
    450,  510 };
  static final int[] _connector32_pointsY = {
    1460,1460 };

  static final int[] _connector34_pointsX = {
    70,  70, 30, 30, 70, 70 };
  static final int[] _connector34_pointsY = {
    1150,1180, 1180, 1390, 1390, 1410 };

  static final int[] _connector35_pointsX = {
    300,  320, 330 };
  static final int[] _connector35_pointsY = {
    1450,1450, 1440 };

  static final int[] _connector36_pointsX = {
    300,  320, 330 };
  static final int[] _connector36_pointsY = {
    1470,1470, 1490 };

  static final int[] _connector37_pointsX = {
    370,  410 };
  static final int[] _connector37_pointsY = {
    1440,1460 };

  static final int[] _connector38_pointsX = {
    370,  410 };
  static final int[] _connector38_pointsY = {
    1490,1460 };


  @Override
  public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
    if (!_publicOnly) {
      drawFunction( _panel, _g, -100, 30, 10, 0, "resetStats");
    }
    if (!_publicOnly) {
      drawDataset( _panel, _g, 800, 1500, 10, 0, "histLOS", histLOS );
    }
    // Embedded object "network"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 20 , 1100 , 40,
10, "network", this.network );
    }
    // Embedded object "arrive"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 180 , 1090 , 60,
20, "arrive", this.arrive );
    }
    // Embedded object "networkEnter"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 280 , 1090 , 40,
20, "networkEnter", this.networkEnter );
    }
    // Embedded object "gotoRegistration"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 380 , 1090 , 40,
20, "gotoRegistration", this.gotoRegistration );
    }
    // Embedded object "queueAtReg"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 480 , 1090 , 50,
10, "queueAtReg", this.queueAtReg );
    }
    // Embedded object "registration"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 580 , 1090 , 50,
20, "registration", this.registration );
    }
    // Embedded object "gotoWaitingRoom1"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 680 , 1090 , 40,
20, "gotoWaitingRoom1", this.gotoWaitingRoom1 );
    }
    // Embedded object "gotoExit"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 560 , 1410 , 50,
20, "gotoExit", this.gotoExit );
    }
    // Embedded object "networkExit"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 660 , 1410 , 50,
20, "networkExit", this.networkExit );
    }
    // Embedded object "leave"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 760 , 1410 , 60,
20, "leave", this.leave );
    }
    // Embedded object "triageRoom"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 20 , 1150 , 20,
80, "triageRoom", this.triageRoom );
    }
    // Embedded object "nurse"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 20 , 1220 , 30,
80, "nurse", this.nurse );
    }
    // Embedded object "seizeTriageRoom"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 280 , 1190 , 40,
20, "seizeTriageRoom", this.seizeTriageRoom );
    }
    // Embedded object "gotoTriageRoom"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 480 , 1190 , 50,
20, "gotoTriageRoom", this.gotoTriageRoom );
    }
    // Embedded object "triage"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 590 , 1190 , 50,
20, "triage", this.triage );
    }
    // Embedded object "gotoWaitingRoom2"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 280 , 1290 , 30,
20, "gotoWaitingRoom2", this.gotoWaitingRoom2 );
    }
    // Embedded object "releaseTriageRoom"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 680 , 1190 , 40,
20, "releaseTriageRoom", this.releaseTriageRoom );
    }
    // Embedded object "releaseNurse"
```

```
   if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 580 , 1290 , 50,
20, "releaseNurse", this.releaseNurse );
   }
   // Embedded object "seizeECRoom"
   if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 380 , 1290 , 50,
20, "seizeECRoom", this.seizeECRoom );
   }
   // Embedded object "ECRoom"
   if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 90 , 1150 , 20,
80, "ECRoom", this.ECRoom );
   }
   // Embedded object "gotoECRoom"
   if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 480 , 1290 , 50,
20, "gotoECRoom", this.gotoECRoom );
   }
   // Embedded object "releaseECRoom"
   if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 460 , 1410 , 40,
20, "releaseECRoom", this.releaseECRoom );
   }
   // Embedded object "PA"
   if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 90 , 1220 , 40,
80, "PA", this.PA );
   }
   // Embedded object "technician"
   if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 20 , 1290 , 20,
80, "technician", this.technician );
   }
   // Embedded object "XRay"
   if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 90 , 1290 , 40,
80, "XRay", this.XRay );
   }
   // Embedded object "releasePA"
   if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 360 , 1410 , 50,
20, "releasePA", this.releasePA );
   }
   // Embedded object "xRayProcess"
   if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 280 , 1390 , 40,
110, null, this.xRayProcess );
   }
   // Embedded object "USound"
   if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 20 , 1360 , 30,
80, "USound", this.USound );
   }
   // Embedded object "uSoundProcess"
   if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 280 , 1340 , 50,
80, null, this.uSoundProcess );
   }
   // Embedded object "selectProcess"
   if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 220 , 1410 , 10,
70, "selectProcess", this.selectProcess );
   }
   // Embedded object "clock"
   if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 120 , 100 , 0, -
30, null, this.clock );
   }
   // Embedded object "callNurse"
   if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 380 , 1190 , 50,
20, "callNurse", this.callNurse );
   }
   // Embedded object "callPA"
   if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 680 , 1290 , 50,
20, "callPA", this.callPA );
   }
   if (!_publicOnly) {
      drawConnector( _panel, _g, _connector_pointsX,
_connector_pointsY, false );
      }
   if (!_publicOnly) {
      drawConnector( _panel, _g, _connector1_pointsX,
_connector1_pointsY, false );
      }
   if (!_publicOnly) {
      drawConnector( _panel, _g, _connector2_pointsX,
_connector2_pointsY, false );
      }
   if (!_publicOnly) {
      drawConnector( _panel, _g, _connector3_pointsX,
_connector3_pointsY, false );
      }
   if (!_publicOnly) {
      drawConnector( _panel, _g, _connector4_pointsX,
_connector4_pointsY, false );
      }
   if (!_publicOnly) {
      drawConnector( _panel, _g, _connector5_pointsX,
_connector5_pointsY, false );
      }
   if (!_publicOnly) {
      drawConnector( _panel, _g, _connector6_pointsX,
_connector6_pointsY, false );
      }
   if (!_publicOnly) {
      drawConnector( _panel, _g, _connector8_pointsX,
_connector8_pointsY, false );
      }
   if (!_publicOnly) {
      drawConnector( _panel, _g, _connector9_pointsX,
_connector9_pointsY, false );
      }
   if (!_publicOnly) {
      drawConnector( _panel, _g, _connector10_pointsX,
_connector10_pointsY, false );
      }
   if (!_publicOnly) {
      drawConnector( _panel, _g, _connector11_pointsX,
_connector11_pointsY, false );
      }
   if (!_publicOnly) {
      drawConnector( _panel, _g, _connector12_pointsX,
_connector12_pointsY, false );
      }
   if (!_publicOnly) {
      drawConnector( _panel, _g, _connector13_pointsX,
_connector13_pointsY, false );
      }
   if (!_publicOnly) {
      drawConnector( _panel, _g, _connector14_pointsX,
_connector14_pointsY, false );
      }
   if (!_publicOnly) {
      drawConnector( _panel, _g, _connector15_pointsX,
_connector15_pointsY, false );
      }
```

```
    if (!_publicOnly) {
      drawConnector(        _panel,        _g,        _connector16_pointsX,
_connector16_pointsY, false );
    }
    if (!_publicOnly) {
      drawConnector(        _panel,        _g,        _connector17_pointsX,
_connector17_pointsY, false );
    }
    if (!_publicOnly) {
      drawConnector(        _panel,        _g,        _connector18_pointsX,
_connector18_pointsY, false );
    }
    if (!_publicOnly) {
      drawConnector(        _panel,        _g,        _connector19_pointsX,
_connector19_pointsY, false );
    }
    if (!_publicOnly) {
      drawConnector(        _panel,        _g,        _connector20_pointsX,
_connector20_pointsY, false );
    }
    if (!_publicOnly) {
      drawConnector(        _panel,        _g,        _connector22_pointsX,
_connector22_pointsY, false );
    }
    if (!_publicOnly) {
      drawConnector(        _panel,        _g,        _connector23_pointsX,
_connector23_pointsY, false );
    }
    if (!_publicOnly) {
      drawConnector(        _panel,        _g,        _connector24_pointsX,
_connector24_pointsY, false );
    }
    if (!_publicOnly) {
      drawConnector(        _panel,        _g,        _connector25_pointsX,
_connector25_pointsY, false );
    }
    if (!_publicOnly) {
      drawConnector(        _panel,        _g,        _connector27_pointsX,
_connector27_pointsY, false );
    }
    if (!_publicOnly) {
      drawConnector(        _panel,        _g,        _connector32_pointsX,
_connector32_pointsY, false );
    }
    if (!_publicOnly) {
      drawConnector(        _panel,        _g,        _connector34_pointsX,
_connector34_pointsY, false );
    }
    if (!_publicOnly) {
      drawConnector(        _panel,        _g,        _connector35_pointsX,
_connector35_pointsY, false );
    }
    if (!_publicOnly) {
      drawConnector(        _panel,        _g,        _connector36_pointsX,
_connector36_pointsY, false );
    }
    if (!_publicOnly) {
      drawConnector(        _panel,        _g,        _connector37_pointsX,
_connector37_pointsY, false );
    }
    if (!_publicOnly) {
      drawConnector(        _panel,        _g,        _connector38_pointsX,
_connector38_pointsY, false );
    }
  }

  @Override
  public boolean onClickModelAt( Panel panel, double x, double y, int
clickCount, boolean publicOnly ) {
    if( !publicOnly && modelElementContains(x, y, 800, 1500) ) {
      panel.addInspect( 800, 1500, this, "histLOS" );
      return true;
    }
    if ( network.onClickIconAt( x - 20, y - 1100, true ) ) {
      if ( clickCount == 2 ) {
        panel.browseEmbeddedObject( x, y, this, "network" );
      } else {
        panel.addInspect( x, y, this, "network" );
      }
      return true;
    }
    if ( arrive.onClickIconAt( x - 180, y - 1090, true ) ) {
      if ( clickCount == 2 ) {
        panel.browseEmbeddedObject( x, y, this, "arrive" );
      } else {
        panel.addInspect( x, y, this, "arrive" );
      }
      return true;
    }
    if ( networkEnter.onClickIconAt( x - 280, y - 1090, true ) ) {
      if ( clickCount == 2 ) {
        panel.browseEmbeddedObject( x, y, this, "networkEnter" );
      } else {
        panel.addInspect( x, y, this, "networkEnter" );
      }
      return true;
    }
    if ( gotoRegistration.onClickIconAt( x - 380, y - 1090, true ) ) {
      if ( clickCount == 2 ) {
        panel.browseEmbeddedObject( x, y, this, "gotoRegistration" );
      } else {
        panel.addInspect( x, y, this, "gotoRegistration" );
      }
      return true;
    }
    if ( queueAtReg.onClickIconAt( x - 480, y - 1090, true ) ) {
      if ( clickCount == 2 ) {
        panel.browseEmbeddedObject( x, y, this, "queueAtReg" );
      } else {
        panel.addInspect( x, y, this, "queueAtReg" );
      }
      return true;
    }
    if ( registration.onClickIconAt( x - 580, y - 1090, true ) ) {
      if ( clickCount == 2 ) {
        panel.browseEmbeddedObject( x, y, this, "registration" );
      } else {
        panel.addInspect( x, y, this, "registration" );
      }
      return true;
    }
    if ( gotoWaitingRoom1.onClickIconAt( x - 680, y - 1090, true ) ) {
      if ( clickCount == 2 ) {
        panel.browseEmbeddedObject( x, y, this, "gotoWaitingRoom1" );
      } else {
        panel.addInspect( x, y, this, "gotoWaitingRoom1" );
      }
      return true;
    }
    if ( gotoExit.onClickIconAt( x - 560, y - 1410, true ) ) {
      if ( clickCount == 2 ) {
        panel.browseEmbeddedObject( x, y, this, "gotoExit" );
      } else {
        panel.addInspect( x, y, this, "gotoExit" );
      }
```

```
    return true;
  }
  if ( networkExit.onClickIconAt( x - 660, y - 1410, true ) ) {
    if ( clickCount == 2 ) {
      panel.browseEmbeddedObject( x, y, this, "networkExit" );
    } else {
      panel.addInspect( x, y, this, "networkExit" );
    }
    return true;
  }
  if ( leave.onClickIconAt( x - 760, y - 1410, true ) ) {
    if ( clickCount == 2 ) {
      panel.browseEmbeddedObject( x, y, this, "leave" );
    } else {
      panel.addInspect( x, y, this, "leave" );
    }
    return true;
  }
  if ( triageRoom.onClickIconAt( x - 20, y - 1150, true ) ) {
    if ( clickCount == 2 ) {
      panel.browseEmbeddedObject( x, y, this, "triageRoom" );
    } else {
      panel.addInspect( x, y, this, "triageRoom" );
    }
    return true;
  }
  if ( nurse.onClickIconAt( x - 20, y - 1220, true ) ) {
    if ( clickCount == 2 ) {
      panel.browseEmbeddedObject( x, y, this, "nurse" );
    } else {
      panel.addInspect( x, y, this, "nurse" );
    }
    return true;
  }
  if ( seizeTriageRoom.onClickIconAt( x - 280, y - 1190, true ) ) {
    if ( clickCount == 2 ) {
      panel.browseEmbeddedObject( x, y, this, "seizeTriageRoom" );
    } else {
      panel.addInspect( x, y, this, "seizeTriageRoom" );
    }
    return true;
  }
  if ( gotoTriageRoom.onClickIconAt( x - 480, y - 1190, true ) ) {
    if ( clickCount == 2 ) {
      panel.browseEmbeddedObject( x, y, this, "gotoTriageRoom" );
    } else {
      panel.addInspect( x, y, this, "gotoTriageRoom" );
    }
    return true;
  }
  if ( triage.onClickIconAt( x - 590, y - 1190, true ) ) {
    if ( clickCount == 2 ) {
      panel.browseEmbeddedObject( x, y, this, "triage" );
    } else {
      panel.addInspect( x, y, this, "triage" );
    }
    return true;
  }
  if ( gotoWaitingRoom2.onClickIconAt( x - 280, y - 1290, true ) ) {
    if ( clickCount == 2 ) {
      panel.browseEmbeddedObject( x, y, this, "gotoWaitingRoom2" );
    } else {
      panel.addInspect( x, y, this, "gotoWaitingRoom2" );
    }
    return true;
  }
  if ( releaseTriageRoom.onClickIconAt( x - 680, y - 1190, true ) ) {
    if ( clickCount == 2 ) {
      panel.browseEmbeddedObject( x, y, this, "releaseTriageRoom" );
    } else {
      panel.addInspect( x, y, this, "releaseTriageRoom" );
    }
    return true;
  }
  if ( releaseNurse.onClickIconAt( x - 580, y - 1290, true ) ) {
    if ( clickCount == 2 ) {
      panel.browseEmbeddedObject( x, y, this, "releaseNurse" );
    } else {
      panel.addInspect( x, y, this, "releaseNurse" );
    }
    return true;
  }
  if ( seizeECRoom.onClickIconAt( x - 380, y - 1290, true ) ) {
    if ( clickCount == 2 ) {
      panel.browseEmbeddedObject( x, y, this, "seizeECRoom" );
    } else {
      panel.addInspect( x, y, this, "seizeECRoom" );
    }
    return true;
  }
  if ( ECRoom.onClickIconAt( x - 90, y - 1150, true ) ) {
    if ( clickCount == 2 ) {
      panel.browseEmbeddedObject( x, y, this, "ECRoom" );
    } else {
      panel.addInspect( x, y, this, "ECRoom" );
    }
    return true;
  }
  if ( gotoECRoom.onClickIconAt( x - 480, y - 1290, true ) ) {
    if ( clickCount == 2 ) {
      panel.browseEmbeddedObject( x, y, this, "gotoECRoom" );
    } else {
      panel.addInspect( x, y, this, "gotoECRoom" );
    }
    return true;
  }
  if ( releaseECRoom.onClickIconAt( x - 460, y - 1410, true ) ) {
    if ( clickCount == 2 ) {
      panel.browseEmbeddedObject( x, y, this, "releaseECRoom" );
    } else {
      panel.addInspect( x, y, this, "releaseECRoom" );
    }
    return true;
  }
  if ( PA.onClickIconAt( x - 90, y - 1220, true ) ) {
    if ( clickCount == 2 ) {
      panel.browseEmbeddedObject( x, y, this, "PA" );
    } else {
      panel.addInspect( x, y, this, "PA" );
    }
    return true;
  }
  if ( technician.onClickIconAt( x - 20, y - 1290, true ) ) {
    if ( clickCount == 2 ) {
      panel.browseEmbeddedObject( x, y, this, "technician" );
    } else {
      panel.addInspect( x, y, this, "technician" );
    }
    return true;
  }
```

```
  if ( XRay.onClickIconAt( x - 90, y - 1290, true ) ) {
   if ( clickCount == 2 ) {
    panel.browseEmbeddedObject( x, y, this, "XRay" );
   } else {
    panel.addInspect( x, y, this, "XRay" );
   }
   return true;
  }
  if ( releasePA.onClickIconAt( x - 360, y - 1410, true ) ) {
   if ( clickCount == 2 ) {
    panel.browseEmbeddedObject( x, y, this, "releasePA" );
   } else {
    panel.addInspect( x, y, this, "releasePA" );
   }
   return true;
  }
  if ( xRayProcess.onClickIconAt( x - 280, y - 1390, true ) ) {
   if ( clickCount == 2 ) {
    panel.browseEmbeddedObject( x, y, this, "xRayProcess" );
   } else {
    panel.addInspect( x, y, this, "xRayProcess" );
   }
   return true;
  }
  if ( USound.onClickIconAt( x - 20, y - 1360, true ) ) {
   if ( clickCount == 2 ) {
    panel.browseEmbeddedObject( x, y, this, "USound" );
   } else {
    panel.addInspect( x, y, this, "USound" );
   }
   return true;
  }
  if ( uSoundProcess.onClickIconAt( x - 280, y - 1340, true ) ) {
   if ( clickCount == 2 ) {
    panel.browseEmbeddedObject( x, y, this, "uSoundProcess" );
   } else {
    panel.addInspect( x, y, this, "uSoundProcess" );
   }
   return true;
  }
  if ( selectProcess.onClickIconAt( x - 220, y - 1410, true ) ) {
   if ( clickCount == 2 ) {
    panel.browseEmbeddedObject( x, y, this, "selectProcess" );
   } else {
    panel.addInspect( x, y, this, "selectProcess" );
   }
   return true;
  }
  if ( clock.onClickIconAt( x - 120, y - 100, true ) ) {
   if ( clickCount == 2 ) {
    panel.browseEmbeddedObject( x, y, this, "clock" );
   } else {
    panel.addInspect( x, y, this, "clock" );
   }
   return true;
  }
  if ( callNurse.onClickIconAt( x - 380, y - 1190, true ) ) {
   if ( clickCount == 2 ) {
    panel.browseEmbeddedObject( x, y, this, "callNurse" );
   } else {
    panel.addInspect( x, y, this, "callNurse" );
   }
   return true;
  }
  if ( callPA.onClickIconAt( x - 680, y - 1290, true ) ) {
   if ( clickCount == 2 ) {
    panel.browseEmbeddedObject( x, y, this, "callPA" );
   } else {
    panel.addInspect( x, y, this, "callPA" );
   }
   return true;
  }
  return false;
 }

 /**
  * Constructor
  */
 public Main( Engine engine, ActiveObject owner,
ActiveObjectCollection<? extends Main> collection ) {
  super( engine, owner, collection );
 }

 @Override
 public void create() {
  // Creating embedded object instances
  network = instantiate_network_xjal();
  arrive = instantiate_arrive_xjal();
  networkEnter = instantiate_networkEnter_xjal();
  gotoRegistration = instantiate_gotoRegistration_xjal();
  queueAtReg = instantiate_queueAtReg_xjal();
  registration = instantiate_registration_xjal();
  gotoWaitingRoom1 = instantiate_gotoWaitingRoom1_xjal();
  gotoExit = instantiate_gotoExit_xjal();
  networkExit = instantiate_networkExit_xjal();
  leave = instantiate_leave_xjal();
  triageRoom = instantiate_triageRoom_xjal();
  nurse = instantiate_nurse_xjal();
  seizeTriageRoom = instantiate_seizeTriageRoom_xjal();
  gotoTriageRoom = instantiate_gotoTriageRoom_xjal();
  triage = instantiate_triage_xjal();
  gotoWaitingRoom2 = instantiate_gotoWaitingRoom2_xjal();
  releaseTriageRoom = instantiate_releaseTriageRoom_xjal();
  releaseNurse = instantiate_releaseNurse_xjal();
  seizeECRoom = instantiate_seizeECRoom_xjal();
  ECRoom = instantiate_ECRoom_xjal();
  gotoECRoom = instantiate_gotoECRoom_xjal();
  releaseECRoom = instantiate_releaseECRoom_xjal();
  PA = instantiate_PA_xjal();
  technician = instantiate_technician_xjal();
  XRay = instantiate_XRay_xjal();
  releasePA = instantiate_releasePA_xjal();
  xRayProcess = instantiate_xRayProcess_xjal();
  USound = instantiate_USound_xjal();
  uSoundProcess = instantiate_uSoundProcess_xjal();
  selectProcess = instantiate_selectProcess_xjal();
  clock = instantiate_clock_xjal();
  callNurse = instantiate_callNurse_xjal();
  callPA = instantiate_callPA_xjal();
  // Dynamic initialization of persistent elements
  {
  List<DataItem> _items = new ArrayList<DataItem>( 7 );
  _items.add( new DataItem() {
   @Override
   public void update() {
    setValue( _chart_DataItem0Value() );
   }
  });
  _items.add( new DataItem() {
   @Override
   public void update() {
```

```
      setValue( _chart_DataItem1Value() );
     }
    });
    _items.add( new DataItem() {
     @Override
     public void update() {
      setValue( _chart_DataItem2Value() );
     }
    });
    _items.add( new DataItem() {
     @Override
     public void update() {
      setValue( _chart_DataItem3Value() );
     }
    });
    _items.add( new DataItem() {
     @Override
     public void update() {
      setValue( _chart_DataItem4Value() );
     }
    });
    _items.add( new DataItem() {
     @Override
     public void update() {
      setValue( _chart_DataItem5Value() );
     }
    });
    _items.add( new DataItem() {
     @Override
     public void update() {
      setValue( _chart_DataItem6Value() );
     }
    });
    List<String> _titles = new ArrayList<String>( 7 );
    _titles.add( "Triage R." );
    _titles.add( "EC R." );
    _titles.add( "X-Ray" );
    _titles.add( "U. Snd" );
    _titles.add( "Nurses" );
    _titles.add( "PAs" );
    _titles.add( "Techs" );
    List<Color> _colors = new ArrayList<Color>( 7 );
    _colors.add( gold );
    _colors.add( chocolate );
    _colors.add( blueViolet );
    _colors.add( navy );
    _colors.add( lightSteelBlue );
    _colors.add( yellowGreen );
    _colors.add( royalBlue );
    chart = new BarChart(
                        Main.this, true, 10, 460,
                                    640, 140,
          null, null,
          40, 20,
                                    580, 70, null, null, black,
          40, Chart.SOUTH,
          Chart.NORTH, Chart.SCALE_FIXED,

0

          ,
1
          , 0.32,
          Chart.GRID_DEFAULT,
          darkGray, darkGray, _items, _titles, _colors
    );
    }
    {
    HistogramData _item;
    List<HistogramData> _items = new ArrayList<HistogramData>( 1 );
    _item =
histLOS
;
    _items.add( _item );
    List<String> _titles = new ArrayList<String>( 1 );
    _titles.add( "Length of stay" );
    List<Histogram.Appearance>  _appearances  =  new
ArrayList<Histogram.Appearance>( 1 );
    _appearances.add( new Histogram.Appearance( crimson, goldenRod,
steelBlue, mediumOrchid, 0, slateBlue ) );
    chart1 = new Histogram(
                                Main.this, true, 670, 470,
                                           210, 100,
          null, null,
          40, 10,
                                           150, 70, null, null, black,
          10, Chart.NONE,
          Chart.GRID_DEFAULT, Chart.GRID_DEFAULT,
          darkGray, black,
          true, false, false, 0.7,
          _items, _titles, _appearances
    );
    }
    {
                          group1  =  new  ShapeGroup(  Main.this,
true,495, 115, 0.0

                    ,rectXRayFloor
                    ,arc
                    ,arc2
                    ,arc1
                    ,oval6
    ) {

     @Override
     public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
       _group1_SetDynamicParams( this );
       super.draw( panel, graphics, xform, publicOnly );
     }
    };
    }
    {
                          group  =  new  ShapeGroup(  Main.this,
true,200, 50, 0.0

                    ,waitingRoom
                    ,rectangle3
                    ,rectangle4
                    ,rectangle6
                    ,rectangle7
                    ,rectangle8
                    ,rectangle9
                    ,rectangle10
                    ,xRayRoom
                    ,USoundStorage
                    ,techHomeRoom
                    ,nurseHomeRoom
                    ,doctorHomeRoom
                    ,entryDoor
                    ,exitDoor
                    ,rectangle11
                    ,rectangle12
```

```
                ,rectangle13
                ,rectangle15
                ,rectangle16
                ,rectangle17
                ,rectangle18
                ,rectangle19
                ,rectangle20
                ,rectangle21
                ,rectangle22
                ,rectangle23
                ,rectangle24
                ,polyline
                ,line
                ,line1
                ,line4
                ,line5
                ,polyline1
                ,line6
                ,line7
                ,line8
                ,line9
                ,polyline2
                ,polyline3
                ,line10
                ,line32
                ,rectangle27
                ,line39
                ,rectangle29
                ,line40
                ,registrationDesk
                ,rectangle
                ,line47
  );
  }
  {
                shapeBusy = new ShapeGroup( Main.this,
true,135, -20, 0.0

                ,oval1
                ,oval
  );
  }
  {
                shapeIdle = new ShapeGroup( Main.this,
true,150, -20, 0.0

                ,oval2
                ,oval3
  );
  }
  {
                shapeUSound = new ShapeGroup( Main.this,
true,175, -20, 0.0

                ,line50
                ,oval5
                ,rectangle32
                ,polyline19
                ,oval4
  );
  }
  presentation = new ShapeGroup( Main.this, true, 0, 0, 0, radio,
radio1, radio2, radio3, group1, rectangle41, rectangle38, rectangle37,
rectangle36, rectangle35, rectangle33, polyline18, polyline4, polyline5,
polyline6, line11, line12, line14, line15, line16, line17, line18, line19,
line20, line21, line22, line23, line24, line26, polyline7, polyline8,
polyline9, line27, line29, line30, line31, polyline10, polyline11,
rectangle25, line33, line34, rectangle26, text, text1, text2, text7, text8,
text9, line35, polyline12, polyline13, line36, polyline14, polyline15,
line37, line38, polyQueueAtReg, polyTriageRooms, polyline16,
polyECRooms, shapePatient, shapeNurse, shapePA, shapeTech,
polyline17, line41, line42, curve, line43, line44, line45, curve1, curve2,
curve3, curve4, curve5, curve6, curve7, NurseHomePath, PAHomePath,
TechHomePath, line46, line49, line48, rectangle31, rectangle30, group,
shapeBusy, shapeIdle, XRayHomePath, USoundHomePath,
shapeUSound, text15, text16, line51, line52, text17, rectangle34, text18,
line53, line54, text19, text20, polyline20, polyline21, text21, text22,
line55, line56, rectangle39, text24, polyline22, rectangle40, text25,
polyline23, text26, text27, text28, text29, text30, text31, rectangle42,
rectangle43, text32, polyline24, text6, text10, text11, text12, text14,
text3, text5, text13, text23, chart, chart1 );
  icon = new ShapeGroup( Main.this, true, 0, 0, 0
);
  // Creating non-replicated embedded objects
  setupParameters_network_xjal( network );
  create_network_xjal( network );
  setupParameters_arrive_xjal( arrive );
  create_arrive_xjal( arrive );
  setupParameters_networkEnter_xjal( networkEnter );
  create_networkEnter_xjal( networkEnter );
  setupParameters_gotoRegistration_xjal( gotoRegistration );
  create_gotoRegistration_xjal( gotoRegistration );
  setupParameters_queueAtReg_xjal( queueAtReg );
  create_queueAtReg_xjal( queueAtReg );
  setupParameters_registration_xjal( registration );
  create_registration_xjal( registration );
  setupParameters_gotoWaitingRoom1_xjal( gotoWaitingRoom1 );
  create_gotoWaitingRoom1_xjal( gotoWaitingRoom1 );
  setupParameters_gotoExit_xjal( gotoExit );
  create_gotoExit_xjal( gotoExit );
  setupParameters_networkExit_xjal( networkExit );
  create_networkExit_xjal( networkExit );
  setupParameters_leave_xjal( leave );
  create_leave_xjal( leave );
  setupParameters_triageRoom_xjal( triageRoom );
  create_triageRoom_xjal( triageRoom );
  setupParameters_nurse_xjal( nurse );
  create_nurse_xjal( nurse );
  setupParameters_seizeTriageRoom_xjal( seizeTriageRoom );
  create_seizeTriageRoom_xjal( seizeTriageRoom );
  setupParameters_gotoTriageRoom_xjal( gotoTriageRoom );
  create_gotoTriageRoom_xjal( gotoTriageRoom );
  setupParameters_triage_xjal( triage );
  create_triage_xjal( triage );
  setupParameters_gotoWaitingRoom2_xjal( gotoWaitingRoom2 );
  create_gotoWaitingRoom2_xjal( gotoWaitingRoom2 );
  setupParameters_releaseTriageRoom_xjal( releaseTriageRoom );
  create_releaseTriageRoom_xjal( releaseTriageRoom );
  setupParameters_releaseNurse_xjal( releaseNurse );
  create_releaseNurse_xjal( releaseNurse );
  setupParameters_seizeECRoom_xjal( seizeECRoom );
  create_seizeECRoom_xjal( seizeECRoom );
  setupParameters_ECRoom_xjal( ECRoom );
  create_ECRoom_xjal( ECRoom );
  setupParameters_gotoECRoom_xjal( gotoECRoom );
  create_gotoECRoom_xjal( gotoECRoom );
  setupParameters_releaseECRoom_xjal( releaseECRoom );
  create_releaseECRoom_xjal( releaseECRoom );
  setupParameters_PA_xjal( PA );
  create_PA_xjal( PA );
  setupParameters_technician_xjal( technician );
  create_technician_xjal( technician );
  setupParameters_XRay_xjal( XRay );
  create_XRay_xjal( XRay );
  setupParameters_releasePA_xjal( releasePA );
  create_releasePA_xjal( releasePA );
  setupParameters_xRayProcess_xjal( xRayProcess );
```

```
    create_xRayProcess_xjal( xRayProcess );
    setupParameters_USound_xjal( USound );
    create_USound_xjal( USound );
    setupParameters_uSoundProcess_xjal( uSoundProcess );
    create_uSoundProcess_xjal( uSoundProcess );
    setupParameters_selectProcess_xjal( selectProcess );
    create_selectProcess_xjal( selectProcess );
    setupParameters_clock_xjal( clock );
    create_clock_xjal( clock );
    setupParameters_callNurse_xjal( callNurse );
    create_callNurse_xjal( callNurse );
    setupParameters_callPA_xjal( callPA );
    create_callPA_xjal( callPA );
                // Port connectors with non-replicated objects
    arrive.out.connect( networkEnter.in ); // connector
    networkEnter.out.connect( gotoRegistration.in ); // connector1
    gotoRegistration.out.connect( queueAtReg.in ); // connector2
    queueAtReg.out.connect( registration.in ); // connector3
    registration.out.connect( gotoWaitingRoom1.in ); // connector4
    gotoExit.out.connect( networkExit.in ); // connector5
    networkExit.out.connect( leave.in ); // connector6
    network.access.connect( triageRoom.port ); // connector8
    network.access.connect( nurse.port ); // connector9
    gotoWaitingRoom1.out.connect( seizeTriageRoom.in ); // connector10
    seizeTriageRoom.out.connect( callNurse.in ); // connector11
    callNurse.out.connect( gotoTriageRoom.in ); // connector12
    gotoTriageRoom.out.connect( triage.in ); // connector13
    triage.out.connect( releaseTriageRoom.in ); // connector14
    releaseTriageRoom.out.connect( gotoWaitingRoom2.in ); //
connector15
    gotoWaitingRoom2.out.connect( seizeECRoom.in ); // connector16
    network.access.connect( ECRoom.port ); // connector17
    seizeECRoom.out.connect( gotoECRoom.in ); // connector18
    gotoECRoom.out.connect( releaseNurse.in ); // connector19
    releaseECRoom.out.connect( gotoExit.in ); // connector20
    network.access.connect( PA.port ); // connector22
    network.access.connect( technician.port ); // connector23
    releaseNurse.out.connect( callPA.in ); // connector24
    callPA.out.connect( selectProcess.in ); // connector25
    network.access.connect( XRay.port ); // connector27
    releasePA.out.connect( releaseECRoom.in ); // connector32
    network.access.connect( USound.port ); // connector34
    selectProcess.outT.connect( uSoundProcess.in ); // connector35
    selectProcess.outF.connect( xRayProcess.in ); // connector36
    uSoundProcess.out.connect( releasePA.in ); // connector37
    xRayProcess.out.connect( releasePA.in ); // connector38
    // Creating replicated embedded objects
    assignInitialConditions();
    radio.setValueToDefault();
    radio1.setValueToDefault();
    radio2.setValueToDefault();
    radio3.setValueToDefault();
    onCreate();
  }

  @Override
  public void start() {
    _histLOS_autoUpdateEvent_xjal.start();
    _chart_autoUpdateEvent_xjal.start();
    network.start();
    arrive.start();
    networkEnter.start();
    gotoRegistration.start();
    queueAtReg.start();
    registration.start();
    gotoWaitingRoom1.start();
    gotoExit.start();
    networkExit.start();
    leave.start();
    triageRoom.start();
    nurse.start();
    seizeTriageRoom.start();
    gotoTriageRoom.start();
    triage.start();
    gotoWaitingRoom2.start();
    releaseTriageRoom.start();
    releaseNurse.start();
    seizeECRoom.start();
    ECRoom.start();
    gotoECRoom.start();
    releaseECRoom.start();
    PA.start();
    technician.start();
    XRay.start();
    releasePA.start();
    xRayProcess.start();
    USound.start();
    uSoundProcess.start();
    selectProcess.start();
    clock.start();
    callNurse.start();
    callPA.start();
    onStartup();
  }


  public List<Object> getEmbeddedObjects() {
    LinkedList<Object> list = new LinkedList<Object>();
    list.add( network );
    list.add( arrive );
    list.add( networkEnter );
    list.add( gotoRegistration );
    list.add( queueAtReg );
    list.add( registration );
    list.add( gotoWaitingRoom1 );
    list.add( gotoExit );
    list.add( networkExit );
    list.add( leave );
    list.add( triageRoom );
    list.add( nurse );
    list.add( seizeTriageRoom );
    list.add( gotoTriageRoom );
    list.add( triage );
    list.add( gotoWaitingRoom2 );
    list.add( releaseTriageRoom );
    list.add( releaseNurse );
    list.add( seizeECRoom );
    list.add( ECRoom );
    list.add( gotoECRoom );
    list.add( releaseECRoom );
    list.add( PA );
    list.add( technician );
    list.add( XRay );
    list.add( releasePA );
    list.add( xRayProcess );
    list.add( USound );
    list.add( uSoundProcess );
    list.add( selectProcess );
    list.add( clock );
    list.add( callNurse );
    list.add( callPA );
```

```
    return list;
  }

  public void onDestroy() {
    super.onDestroy();
    _histLOS_autoUpdateEvent_xjal.onDestroy();
    _chart_autoUpdateEvent_xjal.onDestroy();
    network.onDestroy();
    arrive.onDestroy();
    networkEnter.onDestroy();
    gotoRegistration.onDestroy();
    queueAtReg.onDestroy();
    registration.onDestroy();
    gotoWaitingRoom1.onDestroy();
    gotoExit.onDestroy();
    networkExit.onDestroy();
    leave.onDestroy();
    triageRoom.onDestroy();
    nurse.onDestroy();
    seizeTriageRoom.onDestroy();
    gotoTriageRoom.onDestroy();
    triage.onDestroy();
    gotoWaitingRoom2.onDestroy();
    releaseTriageRoom.onDestroy();
    releaseNurse.onDestroy();
    seizeECRoom.onDestroy();
    ECRoom.onDestroy();
    gotoECRoom.onDestroy();
    releaseECRoom.onDestroy();
    PA.onDestroy();
    technician.onDestroy();
    XRay.onDestroy();
    releasePA.onDestroy();
    xRayProcess.onDestroy();
    USound.onDestroy();
    uSoundProcess.onDestroy();
    selectProcess.onDestroy();
    clock.onDestroy();
    callNurse.onDestroy();
    callPA.onDestroy();
  }
}
```

USoundProcess.java

```
package emergency_department;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import com.xj.anylogic.libraries.enterprise.*;
import java.awt.geom.Arc2D;

public class USoundProcess extends ActiveObject
{
  // Plain Variables

  public Main  ed;

  // Collection Variables

  // Flow/Auxiliary Variables

  // Stock Variables


  // Embedded Objects

  public Delay<Entity > doUSound;
  public NetworkSeize<Entity > seizeTechUSound;
  public NetworkSendTo<Entity > sendTechToUS;
  public NetworkSendTo<Entity > bringUSound;
  public NetworkSendTo<Entity > returnUSound;
  public NetworkRelease<Entity > releaseTechUSound;

  public String getNameOf( ActiveObject ao ) {
    if ( ao == doUSound ) return "doUSound";
    if ( ao == seizeTechUSound ) return "seizeTechUSound";
    if ( ao == sendTechToUS ) return "sendTechToUS";
    if ( ao == bringUSound ) return "bringUSound";
    if ( ao == returnUSound ) return "returnUSound";
    if ( ao == releaseTechUSound ) return "releaseTechUSound";
    return null;
  }


  public String getNameOf( ActiveObjectCollection<?> aolist ) {
    return null;
  }
```

```
 /**
  * Creates an embedded object instance<br>
  * <i>This method should not be called by user</i>
  */
 private Delay<
Entity > instantiate_doUSound_xjal() {Delay<
Entity > object = new Delay<
Entity >( getEngine(), this, null ) {
    @Override
    public double delayTime( Entity entity ) {
      return _doUSound_delayTime_xjal( entity );
    }
  };

  return object;
 }

 /**
  * Setups parameters of an embedded object instance<br>
  * This method should not be called by user
  */
 private void setupParameters_doUSound_xjal(Delay<
Entity > object ) {
  object.delayTimeDefinedByPath                                    =
object._delayTimeDefinedByPath_DefaultValue_xjal();
  object.capacity = object._capacity_DefaultValue_xjal();
  object.maximumCapacity =
true
;
  object.animationGuide = object._animationGuide_DefaultValue_xjal();
  object.animationType = object._animationType_DefaultValue_xjal();
  object.animationForward                                          =
object._animationForward_DefaultValue_xjal();
  object.enableStats = object._enableStats_DefaultValue_xjal();
 }

 /**
  * Setups an embedded object instance<br>
  * This method should not be called by user
  */
 private void create_doUSound_xjal(Delay<
Entity > object ) {
  object.create();

 }
 /**
  * Creates an embedded object instance<br>
  * <i>This method should not be called by user</i>
  */
 private NetworkSeize<
Entity > instantiate_seizeTechUSound_xjal() {NetworkSeize<
Entity > object = new NetworkSeize<
Entity >( getEngine(), this, null ) {
    @Override
    public NetworkResourcePool[] resources( Entity entity ) {
      return _seizeTechUSound_resources_xjal( entity );
    }
  };

  return object;
 }

 /**
  * Setups parameters of an embedded object instance<br>
  * This method should not be called by user
  */
 private void setupParameters_seizeTechUSound_xjal(NetworkSeize<
Entity > object ) {
  object.capacity = object._capacity_DefaultValue_xjal();
  object.maximumCapacity =
true
;
  object.enableTimeout = object._enableTimeout_DefaultValue_xjal();
  object.enablePreemption                                          =
object._enablePreemption_DefaultValue_xjal();
  object.sendResources = object._sendResources_DefaultValue_xjal();
  object.destinationType                                           =
object._destinationType_DefaultValue_xjal();
  object.attachResources                                           =
object._attachResources_DefaultValue_xjal();
  object.animationGuide = object._animationGuide_DefaultValue_xjal();
  object.animationType = object._animationType_DefaultValue_xjal();
  object.animationForward                                          =
object._animationForward_DefaultValue_xjal();
  object.enableStats = object._enableStats_DefaultValue_xjal();
 }

 /**
  * Setups an embedded object instance<br>
  * This method should not be called by user
  */
 private void create_seizeTechUSound_xjal(NetworkSeize<
Entity > object ) {
  object.create();

 }
 /**
  * Creates an embedded object instance<br>
  * <i>This method should not be called by user</i>
  */
 private NetworkSendTo<
Entity > instantiate_sendTechToUS_xjal() {NetworkSendTo<
Entity > object = new NetworkSendTo<
Entity >( getEngine(), this, null ) {
    @Override
    public NetworkResourcePool[] resources( Entity entity ) {
      return _sendTechToUS_resources_xjal( entity );
    }
    @Override
    public NetworkResourcePool destinationResource(  ) {
      return _sendTechToUS_destinationResource_xjal(  );
    }
  };

  return object;
 }

 /**
  * Setups parameters of an embedded object instance<br>
  * This method should not be called by user
  */
 private void setupParameters_sendTechToUS_xjal(NetworkSendTo<
Entity > object ) {
  object.destinationType =
NetworkSendTo.DEST_RESOURCE
;
 }

 /**
  * Setups an embedded object instance<br>
  * This method should not be called by user
  */
 private void create_sendTechToUS_xjal(NetworkSendTo<
```

```
Entity > object ) {
   object.create();

 }
 /**
  * Creates an embedded object instance<br>
  * <i>This method should not be called by user</i>
  */
 private NetworkSendTo<
Entity > instantiate_bringUSound_xjal() {NetworkSendTo<
Entity > object = new NetworkSendTo<
Entity >( getEngine(), this, null ) {
     @Override
     public NetworkResourcePool[] resources( Entity entity ) {
       return _bringUSound_resources_xjal( entity );
     }
   };

   return object;
 }

 /**
  * Setups parameters of an embedded object instance<br>
  * This method should not be called by user
  */
 private void setupParameters_bringUSound_xjal(NetworkSendTo<
Entity > object ) {
   object.destinationType =
NetworkSendTo.DEST_ENTITY
;
 }

 /**
  * Setups an embedded object instance<br>
  * This method should not be called by user
  */
 private void create_bringUSound_xjal(NetworkSendTo<
Entity > object ) {
   object.create();

 }
 /**
  * Creates an embedded object instance<br>
  * <i>This method should not be called by user</i>
  */
 private NetworkSendTo<
Entity > instantiate_returnUSound_xjal() {NetworkSendTo<
Entity > object = new NetworkSendTo<
Entity >( getEngine(), this, null ) {
     @Override
     public NetworkResourcePool[] resources( Entity entity ) {
       return _returnUSound_resources_xjal( entity );
     }
     @Override
     public NetworkResourcePool destinationResource( ) {
       return _returnUSound_destinationResource_xjal( );
     }
   };

   return object;
 }

 /**
  * Setups parameters of an embedded object instance<br>
  * This method should not be called by user
  */
 private void setupParameters_returnUSound_xjal(NetworkSendTo<
Entity > object ) {
   object.destinationType =
NetworkSendTo.DEST_RESOURCE_HOME
;
 }

 /**
  * Setups an embedded object instance<br>
  * This method should not be called by user
  */
 private void create_returnUSound_xjal(NetworkSendTo<
Entity > object ) {
   object.create();

 }
 /**
  * Creates an embedded object instance<br>
  * <i>This method should not be called by user</i>
  */
 private NetworkRelease<
Entity > instantiate_releaseTechUSound_xjal() {NetworkRelease<
Entity > object = new NetworkRelease<
Entity >( getEngine(), this, null ) {
     @Override
     public NetworkResourcePool[] resources( Entity entity ) {
       return _releaseTechUSound_resources_xjal( entity );
     }
   };

   return object;
 }

 /**
  * Setups parameters of an embedded object instance<br>
  * This method should not be called by user
  */
 private void setupParameters_releaseTechUSound_xjal(NetworkRelease<
Entity > object ) {
   object.releaseAll = object._releaseAll_DefaultValue_xjal();
   object.movingGoHome =
object._movingGoHome_DefaultValue_xjal();
 }

 /**
  * Setups an embedded object instance<br>
  * This method should not be called by user
  */
 private void create_releaseTechUSound_xjal(NetworkRelease<
Entity > object ) {
   object.create();

 }

 public double _doUSound_delayTime_xjal( Entity entity ) {
   return
triangular( 10, 25, 30 )
;
 }
 public NetworkResourcePool[] _seizeTechUSound_resources_xjal(
Entity entity ) {
   return new NetworkResourcePool[]
{ ed.technician, ed.USound }
;
```

```
  }
  public NetworkResourcePool[] _sendTechToUS_resources_xjal( Entity entity ) {
    return new NetworkResourcePool[]
{ ed.technician }
;
  }
  public NetworkResourcePool _sendTechToUS_destinationResource_xjal( ) {
    return
ed.USound
;
  }
  public NetworkResourcePool[] _bringUSound_resources_xjal( Entity entity ) {
    return new NetworkResourcePool[]
{ ed.technician, ed.USound }
;
  }
  public NetworkResourcePool[] _returnUSound_resources_xjal( Entity entity ) {
    return new NetworkResourcePool[]
{ ed.technician, ed.USound }
;
  }
  public NetworkResourcePool _returnUSound_destinationResource_xjal( ) {
    return
ed.USound
;
  }
  public NetworkResourcePool[] _releaseTechUSound_resources_xjal( Entity entity ) {
    return new NetworkResourcePool[]
{ ed.technician, ed.USound }
;
  }
  static final Font _text_Font = new Font("SansSerif", 1, 18 );
  static final Font _text1_Font = new Font("SansSerif", 0, 18 );
  static final Font _text2_Font = _text1_Font;
  static final int _line = 1;
  static final int _oval = 2;
  static final int _text = 3;
  static final int _rectangle = 4;
  static final int _line1 = 5;
  static final int _text1 = 6;
  static final int _line2 = 7;
  static final int _text2 = 8;

  /**
   * Top-level presentation group id
   */
  static final int _presentation = 0;

  /**
   * Top-level icon group id
   */
  static final int _icon = -1;


  @Override
  public boolean onShapeClick( int _shape, int index, double clickx, double clicky ){
    switch( _shape ){
      case _rectangle:
        if (true) {
getPresentation().setPresentable( getOwner() );
getPresentation().getPanel().setOffsets( 0, -1000 );
        }
        break;
    default: return super.onShapeClick( _shape, index, clickx, clicky );
              }
              return false;
  }

  ShapeLine line;
  ShapeOval oval;
  ShapeText text;
  ShapeRectangle rectangle;
  ShapeLine line1;
  ShapeText text1;
  ShapeLine line2;
  ShapeText text2;

  // Static initialization of persistent elements
  {
    line = new ShapeLine(
                          true,50, 100,
      black,
                          40,          0,
                          1, LINE_STYLE_SOLID
    );
    oval = new ShapeOval(
      true,70, 100, 0.0,
      black, lime,
                10, 10,
                1, LINE_STYLE_SOLID
    );
    text = new ShapeText(
       true,70, 90, 0.0,
       black,"U",
      _text_Font, ALIGNMENT_CENTER
    );
    rectangle = new ShapeRectangle(
      false,15, 10, 0.0,
          null, null,
                                        50, 30,
                                        1, LINE_STYLE_SOLID
    ) {

      @Override
      public boolean onClick( double clickx, double clicky ) {
        return onShapeClick( _rectangle, 0, clickx, clicky );
      }
    };
    line1 = new ShapeLine(
                          false,20, 34,
      royalBlue,
                          40,          0,
                          1, LINE_STYLE_SOLID
    );
    text1 = new ShapeText(
      false,80, 15, 0.0,
      black,"Ultra Sound Process",
      _text1_Font, ALIGNMENT_LEFT
    );
    line2 = new ShapeLine(
                          false,70, 10,
      black,
                          0,           30,
                          1, LINE_STYLE_SOLID
```

```
    );
    text2 = new ShapeText(
        false,20, 15, 0.0,
        royalBlue,"Back",
        _text2_Font, ALIGNMENT_LEFT
    );
  }
  ShapeGroup presentation;
  ShapeGroup icon;

  @Override
  public Object getPersistentShape( int _shape ) {
    switch(_shape){
      case _presentation: return presentation;
      case _icon: return icon;

      case _line: return line;
      case _oval: return oval;
      case _text: return text;
      case _rectangle: return rectangle;
      case _line1: return line1;
      case _text1: return text1;
      case _line2: return line2;
      case _text2: return text2;
      default: return null;
    }
  }

  static final int[] _connector_pointsX = {
    400,  450 };
  static final int[] _connector_pointsY = {
    200,200 };

  static final int[] _connector3_pointsX = {
    300,  350 };
  static final int[] _connector3_pointsY = {
    200,200 };

  static final int[] _connector8_pointsX = {
    100,  150 };
  static final int[] _connector8_pointsY = {
    200,200 };

  static final int[] _connector9_pointsX = {
    200,  250 };
  static final int[] _connector9_pointsY = {
    200,200 };

  static final int[] _connector10_pointsX = {
    500,  550 };
  static final int[] _connector10_pointsY = {
    200,200 };


  @Override
  public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
      drawPort( _panel, _g, 50, 100, -40, 0, null, in );
      drawPort( _panel, _g, 90, 100, 10, 0, null, out );
    // Embedded object "doUSound"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 300 , 150 , 50,
20, "doUSound", this.doUSound );
    }
    // Embedded object "seizeTechUSound"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 0 , 150 , 30, 20,
"seizeTechUSound", this.seizeTechUSound );
    }
    // Embedded object "sendTechToUS"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 100 , 150 , 30,
20, "sendTechToUS", this.sendTechToUS );
    }
    // Embedded object "bringUSound"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 200 , 150 , 40,
20, "bringUSound", this.bringUSound );
    }
    // Embedded object "returnUSound"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 400 , 150 , 40,
20, "returnUSound", this.returnUSound );
    }
    // Embedded object "releaseTechUSound"
    if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 500 , 150 , 20,
20, "releaseTechUSound", this.releaseTechUSound );
    }
    if (!_publicOnly) {
      drawConnector( _panel, _g, _connector_pointsX,
_connector_pointsY, false );
      }
    if (!_publicOnly) {
      drawConnector( _panel, _g, _connector3_pointsX,
_connector3_pointsY, false );
      }
    if (!_publicOnly) {
      drawConnector( _panel, _g, _connector8_pointsX,
_connector8_pointsY, false );
      }
    if (!_publicOnly) {
      drawConnector( _panel, _g, _connector9_pointsX,
_connector9_pointsY, false );
      }
    if (!_publicOnly) {
      drawConnector( _panel, _g, _connector10_pointsX,
_connector10_pointsY, false );
      }
  }


  @Override
  public boolean onClickModelAt( Panel panel, double x, double y, int
clickCount, boolean publicOnly ) {
    if( modelElementContains(x, y, 50, 100) ) {
      panel.addInspect( 50, 100, this, "in" );
      return true;
    }
    if( modelElementContains(x, y, 90, 100) ) {
      panel.addInspect( 90, 100, this, "out" );
      return true;
    }
    if ( doUSound.onClickIconAt( x - 300, y - 150, true ) ) {
      if ( clickCount == 2 ) {
        panel.browseEmbeddedObject( x, y, this, "doUSound" );
      } else {
        panel.addInspect( x, y, this, "doUSound" );
      }
      return true;
    }
    if ( seizeTechUSound.onClickIconAt( x - 0, y - 150, true ) ) {
      if ( clickCount == 2 ) {
        panel.browseEmbeddedObject( x, y, this, "seizeTechUSound" );
      } else {
        panel.addInspect( x, y, this, "seizeTechUSound" );
```

```
    }
    return true;
   }
   if ( sendTechToUS.onClickIconAt( x - 100, y - 150, true ) ) {
    if ( clickCount == 2 ) {
     panel.browseEmbeddedObject( x, y, this, "sendTechToUS" );
    } else {
     panel.addInspect( x, y, this, "sendTechToUS" );
    }
    return true;
   }
   if ( bringUSound.onClickIconAt( x - 200, y - 150, true ) ) {
    if ( clickCount == 2 ) {
     panel.browseEmbeddedObject( x, y, this, "bringUSound" );
    } else {
     panel.addInspect( x, y, this, "bringUSound" );
    }
    return true;
   }
   if ( returnUSound.onClickIconAt( x - 400, y - 150, true ) ) {
    if ( clickCount == 2 ) {
     panel.browseEmbeddedObject( x, y, this, "returnUSound" );
    } else {
     panel.addInspect( x, y, this, "returnUSound" );
    }
    return true;
   }
   if ( releaseTechUSound.onClickIconAt( x - 500, y - 150, true ) ) {
    if ( clickCount == 2 ) {
     panel.browseEmbeddedObject( x, y, this, "releaseTechUSound" );
    } else {
     panel.addInspect( x, y, this, "releaseTechUSound" );
    }
    return true;
   }
   return false;
  }

  // Ports

  public Port<
Object
,
Object > in = new Port< Object, Object >( this );
  public Port<
Object
,
Object > out = new Port< Object, Object >( this );

  public String getNameOf( Port<?, ?> _p ) {
   if( _p == this.in ) return "in";
   if( _p == this.out ) return "out";
   return null;
  }

  /**
   * Constructor
   */
  public USoundProcess( Engine engine, ActiveObject owner,
ActiveObjectCollection<? extends USoundProcess> collection ) {
   super( engine, owner, collection );
  }

  @Override
  public void create() {
   // Creating embedded object instances
   doUSound = instantiate_doUSound_xjal();
   seizeTechUSound = instantiate_seizeTechUSound_xjal();
   sendTechToUS = instantiate_sendTechToUS_xjal();
   bringUSound = instantiate_bringUSound_xjal();
   returnUSound = instantiate_returnUSound_xjal();
   releaseTechUSound = instantiate_releaseTechUSound_xjal();
   // Assigning initial values for plain variables
   ed =
get_Main()
;
   // Dynamic initialization of persistent elements
   presentation = new ShapeGroup( USoundProcess.this, true, 0, 0, 0,
rectangle, line1, text1, line2, text2 );
   icon = new ShapeGroup( USoundProcess.this, true, 0, 0, 0
     ,line
     ,oval
     ,text          );
   // Creating non-replicated embedded objects
   setupParameters_doUSound_xjal( doUSound );
   create_doUSound_xjal( doUSound );
   setupParameters_seizeTechUSound_xjal( seizeTechUSound );
   create_seizeTechUSound_xjal( seizeTechUSound );
   setupParameters_sendTechToUS_xjal( sendTechToUS );
   create_sendTechToUS_xjal( sendTechToUS );
   setupParameters_bringUSound_xjal( bringUSound );
   create_bringUSound_xjal( bringUSound );
   setupParameters_returnUSound_xjal( returnUSound );
   create_returnUSound_xjal( returnUSound );
   setupParameters_releaseTechUSound_xjal( releaseTechUSound );
   create_releaseTechUSound_xjal( releaseTechUSound );
            // Port connectors with non-replicated objects
   doUSound.out.connect( returnUSound.in ); // connector
   bringUSound.out.connect( doUSound.in ); // connector3
   in.map( seizeTechUSound.in ); // connector6
   out.map( releaseTechUSound.out ); // connector7
   seizeTechUSound.out.connect( sendTechToUS.in ); // connector8
   sendTechToUS.out.connect( bringUSound.in ); // connector9
   returnUSound.out.connect( releaseTechUSound.in ); // connector10
   // Creating replicated embedded objects
   assignInitialConditions();
   onCreate();
  }

  @Override
  public void start() {
   doUSound.start();
   seizeTechUSound.start();
   sendTechToUS.start();
   bringUSound.start();
   returnUSound.start();
   releaseTechUSound.start();
   onStartup();
  }


// User API ----------------------------------
  public Main get_Main() {
   ActiveObject owner = getOwner();
   if ( owner instanceof Main ) return (Main) owner;
   return null;
  }

  public List<Object> getEmbeddedObjects() {
   LinkedList<Object> list = new LinkedList<Object>();
   list.add( doUSound );
   list.add( seizeTechUSound );
```

```
    list.add( sendTechToUS );
    list.add( bringUSound );
    list.add( returnUSound );
    list.add( releaseTechUSound );
    return list;
  }

  public void onDestroy() {
    super.onDestroy();
    doUSound.onDestroy();
    seizeTechUSound.onDestroy();
    sendTechToUS.onDestroy();
    bringUSound.onDestroy();
    returnUSound.onDestroy();
    releaseTechUSound.onDestroy();
  }
}
```

XRayProcess.java

```
package emergency_department;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import com.xj.anylogic.libraries.enterprise.*;
import java.awt.geom.Arc2D;

public class XRayProcess extends ActiveObject
{
  // Plain Variables

  public Main  ed;

  // Collection Variables

  // Flow/Auxiliary Variables

  // Stock Variables

  // Embedded Objects

  public NetworkSeize<Entity > seizeXRay;
  public Delay<Entity > doExamination;
  public NetworkMoveTo<Entity > gotoXRay;
  public Delay<Entity > doXRay;
  public NetworkMoveTo<Entity > gotoECRoom;
  public NetworkRelease<Entity > releaseTechXRay;
  public NetworkSeize<Entity > callTech;

  public String getNameOf( ActiveObject ao ) {
    if ( ao == seizeXRay ) return "seizeXRay";
    if ( ao == doExamination ) return "doExamination";
    if ( ao == gotoXRay ) return "gotoXRay";
    if ( ao == doXRay ) return "doXRay";
    if ( ao == gotoECRoom ) return "gotoECRoom";
    if ( ao == releaseTechXRay ) return "releaseTechXRay";
    if ( ao == callTech ) return "callTech";
    return null;
  }

  public String getNameOf( ActiveObjectCollection<?> aolist ) {
    return null;
  }


  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkSeize<
Entity > instantiate_seizeXRay_xjal() {NetworkSeize<
Entity > object = new NetworkSeize<
Entity >( getEngine(), this, null ) {
      @Override
      public NetworkResourcePool[] resources( Entity entity ) {
        return _seizeXRay_resources_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_seizeXRay_xjal(NetworkSeize<
Entity > object ) {
```

```
    object.capacity = object._capacity_DefaultValue_xjal();
    object.maximumCapacity =
object._maximumCapacity_DefaultValue_xjal();
    object.enableTimeout = object._enableTimeout_DefaultValue_xjal();
    object.enablePreemption =
object._enablePreemption_DefaultValue_xjal();
    object.sendResources = object._sendResources_DefaultValue_xjal();
    object.destinationType =
object._destinationType_DefaultValue_xjal();
    object.attachResources =
object._attachResources_DefaultValue_xjal();
    object.animationGuide = object._animationGuide_DefaultValue_xjal();
    object.animationType = object._animationType_DefaultValue_xjal();
    object.animationForward =
object._animationForward_DefaultValue_xjal();
    object.enableStats = object._enableStats_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_seizeXRay_xjal(NetworkSeize<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private Delay<
Entity > instantiate_doExamination_xjal() {Delay<
Entity > object = new Delay<
Entity >( getEngine(), this, null ) {
      @Override
      public double delayTime( Entity entity ) {
        return _doExamination_delayTime_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_doExamination_xjal(Delay<
Entity > object ) {
    object.delayTimeDefinedByPath =
object._delayTimeDefinedByPath_DefaultValue_xjal();
    object.capacity = object._capacity_DefaultValue_xjal();
    object.maximumCapacity =
true
;
    object.animationGuide = object._animationGuide_DefaultValue_xjal();
    object.animationType = object._animationType_DefaultValue_xjal();
    object.animationForward =
object._animationForward_DefaultValue_xjal();
    object.enableStats = object._enableStats_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_doExamination_xjal(Delay<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkMoveTo<
Entity > instantiate_gotoXRay_xjal() {NetworkMoveTo<
Entity > object = new NetworkMoveTo<
Entity >( getEngine(), this, null ) {
      @Override
      public NetworkResourcePool destinationResource( ) {
        return _gotoXRay_destinationResource_xjal( );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_gotoXRay_xjal(NetworkMoveTo<
Entity > object ) {
    object.destinationIsNode =
false
;
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_gotoXRay_xjal(NetworkMoveTo<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private Delay<
Entity > instantiate_doXRay_xjal() {Delay<
Entity > object = new Delay<
Entity >( getEngine(), this, null ) {
      @Override
      public double delayTime( Entity entity ) {
        return _doXRay_delayTime_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_doXRay_xjal(Delay<
Entity > object ) {
```

```
    object.delayTimeDefinedByPath =
object._delayTimeDefinedByPath_DefaultValue_xjal();
    object.capacity = object._capacity_DefaultValue_xjal();
    object.maximumCapacity =
object._maximumCapacity_DefaultValue_xjal();
    object.animationGuide = object._animationGuide_DefaultValue_xjal();
    object.animationType = object._animationType_DefaultValue_xjal();
    object.animationForward =
object._animationForward_DefaultValue_xjal();
    object.enableStats = object._enableStats_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_doXRay_xjal(Delay<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkMoveTo<
Entity > instantiate_gotoECRoom_xjal() {NetworkMoveTo<
Entity > object = new NetworkMoveTo<
Entity >( getEngine(), this, null ) {
      @Override
      public NetworkResourcePool destinationResource(  ) {
        return _gotoECRoom_destinationResource_xjal(  );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_gotoECRoom_xjal(NetworkMoveTo<
Entity > object ) {
    object.destinationIsNode =
false
;
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_gotoECRoom_xjal(NetworkMoveTo<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkRelease<
Entity > instantiate_releaseTechXRay_xjal() {NetworkRelease<
Entity > object = new NetworkRelease<
Entity >( getEngine(), this, null ) {
      @Override
      public NetworkResourcePool[] resources( Entity entity ) {
        return _releaseTechXRay_resources_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_releaseTechXRay_xjal(NetworkRelease<
Entity > object ) {
    object.releaseAll = object._releaseAll_DefaultValue_xjal();
    object.movingGoHome =
object._movingGoHome_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_releaseTechXRay_xjal(NetworkRelease<
Entity > object ) {
    object.create();

  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private NetworkSeize<
Entity > instantiate_callTech_xjal() {NetworkSeize<
Entity > object = new NetworkSeize<
Entity >( getEngine(), this, null ) {
      @Override
      public NetworkResourcePool[] resources( Entity entity ) {
        return _callTech_resources_xjal( entity );
      }
    };

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_callTech_xjal(NetworkSeize<
Entity > object ) {
    object.capacity = object._capacity_DefaultValue_xjal();
    object.maximumCapacity =
true
;
    object.enableTimeout = object._enableTimeout_DefaultValue_xjal();
    object.enablePreemption =
object._enablePreemption_DefaultValue_xjal();
    object.sendResources =
true
;
    object.destinationType =
NetworkSendTo.DEST_ENTITY
;
    object.attachResources =
```

```
true
;
    object.animationGuide = object._animationGuide_DefaultValue_xjal();
    object.animationType = object._animationType_DefaultValue_xjal();
    object.animationForward                                          =
object._animationForward_DefaultValue_xjal();
    object.enableStats = object._enableStats_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_callTech_xjal(NetworkSeize<
Entity > object ) {
    object.create();

  }

  public NetworkResourcePool[] _seizeXRay_resources_xjal( Entity entity
) {
    return new NetworkResourcePool[]
{ ed.XRay }
;
  }
  public double _doExamination_delayTime_xjal( Entity entity ) {
    return
triangular( 3, 4, 6 )
;
  }
  public NetworkResourcePool _gotoXRay_destinationResource_xjal( ) {
    return
ed.XRay
;
  }
  public double _doXRay_delayTime_xjal( Entity entity ) {
    return
triangular( 5, 7, 10 )
;
  }
  public NetworkResourcePool _gotoECRoom_destinationResource_xjal(
) {
    return
ed.ECRoom
;
  }
  public NetworkResourcePool[] _releaseTechXRay_resources_xjal( Entity
entity ) {
    return new NetworkResourcePool[]
{ ed.technician, ed.XRay }
;
  }
  public NetworkResourcePool[] _callTech_resources_xjal( Entity entity )
{
    return new NetworkResourcePool[]
{ ed.technician }
;
  }
  static final Font _text_Font = new Font("SansSerif", 1, 18 );
  static final Font _text1_Font = new Font("SansSerif", 0, 18 );
  static final Font _text2_Font = _text1_Font;
  static final int _line = 1;
  static final int _oval = 2;
  static final int _text = 3;
  static final int _text1 = 4;
  static final int _line1 = 5;
  static final int _text2 = 6;
  static final int _line2 = 7;
  static final int _rectangle = 8;

  /**
   * Top-level presentation group id
   */
  static final int _presentation = 0;

  /**
   * Top-level icon group id
   */
  static final int _icon = -1;


  @Override
  public boolean onShapeClick( int _shape, int index, double clickx,
double clicky ){
    switch( _shape ){
      case _rectangle:
        if (true) {

getPresentation().setPresentable( getOwner() );
getPresentation().getPanel().setOffsets( 0, -1000 );
        }
        break;
      default: return super.onShapeClick( _shape, index, clickx, clicky );
                }
                return false;
  }

  ShapeLine line;
  ShapeOval oval;
  ShapeText text;
  ShapeText text1;
  ShapeLine line1;
  ShapeText text2;
  ShapeLine line2;
  ShapeRectangle rectangle;

  // Static initialization of persistent elements
  {
    line = new ShapeLine(
                              true,50, 100,
      black,
                              40,              0,
                              1, LINE_STYLE_SOLID
    );
    oval = new ShapeOval(
      true,70, 100, 0.0,
      black, silver,
                  10, 10,
                  1, LINE_STYLE_SOLID
    );
    text = new ShapeText(
      true,70, 89, 0.0,
      black,"X",
      _text_Font, ALIGNMENT_CENTER
    );
    text1 = new ShapeText(
      false,20, 15, 0.0,
      royalBlue,"Back",
      _text1_Font, ALIGNMENT_LEFT
    );
    line1 = new ShapeLine(
                              false,70, 10,
      black,
```

```
                        0,          30,
                        1, LINE_STYLE_SOLID
  );
  text2 = new ShapeText(
    false,80, 15, 0.0,
    black,"X-Ray Process",
    _text2_Font, ALIGNMENT_LEFT
  );
  line2 = new ShapeLine(
                        false,20, 34,
    royalBlue,
                        40,         0,
                        1, LINE_STYLE_SOLID
  );
  rectangle = new ShapeRectangle(
    false,15, 10, 0.0,
      null, null,
                                    50, 30,
                                    1, LINE_STYLE_SOLID
  ) {

   @Override
   public boolean onClick( double clickx, double clicky ) {
    return onShapeClick( _rectangle, 0, clickx, clicky );
   }
  };
}
ShapeGroup presentation;
ShapeGroup icon;

@Override
public Object getPersistentShape( int _shape ) {
 switch(_shape){
  case _presentation: return presentation;
  case _icon: return icon;

  case _line: return line;
  case _oval: return oval;
  case _text: return text;
  case _text1: return text1;
  case _line1: return line1;
  case _text2: return text2;
  case _line2: return line2;
  case _rectangle: return rectangle;
  default: return null;
 }
}

static final int[] _connector_pointsX = {
 200,  250 };
static final int[] _connector_pointsY = {
 200,200 };

static final int[] _connector1_pointsX = {
 590,  650 };
static final int[] _connector1_pointsY = {
 200,200 };

static final int[] _connector2_pointsX = {
 400,  450 };
static final int[] _connector2_pointsY = {
 200,200 };

static final int[] _connector3_pointsX = {
 100,  150 };
static final int[] _connector3_pointsY = {
 200,200 };

static final int[] _connector4_pointsX = {
 300,  350 };
static final int[] _connector4_pointsY = {
 200,200 };

static final int[] _connector5_pointsX = {
 500,  550 };
static final int[] _connector5_pointsY = {
 200,200 };


@Override
public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
    drawPort( _panel, _g, 50, 100, -40, 0, null, in );
    drawPort( _panel, _g, 90, 100, 10, 0, null, out );
  // Embedded object "seizeXRay"
  if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 200 , 150 , 50,
20, "seizeXRay", this.seizeXRay );
  }
  // Embedded object "doExamination"
  if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 100 , 150 , 50,
20, "doExamination", this.doExamination );
  }
  // Embedded object "gotoXRay"
  if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 300 , 150 , 50,
20, "gotoXRay", this.gotoXRay );
  }
  // Embedded object "doXRay"
  if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 400 , 150 , 60,
20, "doXRay", this.doXRay );
  }
  // Embedded object "gotoECRoom"
  if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 600 , 150 , 40,
20, "gotoECRoom", this.gotoECRoom );
  }
  // Embedded object "releaseTechXRay"
  if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 500 , 150 , 40,
20, "releaseTechXRay", this.releaseTechXRay );
  }
  // Embedded object "callTech"
  if (!_publicOnly) {
                drawEmbeddedObjectModel( _panel, _g, 0 , 150 , 60, 20,
"callTech", this.callTech );
  }
  if (!_publicOnly) {
    drawConnector(      _panel,      _g,      _connector_pointsX,
_connector_pointsY, false );
    }
  if (!_publicOnly) {
    drawConnector(      _panel,      _g,      _connector1_pointsX,
_connector1_pointsY, false );
    }
  if (!_publicOnly) {
    drawConnector(      _panel,      _g,      _connector2_pointsX,
_connector2_pointsY, false );
    }
  if (!_publicOnly) {
    drawConnector(      _panel,      _g,      _connector3_pointsX,
_connector3_pointsY, false );
```

```
    }
   if (!_publicOnly) {
    drawConnector( _panel, _g, _connector4_pointsX,
_connector4_pointsY, false );
    }
   if (!_publicOnly) {
    drawConnector( _panel, _g, _connector5_pointsX,
_connector5_pointsY, false );
    }
  }

  @Override
  public boolean onClickModelAt( Panel panel, double x, double y, int
clickCount, boolean publicOnly ) {
   if( modelElementContains(x, y, 50, 100) ) {
    panel.addInspect( 50, 100, this, "in" );
    return true;
   }
   if( modelElementContains(x, y, 90, 100) ) {
    panel.addInspect( 90, 100, this, "out" );
    return true;
   }
   if ( seizeXRay.onClickIconAt( x - 200, y - 150, true ) ) {
    if ( clickCount == 2 ) {
     panel.browseEmbeddedObject( x, y, this, "seizeXRay" );
    } else {
     panel.addInspect( x, y, this, "seizeXRay" );
    }
    return true;
   }
   if ( doExamination.onClickIconAt( x - 100, y - 150, true ) ) {
    if ( clickCount == 2 ) {
     panel.browseEmbeddedObject( x, y, this, "doExamination" );
    } else {
     panel.addInspect( x, y, this, "doExamination" );
    }
    return true;
   }
   if ( gotoXRay.onClickIconAt( x - 300, y - 150, true ) ) {
    if ( clickCount == 2 ) {
     panel.browseEmbeddedObject( x, y, this, "gotoXRay" );
    } else {
     panel.addInspect( x, y, this, "gotoXRay" );
    }
    return true;
   }
   if ( doXRay.onClickIconAt( x - 400, y - 150, true ) ) {
    if ( clickCount == 2 ) {
     panel.browseEmbeddedObject( x, y, this, "doXRay" );
    } else {
     panel.addInspect( x, y, this, "doXRay" );
    }
    return true;
   }
   if ( gotoECRoom.onClickIconAt( x - 600, y - 150, true ) ) {
    if ( clickCount == 2 ) {
     panel.browseEmbeddedObject( x, y, this, "gotoECRoom" );
    } else {
     panel.addInspect( x, y, this, "gotoECRoom" );
    }
    return true;
   }
   if ( releaseTechXRay.onClickIconAt( x - 500, y - 150, true ) ) {
    if ( clickCount == 2 ) {
     panel.browseEmbeddedObject( x, y, this, "releaseTechXRay" );
    } else {
     panel.addInspect( x, y, this, "releaseTechXRay" );
    }
    return true;
   }
   if ( callTech.onClickIconAt( x - 0, y - 150, true ) ) {
    if ( clickCount == 2 ) {
     panel.browseEmbeddedObject( x, y, this, "callTech" );
    } else {
     panel.addInspect( x, y, this, "callTech" );
    }
    return true;
   }
   return false;
  }

  // Ports

  public Port< Object , Object > in = new Port< Object, Object >( this );
  public Port< Object , Object > out = new Port< Object, Object >( this
);

  public String getNameOf( Port<?, ?> _p ) {
   if( _p == this.in ) return "in";
   if( _p == this.out ) return "out";
   return null;
  }

  /**
   * Constructor
   */
  public XRayProcess( Engine engine, ActiveObject owner,
ActiveObjectCollection<? extends XRayProcess> collection ) {
   super( engine, owner, collection );
  }

  @Override
  public void create() {
   // Creating embedded object instances
   seizeXRay = instantiate_seizeXRay_xjal();
   doExamination = instantiate_doExamination_xjal();
   gotoXRay = instantiate_gotoXRay_xjal();
   doXRay = instantiate_doXRay_xjal();
   gotoECRoom = instantiate_gotoECRoom_xjal();
   releaseTechXRay = instantiate_releaseTechXRay_xjal();
   callTech = instantiate_callTech_xjal();
   // Assigning initial values for plain variables
   ed =
get_Main()
;
   // Dynamic initialization of persistent elements
   presentation = new ShapeGroup( XRayProcess.this, true, 0, 0, 0,
text1, line1, text2, line2, rectangle );
   icon = new ShapeGroup( XRayProcess.this, true, 0, 0, 0
     ,line
     ,oval
     ,text                );
   // Creating non-replicated embedded objects
   setupParameters_seizeXRay_xjal( seizeXRay );
   create_seizeXRay_xjal( seizeXRay );
   setupParameters_doExamination_xjal( doExamination );
   create_doExamination_xjal( doExamination );
   setupParameters_gotoXRay_xjal( gotoXRay );
   create_gotoXRay_xjal( gotoXRay );
   setupParameters_doXRay_xjal( doXRay );
   create_doXRay_xjal( doXRay );
   setupParameters_gotoECRoom_xjal( gotoECRoom );
```

```
    create_gotoECRoom_xjal( gotoECRoom );
    setupParameters_releaseTechXRay_xjal( releaseTechXRay );
    create_releaseTechXRay_xjal( releaseTechXRay );
    setupParameters_callTech_xjal( callTech );
    create_callTech_xjal( callTech );
                // Port connectors with non-replicated objects
    doExamination.out.connect( seizeXRay.in ); // connector
    releaseTechXRay.out.connect( gotoECRoom.in ); // connector1
    gotoXRay.out.connect( doXRay.in ); // connector2
    callTech.out.connect( doExamination.in ); // connector3
    seizeXRay.out.connect( gotoXRay.in ); // connector4
    doXRay.out.connect( releaseTechXRay.in ); // connector5
    in.map( callTech.in ); // connector6
    out.map( gotoECRoom.out ); // connector7
    // Creating replicated embedded objects
    assignInitialConditions();
    onCreate();
  }

  @Override
  public void start() {
    seizeXRay.start();
    doExamination.start();
    gotoXRay.start();
    doXRay.start();
    gotoECRoom.start();
    releaseTechXRay.start();
    callTech.start();
    onStartup();
  }

// User API -----------------------------------
  public Main get_Main() {
    ActiveObject owner = getOwner();
    if ( owner instanceof Main ) return (Main) owner;
    return null;
  }

  public List<Object> getEmbeddedObjects() {
    LinkedList<Object> list = new LinkedList<Object>();
    list.add( seizeXRay );
    list.add( doExamination );
    list.add( gotoXRay );
    list.add( doXRay );
    list.add( gotoECRoom );
    list.add( releaseTechXRay );
    list.add( callTech );
    return list;
  }

  public void onDestroy() {
    super.onDestroy();
    seizeXRay.onDestroy();
    doExamination.onDestroy();
    gotoXRay.onDestroy();
    doXRay.onDestroy();
    gotoECRoom.onDestroy();
    releaseTechXRay.onDestroy();
    callTech.onDestroy();
  }


}
```

Simulation.java

```
package emergency_department;

import java.sql.Connection;
import java.sql.SQLException;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import com.xj.anylogic.libraries.enterprise.*;
import javax.swing.JApplet;

public class Simulation extends ExperimentSimulation<Main> {
  static final Font _button_Font = new Font("Dialog", 0, 11 );
  static final Font _text_Font = new Font("SansSerif", 1, 28 );
  static final Font _text1_Font = new Font("SansSerif", 0, 9 );
  static final Font _text2_Font = new Font("SansSerif", 1, 11 );
  static final Color _text1_Color = new Color( 0xFF333333, true );
  static final Color _text2_Color = new Color( 0xFF454545, true );
  static final int _button = 1;
  static final int _text = 2;
  static final int _image = 3;
  static final int _text1 = 4;
  static final int _line = 5;
  static final int _text2 = 6;
```

```
 /**
  * Top-level presentation group id
  */
 static final int _presentation = 0;

 /**
  * Top-level icon group id
  */
 static final int _icon = -1;


 @Override
 public void executeShapeControlAction( int _shape, int index ) {
   switch( _shape ) {
     case _button: {
run();

getEngine().getPresentation().setPresentable( getEngine().getRoot() );
;}
       break;
     default:
       super.executeShapeControlAction( _shape, index );
       break;
   }
 }


 /**
  * <i>This method should not be called by user</i>
  */
 private void _button_SetDynamicParams( ShapeButton shape ) {
   boolean _visible;
   shape.setEnabled(
getState() == IDLE
);
 }

 ShapeButton button;
 ShapeText text;
 ShapeImage image;
 ShapeText text1;
 ShapeLine line;
 ShapeText text2;

 // Static initialization of persistent elements
 {
   button = new ShapeButton(
                                    Simulation.this, true, 50, 440,
                                    70, 30,
         controlDefault, black,
         _button_Font,
                                    "Run"
   ) {

     @Override
     public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
       _button_SetDynamicParams( this );
       super.draw( panel, graphics, xform, publicOnly );
     }

     @Override
     public void action(){
       executeShapeControlAction( _button, 0 );
     }
   };
   text = new ShapeText(
     true,50, 26, 0.0,
     navy,"Emergency Department",
     _text_Font, ALIGNMENT_LEFT
   );
   image = new ShapeImage(
                          Simulation.this, true, 50, 500, 0.0,
                                  116,                    40,
"/emergency_department/",
                                  new      String[]{"xjlogo.png",}

   );
   text1 = new ShapeText(
     true,90, 540, 0.0,
     _text1_Color,"This AnyLogic™ Model is\r\n© 1992-2007 XJ
Technologies\r\nwww.anylogic.com\r\n\r\n",
     _text1_Font, ALIGNMENT_LEFT
   );
   line = new ShapeLine(
                          true,0, 60,
     steelBlue,
                          900,      0,
                          1, LINE_STYLE_SOLID
   );
   text2 = new ShapeText(
     true,50, 100, 0.0,
     _text2_Color,"This simplistic model of an Emergency Department
was designed primarily to demonstrate the usage of Network part of
the\r\nEnterprise Library. A network of nodes and paths is defined on the
basis of a given facility layout. Resources of different\r\nkinds are placed
in the network. In this case the resources are: \r\n\r\n  - Nurses, PAs and
Technicians - of type moving [can move on their own]\r\n  - Triage
rooms, Express care rooms, XRay - of type static [are bound to their
home locations and cannot be moved]\r\n  - Ultra Sound devices - of
type portable [can be moved by resources of type moving]\r\n\r\nUpon
arrival, a patient registers and proceeds to the waiting room, from where
he is escorted by the nurse to a triage room.\r\nAfter triage the patient
goes to an express care room and then either x-ray or ultra sound is
done with the help of a\r\ntechnician and a PA. X Ray process requires
the patient to go the X-Ray room, whereas ultra sound device is moved
to the EC\r\nroom where the patient is located.\r\n\r\nThis model shows
several important features of the Enterprise Library:\r\n\r\n  - The
network objects can interleave in the process flowchart with regular
objects such as delays, queues, etc. \r\n  - An entity can be animated
either by the network or by regular objects\r\n  - The model can be
made hierarchical with subprocesses defined in different active object
classes (such as XRayProcess and\r\n      USoundProcess)\r\n  - The
parameters of the process (such as resource capacities) can be changed
dynamically\r\n\r\nThe model presentation screen is organized as a
number of pages (animation, main flowchart, etc.) with
hyperlinks\r\nbetween them to enable easy navigation.",
     _text2_Font, ALIGNMENT_LEFT
   );
 }
 ShapeGroup presentation;
 ShapeGroup icon;

 @Override
 public Object getPersistentShape( int _shape ) {
   switch(_shape){
     case _presentation: return presentation;
     case _icon: return icon;

     case _button: return button;
     case _text: return text;
     case _image: return image;
     case _text1: return text1;
     case _line: return line;
     case _text2: return text2;
     default: return null;
   }
 }
```

```
 @Override
 public int getWindowWidth() {
  return 900;
 }

 @Override
 public int getWindowHeight() {
  return 600;
 }

 /**
  * Applet class to run experiment as java applet
  */
 public static class Applet extends JApplet {

  Simulation ex;

  @Override
  public void init() {
   ex = new Simulation();
   ex.setup( this );
  }

  @Override
  public void destroy() {
   ex.close();
  }

 }

 @Override
 public void initDefaultRandomNumberGenerator(Engine engine) {
  engine.getDefaultRandomGenerator().setSeed( 1 );
 }

 @Override
 public Main createRoot( Engine engine ) {
  // Create the root object
  return new Main( engine, null, null );
 }

 @Override
 public void setupRootParameters( Main root, boolean
callOnChangeActions ) {
 }
  /**
  * Engine setup
  */
 @Override
 public void setupEngine(Engine engine) {
  engine.setATOL( 1.0E-5 );
  engine.setRTOL( 1.0E-5 );
  engine.setTTOL( 1.0E-5 );
  engine.setHTOL( 0.0010 );
  engine.setSolverODE( Engine.SOLVER_ODE_EULER );
  engine.setSolverNAE( Engine.SOLVER_NAE_MODIFIED_NEWTON );
  engine.setSolverDAE( Engine.SOLVER_DAE_RK45_NEWTON );
  engine.setVMethods( 16032 );

  engine.setStartTime( 0.0 );
  engine.setTimeUnit( TIME_UNIT_MINUTE );
  engine.setStartDate( toDate( 2007, JANUARY, 1, 0, 0, 0 ) );
  engine.setRealTimeMode( true );
  engine.setRealTimeScale( 1.0 );
 }
```

```
 /**
  * Experiment setup
  */
 @Override
 public void setup( JApplet applet ) {
  setName( "Emergency Department : Simulation" );

  // Dynamic initialization of persistent elements
  presentation = new ShapeGroup( Simulation.this, true, 0, 0, 0, button,
text, image, text1, line, text2 );
  icon = new ShapeGroup( Simulation.this, true, 0, 0, 0 );
  // Setup presentation
  Presentation _p = new Presentation( this, applet != null ?
Presentation.MODE_APPLET :
      Presentation.MODE_APPLICATION, applet );
  _p.start();

  Panel _panel = _p.getPanel();
  ToolBar _tb = _p.getToolBar();
  StatusBar _sb = _p.getStatusBar();

  _panel.setZoomEnabled( false );
  _panel.setPanningEnabled( false );
  _panel.setFrameManagementBalance( 2.0 );

  _sb.setSectionVisible( StatusBar.DATE, false );
  _sb.setSectionVisible( StatusBar.EPS, true );
  _sb.setSectionVisible( StatusBar.EXPERIMENT, false );
  _sb.setSectionVisible( StatusBar.FPS, true );
  _sb.setSectionVisible( StatusBar.MEMORY, true );
  _sb.setSectionVisible( StatusBar.SECONDS, true );
  _sb.setSectionVisible( StatusBar.SIMULATION, false );
  _sb.setSectionVisible( StatusBar.STATUS, true );
  _sb.setSectionVisible( StatusBar.STEP, false );
  _sb.setSectionVisible( StatusBar.TIME, false );
  _tb.setSectionVisible( ToolBar.ANIMATION, false );
  _tb.setSectionVisible( ToolBar.EXECUTION, true );
  _tb.setSectionVisible( ToolBar.FILE, false );
  _tb.setSectionVisible( ToolBar.NAVIGATION, true );
  _tb.setSectionVisible( ToolBar.TIME_SCALE, true );
  _tb.setSectionVisible( ToolBar.VIEW, true );
 }
}
```

Patient.java

```
/**
 *  Patient
 */
public class Patient  extends Entity{

        double timeAdmitted;


  /**
   * Default constructor
   */
  public Patient(){
  }

  /**
   * Constructor initializing the fields
   */
  public Patient(double timeAdmitted){
            this.timeAdmitted = timeAdmitted;
```

```
	}

		@Override
		public String toString() {
			return
					"timeAdmitted = " +
timeAdmitted +" ";
		}

		/**
		 * This number is here for model snapshot storing
purpose<br>
		 * It needs to be changed when this class gets changed
		 */
		private static final long serialVersionUID = -
7006651055925022762L;

}
```

Παράρτημα 7

UML – Διαγράμματα κλάσεων

SDModel
+People_Birth: double
+Contact_Rate: double
+People_Reaction_Delay_On_Anti_Regime_Messages: double
+Regime_Social_and_Economic_Efforts: double
+Avg_Time_as_Dissident: double
+Appeasement_Fraction: double
+Desired_Time_to_Remove_Insurgents: double
+Removal_Effectiveness: double
+Government_Supporters: double
+Dissidents: double
+Insurgents: double
+Removed_Insurgents: double
+Dissidents_Fraction: double
+Insurgents_Fraction: double
+Becoming_Dissident: double
+Removing_Insurgents: double
+Recruits_Through_Social_Network: double
+Insurgent_Recruitment: double
+Births: double
+Appeasement_Rate: double
+Regime_Opponents: double
+Violent_Incident_Intensity: double
+Propensity_to_Commit_Violence: double
+Propensity_to_Protest: double
+Protest_Intensity: double
+Incident_Intensity: double
+Relative_Strength_of_Violent_Incidents: double
+Effect_of_Incidents_on_Anti_Regime_Messages: double
+Effect_of_Anti_Regime_Messages_on_Recruitment: double
+Strength_of_Incident_Effect_on_Anti_Regime_Messages: double
+Propensity_to_be_Recruited: double
+Normal_Propensity_to_be_Recruited: double
+Normal_Frequency_of_Anti_Regime_Messages: double
+Perceived_Intencity_of_Anti_Regime_Messages: double
+Indicated_Force_Strength: double
~integrationManager_xjal: ActiveObjectIntegrationManager = new ActiveObjectIntegrationManager(5, 0, 15)
+_plot1_autoUpdateEvent_xjal: EventTimeout = new EventTimeout(this)
+_autoCreatedDS_xjal: EventTimeout = new EventTimeout(this)
+_ds_Becoming_Dissident: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Becoming_Dissident); }}
+_ds_Insurgent_Recruitment: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Insurgent_Recruitment); }}
+_ds_Removing_Insurgents: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Removing_Insurgents); }}
+_ds_Births: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Births); }}
+_ds_Appeasement_Rate: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Appeasement_Rate); }}
+_ds_Recruits_Through_Social_Network: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Recruits_Through_Social_Network); }}
+_ds_Regime_Opponents: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Regime_Opponents); }}
+_ds_Violent_Incident_Intensity: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Violet_Incident_Intensity); }}
+_ds_Propensity_to_Commit_Violence: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Propensity_to_Commit_Violence); }}
+_ds_Propensity_to_Protest: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Propensity_to_Protest); }}
+_ds_Protest_Intensity: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Protest_Intensity); }}
+_ds_Incident_Intensity: DataSet = new DataSet(100) { @Override public void update() { add(time(),

SDModel.this.Incident_Intensity); }}

+_ds_Relative_Strength_of_Violent_Incidents: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Relative_Strength_of_Violent_Incidents); }}

+_ds_Effect_of_Incidents_on_Anti_Regime_Messages: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Effect_of_Incidents_on_Anti_Regime_Messages); }}

+_ds_Strength_of_Incident_Effect_on_Anti_Regime_Messages: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Strength_of_Incident_Effect_on_Anti_Regime_Messages); }}

+_ds_Effect_of_Anti_Regime_Messages_on_Recruitment: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Effect_of_Anti_Regime_Messages_on_Recruitment); }}

+_ds_Propensity_to_be_Recruited: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Propensity_to_be_Recruited); }}

+_ds_Normal_Propensity_to_be_Recruited: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Normal_Propensity_to_be_Recruited); }}

+_ds_Dissidents_Fraction: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Dissidents_Fraction); }}

+_ds_Insurgents_Fraction: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Insurgents_Fraction); }}

+_ds_Indicated_Force_Strength: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Indicated_Force_Strength); }}

+_ds_Normal_Frequency_of_Anti_Regime_Messages: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Normal_Frequency_of_Anti_Regime_Messages); }}

+_ds_Government_Supporters: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Government_Supporters); }}

+_ds_Dissidents: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Dissidents); }}

+_ds_Insurgents: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Insurgents); }}

+ds_Removed_Insurgents: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Removed_Insurgents); }}

+_ds_Perceived_Intencity_of_Anti_Regime_Messages: DataSet = new DataSet(100) { @Override public void update() { add(time(), SDModel.this.Perceived_Intencity_of_Anti_Regime_Messages); }}

+dsGovernment_Supporters: DataSet = new DataSet(10000) { @Override public void update() { add(time(), _dsGovernment_Supporters_YValue()); }}

+dsDissidents: DataSet = new DataSet(10000) { @Override public void update() { add(time(), _dsDissidents_YValue()); }}

+dsInsurgents: DataSet = new DataSet(10000) { @Override public void update() { add(time(), _dsInsurgents_YValue()); }}

+dsRemoved_Insurgents: DataSet = new DataSet(10000) { @Override public void update() { add(time(), _dsRemoved_Insurgents_YValue()); }}

~_text_Font: Font = new Font("SansSerif", 0, 32)

~_text17_Font: Font = new Font("SansSerif", 0, 18)

~_text18_Font: Font = new Font("SansSerif", 0, 18)

~_text19_Font: Font = new Font("SansSerif", 0, 18)

~_text27_Font: Font = new Font("SansSerif", 0, 18)

~_text17: int=1

~_rectangle34: int=2

~_text19: int=3

~_rectangle37: int=4

~_line55: int=5

~_line54: int=6

~_line53: int=7

~_text18: int=8

~_line57: int=9

~_line66: int=10

~_line67: int=11

~_text27: int=12

~_rectangle40: int=13

~_text: int=14

~line1: int=15

~_menu_group: int=16

~_plot1: int=17

~_presentation: int=0

~_icon: int=-1

~plot1: Plot

~text: ShapeText

~text17: ShapeText

~text18: ShapeText

~text19: ShapeText

~text27: ShapeText

~rectangle34: ShapeRectangle

~rectangle37: ShapeRectangle

~rectangle40: ShapeRectangle ~line53: ShapeLine ~line54: ShapeLine ~line55: ShapeLine ~line57: ShapeLine ~line66: ShapeLine ~line67: ShapeLine ~menu_group: ShapeGroup ~presentation: ShapeGroup ~icon: ShapeGroup ~arc_PD_1207137222568: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137222594: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137222599: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137222609: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137222656: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137222657: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137222658: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137222765: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137222781: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137222828: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137222829: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137222844: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137222847: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137222906: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137222968: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137222984: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137223031: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137223140: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207137223156: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207216927734: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207216941078: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207228547388: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207119091875: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207576881921: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207592083312: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207662497453: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207726841718: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207728954437: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207729293846: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207736540218: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207748231921: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207830555656: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207830555657: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207830591078: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207830591079: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207830615421: Arc2D.Double = new Arc2D.Double() ~arc_PD_1207830635796: Arc2D.Double = new Arc2D.Double() ~arc_PD_1208247790296: Arc2D.Double = new Arc2D.Double() ~arc_PD_1208432283798: Arc2D.Double = new Arc2D.Double() ~arc_PD_1208432283799: Arc2D.Double = new Arc2D.Double() ~arc_PD_1208947334609: Arc2D.Double = new Arc2D.Double()
<<create>>+SDModel(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection) +_People_Birth_DefaultValue_xjal(): double +set_People_Birth(People_Birth: double) ~onChange_People_Birth() +_Contact_Rate_DefaultValue_xjal(): double +set_Contact_Rate(Contact_Rate: double) ~onChange_Contact_Rate() +_People_Reaction_Delay_On_Anti_Regime_Messages_DefaultValue_xjal(): double +set_People_Reaction_Delay_On_Anti_Regime_Messages(People_Reaction_Delay_On_Anti_Regime_Messages: double) ~onChange_ People_Reaction_Delay_On_Anti_Regime_Messages () +_Regime_Social_and_Economic_Efforts_DefaultValue_xjal(): double +set_Regime_Social_and_Economic_Efforts(Regime_Social_and_Economic_Efforts: double) ~onChange_Regime_Social_and_Economic_Efforts()

+_Avg_Time_as_Dissident_DefaultValue_xjal(): double
+set_Avg_Time_as_Dissident(Avg_Time_as_Dissident: double)
~onChange_Avg_Time_as_Dissident()
+_Appeasement_Fraction_DefaultValue_xjal(): double
+set_Appeasement_Fraction(Appeasement_Fraction: double)
~onChange_Appeasement_Fraction()
+_Desired_Time_to_Remove_Insurgents_DefaultValue_xjal(): double
+set_Desired_Time_to_Remove_Insurgents(Desired_Time_to_Remove_Insurgents: double)
~onChange_Desired_Time_to_Remove_Insurgents()
+_Removal_Effectiveness_DefaultValue_xjal(): double
+set_Removal_Effectiveness(Removal_Effectiveness: double)
~onChange_Removal_Effectiveness()
+getScalarPhaseVector(_d: double, _a: double)
+putScalarPhaseVector(_d: double, _a: double)
+assignInitialConditions()
+assignInitialConditions(_variableNameCode: int)
+_Government_Supporters_Initial_Value(): double
+_Propensity_to_Protest_Value(): double
+_Incident_Intensity_Formula(): double
+_Effect_of_Incidents_on_Anti_Regime_Messages_Formula(): double
+_Relative_Strength_of_Violent_Incidents_Value(): double
+_Dissidents_Fraction_Value(): double
+_Dissidents_Expression(): double
+_Normal_Frequency_of_Anti_Regime_Messages_Formula(): double
+_Violent_Incident_Intensity_Formula(): double
+_Effect_of_Anti_Regime_Messages_on_Recruitment_Formula(): double
+_Government_Supporters_Expression(): double
+_Becoming_Dissident_Formula(): double
+_Appeasement_Rate_Formula(): double
+_Removed_Insurgents_Expression(): double
+_Removing_Insurgents_Formula(): double
+_Normal_Propensity_to_be_Recruited_Value(): double
+_Births_Formula(): double
+_Perceived_Intencity_of_Anti_Regime_Messages_Expression(): double
+_Insurgents_Fraction_Value(): double
+_Propensity_to_be_Recruited_Formula(): double
+_Strength_of_Incident_Effect_on_Anti_Regime_Messages_Value(): double
+_Indicated_Force_Strength_Formula(): double
+_Insurgents_Initial_Value(): double
+_Protest_Intensity_Formula(): double
+_Recruits_Through_Social_Network_Formula(): double
+_Propensity_to_Commit_Violence_Value(): double
+_Insurgents_Expression(): double
+_Insurgent_Recruitment_Formula(): double
+_Regime_Opponents_Formula(): double
+_Dissidents_Initial_Value(): double
+formulasExecute()
+formulasExecute(_variableNameCode: int)
+getScalarRightPart(_dr: double, _ar: double)
+getIntegrationManager(): ActiveObjectIntegrationManager
+getNameOf(_e: EventTimeout): String
+getModeOf(_e: EventTimeout): int
+getFirstOccurenceTime(_e: EventTimeout): double
+evaluateTimeout(_e: EventTimeout): double
+executeActionOf(_e: EventTimeout)
-_dsInsurgents_YValue(): double
-_dsGovernment_Supporters_YValue(): double
-_dsDissidents_YValue(): double
-_dsRemoved_Insurgents_YValue(): double
+onShapeClick(_shape: int, index: int, clickx: double, clicky: double): boolean
+getNameOfShape(_shape: int): String
+getShapeType(_shape: int): int
+getShapeX(_shape: int, index: int): double
+getShapeY(_shape: int, index: int): double
+getShapeLineColor(_shape: int, index: int): Color

+getShapeLineStyle(_shape: int, index: int): int
+getShapeLineDx(_shape: int, index: int): double
+getShapeLineDy(_shape: int, index: int): double
+getPersistentShape(_shape: int): Object
+drawModelElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean)
+onClickModelAt(panel: Panel, x: double, y: double, clickCount: int, publicOnly: boolean): boolean
+create()
+start()
+get_Main(): Main
+onDestroy()

ABModel

+People_Birth: double
+Contact_Rate: double
+People_Reaction_Delay_On_Anti_Regime_Messages: double
+Regime_Social_and_Economic_Efforts: double
+Avg_Time_as_Dissident: double
+Appeasement_Fraction: double
+Desired_Time_to_Remove_Insurgents: double
+Removal_Effectiveness: double
+Government_Supporters: double
+Dissidents: int
+Insurgents: int
+Removed_Insurgents: int
+ToDisidents: int
+ToGovernment_Supporters: int
+ToInsurgents: int
+ToRemoved_Insurgents: int
+ToInsurgentsByPreviousStep: int
+ToRemoved_InsurgentsByPreviousStep: int
+ToGovernment_SupportersByPreviousStep: int
+ToDisidentsByPreviousStep: int
+Disidents_Fraction: double
+Insurgents_Fraction: double
+Normal_Propensity_to_be_Recruited: double
+Propensity_to_Commit_Violence: double
+Relative_Strength_of_Violent_Incidents: double
+Propensity_to_Protest: double
+Protest_Intensity: double
+Violet_Incident_Intensity: double
+Regime_Opponents: double
+Incident_Intensity: double
+Normal_Frequency_of_Anti_Regime_Messages: double
+Effect_of_Incidents_on_Anti_Regime_Messages: double
+Effect_of_Anti_Regime_Messages_on_Recruitment: double
+Strength_of_Incident_Effect_on_Anti_Regime_Messages: double
+Propensity_to_be_Recruited: double
+Perceived_Intencity_of_Anti_Regime_Messages: double
~integrationManager_xjal: ActiveObjectIntegrationManager = new ActiveObjectIntegrationManager(1, 0, 8)
+births: EventTimeout = new EventTimeout(this)
+_plot_autoUpdateEvent_xjal: EventTimeout = new EventTimeout(this)
+_autoCreatedDS_xjal: EventTimeout = new EventTimeout(this)
+_ds_Dissidents_Fraction: DataSet = new DataSet(100) { @Override public void update() { add(time(), ABModel.this.Dissidents_Fraction); }}
+_ds_Insurgents_Fraction: DataSet = new DataSet(100) { @Override public void update() { add(time(), ABModel.this.Insurgents_Fraction); }}
+_ds_Normal_Propensity_to_be_Recruited: DataSet = new DataSet(100) { @Override public void update() { add(time(), ABModel.this.Normal_Propensity_to_be_Recruited); }}
+_ds_Propensity_to_Commit_Violence: DataSet = new DataSet(100) { @Override public void update() { add(time(), ABModel.this.Propensity_to_Commit_Violence); }}
+_ds_Relative_Strength_of_Violent_Incidents: DataSet = new DataSet(100) { @Override public void update() { add(time(), ABModel.this.Relative_Strength_of_Violent_Incidents); }}
+_ds_Propensity_to_Protest: DataSet = new DataSet(100) { @Override public void update() { add(time(), ABModel.this.Propensity_to_Protest); }}
+_ds_Protest_Intensity: DataSet = new DataSet(100) { @Override public void update() { add(time(),

ABModel.this.Protest_Intensity); }}

+_ds_Violet_Incident_Intensity: DataSet = new DataSet(100) { @Override public void update() { add(time(), ABModel.this.Violet_Incident_Intensity); }}

+_ds_Regime_Opponents: DataSet = new DataSet(100) { @Override public void update() { add(time(), ABModel.this.Regime_Opponents); }}

+_ds_Incident_Intensity: DataSet = new DataSet(100) { @Override public void update() { add(time(), ABModel.this.Incident_Intensity); }}

+_ds_Normal_Frequency_of_Anti_Regime_Messages: DataSet = new DataSet(100) { @Override public void update() { add(time(), ABModel.this.Normal_Frequency_of_Anti_Regime_Messages); }}

+_ds_Effect_of_Incidents_on_Anti_Regime_Messages: DataSet = new DataSet(100) { @Override public void update() { add(time(), ABModel.this.Effect_of_Incidents_on_Anti_Regime_Messages); }}

+_ds_Effect_of_Anti_Regime_Messages_on_Recruitment: DataSet = new DataSet(100) { @Override public void update() { add(time(), ABModel.this.Effect_of_Anti_Regime_Messages_on_Recruitment); }}

+_ds_Strength_of_Incident_Effect_on_Anti_Regime_Messages: DataSet = new DataSet(100) { @Override public void update() { add(time(), ABModel.this.Strength_of_Incident_Effect_on_Anti_Regime_Messages); }}

+_ds_Propensity_to_be_Recruited: DataSet = new DataSet(100) { @Override public void update() { add(time(), ABModel.this.Propensity_to_be_Recruited); }}

+_ds_Perceived_Intencity_of_Anti_Regime_Messages: DataSet = new DataSet(100) { @Override public void update() { add(time(), ABModel.this.Perceived_Intencity_of_Anti_Regime_Messages); }}

+dsInsurgents: DataSet = new DataSet(10000) { @Override public void update() { add(time(), _dsInsurgents_YValue()); }}

+dsGovernment_Supporters: DataSet = new DataSet(10000) { @Override public void update() { add(time(), _dsGovernment_Supporters_YValue()); }}

+dsDissidents: DataSet = new DataSet(10000) { @Override public void update() { add(time(), _dsDissidents_YValue()); }}

+dsRemoved_Insurgents: DataSet = new DataSet(10000) { @Override public void update() { add(time(), _dsRemoved_Insurgents_YValue()); }}

~_button_Font: Font = new Font("Dialog", 0, 10)

~_text_Font: Font = new Font("SansSerif", 1, 12)

~_text2_Font: Font = new Font("SansSerif", 1, 12)

~_text3_Font: Font = new Font("SansSerif", 0, 12)

~_text4_Font: Font = new Font("SansSerif", 1, 11)

~_text5_Font: Font = new Font("SansSerif", 1, 11)

~_text6_Font: Font = new Font("SansSerif", 1, 11)

~_text7_Font: Font = new Font("SansSerif", 1, 11)

~_text8_Font: Font = new Font("SansSerif", 1, 10)

~_text9_Font: Font = new Font("SansSerif", 1, 10)

~_text10_Font: Font = new Font("SansSerif", 1, 10)

~_text11_Font: Font = new Font("SansSerif", 1, 10)

~_text17_Font: Font = new Font("SansSerif", 0, 18)

~_text18_Font: Font = new Font("SansSerif", 0, 18)

~_text19_Font: Font = new Font("SansSerif", 0, 18)

~_text27_Font: Font = new Font("SansSerif", 0, 18)

~_text13_Font: Font = new Font("SansSerif", 0, 32)

~_text1_Font: Font = new Font("SansSerif", 0, 10)

~_text12_Font: Font = new Font("SansSerif", 0, 10)

~_text14_Font: Font = new Font("SansSerif", 0, 10)

~_text15_Font: Font = new Font("SansSerif", 0, 10)

~_roundRectangle2_FillColor: Color = new Color(0xFFE7FEC1, true)

~_roundRectangle3_FillColor: Color = new Color(0xFFFEF4C3, true)

~_roundRectangle4_FillColor: Color = new Color(0xFFFEDBD1, true)

~_roundRectangle5_FillColor: Color = new Color(0xFFEEF1E8, true)

~_button: int=1

~_roundRectangle: int=2

~_roundRectangle1: int=3

~_text: int=4

~_text2: int=5

~_roundRectangle2: int=6

~_roundRectangle3: int=7

~_roundRectangle4: int=8

~people_presentation: int=9

~_roundRectangle5: int=10

~_polyline: int=11

~_polyline1: int=12

~_polyline2: int=13

~_polyline3: int=14

~_text3: int=15

~_polyline4: int=16

~_text4: int=17

~_text5: int=18
~_text6: int=19
~_text7: int=20
~_text8: int=21
~_text9: int=22
~_text10: int=23
~_text11: int=24
~_text17: int=25
~_rectangle34: int=26
~_text19: int=27
~_rectangle37: int=28
~_line55: int=29
~_line54: int=30
~_line53: int=31
~_text18: int=32
~_line57: int=33
~_rectangle40: int=34
~_line67: int=35
~_text27: int=36
~_line66: int=37
~line1: int=38
~_text13: int=39
~_menu_group: int=40
~_text1: int=41
~_text12: int=42
~_text14: int=43
~_text15: int=44
~_plot: int=45
~_presentation: int=0
~_icon: int=-1
~button: ShapeButton
~plot: Plot
~roundRectangle: ShapeRoundedRectangle
~roundRectangle1: ShapeRoundedRectangle
~roundRectangle2: ShapeRoundedRectangle
~roundRectangle3: ShapeRoundedRectangle
~roundRectangle4: ShapeRoundedRectangle
~roundRectangle5: ShapeRoundedRectangle
~text: ShapeText
~text1: ShapeText
~text2: ShapeText
~text3: ShapeText
~text4: ShapeText
~text5: ShapeText
~text6: ShapeText
~text7: ShapeText
~text8: ShapeText
~text9: ShapeText
~text10: ShapeText
~text11: ShapeText
~text12: ShapeText
~text13: ShapeText
~text14: ShapeText
~text15: ShapeText
~text17: ShapeText
~text18: ShapeText
~text19: ShapeText
~text27: ShapeText
~rectangle34: ShapeRectangle
~rectangle37: ShapeRectangle
~rectangle40: ShapeRectangle
~polyline: ShapePolyline
~polyline1: ShapePolyline
~polyline2: ShapePolyline
~polyline3: ShapePolyline

~polyline4: ShapePolyline
~line53: ShapeLine
~line54: ShapeLine
~line55: ShapeLine
~line57: ShapeLine
~line66: ShapeLine
~line67: ShapeLine
~menu_group: ShapeGroup
~presentation: ShapeGroup
~icon: ShapeGroup
~arc_PD_1207137343218: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207137343221: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207137343239: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207137343250: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207137343251: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207137343257: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207137343258: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207137343259: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207658308437: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207658308500: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207658308501: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207662588265: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207662593000: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207726851859: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207728974046: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207736960250: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207736960251: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207736960254: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207855268234: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207855268235: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207855301875: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207855301876: Arc2D.Double = new Arc2D.Double()
~arc_PD_1207855301877: Arc2D.Double = new Arc2D.Double()
~arc_PD_1208247759218: Arc2D.Double = new Arc2D.Double()
+environment: Environment = new Environment(this)

<<create>>+ABModel(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection)
+_People_Birth_DefaultValue_xjal(): double
+set_People_Birth(People_Birth: double)
~onChange_People_Birth()
+_Contact_Rate_DefaultValue_xjal(): double
+set_Contact_Rate(Contact_Rate: double)
~onChange_Contact_Rate()
+_People_Reaction_Delay_On_Anti_Regime_Messages_DefaultValue_xjal(): double
+set_People_Reaction_Delay_On_Anti_Regime_Messages(People_Reaction_Delay_On_Anti_Regime_Messages: double)
~onChange_People_Reaction_Delay_On_Anti_Regime_Messages ()
+_Regime_Social_and_Economic_Efforts_DefaultValue_xjal(): double
+set_Regime_Social_and_Economic_Efforts(Regime_Social_and_Economic_Efforts: double)
~onChange_Regime_Social_and_Economic_Efforts()
+_Avg_Time_as_Dissident_DefaultValue_xjal(): double
+set_Avg_Time_as_Dissident(Avg_Time_as_Dissident: double)
~onChange_Avg_Time_as_Dissident()
+_Appeasement_Fraction_DefaultValue_xjal(): double
+set_Appeasement_Fraction(Appeasement_Fraction: double)
~onChange_Appeasement_Fraction()
+_Desired_Time_to_Remove_Insurgents_DefaultValue_xjal(): double
+set_Desired_Time_to_Remove_Insurgents(Desired_Time_to_Remove_Insurgents: double)
~onChange_Desired_Time_to_Remove_Insurgents()
+_Removal_Effectiveness_DefaultValue_xjal(): double
+set_Removal_Effectiveness(Removal_Effectiveness: double)
~onChange_Removal_Effectiveness()
+getScalarPhaseVector(_d: double, _a: double)
+putScalarPhaseVector(_d: double, _a: double)
+assignInitialConditions()
+assignInitialConditions(_variableNameCode: int)

+_Protest_Intensity_Formula(): double
+_Incident_Intensity_Formula(): double
+_Normal_Frequency_of_Anti_Regime_Messages_Formula(): double
+_Propensity_to_Commit_Violence_Value(): double
+_Violet_Incident_Intensity_Formula(): double
+_Normal_Propensity_to_be_Recruited_Value(): double
+_Perceived_Intencity_of_Anti_Regime_Messages_Expression(): double
+_Relative_Strength_of_Violent_Incidents_Value(): double
+_Effect_of_Incidents_on_Anti_Regime_Messages_Formula(): double
+_Effect_of_Anti_Regime_Messages_on_Recruitment_Formula(): double
+_Strength_of_Incident_Effect_on_Anti_Regime_Messages_Value(): double
+_Propensity_to_be_Recruited_Formula(): double
+_Insurgents_Fraction_Value(): double
+_Propensity_to_Protest_Value(): double
+_Regime_Opponents_Formula(): double
+_Dissidents_Fraction_Value(): double
+formulasExecute()
+formulasExecute(_variableNameCode: int)
+getScalarRightPart(_dr: double, _ar: double)
+getIntegrationManager(): ActiveObjectIntegrationManager
+getNameOf(_e: EventTimeout): String
+getModeOf(_e: EventTimeout): int
+getFirstOccurenceTime(_e: EventTimeout): double
+evaluateTimeout(_e: EventTimeout): double
+executeActionOf(_e: EventTimeout)
+getNameOf(ao: ActiveObject): String
+getNameOf(aolist: ActiveObjectCollection): String
+add_people(): Person
+remove_people(object: Person): boolean
-instantiate_people_xjal(index: int): Person
-setupParameters_people_xjal(object: Person, index: int)
-create_people_xjal(object: Person, index: int)
-_people_peopleStat_xjal(): int
-_dsInsurgents_YValue(): double
-_dsGovernment_Supporters_YValue(): double
-_dsDissidents_YValue(): double
-_dsRemoved_Insurgents_YValue(): double
+onShapeClick(_shape: int, index: int, clickx: double, clicky: double): boolean
+executeShapeControlAction(_shape: int, index: int)
+getNameOfShape(_shape: int): String
+getShapeType(_shape: int): int
+getShapeReplicator(_shape: int): int
+getShapeX(_shape: int, index: int): double
+getShapeY(_shape: int, index: int): double
+getShapeEmbeddedObject(_shape: int): Object
+getShapeLineColor(_shape: int, index: int): Color
+getShapeLineStyle(_shape: int, index: int): int
+getShapeLineDx(_shape: int, index: int): double
+getShapeLineDy(_shape: int, index: int): double
-text8_SetDynamicParams(shape: ShapeText)
-text9_SetDynamicParams(shape: ShapeText)
-text10_SetDynamicParams(shape: ShapeText)
-text11_SetDynamicParams(shape: ShapeText)
+getPersistentShape(_shape: int): Object
+drawModelElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean)
+onClickModelAt(panel: Panel, x: double, y: double, clickCount: int, publicOnly: boolean): boolean
+create()
+start()
+onStartup()
+get_Main(): Main
+getEmbeddedObjects(): List
onDestroy()

Main
+Initial_Dissident: int
+Initial_Government_Supporters: int
+Initial_Insurgent: int
+People_Reaction_Delay_On_Anti_Regime_Messages: double
+Regime_Social_and_Economic_Efforts: double
+Contact_Rate: double
+People_Birth: int
+Avg_Time_as_Dissident: double
+Desired_Time_to_Remove_Insurgents: double
+_plot_autoUpdateEvent_xjal: EventTimeout = new EventTimeout(this)
+_plot1_autoUpdateEvent_xjal: EventTimeout = new EventTimeout(this)
+_plot2_autoUpdateEvent_xjal: EventTimeout = new EventTimeout(this)
+aM: ABModel
+sD: SDModel
+dsDifferenceGovernment_Supporters: DataSet = new DataSet(10000) { @Override public void update() { add(time(), _dsDifferenceGovernment_Supporters_YValue()); }}
+dsDifferenceDissidents: DataSet = new DataSet(10000) { @Override public void update() { add(time(), _dsDifferenceDissidents_YValue()); }}
+dsDifferenceRemoved_Insurgents: DataSet = new DataSet(10000) { @Override public void update() { add(time(), _dsDifferenceRemoved_Insurgents_YValue()); }}
+dsDifferenceInsurgents: DataSet = new DataSet(10000) { @Override public void update() { add(time(), _dsDifferenceInsurgents_YValue()); }}
~_text7_Font: Font = new Font("SansSerif", 0, 11)
~_text1_Font: Font = new Font("SansSerif", 1, 12)
~_text2_Font: Font = new Font("SansSerif", 1, 12)
~_text3_Font: Font = new Font("SansSerif", 1, 12)
~_text4_Font: Font = new Font("SansSerif", 0, 11)
~_text5_Font: Font = new Font("SansSerif", 1, 12)
~_text6_Font: Font = new Font("SansSerif", 1, 11)
~_text24_Font: Font = new Font("SansSerif", 0, 18)
~_text25_Font: Font = new Font("SansSerif", 0, 18)
~_text26_Font: Font = new Font("SansSerif", 0, 18)
~_text8_Font: Font = new Font("SansSerif", 0, 32)
~_text10_Font: Font = new Font("SansSerif", 1, 12)
~_text11_Font: Font = new Font("SansSerif", 0, 11)
~_text_Font: Font = new Font("SansSerif", 0, 32)
~_text20_Font: Font = new Font("SansSerif", 0, 18)
~_text21_Font: Font = new Font("SansSerif", 0, 18)
~_text22_Font: Font = new Font("SansSerif", 0, 18)
~_text27_Font: Font = new Font("SansSerif", 0, 18)
~_text33_Font: Font = new Font("SansSerif", 0, 18)
~_text9_Font: Font = new Font("SansSerif", 1, 12)
~_text7: int=1
~_rectangle: int=2
~text1: int=3
~text2: int=4
~text3: int=5
~_text4: int=6
_~text5: int=7
~line5: int=8
~line6: int=9
~line7: int=10
~_text6: int=11
~_menu_group: int=12
~_text24: int=13
~_line61: int=14
~_line62: int=15
~_line57: int=16
~_rectangle37: int=17
~_rectangle38: int=18
~_text25: int=19
~_line63: int=20
~_text26: int=21
~_rectangle39: int=22
~_text64: int=23

~line2: int=24
~_text8: int=25
~_text10: int=26
~_text11: int=27
~_text: int=28
~line1: int=29
~_text20: int=30
~_rectangle35: int=31
~_text21: int=32
~_line56: int=33
~_line58: int=34
~_text22: int=35
~_rectangle36: int=36
~_line59: int=37
~_line60: int=38
~_line66: int=39
~_text27: int=40
~_rectangle40: int=41
~_line67: int=42
~_group2: int=43
~_line76: int=44
~_text33: int=45
~_text9: int=46
~line8: int=47
~_plot: int=48
~_plot1: int=49
~_plot2: int=50
~_presentation: int=0
~_icon: int=-1
~plot: Plot
~plot1: Plot
~plot2: Plot
~text7: ShapeText
~rectangle: ShapeRectangle
~text4: ShapeText
~text5: ShapeText
~text6: ShapeText
~menu_group: ShapeGroup
~text24: ShapeText
~line61: ShapeLine
~line62: ShapeLine
~line57: ShapeLine
~rectangle37: ShapeRectangle
~rectangle38: ShapeRectangle
~text25: ShapeText
~line63: ShapeLine
~text26: ShapeText
~rectangle39: ShapeRectangle
~line64: ShapeLine
~text8: ShapeText
~text10: ShapeText
~text11: ShapeText
~text: ShapeText
~text20: ShapeText
~rectangle35: ShapeRectangle
~text21: ShapeText
~line56: ShapeLine
~line58: ShapeLine
~text22: ShapeText
~rectangle36: ShapeRectangle
~line59: ShapeLine
~line60: ShapeLine
~line66: ShapeLine
~text27: ShapeText
~rectangle40: ShapeRectangle

~line67: ShapeLine
~group2: ShapeGroup
~line76: ShapeLine
~text33: ShapeText
~text9: ShapeText
~presentation: ShapeGroup
~icon: ShapeGroup

<<create>>+Main(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection)
+_Initial_Dissident_DefaultValue_xjal(): int
+set_Initial_Dissident(Initial_Dissident: int)
~onChange_Initial_Dissident()
+_Initial_Government_Supporters_DefaultValue_xjal(): int
+set_Initial_Government_Supporters(Initial_Government_Supporters: int)
~onChange_Initial_Government_Supporters()
+_Initial_Insurgent_DefaultValue_xjal(): int
+set_Initial_Insurgent(Initial_Insurgent: int)
~onChange_Initial_Insurgent()
+_People_Reaction_Delay_On_Anti_Regime_Messages_DefaultValue_xjal(): double
+set_People_Reaction_Delay_On_Anti_Regime_Messages(People_Reaction_Delay_On_Anti_Regime_Messages: double)
~onChange_People_Reaction_Delay_On_Anti_Regime_Messages()
+_Regime_Social_and_Economic_Efforts_DefaultValue_xjal(): double
+set_Regime_Social_and_Economic_Efforts(Regime_Social_and_Economic_Efforts: double)
~onChange_Regime_Social_and_Economic_Efforts()
+_Contact_Rate_DefaultValue_xjal(): double
+set_Contact_Rate(Contact_Rate: double)
~onChange_Contact_Rate()
+_People_Birth_DefaultValue_xjal(): int
+set_People_Birth(People_Birth: int)
~onChange_People_Birth()
+_Avg_Time_as_Dissident_DefaultValue_xjal(): double
+set_Avg_Time_as_Dissident(Avg_Time_as_Dissident: double)
~onChange_Avg_Time_as_Dissident()
+_Desired_Time_to_Remove_Insurgents_DefaultValue_xjal(): double
+set_Desired_Time_to_Remove_Insurgents(Desired_Time_to_Remove_Insurgents: double)
~onChange_Desired_Time_to_Remove_Insurgents()
+getNameOf(_e: EventTimeout): String
+getModeOf(_e: EventTimeout): int
+getFirstOccurenceTime(_e: EventTimeout): double
+evaluateTimeout(_e: EventTimeout): double
+executeActionOf(_e: EventTimeout)
+getNameOf(ao: ActiveObject): String
+getNameOf(collection: ActiveObjectCollection): String
-instantiate_aM_xjal(): ABModel
-setupParameters_aM_xjal(object: ABModel)
-create_aM_xjal(object: ABModel)
-instantiate_sD_xjal(): SDModel
-setupParameters_sD_xjal(object: SDModel)
-create_sD_xjal(object: SDModel)
-_dsDifferenceInsurgents_YValue(): double
-_dsDifferenceGovernment_Supporters_YValue(): double
-_dsDifferenceDissidents_YValue(): double
-_dsDifferenceRemoved_Insurgents_YValue(): double
+onShapeClick(_shape: int, index: int, clickx: double, clicky: double): boolean
+getNameOfShape(_shape: int): String
+getShapeType(_shape: int): int
+getShapeX(_shape: int, index: int): double
+getShapeY(_shape: int, index: int): double
+getShapeLineColor(_shape: int, index: int): Color
+getShapeFillColor(_shape: int, index: int): Color
+getShapeLineStyle(_shape: int, index: int): int
+getShapeLineDx(_shape: int, index: int): double
+getShapeLineDy(_shape: int, index: int): double
+getShapeText(_shape: int, index: int): Object
+getShapeFont(_shape: int, index: int): Font
+getShapeTextAlignment(_shape: int, index: int): int

-text6_SetDynamicParams(shape: ShapeText)
+getPersistentShape(_shape: int): Object
+drawModelElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean)
+onClickModelAt(panel: Panel, x: double, y: double, clickCount: int, publicOnly: boolean): boolean
+create()
+start()
+getEmbeddedObjects(): List
onDestroy()

Person
+statechart: Statechart = new Statechart(this, (short)4)
+Government_Supporter: short=0
+Dissident: short=1
+Insurgent: short=2
+Removed_Insurgent: short=3
+Active_Insurgent: short=4
+Inactive_Insurgent: short=5
+Active_Dissident: short=6
+Inactive_Dissident: short=7
+branch: short=8
+transition10: TransitionRate = new TransitionRate(this)
+transition11: TransitionRate = new TransitionRate(this)
+transition3: TransitionRate = new TransitionRate(this)
+transition7: TransitionRate = new TransitionRate(this)
+transition8: TransitionRate = new TransitionRate(this)
+transition2: TransitionRate = new TransitionRate(this)
+transition5: TransitionRate = new TransitionRate(this)
+transition6: TransitionRate = new TransitionRate(this)
+transition1: TransitionRate = new TransitionRate(this)
+transition: TransitionMessage = new TransitionMessage(this)
+transition4: TransitionMessage = new TransitionMessage(this)
+transition9: TransitionMessage = new TransitionMessage(this)
~_button_Font: Font = new Font("Dialog", 0, 11)
~_button1_Font: Font = new Font("Dialog", 0, 11)
~_text2_Font: Font = new Font("SansSerif", 1, 12)
~_text3_Font: Font = new Font("SansSerif", 0, 12)
~_text4_Font: Font = new Font("SansSerif", 0, 12)
~_text5_Font: Font = new Font("SansSerif", 0, 12)
~_text6_Font: Font = new Font("SansSerif", 0, 12)
~_text1_Font: Font = new Font("SansSerif", 0, 18)
~_text7_Font: Font = new Font("SansSerif", 0, 12)
~_text18_Font: Font = new Font("SansSerif", 0, 18)
~_text17_Font: Font = new Font("SansSerif", 0, 18)
~_text21_Font: Font = new Font("SansSerif", 0, 18)
~_text27_Font: Font = new Font("SansSerif", 0, 18)
~_text_Font: Font = new Font("SansSerif", 0, 32)
~_button: int=1
~_button1: int=2
~_agentRectangle: int=3
~_text2: int=4
~_text3: int=5
~_text4: int=6
~_text5: int=7
~_text6: int=8
~_oval: int=9
~_text1: int=10
~_text7: int=11
~_group: int=12
~_line55: int=13
~_rectangle34: int=14
~_line57: int=15
~_rectangle37: int=16
~_text18: int=17

~_text17: int=18
~_line53: int=19
~_line54: int=20
~_line59: int=21
~_text21: int=22
~_text27: int=23
~_line67: int=24
~_rectangle40: int=25
~line1: int=26
~_text: int=27
~_line66: int=28
~_rectangle39: int=29
~_group1: int=30
~_presentation: int=0
~_icon: int=-1
~button: ShapeButton
~button1: ShapeButton
~agentRectangle: ShapeRectangle
~text2: ShapeText
~text3: ShapeText
~text4: ShapeText
~text5: ShapeText
~text6: ShapeText
~oval: ShapeOval
~text1: ShapeText
~text7: ShapeText
~group: ShapeGroup
~line55: ShapeLine
~rectangle34: ShapeRectangle
~line57: ShapeLine
~rectangle37: ShapeRectangle
~text17: ShapeText
~text18: ShapeText
~line53: ShapeLine
~line54: ShapeLine
~line59: ShapeLine
~text21: ShapeText
~text27: ShapeText
~line67: ShapeLine
~rectangle40: ShapeRectangle
~text: ShapeText
~line66: ShapeLine
~rectangle39: ShapeRectangle
~group1: ShapeGroup
~presentation: ShapeGroup
~icon: ShapeGroup
~_transition_pointsX: int[*] = {200, 200}
~_transition_pointsY: int[*] = {200, 220}
~_transition12_pointsX: int[*] = {200, 200}
~_transition12_pointsY: int[*] = {240, 260}
~_transition10_pointsX: int[*] = {300, 300}
~_transition10_pointsY: int[*] = {260, 200}
~_transition11_pointsX: int[*] = {250, 250}
~_transition11_pointsY: int[*] = {370, 400}
~_transition3_pointsX: int[*] = {250, 250}
~_transition3_pointsY: int[*] = {510, 540}
~_transition7_pointsX: int[*] = {240, 240}
~_transition7_pointsY: int[*] = {450, 470}
~_transition8_pointsX: int[*] = {290, 290}
~_transition8_pointsY: int[*] = {470, 490}
~_transition2_pointsX: int[*] = {310, 310, 330}
~_transition2_pointsY: int[*] = {420, 440, 440}
~_transition5_pointsX: int[*] = {240, 240}
~_transition5_pointsY: int[*] = {310, 330}
~_transition6_pointsX: int[*] = {290, 290}

~_transition6_pointsY: int[*] = {330, 310} ~_transition1_pointsX: int[*] = {310, 310, 330} ~_transition1_pointsY: int[*] = {280, 300, 300} ~_transition13_pointsX: int[*] = {212, 250, 250} ~_transition13_pointsY: int[*] = {230, 230, 200} ~_transition4_pointsX: int[*] = {150, 120, 120, 150} ~_transition4_pointsY: int[*] = {190, 190, 320, 320} ~_transition9_pointsX: int[*] = {150, 100, 100, 150} ~_transition9_pointsY: int[*] = {180, 180, 460, 460}
<<create>>+Person(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection) +getNameOf(_s: Statechart): String +executeActionOf(_s: Statechart) +getNameOfState(_state: short): String +stateContainsState(compstate: short, simpstate: short): boolean +getContainerStateOf(_state: short): short +enterState(_state: short, _destination: boolean) +exitState(_state: short, _t: Transition, _source: boolean, _statechart: Statechart) +getNameOf(_t: TransitionRate): String +getStatechartOf(_t: TransitionRate): Statechart +executeActionOf(_t: TransitionRate) +evaluateRateOf(_t: TransitionRate): double +getNameOf(_t: TransitionMessage): String +getStatechartOf(_t: TransitionMessage): Statechart +executeActionOf(_t: TransitionMessage, _msg: Object) +testMessageOf(_t: TransitionMessage, _msg: Object): boolean +onShapeClick(_shape: int, index: int, clickx: double, clicky: double): boolean +executeShapeControlAction(_shape: int, index: int) +getNameOfShape(_shape: int): String +getShapeType(_shape: int): int +isShapePublic(_shape: int): boolean +getShapeX(_shape: int, index: int): double +getShapeY(_shape: int, index: int): double +getShapeLineColor(_shape: int, index: int): Color +getShapeLineStyle(_shape: int, index: int): int +getShapeLineDx(_shape: int, index: int): double +getShapeLineDy(_shape: int, index: int): double -_agentRectangle_SetDynamicParams(shape: ShapeRectangle) -_text5_SetDynamicParams(shape: ShapeText) -_text6_SetDynamicParams(shape: ShapeText) -_group_SetDynamicParams(shape: ShapeText) -_group1_SetDynamicParams(shape: ShapeText) +getPersistentShape(_shape: int): Object +drawModelElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean) +onReceive(msg: Object, sender: Agent) +create() +start() +get_ABModel(): ABModel +onChange() +onDestroy()

Simulation
+Initial_Dissident: int=450 +Initial_Government_Supporters: int=2000 +Initial_Insurgent: int=250 +People_Reaction_Delay_On_Anti_Regime_Messages: int=10 +Regime_Social_and_Economic_Efforts: double=0.12 +Contact_Rate: double=20 +People_Birth: int=1 +Avg_Time_as_Dissident: double=15.0 +Desired_Time_to_Remove_Insurgents: double=15.0 ~_button_Font: Font = Font("Dialog", 0, 11) ~_text_Font: Font = new Font("SansSerif", 0, 32) ~_text1_Font: Font = new Font("SansSerif", 0, 11)

```
~_text2_Font: Font = new Font("SansSerif", 0, 12)
~_text3_Font: Font = new Font("SansSerif", 0, 12)
~_text4_Font: Font = new Font("SansSerif", 0, 12)
~_text5_Font: Font = new Font("SansSerif", 0, 12)
~_text6_Font: Font = new Font("SansSerif", 1, 12)
~_text7_Font: Font = new Font("SansSerif", 1, 12)
~_text8_Font: Font = new Font("SansSerif", 1, 12)
~_text9_Font: Font = new Font("SansSerif", 1, 12)
~_text10_Font: Font = new Font("SansSerif", 1, 12)
~_text11_Font: Font = new Font("SansSerif", 0, 12)
~_text12_Font: Font = new Font("SansSerif", 1, 12)
~_text13_Font: Font = new Font("SansSerif", 1, 12)
~_text14_Font: Font = new Font("SansSerif", 1, 12)
~_text15_Font: Font = new Font("SansSerif", 0, 12)
~_text16_Font: Font = new Font("SansSerif", 0, 12)
~_text17_Font: Font = new Font("SansSerif", 0, 12)
~_text18_Font: Font = new Font("SansSerif", 0, 12)
~_text19_Font: Font = new Font("SansSerif", 1, 12)
~_text20_Font: Font = new Font("SansSerif", 1, 12)
~_text21_Font: Font = new Font("SansSerif", 0, 11)
~_text22_Font: Font = new Font("SansSerif", 1, 12)
~_text23_Font: Font = new Font("SansSerif", 0, 9)
~_text23_Color: Color = new Color(0xFF2C5373, true)
~_button: int=1
~slider1: int=2
~slider0: int=3
~slider2: int=4
~slider3: int=5
~slider4: int=6
~slider5: int=7
~slider6: int=8
~slider7: int=9
~slider8: int=10
~_text: int=11
~_text4: int=12
~_text3: int=13
~text7: int=14
~text6: int=15
~_text2: int=16
~text8: int=17
~text9: int=18
~_text5: int=19
~text10: int=20
~_text11: int=21
~line1: int=22
~text12: int=23
~text13: int=24
~text14: int=25
~_text15: int=26
~_text16: int=27
~line5: int=28
~_text17: int=29
~_text28: int=30
~text19: int=31
~text20: int=32
~_text21: int=33
~_text22: int=34
~_text1: int=35
~_image: int=36
~_text23: int=37
~_polyline: int=38
~_presentation: int=0
~_icon: int=-1
~button: ShapeButton
~text: ShapeText
```

~text1: ShapeText
~text2: ShapeText
~text3: ShapeText
~text4: ShapeText
~text5: ShapeText
~text11: ShapeText
~text15: ShapeText
~text16: ShapeText
~text17: ShapeText
~text18: ShapeText
~text21: ShapeText
~text22: ShapeText
~text23: ShapeText
~image: ShapeImage
~polyline: ShapePolyLine
~presentation: ShapeGroup
~icon: ShapeGroup

+drawModelElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean)
+onClickModelAt(panel: Panel, x: double, y: double, clickCount: int, publicOnly: boolean): boolean
+executeShapeControlAction(_shape: int, index: int)
+executeShapeControlAction(_shape: int, index: int, value: double)
+getShapeControlMinimum(_shape: int, index: int): double
+getShapeControlMaximum(_shape: int, index: int): double
+getShapeControlDefaultValueDouble(_shape: int, index: int): double
+getNameOfShape(_shape: int): String
+getShapeType(_shape: int): int
+isShapePublic(_shape: int): boolean
+getShapeX(_shape: int, index: int): double
+getShapeY(_shape: int, index: int): double
+getShapeLineColor(_shape: int, index: int): Color
+getShapeFillColor(_shape: int, index: int): Color
+getShapeLineStyle(_shape: int, index: int): int
+getShapeLineDx(_shape: int, index: int): double
+getShapeLineDy(_shape: int, index: int): double
+getShapeWidth(_shape: int, index: int): double
+getShapeHeight(_shape: int, index: int): double
+getShapeText(_shape: int, index: int): Object
+getShapeFont(_shape: int, index: int): Font
+getShapeTextAlignment(_shape: int, index: int): int
+isShapeControlEnabled(_shape: int, index: int): boolean
+isShapeControlVertical(_shape: int, index: int): boolean
-_button_SetDynamicParams(shape: ShapeButton)
+getPersistentShape(_shape: int): Object
+getWindowWidth(): int
+getWindowHeight(): int
+initDefaultRandomNumberGenerator(engine: Engine)
+createRoot(engine: Engine): Main
+setupRootParameters(root: Main, callOnChangeActions: boolean)
+setupEngine(engine: Engine)
+setup(applet: JApplet)

Κώδικας Κεφαλαίου 9 – Έργο Insurgency Dynamics

ABModel.java

```
package insurgencySDABModel;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import javax.swing.JApplet;


public class Simulation extends ExperimentSimulation<Main> {
  // Plain Variables
  public int Initial_Government_Supporters = 2000 ;
  public int Initial_Dissident = 450 ;
  public int Initial_Insurgent = 250 ;
  public int People_Reaction_Delay_On_Anti_Regime_Messages = 10 ;
  public double Regime_Social_and_Economic_Efforts = 0.12 ;
  public double Contact_Rate = 20 ;
  public int People_Birth = 1 ;
  public double Avg_Time_as_Dissident = 15.0 ;
  public double Desired_Time_to_Remove_Insurgents = 15.0 ;


  @Override
  public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, -300, 150,
        10, 0,
       "Initial_Government_Supporters",  Initial_Government_Supporters,
false );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, -300, 180,
        10, 0,
        "Initial_Dissident", Initial_Dissident, false );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, -300, 210,
        10, 0,
        "Initial_Insurgent", Initial_Insurgent, false );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, -300, 300,
        10, 0,
        "People_Reaction_Delay_On_Anti_Regime_Messages",
People_Reaction_Delay_On_Anti_Regime_Messages, false );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, -300, 330,
        10, 0,
       "Regime_Social_and_Economic_Efforts",
Regime_Social_and_Economic_Efforts, false );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, -300, 360,
        10, 0,
       "Contact_Rate", Contact_Rate, false );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, -300, 390,
        10, 0,
       "People_Birth", People_Birth, false );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, -300, 420,
        10, 0,
       "Avg_Time_as_Dissident", Avg_Time_as_Dissident, false );
    }
    if (!_publicOnly) {
      drawPlainVariable( _panel, _g, -300, 450,
        10, 0,
       "Desired_Time_to_Remove_Insurgents",
Desired_Time_to_Remove_Insurgents, false );
    }
}
  @Override
```

```
 public boolean onClickModelAt( Panel panel, double x, double y, int clickCount, boolean publicOnly ) {
  if( !publicOnly && modelElementContains(x, y, -300, 150) ){
   panel.addInspect( -300, 150, this, "Initial_Government_Supporters" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -300, 180) ){
   panel.addInspect( -300, 180, this, "Initial_Dissident" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -300, 210) ){
   panel.addInspect( -300, 210, this, "Initial_Insurgent" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -300, 300) ){
   panel.addInspect( -300, 300, this, "People_Reaction_Delay_On_Anti_Regime_Messages" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -300, 330) ){
   panel.addInspect( -300, 330, this, "Regime_Social_and_Economic_Efforts" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -300, 360) ){
   panel.addInspect( -300, 360, this, "Contact_Rate" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -300, 390) ){
   panel.addInspect( -300, 390, this, "People_Birth" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -300, 420) ){
   panel.addInspect( -300, 420, this, "Avg_Time_as_Dissident" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -300, 450) ){
   panel.addInspect( -300, 450, this, "Desired_Time_to_Remove_Insurgents" );
   return true;
  }
  return false;
 }
 static final Font _button_Font = new Font("Dialog", 0, 11 );
 static final Font _text_Font = new Font("SansSerif", 0, 32 );
 static final Font _text4_Font = new Font("SansSerif", 0, 12 );
 static final Font _text3_Font = _text4_Font;
 static final Font _text7_Font = new Font("SansSerif", 1, 12 );
 static final Font _text6_Font = _text7_Font;
 static final Font _text2_Font = _text4_Font;
 static final Font _text8_Font = _text7_Font;
 static final Font _text9_Font = _text7_Font;
 static final Font _text5_Font = _text4_Font;
 static final Font _text10_Font = _text7_Font;
 static final Font _text11_Font = _text4_Font;
 static final Font _text12_Font = _text7_Font;
 static final Font _text13_Font = _text7_Font;
 static final Font _text14_Font = _text7_Font;
 static final Font _text15_Font = _text4_Font;
 static final Font _text16_Font = _text4_Font;
 static final Font _text17_Font = _text4_Font;
 static final Font _text18_Font = _text4_Font;
 static final Font _text19_Font = _text7_Font;
 static final Font _text20_Font = _text7_Font;
 static final Font _text21_Font = new Font("SansSerif", 0, 11 );
 static final Font _text22_Font = _text7_Font;
 static final Font _text1_Font = _text21_Font;
 static final Font _text23_Font = new Font("SansSerif", 0, 9 );
 static final Color _text23_Color = new Color( 0xFF2C5373, true );
 static final int _button = 1;
 static final int slider1 = 2;
 static final int slider0 = 3;
 static final int slider2 = 4;
 static final int slider3 = 5;
 static final int slider4 = 6;
 static final int slider5 = 7;
 static final int slider6 = 8;
 static final int slider7 = 9;
 static final int slider8 = 10;
 static final int _text = 11;
 static final int _text4 = 12;
 static final int _text3 = 13;
 static final int text7 = 14;
 static final int text6 = 15;
 static final int _text2 = 16;
 static final int text8 = 17;
 static final int text9 = 18;
 static final int _text5 = 19;
 static final int text10 = 20;
 static final int _text11 = 21;
 static final int line1 = 22;
 static final int text12 = 23;
 static final int text13 = 24;
 static final int text14 = 25;
 static final int _text15 = 26;
 static final int _text16 = 27;
 static final int line5 = 28;
 static final int _text17 = 29;
 static final int _text18 = 30;
 static final int text19 = 31;
 static final int text20 = 32;
 static final int _text21 = 33;
 static final int _text22 = 34;
 static final int _text1 = 35;
 static final int _image = 36;
 static final int _text23 = 37;
 static final int _polyline = 38;

 /**
  * Top-level presentation group id
  */
 static final int _presentation = 0;

 /**
  * Top-level icon group id
  */
 static final int _icon = -1;

 static final double[] _polyline_pointsDX_xjal() {
   return new double[] { 0, 0, -10,  };
 }
 static final double[] _polyline_pointsDY_xjal() {
   return new double[] { 0, 20, 10,  };
 }

 @Override
 public void executeShapeControlAction( int _shape, int index ) {
   switch( _shape ) {
     case _button: {
run();
getEngine().getPresentation().setPresentable(
((Main)getEngine().getRoot()).aM );
```

```
;}
      break;
    default:
      super.executeShapeControlAction( _shape, index );
      break;
  }
 }

 @Override
 public void executeShapeControlAction( int _shape, int index, double
value ) {
  switch( _shape ) {
    case slider1:
      Initial_Dissident = (int) value;
      break;
    case slider0:
      Initial_Government_Supporters = (int) value;
      break;
    case slider2:
      Initial_Insurgent = (int) value;
      break;
    case slider3:
      Regime_Social_and_Economic_Efforts = value;
      break;
    case slider4:
      People_Reaction_Delay_On_Anti_Regime_Messages = (int) value;
      break;
    case slider5:
      Contact_Rate = value;
      break;
    case slider6:
      People_Birth = (int) value;
      break;
    case slider7:
      Desired_Time_to_Remove_Insurgents = value;
      break;
    case slider8:
      Avg_Time_as_Dissident = value;
      break;
    default:
      super.executeShapeControlAction( _shape, index, value );
      break;
  }
 }

 @Override
 public int getShapeControlValueType( int _shape, int index ) {
  switch( _shape ) {
    case slider1: return ShapeControl.TYPE_INT;
    case slider0: return ShapeControl.TYPE_INT;
    case slider2: return ShapeControl.TYPE_INT;
    case slider3: return ShapeControl.TYPE_DOUBLE;
    case slider4: return ShapeControl.TYPE_INT;
    case slider5: return ShapeControl.TYPE_DOUBLE;
    case slider6: return ShapeControl.TYPE_INT;
    case slider7: return ShapeControl.TYPE_DOUBLE;
    case slider8: return ShapeControl.TYPE_DOUBLE;
    default: return super.getShapeControlValueType( _shape, index );
  }
 }

 @Override
 public double getShapeControlMinimum( int _shape, int index ) {
  switch( _shape ) {
    case slider1: return 300 ;
    case slider0: return 1190 ;
    case slider2: return 200 ;
    case slider3: return 0.001 ;
    case slider4: return 0.01 ;
    case slider5: return 5 ;
    case slider6: return 0 ;
    case slider7: return 10 ;
    case slider8: return 10 ;
    default: return super.getShapeControlMinimum( _shape, index );
  }
 }

 @Override
 public double getShapeControlMaximum( int _shape, int index ) {
  switch( _shape ) {
    case slider1: return 1000 ;
    case slider0: return 10000 ;
    case slider2: return 800 ;
    case slider3: return 1 ;
    case slider4: return 50 ;
    case slider5: return 50 ;
    case slider6: return 5 ;
    case slider7: return 30 ;
    case slider8: return 30 ;
    default: return super.getShapeControlMaximum( _shape, index );
  }
 }

 @Override
 public double getShapeControlDefaultValueDouble( int _shape, int index
) {
  switch(_shape) {
    case slider1: return Initial_Dissident ;
    case slider0: return Initial_Government_Supporters ;
    case slider2: return Initial_Insurgent ;
    case slider3: return Regime_Social_and_Economic_Efforts ;
    case slider4: return
People_Reaction_Delay_On_Anti_Regime_Messages ;
    case slider5: return Contact_Rate ;
    case slider6: return People_Birth ;
    case slider7: return Desired_Time_to_Remove_Insurgents ;
    case slider8: return Avg_Time_as_Dissident ;
    default: return super.getShapeControlDefaultValueDouble( _shape,
index );
  }
 }

 @Override
 public String getNameOfShape( int _shape ) {
  switch( _shape ) {
    case text7: return "text7";
    case text6: return "text6";
    case text8: return "text8";
    case text9: return "text9";
    case text10: return "text10";
    case line1: return "line1";
    case text12: return "text12";
    case text13: return "text13";
    case text14: return "text14";
    case line5: return "line5";
    case text19: return "text19";
    case text20: return "text20";
    case slider1: return "slider1";
    case slider0: return "slider0";
    case slider2: return "slider2";
    case slider3: return "slider3";
    case slider4: return "slider4";
```

```
    case slider5: return "slider5";
    case slider6: return "slider6";
    case slider7: return "slider7";
    case slider8: return "slider8";
    default: return super.getNameOfShape( _shape );
  }
 }

 @Override
 public int getShapeType( int _shape ) {
  switch( _shape ) {
    case text7: return SHAPE_TEXT;
    case text6: return SHAPE_TEXT;
    case text8: return SHAPE_TEXT;
    case text9: return SHAPE_TEXT;
    case text10: return SHAPE_TEXT;
    case line1: return SHAPE_LINE;
    case text12: return SHAPE_TEXT;
    case text13: return SHAPE_TEXT;
    case text14: return SHAPE_TEXT;
    case line5: return SHAPE_LINE;
    case text19: return SHAPE_TEXT;
    case text20: return SHAPE_TEXT;
    case slider1: return SHAPE_SLIDER;
    case slider0: return SHAPE_SLIDER;
    case slider2: return SHAPE_SLIDER;
    case slider3: return SHAPE_SLIDER;
    case slider4: return SHAPE_SLIDER;
    case slider5: return SHAPE_SLIDER;
    case slider6: return SHAPE_SLIDER;
    case slider7: return SHAPE_SLIDER;
    case slider8: return SHAPE_SLIDER;
    default: return super.getShapeType( _shape );
  }
 }

 @Override
 public boolean isShapePublic( int _shape ) {
  switch( _shape ) {
    case text7:
    case text6:
    case text8:
    case text9:
    case text10:
    case text13:
    case text14:
    case text19:
    case text20:
    case slider1:
    case slider0:
    case slider2:
    case slider3:
    case slider4:
    case slider5:
    case slider6:
    case slider7:
    case slider8:
      return false;
    default:
      return true;
  }
 }

 @Override
 public double getShapeX( int _shape, int index ) {
  switch( _shape ) {
    case text7: return 346;
    case text6: return 346;
    case text8: return 346;
    case text9: return 350;
    case text10: return 350;
    case line1: return 30;
    case text12: return 45;
    case text13: return 350;
    case text14: return 350;
    case line5: return 120;
    case text19: return 350;
    case text20: return 350;
    case slider1: return 40;
    case slider0: return 40;
    case slider2: return 40;
    case slider3: return 40;
    case slider4: return 40;
    case slider5: return 40;
    case slider6: return 40;
    case slider7: return 40;
    case slider8: return 40;
    default: return super.getShapeX( _shape, index );
  }
 }

 @Override
 public double getShapeY( int _shape, int index ) {
  switch( _shape ) {
    case text7: return 170;
    case text6: return 130;
    case text8: return 210;
    case text9: return 279;
    case text10: return 319;
    case line1: return 50;
    case text12: return 90;
    case text13: return 359;
    case text14: return 399;
    case line5: return 100;
    case text19: return 439;
    case text20: return 479;
    case slider1: return 181;
    case slider0: return 141;
    case slider2: return 221;
    case slider3: return 331;
    case slider4: return 291;
    case slider5: return 371;
    case slider6: return 411;
    case slider7: return 491;
    case slider8: return 451;
    default: return super.getShapeY( _shape, index );
  }
 }

 @Override
 public Color getShapeLineColor( int _shape, int index ) {
  switch( _shape ) {
    // shapes
    case line1: return slateGray;
    case line5: return slateGray;
    // controls (text color)
    // charts (border color)
    default: return super.getShapeLineColor( _shape, index );
  }
 }
```

```
@Override
public Color getShapeFillColor( int _shape, int index ) {
  switch( _shape ) {
    case text7: return black;
    case text6: return black;
    case text8: return black;
    case text9: return black;
    case text10: return black;
    case text12: return black;
    case text13: return black;
    case text14: return black;
    case text19: return black;
    case text20: return black;
    case slider1: return transparent;
    case slider0: return transparent;
    case slider2: return transparent;
    case slider3: return transparent;
    case slider4: return transparent;
    case slider5: return transparent;
    case slider6: return transparent;
    case slider7: return transparent;
    case slider8: return transparent;
    default: return super.getShapeFillColor( _shape, index );
  }
}

@Override
public int getShapeLineStyle( int _shape, int index ) {
  switch( _shape ) {
    case line1: return LINE_STYLE_SOLID;
    case line5: return LINE_STYLE_SOLID;
    default: return super.getShapeLineStyle( _shape, index );
  }
}

@Override
public double getShapeLineDx( int _shape, int index ) {
  switch( _shape ) {
    case line1: return 970;
    case line5: return 230;
    default: return super.getShapeLineDx( _shape, index );
  }
}

@Override
public double getShapeLineDy( int _shape, int index ) {
  switch( _shape ) {
    case line1: return 0;
    case line5: return 0;
    default: return super.getShapeLineDy( _shape, index );
  }
}

@Override
public double getShapeWidth( int _shape, int index ) {
  switch( _shape ) {
    case slider1: return 316;
    case slider0: return 316;
    case slider2: return 316;
    case slider3: return 316;
    case slider4: return 316;
    case slider5: return 316;
    case slider6: return 316;
    case slider7: return 316;
    case slider8: return 316;
    default: return super.getShapeWidth( _shape, index );
  }
}


@Override
public double getShapeHeight( int _shape, int index ) {
  switch( _shape ) {
    case slider1: return 30;
    case slider0: return 30;
    case slider2: return 30;
    case slider3: return 30;
    case slider4: return 30;
    case slider5: return 30;
    case slider6: return 30;
    case slider7: return 30;
    case slider8: return 30;
    default: return super.getShapeHeight( _shape, index );
  }
}

@Override
public Object getShapeText( int _shape, int index ) {
  switch( _shape ) {
    case text7: return
format(Initial_Dissident)
;
    case text6: return
format(Initial_Government_Supporters)
;
    case text8: return
format(Initial_Insurgent)
;
    case text9: return
format(People_Reaction_Delay_On_Anti_Regime_Messages)
;
    case text10: return
format(Regime_Social_and_Economic_Efforts)
;
    case text12: return "Parameters";
    case text13: return
format(Contact_Rate)
;
    case text14: return
format(People_Birth)
;
    case text19: return
format(Avg_Time_as_Dissident)
;
    case text20: return
format(Desired_Time_to_Remove_Insurgents)
;
    default: return super.getShapeText( _shape, index );
  }
}

@Override
public Font getShapeFont( int _shape, int index ) {
  switch( _shape ) {
    case text7: return _text7_Font;
    case text6: return _text6_Font;
    case text8: return _text8_Font;
    case text9: return _text9_Font;
    case text10: return _text10_Font;
```

```
    case text12: return _text12_Font;
    case text13: return _text13_Font;
    case text14: return _text14_Font;
    case text19: return _text19_Font;
    case text20: return _text20_Font;
    default: return super.getShapeFont( _shape, index );
   }
  }

  @Override
  public int getShapeTextAlignment( int _shape, int index ) {
   switch( _shape ) {
    case text7: return ALIGNMENT_RIGHT;
    case text6: return ALIGNMENT_RIGHT;
    case text8: return ALIGNMENT_RIGHT;
    case text9: return ALIGNMENT_RIGHT;
    case text10: return ALIGNMENT_RIGHT;
    case text12: return ALIGNMENT_LEFT;
    case text13: return ALIGNMENT_RIGHT;
    case text14: return ALIGNMENT_RIGHT;
    case text19: return ALIGNMENT_RIGHT;
    case text20: return ALIGNMENT_RIGHT;
    default: return super.getShapeTextAlignment( _shape, index );
   }
  }

  @Override
  public boolean isShapeControlEnabled( int _shape, int index ) {
   switch( _shape ) {
    case slider1: return true;
    case slider0: return true;
    case slider2: return true;
    case slider3: return true;
    case slider4: return true;
    case slider5: return true;
    case slider6: return true;
    case slider7: return true;
    case slider8: return true;
    default: return super.isShapeControlEnabled( _shape, index );
   }
  }

  @Override
  public boolean isShapeControlVertical( int _shape, int index ) {
   switch( _shape ) {
    case slider1: return false;
    case slider0: return false;
    case slider2: return false;
    case slider3: return false;
    case slider4: return false;
    case slider5: return false;
    case slider6: return false;
    case slider7: return false;
    case slider8: return false;
    default: return super.isShapeControlVertical( _shape, index );
   }
  }


  /**
   * <i>This method should not be called by user</i>
   */
  private void _button_SetDynamicParams( ShapeButton shape ) {
   boolean _visible;
   shape.setEnabled(
getState() == IDLE
);
  }

  ShapeButton button;
  ShapeText text;
  ShapeText text4;
  ShapeText text3;
  ShapeText text2;
  ShapeText text5;
  ShapeText text11;
  ShapeText text15;
  ShapeText text16;
  ShapeText text17;
  ShapeText text18;
  ShapeText text21;
  ShapeText text22;
  ShapeText text1;
  ShapeImage image;
  ShapeText text23;
  ShapePolyLine polyline;

  // Static initialization of persistent elements
  {
   button = new ShapeButton(
                                        Simulation.this, true, 50, 560,
                                        150, 30,
       controlDefault, controlDefault,
       _button_Font,
                                        "Run"
   ) {

    @Override
    public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
     _button_SetDynamicParams( this );
     super.draw( panel, graphics, xform, publicOnly );
    }

    @Override
    public void action(){
     executeShapeControlAction( _button, 0 );
    }
   };
   text = new ShapeText(
     true,30, 10, 0.0,
     slateGray,"SD+AB Insurgency Models",
     _text_Font, ALIGNMENT_LEFT
   );
   text4 = new ShapeText(
     false,46, 130, 0.0,
     black,"Initial government supporters",
     _text4_Font, ALIGNMENT_LEFT
   );
   text3 = new ShapeText(
     false,46, 170, 0.0,
     black,"Initial dissidents",
     _text3_Font, ALIGNMENT_LEFT
   );
   text2 = new ShapeText(
     false,46, 210, 0.0,
     black,"Initial insurgents",
     _text2_Font, ALIGNMENT_LEFT
   );
   text5 = new ShapeText(
     false,50, 279, 0.0,
```

```
    black,"People reaction delay on anti-regime messages",
    _text5_Font, ALIGNMENT_LEFT
  );
  text11 = new ShapeText(
    false,50, 319, 0.0,
    black,"Regime social and economic efforts",
    _text11_Font, ALIGNMENT_LEFT
  );
  text15 = new ShapeText(
    false,50, 399, 0.0,
    black,"People birth",
    _text15_Font, ALIGNMENT_LEFT
  );
  text16 = new ShapeText(
    false,50, 359, 0.0,
    black,"Contact rate",
    _text16_Font, ALIGNMENT_LEFT
  );
  text17 = new ShapeText(
    false,50, 439, 0.0,
    black,"Average time as dissident",
    _text17_Font, ALIGNMENT_LEFT
  );
  text18 = new ShapeText(
    false,50, 479, 0.0,
    black,"Desired time to remove insurgents",
    _text18_Font, ALIGNMENT_LEFT
  );
  text21 = new ShapeText(
    true,420, 120, 0.0,
    black,"This is a combined System Dynamics / Agent Based model of insurgency in a society. The \r\nmodel contains the original System Dynamics version and calculates the difference between \r\nthe mixed model and the SD model. From the dynamics of insurgence viewpoint the \r\npopulation is divided into four groups, namely Government_Supporters, Dissidents, Insurgents and \r\nRemoved_Insurgents. The model starts with Government_Supporters = 2000, Dissidents = 450, \r\nInsurgents = 250. Dissidents and Insurgents communicate with Government_Supporters recruiting \r\npeople to become dissidents. This flow of new dissidents depends on several \r\nfactors: number of active regime opponents (a fraction of dissidents plus insurgents), \r\nfrequency of contacts (Contact_Rate) and probability of recruiting (Propensity_to_be_Recruited). \r\nThere is a process of dissident relaxation: not all the dissidents necessarily become insurgents. \r\nA fraction of the dissident group is appeased by the government and returns to supporting \r\npopulation. This returning flow depends on an average time as dissident (Avg_Time_as_Dissident). \r\nThe remaining part of the dissidents eventually becomes insurgents. The government tries to reduce \r\nthe size of the insurgents group. As a result of these efforts insurgents might be removed from \r\nthe system (Removed_Insurgents). Strength of the efforts depends on resources allocated and \r\nis proportional to the size of the insurgents group. There is a feedback loop from dissidents and \r\ninsurgents groups to the probability of recruiting mentioned at the very beginning of this description. \r\nThe feedback is implemented as a flow of anti-regime messages generated by a certain fraction \r\nof active dissidents and active insurgents. This flow of messages with a certain delay reaches a \r\nsocial network thus making the probability higher.\r\n\r\nRefer to Description page of the model for more details (available at runtime).\r\n\r\nThe model is based on a working paper "Using System Dynamics to Model and Better \r\nUnderstand State Stability" Nazli Choucri, Daniel Goldsmith, Stuart E. Madnick, Dinsha Mistree, \r\nJ. Bradley Morrison, Michael D. Siegel\r\n",
    _text21_Font, ALIGNMENT_LEFT
  );
  text22 = new ShapeText(
    true,420, 90, 0.0,
    black,"Model description",
    _text22_Font, ALIGNMENT_LEFT
  );
  text1 = new ShapeText(
    true,230, 570, 0.0,
    slateGray,"Press \"Run\", the model starts in a while - please wait while agents are created",
    _text1_Font, ALIGNMENT_LEFT
  );
  image = new ShapeImage(
                    Simulation.this, true, 420, 470, 0.0,
                                116,                    40,
"/insurgencySDABModel/",
                                new     String[]{"xjlogo.png",}

  );
  text23 = new ShapeText(
    true,460, 500, 0.0,
    _text23_Color,"This AnyLogic™ model is © 1992-2008 XJ Technologies. www.anylogic.com\r\n",
    _text23_Font, ALIGNMENT_LEFT
  );
  polyline = new ShapePolyLine(
                    true,220, 565,
      null, red,
      3, _polyline_pointsDX_xjal(), _polyline_pointsDY_xjal(),
      false, 1, LINE_STYLE_SOLID
  );
 }
 ShapeGroup presentation;
 ShapeGroup icon;

 @Override
 public Object getPersistentShape( int _shape ) {
  switch(_shape){
   case _presentation: return presentation;
   case _icon: return icon;
   case _button: return button;
   case _text: return text;
   case _text4: return text4;
   case _text3: return text3;
   case _text2: return text2;
   case _text5: return text5;
   case _text11: return text11;
   case _text15: return text15;
   case _text16: return text16;
   case _text17: return text17;
   case _text18: return text18;
   case _text21: return text21;
   case _text22: return text22;
   case _text1: return text1;
   case _image: return image;
   case _text23: return text23;
   case _polyline: return polyline;
   default: return null;
  }
 }

 @Override
 public int getWindowWidth() {
  return 1024;
 }

 @Override
 public int getWindowHeight() {
  return 681;
 }

 /**
  * Applet class to run experiment as java applet
  */
 public static class Applet extends JApplet {

  Simulation ex;
```

```
  @Override
  public void init() {
    ex = new Simulation();
    ex.setup( this );
  }

  @Override
  public void destroy() {
    ex.close();
  }

}

@Override
public void initDefaultRandomNumberGenerator(Engine engine) {
  engine.getDefaultRandomGenerator().setSeed( 1 );
}

@Override
public Main createRoot( Engine engine ) {
  // Create the root object
  return new Main( engine, null, null );
}

@Override
public void setupRootParameters( Main root, boolean
callOnChangeActions ) {
  int Initial_Dissident_xjal =
Initial_Dissident
;
  if (callOnChangeActions) {
    root.set_Initial_Dissident( Initial_Dissident_xjal );
  } else {
    root.Initial_Dissident = Initial_Dissident_xjal;
  }
  int Initial_Government_Supporters_xjal =
Initial_Government_Supporters
;
  if (callOnChangeActions) {
    root.set_Initial_Government_Supporters(
Initial_Government_Supporters_xjal );
  } else {
    root.Initial_Government_Supporters =
Initial_Government_Supporters_xjal;
  }
  int Initial_Insurgent_xjal =
Initial_Insurgent
;
  if (callOnChangeActions) {
    root.set_Initial_Insurgent( Initial_Insurgent_xjal );
  } else {
    root.Initial_Insurgent = Initial_Insurgent_xjal;
  }
  double People_Reaction_Delay_On_Anti_Regime_Messages_xjal =
People_Reaction_Delay_On_Anti_Regime_Messages
;
  if (callOnChangeActions) {
    root.set_People_Reaction_Delay_On_Anti_Regime_Messages(
People_Reaction_Delay_On_Anti_Regime_Messages_xjal );
  } else {
    root.People_Reaction_Delay_On_Anti_Regime_Messages =
People_Reaction_Delay_On_Anti_Regime_Messages_xjal;
  }
  double Regime_Social_and_Economic_Efforts_xjal =
Regime_Social_and_Economic_Efforts
;
  if (callOnChangeActions) {
    root.set_Regime_Social_and_Economic_Efforts(
Regime_Social_and_Economic_Efforts_xjal );
  } else {
    root.Regime_Social_and_Economic_Efforts =
Regime_Social_and_Economic_Efforts_xjal;
  }
  double Contact_Rate_xjal =
Contact_Rate
;
  if (callOnChangeActions) {
    root.set_Contact_Rate( Contact_Rate_xjal );
  } else {
    root.Contact_Rate = Contact_Rate_xjal;
  }
  int People_Birth_xjal =
People_Birth
;
  if (callOnChangeActions) {
    root.set_People_Birth( People_Birth_xjal );
  } else {
    root.People_Birth = People_Birth_xjal;
  }
  double Avg_Time_as_Dissident_xjal =
Avg_Time_as_Dissident
;
  if (callOnChangeActions) {
    root.set_Avg_Time_as_Dissident( Avg_Time_as_Dissident_xjal );
  } else {
    root.Avg_Time_as_Dissident = Avg_Time_as_Dissident_xjal;
  }
  double Desired_Time_to_Remove_Insurgents_xjal =
Desired_Time_to_Remove_Insurgents
;
  if (callOnChangeActions) {
    root.set_Desired_Time_to_Remove_Insurgents(
Desired_Time_to_Remove_Insurgents_xjal );
  } else {
    root.Desired_Time_to_Remove_Insurgents =
Desired_Time_to_Remove_Insurgents_xjal;
  }
}


/**
 * Engine setup
 */
@Override
public void setupEngine(Engine engine) {
  engine.setATOL( 1.0E-5 );
  engine.setRTOL( 1.0E-5 );
  engine.setTTOL( 1.0E-5 );
  engine.setHTOL( 0.0010 );
  engine.setSolverODE( Engine.SOLVER_ODE_EULER );
  engine.setSolverNAE( Engine.SOLVER_NAE_MODIFIED_NEWTON );
  engine.setSolverDAE( Engine.SOLVER_DAE_RK45_NEWTON );
  engine.setVMethods( 16032 );

  engine.setStartTime( 0.0 );
  engine.setTimeUnit( TIME_UNIT_DAY );
  engine.setStopTime( 100.0 );
  engine.setRealTimeMode( true );
  engine.setRealTimeScale( 1.0 );
}

/**
 * Experiment setup
```

```
   */
  @Override
  public void setup( JApplet applet ) {
    setName( "InsurgencyUnitedModel : Simulation" );

    // Dynamic initialization of persistent elements
    presentation = new ShapeGroup( Simulation.this, true, 0, 0, 0, button,
slider1, slider0, slider2, slider3, slider4, slider5, slider6, slider7, slider8,
text, text4, text3, text7, text6, text2, text8, text9, text5, text10, text11,
line1, text12, text13, text14, text15, text16, line5, text17, text18, text19,
text20, text21, text22, text1, image, text23, polyline );
    icon = new ShapeGroup( Simulation.this, true, 0, 0, 0
);
    // Setup presentation
    Presentation _p = new Presentation( this, applet != null ?
Presentation.MODE_APPLET :
        Presentation.MODE_APPLICATION, applet );
    _p.start();

    Panel _panel = _p.getPanel();
    ToolBar _tb = _p.getToolBar();
    StatusBar _sb = _p.getStatusBar();

    _panel.setZoomEnabled( false );
    _panel.setPanningEnabled( false );
    _panel.setFrameManagementBalance( 2.0 );

    _sb.setSectionVisible( StatusBar.DATE, false );
    _sb.setSectionVisible( StatusBar.EPS, false );
    _sb.setSectionVisible( StatusBar.EXPERIMENT, false );
    _sb.setSectionVisible( StatusBar.FPS, false );
    _sb.setSectionVisible( StatusBar.MEMORY, true );
    _sb.setSectionVisible( StatusBar.SECONDS, true );
    _sb.setSectionVisible( StatusBar.SIMULATION, true );
    _sb.setSectionVisible( StatusBar.STATUS, true );
    _sb.setSectionVisible( StatusBar.STEP, false );
    _sb.setSectionVisible( StatusBar.TIME, true );
    _tb.setSectionVisible( ToolBar.ANIMATION, false );
    _tb.setSectionVisible( ToolBar.EXECUTION, true );
    _tb.setSectionVisible( ToolBar.FILE, false );
    _tb.setSectionVisible( ToolBar.NAVIGATION, true );
    _tb.setSectionVisible( ToolBar.TIME_SCALE, true );
    _tb.setSectionVisible( ToolBar.VIEW, false );
  }

}
```

SDModel.java

```
package insurgencySDABModel;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import java.awt.geom.Arc2D;

public class SDModel extends ActiveObject
{
  // Parameters

  public double  People_Birth;

  /**
   * Returns default value for parameter <code>People_Birth</code>.
   * <i>This method should not be called by user</i>
   */
  public
double  _People_Birth_DefaultValue_xjal() {
    return 0.0;
  }

  public void set_People_Birth(
double  People_Birth ) {
    if (People_Birth == this.People_Birth) {
      return;
    }
    this.People_Birth = People_Birth;
    onChange_People_Birth();
    onChange();
  }

  void onChange_People_Birth() {
  }

  public double  Contact_Rate;

  /**
   * Returns default value for parameter <code>Contact_Rate</code>.
   * <i>This method should not be called by user</i>
   */
  public
```

```
double _Contact_Rate_DefaultValue_xjal() {
  return 0.0;
 }

 public void set_Contact_Rate(
double Contact_Rate ) {
  if (Contact_Rate == this.Contact_Rate) {
   return;
  }
  this.Contact_Rate = Contact_Rate;
  onChange_Contact_Rate();
  onChange();
 }

 void onChange_Contact_Rate() {
 }

 public double People_Reaction_Delay_On_Anti_Regime_Messages;

 /**
  * Returns default value for parameter
<code>People_Reaction_Delay_On_Anti_Regime_Messages</code>.
  * <i>This method should not be called by user</i>
  */
 public double
_People_Reaction_Delay_On_Anti_Regime_Messages_DefaultValue_xjal(
) {
  return 0.0;
 }

 public void set_People_Reaction_Delay_On_Anti_Regime_Messages(
double People_Reaction_Delay_On_Anti_Regime_Messages ) {
  if (People_Reaction_Delay_On_Anti_Regime_Messages ==
this.People_Reaction_Delay_On_Anti_Regime_Messages) {
   return;
  }
  this.People_Reaction_Delay_On_Anti_Regime_Messages =
People_Reaction_Delay_On_Anti_Regime_Messages;
  onChange_People_Reaction_Delay_On_Anti_Regime_Messages();
  onChange();
 }

 void onChange_People_Reaction_Delay_On_Anti_Regime_Messages() {
 }

 public double Regime_Social_and_Economic_Efforts;

 /**
  * Returns default value for parameter
<code>Regime_Social_and_Economic_Efforts</code>.
  * <i>This method should not be called by user</i>
  */
 public
double _Regime_Social_and_Economic_Efforts_DefaultValue_xjal() {
  return 0.0;
 }

 public void set_Regime_Social_and_Economic_Efforts(
double Regime_Social_and_Economic_Efforts ) {
  if (Regime_Social_and_Economic_Efforts ==
this.Regime_Social_and_Economic_Efforts) {
   return;
  }
  this.Regime_Social_and_Economic_Efforts =
Regime_Social_and_Economic_Efforts;
  onChange_Regime_Social_and_Economic_Efforts();
  onChange();
 }

 void onChange_Regime_Social_and_Economic_Efforts() {
 }

 public double Avg_Time_as_Dissident;

 /**
  * Returns default value for parameter
<code>Avg_Time_as_Dissident</code>.
  * <i>This method should not be called by user</i>
  */
 public
double _Avg_Time_as_Dissident_DefaultValue_xjal() {
  return 15 ;
 }

 public void set_Avg_Time_as_Dissident(
double Avg_Time_as_Dissident ) {
  if (Avg_Time_as_Dissident == this.Avg_Time_as_Dissident) {
   return;
  }
  this.Avg_Time_as_Dissident = Avg_Time_as_Dissident;
  onChange_Avg_Time_as_Dissident();
  onChange();
 }

 void onChange_Avg_Time_as_Dissident() {
 }

 public double Appeasement_Fraction;

 /**
  * Returns default value for parameter
<code>Appeasement_Fraction</code>.
  * <i>This method should not be called by user</i>
  */
 public
double _Appeasement_Fraction_DefaultValue_xjal() {
  return 0.5 ;
 }

 public void set_Appeasement_Fraction(
double Appeasement_Fraction ) {
  if (Appeasement_Fraction == this.Appeasement_Fraction) {
   return;
  }
  this.Appeasement_Fraction = Appeasement_Fraction;
  onChange_Appeasement_Fraction();
  onChange();
 }

 void onChange_Appeasement_Fraction() {
 }

 public double Desired_Time_to_Remove_Insurgents;

 /**
  * Returns default value for parameter
<code>Desired_Time_to_Remove_Insurgents</code>.
  * <i>This method should not be called by user</i>
  */
 public
double _Desired_Time_to_Remove_Insurgents_DefaultValue_xjal() {
  return 15 ;
```

```
  }

  public void set_Desired_Time_to_Remove_Insurgents(
double  Desired_Time_to_Remove_Insurgents ) {
    if (Desired_Time_to_Remove_Insurgents ==
this.Desired_Time_to_Remove_Insurgents) {
      return;
    }
    this.Desired_Time_to_Remove_Insurgents =
Desired_Time_to_Remove_Insurgents;
    onChange_Desired_Time_to_Remove_Insurgents();
    onChange();
  }

  void onChange_Desired_Time_to_Remove_Insurgents() {
  }



  public double  Removal_Effectiveness;

  /**
   *  Returns  default  value  for  parameter
<code>Removal_Effectiveness</code>.
   * <i>This method should not be called by user</i>
   */
  public
double  _Removal_Effectiveness_DefaultValue_xjal() {
    return 0.5 ;
  }

  public void set_Removal_Effectiveness(
double  Removal_Effectiveness ) {
    if (Removal_Effectiveness == this.Removal_Effectiveness) {
      return;
    }
    this.Removal_Effectiveness = Removal_Effectiveness;
    onChange_Removal_Effectiveness();
    onChange();
  }

  void onChange_Removal_Effectiveness() {
  }

  // Plain Variables



  // Collection Variables

  // Flow/Auxiliary Variables
  public double Becoming_Dissident;
  public double Insurgent_Recruitment;
  public double Removing_Insurgents;
  public double Births;
  public double Appeasement_Rate;
  public double Recruits_Through_Social_Network;
  public double Regime_Opponents;
  public double Violent_Incident_Intensity;
  public final double Propensity_to_Commit_Violence;
  public final double Propensity_to_Protest;
  public double Protest_Intensity;
  public double Incident_Intensity;
  public final double Relative_Strength_of_Violent_Incidents;
  public double Effect_of_Incidents_on_Anti_Regime_Messages;
  public                    final                    double
Strength_of_Incident_Effect_on_Anti_Regime_Messages;
  public double Effect_of_Anti_Regime_Messages_on_Recruitment;
  public double Propensity_to_be_Recruited;
  public final double Normal_Propensity_to_be_Recruited;
  public final double Dissidents_Fraction;
  public final double Insurgents_Fraction;
  public double Indicated_Force_Strength;
  public double Normal_Frequency_of_Anti_Regime_Messages;

  // Stock Variables
  public double Government_Supporters;
  public double Dissidents;
  public double Insurgents;
  public double Removed_Insurgents;
  public double Perceived_Intencity_of_Anti_Regime_Messages;

  /**
   * Writes model variables into given arrays
   */
  public void getScalarPhaseVector(double[] _d, double[] _a){
    _d[0] = Government_Supporters;
    _d[1] = Dissidents;
    _d[2] = Insurgents;
    _d[3] = Removed_Insurgents;
    _d[4] = Perceived_Intencity_of_Anti_Regime_Messages;
  }

  /**
   * Writes given arrays to model variables
   */
  public void putScalarPhaseVector(double[] _d, double[] _a){
    Government_Supporters = _d[0];
    Dissidents = _d[1];
    Insurgents = _d[2];
    Removed_Insurgents = _d[3];
    Perceived_Intencity_of_Anti_Regime_Messages = _d[4];
  }

  public void assignInitialConditions(){
    Normal_Frequency_of_Anti_Regime_Messages =
_Normal_Frequency_of_Anti_Regime_Messages_Formula();
    Dissidents = _Dissidents_Initial_Value();
    Insurgents = _Insurgents_Initial_Value();
    Regime_Opponents = _Regime_Opponents_Formula();
    Effect_of_Anti_Regime_Messages_on_Recruitment =
_Effect_of_Anti_Regime_Messages_on_Recruitment_Formula();
    Violent_Incident_Intensity = _Violent_Incident_Intensity_Formula();
    Protest_Intensity = _Protest_Intensity_Formula();
    Propensity_to_be_Recruited =
_Propensity_to_be_Recruited_Formula();
    Government_Supporters = _Government_Supporters_Initial_Value();
    Recruits_Through_Social_Network =
_Recruits_Through_Social_Network_Formula();
    Incident_Intensity = _Incident_Intensity_Formula();
    Indicated_Force_Strength = _Indicated_Force_Strength_Formula();
    Becoming_Dissident = _Becoming_Dissident_Formula();
    Insurgent_Recruitment = _Insurgent_Recruitment_Formula();
    Removing_Insurgents = _Removing_Insurgents_Formula();
    Births = _Births_Formula();
    Appeasement_Rate = _Appeasement_Rate_Formula();
    Effect_of_Incidents_on_Anti_Regime_Messages =
_Effect_of_Incidents_on_Anti_Regime_Messages_Formula();
  }

  public void assignInitialConditions( int _variableNameCode ) {
    switch (_variableNameCode) {
      case 1923161369:
        break;
      case -1863142898:
```

```
      Normal_Frequency_of_Anti_Regime_Messages =
_Normal_Frequency_of_Anti_Regime_Messages_Formula();
      break;
    case -637208952:
      Dissidents = _Dissidents_Initial_Value();
      break;
    case -31673916:
      Insurgents = _Insurgents_Initial_Value();
      break;
    case 811998116:
      Regime_Opponents = _Regime_Opponents_Formula();
      break;
    case 321296160:
      Effect_of_Anti_Regime_Messages_on_Recruitment =
_Effect_of_Anti_Regime_Messages_on_Recruitment_Formula();
      break;
    case 1345681610:
      Violent_Incident_Intensity =
_Violent_Incident_Intensity_Formula();
      break;
    case 960669811:
      Protest_Intensity = _Protest_Intensity_Formula();
      break;
    case 822880013:
      Propensity_to_be_Recruited =
_Propensity_to_be_Recruited_Formula();
      break;
    case 181598789:
      Government_Supporters =
_Government_Supporters_Initial_Value();
      break;
    case -870908256:
      Recruits_Through_Social_Network =
_Recruits_Through_Social_Network_Formula();
      break;
    case 599708934:
      Incident_Intensity = _Incident_Intensity_Formula();
      break;
    case 1416004969:
      Indicated_Force_Strength = _Indicated_Force_Strength_Formula();
      break;
    case 438087088:
      Becoming_Dissident = _Becoming_Dissident_Formula();
      break;
    case 131061196:
      Insurgent_Recruitment = _Insurgent_Recruitment_Formula();
      break;
    case 2005944418:
      Removing_Insurgents = _Removing_Insurgents_Formula();
      break;
    case 1990004660:
      Births = _Births_Formula();
      break;
    case 1375878834:
      Appeasement_Rate = _Appeasement_Rate_Formula();
      break;
    case 83020585:
      Effect_of_Incidents_on_Anti_Regime_Messages =
_Effect_of_Incidents_on_Anti_Regime_Messages_Formula();
      break;
    case 1095348003:
      break;
    default:
      break;
   }
 }
```

```
 public double _Normal_Propensity_to_be_Recruited_Value() {
  return 0.5 ;
 }

 public double _Propensity_to_Commit_Violence_Value() {
  return 0.05 ;
 }
 public double _Removing_Insurgents_Formula() {
  return Indicated_Force_Strength ;
 }

 public double
_Effect_of_Anti_Regime_Messages_on_Recruitment_Formula() {
  return
Perceived_Intencity_of_Anti_Regime_Messages /
People_Reaction_Delay_On_Anti_Regime_Messages * ( 1 +
Normal_Frequency_of_Anti_Regime_Messages ) ;
 }

 public double _Dissidents_Initial_Value() {
  return get_Main().Initial_Dissident ;
 }

 public double _Government_Supporters_Initial_Value() {
  return get_Main().Initial_Government_Supporters ;
 }

 public double
_Strength_of_Incident_Effect_on_Anti_Regime_Messages_Value() {
  return 0.1 ;
 }

 public double _Becoming_Dissident_Formula() {
  return Recruits_Through_Social_Network ;
 }

 public double _Removed_Insurgents_Expression() {
  return Removing_Insurgents ;
 }

 public double _Propensity_to_be_Recruited_Formula() {
  return
1 - 1 / ( ( Effect_of_Anti_Regime_Messages_on_Recruitment *
Normal_Propensity_to_be_Recruited /
Regime_Social_and_Economic_Efforts / ( Dissidents + Insurgents ) ) * (
Effect_of_Anti_Regime_Messages_on_Recruitment *
Normal_Propensity_to_be_Recruited /
Regime_Social_and_Economic_Efforts / ( Dissidents + Insurgents ) ) + 1
)
;
 }

 public double
_Perceived_Intencity_of_Anti_Regime_Messages_Expression() {
  return
Effect_of_Incidents_on_Anti_Regime_Messages -
Effect_of_Anti_Regime_Messages_on_Recruitment ;
 }

 public double _Appeasement_Rate_Formula() {
  return
Dissidents * Appeasement_Fraction / Avg_Time_as_Dissident ;
 }

 public double _Incident_Intensity_Formula() {
  return
Violent_Incident_Intensity * Relative_Strength_of_Violent_Incidents +
Protest_Intensity ;
 }
```

```
  public double _Dissidents_Fraction_Value() {
    return 0.9 ;
  }

  public double _Government_Supporters_Expression() {
    return
Births - Becoming_Dissident + Appeasement_Rate ;
  }

  public double _Propensity_to_Protest_Value() {
    return 0.3 ;
  }

  public double _Dissidents_Expression() {
    return
Becoming_Dissident - Insurgent_Recruitment - Appeasement_Rate ;
  }

  public                                                    double
_Effect_of_Incidents_on_Anti_Regime_Messages_Formula() {
    return
Incident_Intensity                                          *
Strength_of_Incident_Effect_on_Anti_Regime_Messages  *  (  1  +
Normal_Frequency_of_Anti_Regime_Messages ) ;
  }

  public double _Births_Formula() {
    return People_Birth / 1000 * Government_Supporters ;
  }

  public double _Insurgent_Recruitment_Formula() {
    return
Dissidents / Avg_Time_as_Dissident * ( 1 - Appeasement_Fraction ) ;
  }

  public double _Regime_Opponents_Formula() {
    return
Dissidents * Dissidents_Fraction + Insurgents * Insurgents_Fraction ;
  }

  public                                                    double
_Normal_Frequency_of_Anti_Regime_Messages_Formula() {
    return
Perceived_Intencity_of_Anti_Regime_Messages                 /
get_Main().Initial_Government_Supporters ;
  }

  public double _Indicated_Force_Strength_Formula() {
    return
Removal_Effectiveness       *       Insurgents       /
Desired_Time_to_Remove_Insurgents ;
  }

  public double _Insurgents_Initial_Value() {
    return get_Main().Initial_Insurgent ;
  }

  public double _Recruits_Through_Social_Network_Formula() {
    return
(    Propensity_to_be_Recruited    *    Government_Supporters    *
Regime_Opponents  *  Contact_Rate  )  /  (  Regime_Opponents  +
Government_Supporters ) ;
  }

  public double _Violent_Incident_Intensity_Formula() {
    return
Insurgents * Propensity_to_Commit_Violence * Insurgents_Fraction
;
  }

  public double _Relative_Strength_of_Violent_Incidents_Value() {
    return 1 ;
  }

  public double _Insurgents_Fraction_Value() {
    return 0.9 ;
  }

  public double _Insurgents_Expression() {
    return
Insurgent_Recruitment - Removing_Insurgents ;
  }

  public double _Protest_Intensity_Formula() {
    return
Regime_Opponents * Propensity_to_Protest ;
  }


  public void formulasExecute(){
    Normal_Frequency_of_Anti_Regime_Messages                =
_Normal_Frequency_of_Anti_Regime_Messages_Formula();
    Regime_Opponents = _Regime_Opponents_Formula();
    Effect_of_Anti_Regime_Messages_on_Recruitment           =
_Effect_of_Anti_Regime_Messages_on_Recruitment_Formula();
    Violent_Incident_Intensity = _Violent_Incident_Intensity_Formula();
    Protest_Intensity = _Protest_Intensity_Formula();
    Propensity_to_be_Recruited                              =
_Propensity_to_be_Recruited_Formula();
    Recruits_Through_Social_Network                         =
_Recruits_Through_Social_Network_Formula();
    Incident_Intensity = _Incident_Intensity_Formula();
    Indicated_Force_Strength = _Indicated_Force_Strength_Formula();
    Becoming_Dissident = _Becoming_Dissident_Formula();
    Insurgent_Recruitment = _Insurgent_Recruitment_Formula();
    Removing_Insurgents = _Removing_Insurgents_Formula();
    Births = _Births_Formula();
    Appeasement_Rate = _Appeasement_Rate_Formula();
    Effect_of_Incidents_on_Anti_Regime_Messages             =
_Effect_of_Incidents_on_Anti_Regime_Messages_Formula();
  }

  public void formulasExecute( int _variableNameCode ) {
    switch (_variableNameCode) {
      case -1863142898:
        Normal_Frequency_of_Anti_Regime_Messages            =
_Normal_Frequency_of_Anti_Regime_Messages_Formula();
        break;
      case 811998116:
        Regime_Opponents = _Regime_Opponents_Formula();
        break;
      case 321296160:
        Effect_of_Anti_Regime_Messages_on_Recruitment       =
_Effect_of_Anti_Regime_Messages_on_Recruitment_Formula();
        break;
      case 1345681610:
        Violent_Incident_Intensity                          =
_Violent_Incident_Intensity_Formula();
        break;
      case 960669811:
        Protest_Intensity = _Protest_Intensity_Formula();
        break;
      case 822880013:
        Propensity_to_be_Recruited                          =
_Propensity_to_be_Recruited_Formula();
```

```
      break;
    case -870908256:
      Recruits_Through_Social_Network = _Recruits_Through_Social_Network_Formula();
      break;
    case 599708934:
      Incident_Intensity = _Incident_Intensity_Formula();
      break;
    case 1416004969:
      Indicated_Force_Strength = _Indicated_Force_Strength_Formula();
      break;
    case 438087088:
      Becoming_Dissident = _Becoming_Dissident_Formula();
      break;
    case 131061196:
      Insurgent_Recruitment = _Insurgent_Recruitment_Formula();
      break;
    case 2005944418:
      Removing_Insurgents = _Removing_Insurgents_Formula();
      break;
    case 1990004660:
      Births = _Births_Formula();
      break;
    case 1375878834:
      Appeasement_Rate = _Appeasement_Rate_Formula();
      break;
    case 83020585:
      Effect_of_Incidents_on_Anti_Regime_Messages = _Effect_of_Incidents_on_Anti_Regime_Messages_Formula();
      break;
    default:
      break;
   }
  }

  public void getScalarRightPart( double[] _dr, double[] _ar ) {
    _dr[ 0 ] = _Government_Supporters_Expression();
    _dr[ 1 ] = _Dissidents_Expression();
    _dr[ 2 ] = _Insurgents_Expression();
    _dr[ 3 ] = _Removed_Insurgents_Expression();
    _dr[ 4 ] = _Perceived_Intencity_of_Anti_Regime_Messages_Expression();
  }

  static ActiveObjectIntegrationManager integrationManager_xjal = new ActiveObjectIntegrationManager( 5, 0, 15 );

  public ActiveObjectIntegrationManager getIntegrationManager(){
    return integrationManager_xjal;
  }

  // Events

  public EventTimeout _plot2_autoUpdateEvent_xjal = new EventTimeout(this);
  public EventTimeout _autoCreatedDS_xjal = new EventTimeout(this);


  @Override
  public String getNameOf( EventTimeout _e ) {
    if ( _e == _plot2_autoUpdateEvent_xjal ) return "plot2 auto update event";
    if ( _e == _autoCreatedDS_xjal ) return "Auto-created DataSets auto update event";
     return super.getNameOf( _e );
  }

  @Override
  public int getModeOf( EventTimeout _e ) {
    if ( _e == _plot2_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
    if ( _e == _autoCreatedDS_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
    return super.getModeOf( _e );
  }
  @Override
  public double getFirstOccurrenceTime( EventTimeout _e ) {
    if (
       _e == _plot2_autoUpdateEvent_xjal
       || _e == _autoCreatedDS_xjal
     ) return getEngine().getStartTime();
    return super.getFirstOccurrenceTime( _e );
  }

  @Override
  public double evaluateTimeoutOf( EventTimeout _e ) {
    if ( _e == _plot2_autoUpdateEvent_xjal ) return
1
;
    if ( _e == _autoCreatedDS_xjal ) return
1
;
    return super.evaluateTimeoutOf( _e );
  }

  @Override
  public void executeActionOf( EventTimeout _e ) {
    if ( _e == _plot2_autoUpdateEvent_xjal ) {
      plot2.updateData();
      return;
    }
    if ( _e == _autoCreatedDS_xjal ) {
      _ds_Becoming_Dissident.update();
      _ds_Insurgent_Recruitment.update();
      _ds_Removing_Insurgents.update();
      _ds_Births.update();
      _ds_Appeasement_Rate.update();
      _ds_Recruits_Through_Social_Network.update();
      _ds_Regime_Opponents.update();
      _ds_Violent_Incident_Intensity.update();
      _ds_Propensity_to_Commit_Violence.update();
      _ds_Propensity_to_Protest.update();
      _ds_Protest_Intensity.update();
      _ds_Incident_Intensity.update();
      _ds_Relative_Strength_of_Violent_Incidents.update();
      _ds_Effect_of_Incidents_on_Anti_Regime_Messages.update();
      _ds_Strength_of_Incident_Effect_on_Anti_Regime_Messages.update();
      _ds_Effect_of_Anti_Regime_Messages_on_Recruitment.update();
      _ds_Propensity_to_be_Recruited.update();
      _ds_Normal_Propensity_to_be_Recruited.update();
      _ds_Dissidents_Fraction.update();
      _ds_Insurgents_Fraction.update();
      _ds_Indicated_Force_Strength.update();
      _ds_Normal_Frequency_of_Anti_Regime_Messages.update();
      _ds_Government_Supporters.update();
      _ds_Dissidents.update();
      _ds_Insurgents.update();
      _ds_Removed_Insurgents.update();
      _ds_Perceived_Intencity_of_Anti_Regime_Messages.update();
    }
    super.executeActionOf( _e );
  }
```

```
 /**
  * Auto-created data set(s) for Becoming_Dissident
  */
 public DataSet _ds_Becoming_Dissident = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Becoming_Dissident );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Insurgent_Recruitment
  */
 public DataSet _ds_Insurgent_Recruitment = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Insurgent_Recruitment );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Removing_Insurgents
  */
 public DataSet _ds_Removing_Insurgents = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Removing_Insurgents );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Births
  */
 public DataSet _ds_Births = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Births );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Appeasement_Rate
  */
 public DataSet _ds_Appeasement_Rate = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Appeasement_Rate );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Recruits_Through_Social_Network
  */
 public DataSet _ds_Recruits_Through_Social_Network = new DataSet(
100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Recruits_Through_Social_Network );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Regime_Opponents
  */
 public DataSet _ds_Regime_Opponents = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Regime_Opponents );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Violent_Incident_Intensity
  */
 public DataSet _ds_Violent_Incident_Intensity = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Violent_Incident_Intensity );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Propensity_to_Commit_Violence
  */
 public DataSet _ds_Propensity_to_Commit_Violence = new DataSet(
100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Propensity_to_Commit_Violence );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Propensity_to_Protest
  */
 public DataSet _ds_Propensity_to_Protest = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Propensity_to_Protest );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Protest_Intensity
  */
 public DataSet _ds_Protest_Intensity = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Protest_Intensity );
   _lastUpdateTime = time();
```

```
  }
 };
 /**
  * Auto-created data set(s) for Incident_Intensity
  */
 public DataSet _ds_Incident_Intensity = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Incident_Intensity );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Relative_Strength_of_Violent_Incidents
  */
 public DataSet _ds_Relative_Strength_of_Violent_Incidents = new
DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Relative_Strength_of_Violent_Incidents );
   _lastUpdateTime = time();
  }
 };
 /**
  *          Auto-created          data          set(s)          for
Effect_of_Incidents_on_Anti_Regime_Messages
  */
 public DataSet _ds_Effect_of_Incidents_on_Anti_Regime_Messages =
new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add(                                                    time(),
SDModel.this.Effect_of_Incidents_on_Anti_Regime_Messages );
   _lastUpdateTime = time();
  }
 };
 /**
  *          Auto-created          data          set(s)          for
Strength_of_Incident_Effect_on_Anti_Regime_Messages
  */
 public                                                   DataSet
_ds_Strength_of_Incident_Effect_on_Anti_Regime_Messages   =   new
DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add(                                                    time(),
SDModel.this.Strength_of_Incident_Effect_on_Anti_Regime_Messages );
   _lastUpdateTime = time();
  }
 };
 /**
  *          Auto-created          data          set(s)          for
Effect_of_Anti_Regime_Messages_on_Recruitment
  */
 public DataSet _ds_Effect_of_Anti_Regime_Messages_on_Recruitment
= new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add(                                                    time(),
SDModel.this.Effect_of_Anti_Regime_Messages_on_Recruitment );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Propensity_to_be_Recruited
  */
 public DataSet _ds_Propensity_to_be_Recruited = new DataSet( 100 )
{
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Propensity_to_be_Recruited );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Normal_Propensity_to_be_Recruited
  */
 public DataSet _ds_Normal_Propensity_to_be_Recruited = new
DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Normal_Propensity_to_be_Recruited );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Dissidents_Fraction
  */
 public DataSet _ds_Dissidents_Fraction = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Dissidents_Fraction );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Insurgents_Fraction
  */
 public DataSet _ds_Insurgents_Fraction = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Insurgents_Fraction );
   _lastUpdateTime = time();
  }
 };
 /**
  * Auto-created data set(s) for Indicated_Force_Strength
  */
 public DataSet _ds_Indicated_Force_Strength = new DataSet( 100 ) {
  double _lastUpdateTime = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateTime ) { return; }
   add( time(), SDModel.this.Indicated_Force_Strength );
   _lastUpdateTime = time();
  }
```

```
  };
  /**
   *  Auto-created  data  set(s)  for
Normal_Frequency_of_Anti_Regime_Messages
   */
  public  DataSet  _ds_Normal_Frequency_of_Anti_Regime_Messages  =
new DataSet( 100 ) {
    double _lastUpdateTime = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateTime ) { return; }
      add(                                                  time(),
SDModel.this.Normal_Frequency_of_Anti_Regime_Messages );
      _lastUpdateTime = time();
    }
  };
  /**
   * Auto-created data set(s) for Government_Supporters
   */
  public DataSet _ds_Government_Supporters = new DataSet( 100 ) {
    double _lastUpdateTime = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateTime ) { return; }
      add( time(), SDModel.this.Government_Supporters );
      _lastUpdateTime = time();
    }
  };
  /**
   * Auto-created data set(s) for Dissidents
   */
  public DataSet _ds_Dissidents = new DataSet( 100 ) {
    double _lastUpdateTime = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateTime ) { return; }
      add( time(), SDModel.this.Dissidents );
      _lastUpdateTime = time();
    }
  };
  /**
   * Auto-created data set(s) for Insurgents
   */
  public DataSet _ds_Insurgents = new DataSet( 100 ) {
    double _lastUpdateTime = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateTime ) { return; }
      add( time(), SDModel.this.Insurgents );
      _lastUpdateTime = time();
    }
  };
  /**
   * Auto-created data set(s) for Removed_Insurgents
   */
  public DataSet _ds_Removed_Insurgents = new DataSet( 100 ) {
    double _lastUpdateTime = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateTime ) { return; }
      add( time(), SDModel.this.Removed_Insurgents );
      _lastUpdateTime = time();
    }
  };
  /**
   *  Auto-created  data  set(s)  for
Perceived_Intencity_of_Anti_Regime_Messages
   */
  public  DataSet  _ds_Perceived_Intencity_of_Anti_Regime_Messages  =
new DataSet( 100 ) {
    double _lastUpdateTime = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateTime ) { return; }
      add(                                                  time(),
SDModel.this.Perceived_Intencity_of_Anti_Regime_Messages );
      _lastUpdateTime = time();
    }
  };
  public DataSet _plot2_expression0_dataSet_xjal = new DataSet( 100 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
Government_Supporters
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  public DataSet _plot2_expression1_dataSet_xjal = new DataSet( 100 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
Dissidents
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  public DataSet _plot2_expression2_dataSet_xjal = new DataSet( 100 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
Insurgents
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  public DataSet _plot2_expression3_dataSet_xjal = new DataSet( 100 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
Removed_Insurgents
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  // View areas
  public ViewArea SDView = new ViewArea( this, "SDView", 0, 0,
ViewArea.TOP_LEFT, ViewArea.NONE, 1.0, 100, 100 );
```

```
 static final Font _text_Font = new Font("SansSerif", 0, 32 );
 static final Font _text27_Font = new Font("SansSerif", 0, 18 );
 static final Font _text18_Font = _text27_Font;
 static final Font _text19_Font = _text27_Font;
 static final Font _text17_Font = _text27_Font;
 static final int line1 = 1;
 static final int _text = 2;
 static final int _rectangle40 = 3;
 static final int _text27 = 4;
 static final int _line67 = 5;
 static final int _line66 = 6;
 static final int _line57 = 7;
 static final int _text18 = 8;
 static final int _line53 = 9;
 static final int _line54 = 10;
 static final int _line55 = 11;
 static final int _rectangle37 = 12;
 static final int _text19 = 13;
 static final int _rectangle34 = 14;
 static final int _text17 = 15;
 static final int _plot2 = 16;

 /**
  * Top-level presentation group id
  */
 static final int _presentation = 0;

 /**
  * Top-level icon group id
  */
 static final int _icon = -1;


 @Override
 public boolean onShapeClick( int _shape, int index, double clickx,
double clicky ){
   switch( _shape ){
     case _rectangle40:
       if (true) {

get_Main().DescriptionView.navigateTo();
       }
       break;
     case _rectangle37:
       if (true) {

get_Main().aMLABView.navigateTo();
       }
       break;
     case _rectangle34:
       if (true) {

get_Main().MainView.navigateTo();
       }
       break;
     default: return super.onShapeClick( _shape, index, clickx, clicky );
               }
               return false;
 }


 @Override
 public String getNameOfShape( int _shape ) {
   switch( _shape ) {
     case line1: return "line1";
     default: return super.getNameOfShape( _shape );
   }
 }

 @Override
 public int getShapeType( int _shape ) {
   switch( _shape ) {
     case line1: return SHAPE_LINE;
     default: return super.getShapeType( _shape );
   }
 }

 @Override
 public double getShapeX( int _shape, int index ) {
   switch( _shape ) {
     case line1: return 30;
     default: return super.getShapeX( _shape, index );
   }
 }

 @Override
 public double getShapeY( int _shape, int index ) {
   switch( _shape ) {
     case line1: return 50;
     default: return super.getShapeY( _shape, index );
   }
 }

 @Override
 public Color getShapeLineColor( int _shape, int index ) {
   switch( _shape ) {
     // shapes
     case line1: return slateGray;
     // controls (text color)
     // charts (border color)
     default: return super.getShapeLineColor( _shape, index );
   }
 }

 @Override
 public int getShapeLineStyle( int _shape, int index ) {
   switch( _shape ) {
     case line1: return LINE_STYLE_SOLID;
     default: return super.getShapeLineStyle( _shape, index );
   }
 }

 @Override
 public double getShapeLineDx( int _shape, int index ) {
   switch( _shape ) {
     case line1: return 970;
     default: return super.getShapeLineDx( _shape, index );
   }
 }

 @Override
 public double getShapeLineDy( int _shape, int index ) {
   switch( _shape ) {
     case line1: return 0;
     default: return super.getShapeLineDy( _shape, index );
   }
 }


 TimePlot plot2;
```

```
ShapeText text;
ShapeRectangle rectangle40;
ShapeText text27;
ShapeLine line67;
ShapeLine line66;
ShapeLine line57;
ShapeText text18;
ShapeLine line53;
ShapeLine line54;
ShapeLine line55;
ShapeRectangle rectangle37;
ShapeText text19;
ShapeRectangle rectangle34;
ShapeText text17;

// Static initialization of persistent elements
{
 text = new ShapeText(
    true,30, 10, 0.0,
    slateGray,"SD+AB Insurgency Models",
    _text_Font, ALIGNMENT_LEFT
 );
 rectangle40 = new ShapeRectangle(
   true,349, 70, 0.0,
       null, null,
                                105, 30,
                                1, LINE_STYLE_SOLID
 ) {

  @Override
  public boolean onClick( double clickx, double clicky ) {
   return onShapeClick( _rectangle40, 0, clickx, clicky );
  }
 };
 text27 = new ShapeText(
    true,354, 75, 0.0,
    royalBlue,"Description",
    _text27_Font, ALIGNMENT_LEFT
 );
 line67 = new ShapeLine(
                        true,355, 95,
    royalBlue,
                        94,         0,
                        1, LINE_STYLE_SOLID
 );
 line66 = new ShapeLine(
                        false,345, 70,
    black,
                        0,          30,
                        1, LINE_STYLE_SOLID
 );
 line57 = new ShapeLine(
                        true,235, 95,
    royalBlue,
                        100,        0,
                        1, LINE_STYLE_SOLID
 );
 text18 = new ShapeText(
    true,135, 75, 0.0,
    black,"SD model",
    _text18_Font, ALIGNMENT_LEFT
 );
 line53 = new ShapeLine(
                        true,125, 70,
    black,
                        0,          30,
                        1, LINE_STYLE_SOLID
 );
 line54 = new ShapeLine(
                        true,36, 95,
    royalBlue,
                        80,         0,
                        1, LINE_STYLE_SOLID
 );
 line55 = new ShapeLine(
                        true,225, 70,
    black,
                        0,          30,
                        1, LINE_STYLE_SOLID
 );
 rectangle37 = new ShapeRectangle(
   true,30, 70, 0.0,
       null, null,
                                90, 30,
                                1, LINE_STYLE_SOLID
 ) {

  @Override
  public boolean onClick( double clickx, double clicky ) {
   return onShapeClick( _rectangle37, 0, clickx, clicky );
  }
 };
 text19 = new ShapeText(
    true,35, 75, 0.0,
    royalBlue,"AB model",
    _text19_Font, ALIGNMENT_LEFT
 );
 rectangle34 = new ShapeRectangle(
   true,230, 70, 0.0,
       null, null,
                                111, 30,
                                1, LINE_STYLE_SOLID
 ) {

  @Override
  public boolean onClick( double clickx, double clicky ) {
   return onShapeClick( _rectangle34, 0, clickx, clicky );
  }
 };
 text17 = new ShapeText(
    true,235, 75, 0.0,
    royalBlue,"Comparison",
    _text17_Font, ALIGNMENT_LEFT
 );
}
ShapeGroup presentation;
ShapeGroup icon;

@Override
public Object getPersistentShape( int _shape ) {
 switch(_shape){
   case _presentation: return presentation;
   case _icon: return icon;

   case _plot2: return plot2;
   case _text: return text;
   case _rectangle40: return rectangle40;
   case _text27: return text27;
   case _line67: return line67;
   case _line66: return line66;
```

```
    case _line57: return line57;
    case _text18: return text18;
    case _line53: return line53;
    case _line54: return line54;
    case _line55: return line55;
    case _rectangle37: return rectangle37;
    case _text19: return text19;
    case _rectangle34: return rectangle34;
    case _text17: return text17;
    default: return null;
  }
 }


 static final Arc2D.Double arc_PD_1207137222568 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137222568, 408,330, 528, 330, 10.0 );}
 static final Arc2D.Double arc_PD_1207137222594 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137222594, 60,280, 58, 330, -18.027068612132567 );}
 static final Arc2D.Double arc_PD_1207137222599 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137222599, 408,330, 288, 270, -17.737176262694632 );}
 static final Arc2D.Double arc_PD_1207137222609 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137222609, 238,390, 288, 330, 10.0 );}
 static final Arc2D.Double arc_PD_1207137222656 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137222656, 460,400, 238, 390, 10.0 );}
 static final Arc2D.Double arc_PD_1207137222657 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137222657, 408,330, 460, 400, -15.601692918869809 );}
 static final Arc2D.Double arc_PD_1207137222658 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137222658, 648,330, 460, 400, 14.815906454474089 );}
 static final Arc2D.Double arc_PD_1207137222765 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137222765, 648,330, 720, 430, 10.0 );}
 static final Arc2D.Double arc_PD_1207137222781 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137222781, 810,450, 720, 430, 10.0 );}
 static final Arc2D.Double arc_PD_1207137222828 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137222828, 460,400, 628, 470, 10.0 );}
 static final Arc2D.Double arc_PD_1207137222829 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137222829, 560,500, 628, 470, 10.0 );}
 static final Arc2D.Double arc_PD_1207137222844 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137222844, 628,470, 720, 500, 8.19090676458768 );}
 static final Arc2D.Double arc_PD_1207137222847 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137222847, 810,540, 720, 500, 10.0 );}
 static final Arc2D.Double arc_PD_1207137222906 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137222906, 720,500, 750, 610, 10.0 );}
 static final Arc2D.Double arc_PD_1207137222968 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137222968, 200,530, 238, 390, 10.0 );}
 static final Arc2D.Double arc_PD_1207137222984 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137222984, 310,610, 200, 530, 10.0 );}
 static final Arc2D.Double arc_PD_1207137223031 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137223031, 90,460, 200, 530, 10.0 );}
 static final Arc2D.Double arc_PD_1207137223140 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137223140, 440,470, 460, 400, 10.0 );}
 static final Arc2D.Double arc_PD_1207137223156 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207137223156, 620,390, 460, 400, 5.295702537893931 );}
 static final Arc2D.Double arc_PD_1207216927734 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207216927734, 408,330, 200, 530, 48.4445532977476 );}
 static final Arc2D.Double arc_PD_1207216941078 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207216941078, 648,330, 200, 530, 67.609035521321 );}
 static final Arc2D.Double arc_PD_1207228547388 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207228547388, 690,270, 768, 330, 10.0 );}
 static final Arc2D.Double arc_PD_1207119093875 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207119093875, 528,610, 520, 560, 10.0 );}
 static final Arc2D.Double arc_PD_1207576881921 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207576881921, 648,330, 690, 270, 10.0 );}
 static final Arc2D.Double arc_PD_1207592083312 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207592083312, 528,610, 310, 610, 29.383396274006856 );}
 static final Arc2D.Double arc_PD_1207662497453 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207662497453, 520,560, 750, 610, 10.0 );}
 static final Arc2D.Double arc_PD_1207726841718 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207726841718, 620,390, 720, 430, 10.0 );}
 static final Arc2D.Double arc_PD_1207728954437 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207728954437, 700,650, 750, 610, 8.6768794117026 );}
 static final Arc2D.Double arc_PD_1207729293846 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207729293846, 520,560, 310, 610, -12.859107343390974 );}
 static final Arc2D.Double arc_PD_1207736540218 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207736540218, 100,570, 200, 530, 17.04180910638354 );}
 static final Arc2D.Double arc_PD_1207748231921 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207748231921, 300,450, 238, 390, 10.0 );}
 static final Arc2D.Double arc_PD_1207830555656 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207830555656, 370,230, 528, 330, 10.0 );}
 static final Arc2D.Double arc_PD_1207830555657 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207830555657, 370,230, 288, 270, -7.155718817544482 );}
 static final Arc2D.Double arc_PD_1207830591078 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207830591078, 510,270, 528, 330, 10.0 );}
```

```
 static final Arc2D.Double arc_PD_1207830591079 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207830591079, 510,270, 288, 270, -16.907723435898106 );}
 static final Arc2D.Double arc_PD_1207830615421 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207830615421, 820,250, 690, 270, 10.0 );}
 static final Arc2D.Double arc_PD_1207830635796 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1207830635796, 860,290, 690, 270, 10.0 );}
 static final Arc2D.Double arc_PD_1208247790296 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1208247790296, 290,650, 310, 610, 11.86562168052566 );}
 static final Arc2D.Double arc_PD_1208432283798 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1208432283798, 168,330, 58, 330, -20.14693182963201 );}
 static final Arc2D.Double arc_PD_1208432283799 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1208432283799, 168,330, 238, 390, 10.0 );}
 static final Arc2D.Double arc_PD_1208947334609 = new Arc2D.Double();
 static { setupPlainDependencyArc( arc_PD_1208947334609, 720,430, 720, 500, 10.0 );}


 @Override
 public void drawModelElements(Panel _panel, Graphics2D _g, boolean _publicOnly ) {
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137222568, null );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137222594, null );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137222599, null );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137222609, red );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137222656, blueViolet );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137222657, red );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137222658, blueViolet );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137222765, red );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137222781, silver );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137222828, red );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137222829, silver );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137222844, red );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137222847, silver );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137222906, red );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137222968, red );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137222984, red );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137223031, silver );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137223140, silver );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207137223156, silver );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207216927734, blueViolet );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207216941078, blueViolet );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207228547388, null );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207119093875, null );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207576881921, null );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207592083312, null );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207662497453, null );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207726841718, silver );
   }
   if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207728954437, silver );
   }
```

```
  if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207729293846, null
);
  }
  if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207736540218,
silver );
  }
  if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207748231921,
silver );
  }
  if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207830555656, null
);
  }
  if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207830555657, null
);
  }
  if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207830591078, null
);
  }
  if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207830591079, null
);
  }
  if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207830615421, null
);
  }
  if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1207830635796, null
);
  }
  if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1208247790296,
silver );
  }
  if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1208432283798, null
);
  }
  if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1208432283799, null
);
  }
  if (!_publicOnly) {
    drawPlainDependencyArc( _panel, _g, arc_PD_1208947334609, red
);
  }
  if (!_publicOnly) {
    drawFlowStockDependencyLine( _panel, _g, 183, 330, 270, 330, null
);
  }
  if (!_publicOnly) {
    drawFlowStockDependencyLine( _panel, _g, 303, 330, 390, 330, null
);
  }
  if (!_publicOnly) {
    drawFlowStockDependencyLine( _panel, _g, 423, 330, 510, 330, null
);
  }
  if (!_publicOnly) {
    drawFlowStockDependencyLine( _panel, _g, 543, 330, 630, 330, null
);
  }
  if (!_publicOnly) {
    drawFlowStockDependencyLine( _panel, _g, 73, 330, 150, 330, null
);
  }
  if (!_publicOnly) {
    drawFlowStockDependencyLine( _panel, _g, 275, 276, 184, 322, null
);
  }
  if (!_publicOnly) {
    drawFlowStockDependencyLine( _panel, _g, 395, 324, 304, 278, null
);
  }
  if (!_publicOnly) {
    drawFlowStockDependencyLine( _panel, _g, 735, 610, 546, 610,
orangeRed );
  }
  if (!_publicOnly) {
    drawFlowStockDependencyLine( _panel, _g, 513, 610, 328, 610,
orangeRed );
  }
  if (!_publicOnly) {
    drawFlowStockDependencyLine( _panel, _g, 663, 330, 750, 330, null
);
  }
  if (!_publicOnly) {
    drawFlowStockDependencyLine( _panel, _g, 783, 330, 870, 330, null
);
  }
  if (!_publicOnly) {
    drawParameter( _panel, _g, 60, 280, -30, -20, "People_Birth",
People_Birth, false, false );
  }
  if (!_publicOnly) {
    drawParameter( _panel, _g, 300, 450, 10, 0, "Contact_Rate",
Contact_Rate, false, false );
  }
  if (!_publicOnly) {
    drawParameter( _panel, _g, 290, 650, -270, 0,
"People_Reaction_Delay_On_Anti_Regime_Messages",
People_Reaction_Delay_On_Anti_Regime_Messages, false, false );
  }
  if (!_publicOnly) {
    drawParameter( _panel, _g, 100, 570, -60, -20,
"Regime_Social_and_Economic_Efforts",
Regime_Social_and_Economic_Efforts, false, false );
  }
  if (!_publicOnly) {
    drawParameter( _panel, _g, 370, 230, -40, -20,
"Avg_Time_as_Dissident", Avg_Time_as_Dissident, false, false );
  }
  if (!_publicOnly) {
    drawParameter( _panel, _g, 510, 270, -30, -20,
"Appeasement_Fraction", Appeasement_Fraction, false, false );
  }
  if (!_publicOnly) {
    drawParameter( _panel, _g, 820, 250, -60, -20,
"Desired_Time_to_Remove_Insurgents",
Desired_Time_to_Remove_Insurgents, false, false );
  }
  if (!_publicOnly) {
    drawParameter( _panel, _g, 860, 290, -30, -20,
"Removal_Effectiveness", Removal_Effectiveness, false, false );
  }
  if (!_publicOnly) {
    drawStock( _panel, _g, 168, 330, -98, 20,
"Government_Supporters", Government_Supporters, false );
  }
  if (!_publicOnly) {
    drawStock( _panel, _g, 408, 330, -10, -20, "Dissidents", Dissidents,
false );
  }
  if (!_publicOnly) {
    drawStock( _panel, _g, 648, 330, -10, -20, "Insurgents", Insurgents,
false );
```

```
  }
  if (!_publicOnly) {
    drawStock( _panel, _g, 888, 330, -18, -20, "Removed_Insurgents", Removed_Insurgents, false );
  }
  if (!_publicOnly) {
    drawStock( _panel, _g, 528, 610, -88, -20, "Perceived_Intencity_of_Anti_Regime_Messages", Perceived_Intencity_of_Anti_Regime_Messages, false );
  }
  if (!_publicOnly) {
    drawFlow( _panel, _g, 288, 330, -58, -20, "Becoming_Dissident", Becoming_Dissident, false, false );
  }
  if (!_publicOnly) {
    drawFlow( _panel, _g, 528, 330, -58, 20, "Insurgent_Recruitment", Insurgent_Recruitment, false, false );
  }
  if (!_publicOnly) {
    drawFlow( _panel, _g, 768, 330, -40, -20, "Removing_Insurgents", Removing_Insurgents, false, false );
  }
  if (!_publicOnly) {
    drawFlow( _panel, _g, 58, 330, -48, 0, "Births", Births, false, false );
  }
  if (!_publicOnly) {
    drawFlow( _panel, _g, 288, 270, -88, -20, "Appeasement_Rate", Appeasement_Rate, false, false );
  }
  if (!_publicOnly) {
    drawAuxiliaryVariable( _panel, _g, 238, 390, -50, -20, "Recruits_Through_Social_Network", Recruits_Through_Social_Network, false, false, false );
  }
  if (!_publicOnly) {
    drawAuxiliaryVariable( _panel, _g, 460, 400, -30, -20, "Regime_Opponents", Regime_Opponents, false, false, false );
  }
  if (!_publicOnly) {
    drawAuxiliaryVariable( _panel, _g, 720, 430, 10, -10, "Violent_Incident_Intensity", Violent_Incident_Intensity, false, false, false );
  }
  if (!_publicOnly) {
    drawAuxiliaryVariable( _panel, _g, 810, 450, 20, 0, "Propensity_to_Commit_Violence", Propensity_to_Commit_Violence, true, false, false );
  }
  if (!_publicOnly) {
    drawAuxiliaryVariable( _panel, _g, 560, 500, -130, 10, "Propensity_to_Protest", Propensity_to_Protest, true, false, false );
  }
  if (!_publicOnly) {
    drawAuxiliaryVariable( _panel, _g, 628, 470, -30, -20, "Protest_Intensity", Protest_Intensity, false, false, false );
  }
  if (!_publicOnly) {
    drawAuxiliaryVariable( _panel, _g, 720, 500, -30, -20, "Incident_Intensity", Incident_Intensity, false, false, false );
  }
  if (!_publicOnly) {
    drawAuxiliaryVariable( _panel, _g, 810, 540, -60, -20, "Relative_Strength_of_Violent_Incidents", Relative_Strength_of_Violent_Incidents, true, false, false );
  }
  if (!_publicOnly) {
    drawFlow( _panel, _g, 750, 610, -30, -20, "Effect_of_Incidents_on_Anti_Regime_Messages", Effect_of_Incidents_on_Anti_Regime_Messages, false, false );
  }
  if (!_publicOnly) {
    drawAuxiliaryVariable( _panel, _g, 700, 650, -290, 0, "Strength_of_Incident_Effect_on_Anti_Regime_Messages", Strength_of_Incident_Effect_on_Anti_Regime_Messages, true, false, false );
  }
  if (!_publicOnly) {
    drawFlow( _panel, _g, 310, 610, -280, 0, "Effect_of_Anti_Regime_Messages_on_Recruitment", Effect_of_Anti_Regime_Messages_on_Recruitment, false, false );
  }
  if (!_publicOnly) {
    drawAuxiliaryVariable( _panel, _g, 200, 530, -50, -20, "Propensity_to_be_Recruited", Propensity_to_be_Recruited, false, false, false );
  }
  if (!_publicOnly) {
    drawAuxiliaryVariable( _panel, _g, 90, 460, -60, -20, "Normal_Propensity_to_be_Recruited", Normal_Propensity_to_be_Recruited, true, false, false );
  }
  if (!_publicOnly) {
    drawAuxiliaryVariable( _panel, _g, 440, 470, -30, -20, "Dissidents_Fraction", Dissidents_Fraction, true, false, false );
  }
  if (!_publicOnly) {
    drawAuxiliaryVariable( _panel, _g, 620, 390, 10, -10, "Insurgents_Fraction", Insurgents_Fraction, true, false, false );
  }
  if (!_publicOnly) {
    drawAuxiliaryVariable( _panel, _g, 690, 270, -40, -20, "Indicated_Force_Strength", Indicated_Force_Strength, false, false, false );
  }
  if (!_publicOnly) {
    drawAuxiliaryVariable( _panel, _g, 520, 560, -60, -20, "Normal_Frequency_of_Anti_Regime_Messages", Normal_Frequency_of_Anti_Regime_Messages, false, false, false );
  }
 }

 @Override
 public boolean onClickModelAt( Panel panel, double x, double y, int clickCount, boolean publicOnly ) {
  if( !publicOnly && modelElementContains(x, y, 60, 280) ) {
   panel.addInspect( 60, 280, this, "People_Birth" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 300, 450) ) {
   panel.addInspect( 300, 450, this, "Contact_Rate" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 290, 650) ) {
   panel.addInspect( 290, 650, this, "People_Reaction_Delay_On_Anti_Regime_Messages" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 100, 570) ) {
   panel.addInspect( 100, 570, this, "Regime_Social_and_Economic_Efforts" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 370, 230) ) {
   panel.addInspect( 370, 230, this, "Avg_Time_as_Dissident" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 510, 270) ) {
   panel.addInspect( 510, 270, this, "Appeasement_Fraction" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, 820, 250) ) {
```

```
    panel.addInspect( 820, 250, this,
"Desired_Time_to_Remove_Insurgents" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 860, 290) ) {
    panel.addInspect( 860, 290, this, "Removal_Effectiveness" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 288, 330) ) {
    panel.addInspect( 288, 330, this, "Becoming_Dissident" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 528, 330) ) {
    panel.addInspect( 528, 330, this, "Insurgent_Recruitment" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 768, 330) ) {
    panel.addInspect( 768, 330, this, "Removing_Insurgents" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 58, 330) ) {
    panel.addInspect( 58, 330, this, "Births" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 288, 270) ) {
    panel.addInspect( 288, 270, this, "Appeasement_Rate" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 238, 390) ) {
    panel.addInspect( 238, 390, this,
"Recruits_Through_Social_Network" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 460, 400) ) {
    panel.addInspect( 460, 400, this, "Regime_Opponents" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 720, 430) ) {
    panel.addInspect( 720, 430, this, "Violent_Incident_Intensity" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 810, 450) ) {
    panel.addInspect( 810, 450, this, "Propensity_to_Commit_Violence"
);
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 560, 500) ) {
    panel.addInspect( 560, 500, this, "Propensity_to_Protest" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 628, 470) ) {
    panel.addInspect( 628, 470, this, "Protest_Intensity" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 720, 500) ) {
    panel.addInspect( 720, 500, this, "Incident_Intensity" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 810, 540) ) {
    panel.addInspect( 810, 540, this,
"Relative_Strength_of_Violent_Incidents" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 750, 610) ) {
    panel.addInspect( 750, 610, this,
"Effect_of_Incidents_on_Anti_Regime_Messages" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 700, 650) ) {
    panel.addInspect( 700, 650, this,
"Strength_of_Incident_Effect_on_Anti_Regime_Messages" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 310, 610) ) {
    panel.addInspect( 310, 610, this,
"Effect_of_Anti_Regime_Messages_on_Recruitment" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 200, 530) ) {
    panel.addInspect( 200, 530, this, "Propensity_to_be_Recruited" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 90, 460) ) {
    panel.addInspect( 90, 460, this,
"Normal_Propensity_to_be_Recruited" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 440, 470) ) {
    panel.addInspect( 440, 470, this, "Dissidents_Fraction" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 620, 390) ) {
    panel.addInspect( 620, 390, this, "Insurgents_Fraction" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 690, 270) ) {
    panel.addInspect( 690, 270, this, "Indicated_Force_Strength" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 520, 560) ) {
    panel.addInspect( 520, 560, this,
"Normal_Frequency_of_Anti_Regime_Messages" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 168, 330) ) {
    panel.addInspect( 168, 330, this, "Government_Supporters" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 408, 330) ) {
    panel.addInspect( 408, 330, this, "Dissidents" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 648, 330) ) {
    panel.addInspect( 648, 330, this, "Insurgents" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 888, 330) ) {
    panel.addInspect( 888, 330, this, "Removed_Insurgents" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, 528, 610) ) {
    panel.addInspect( 528, 610, this,
"Perceived_Intencity_of_Anti_Regime_Messages" );
    return true;
   }
   return false;
  }

  /**
   * Constructor
   */
  public SDModel( Engine engine, ActiveObject owner,
ActiveObjectCollection<? extends SDModel> collection ) {
   super( engine, owner, collection );
```

```
    // Assigning initial conditions:
    Dissidents_Fraction = _Dissidents_Fraction_Value();
    Insurgents_Fraction = _Insurgents_Fraction_Value();
    Propensity_to_Commit_Violence =
_Propensity_to_Commit_Violence_Value();
    Propensity_to_Protest = _Propensity_to_Protest_Value();
    Normal_Propensity_to_be_Recruited =
_Normal_Propensity_to_be_Recruited_Value();
    Relative_Strength_of_Violent_Incidents =
_Relative_Strength_of_Violent_Incidents_Value();
    Strength_of_Incident_Effect_on_Anti_Regime_Messages =
_Strength_of_Incident_Effect_on_Anti_Regime_Messages_Value();
    // Registering in Engine continuous part
    getEngine().registerActiveObjectWithEquations( this );
  }

  @Override
  public void create() {
    // Dynamic initialization of persistent elements
    {
    DataSet _item;
    List<DataSet> _items = new ArrayList<DataSet>( 4 );
    _items.add( _plot2_expression0_dataSet_xjal );
    _items.add( _plot2_expression1_dataSet_xjal );
    _items.add( _plot2_expression2_dataSet_xjal );
    _items.add( _plot2_expression3_dataSet_xjal );
    List<String> _titles = new ArrayList<String>( 4 );
    _titles.add( "Government supporters" );
    _titles.add( "Dissidents" );
    _titles.add( "Insurgents" );
    _titles.add( "Removed insurgents" );
    List<Chart2DPlot.Appearance> _appearances = new
ArrayList<Chart2DPlot.Appearance>( 4 );
    _appearances.add( new Chart2DPlot.Appearance( limeGreen, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
    _appearances.add( new Chart2DPlot.Appearance( gold, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
    _appearances.add( new Chart2DPlot.Appearance( orangeRed, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
    _appearances.add( new Chart2DPlot.Appearance( yellowGreen, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
    plot2 = new TimePlot(
                                SDModel.this, true, 450, 50,
                                        590, 150,
          null, null,
          60, 20,
                                        320, 110, white, black, black,
          190, Chart.EAST,

100
          , null, Chart.SCALE_AUTO,
          0, 0, Chart.GRID_DEFAULT, Chart.GRID_DEFAULT,
          darkGray, darkGray, _items, _titles, _appearances
    );
    }
    presentation = new ShapeGroup( SDModel.this, true, 0, 0, 0, line1,
text, rectangle40, text27, line67, line57, text18, line53, line54, line55,
rectangle37, text19, rectangle34, text17, plot2 );
    icon = new ShapeGroup( SDModel.this, true, 0, 0, 0
      ,line66                           );
                // Port connectors with non-replicated objects
    // Creating replicated embedded objects
    assignInitialConditions();
    onCreate();
  }

  @Override
  public void start() {
    _plot2_autoUpdateEvent_xjal.start();
    _autoCreatedDS_xjal.start();
    onStartup();
  }


// User API ---------------------------------------
  public Main get_Main() {
    ActiveObject owner = getOwner();
    if ( owner instanceof Main ) return (Main) owner;
    return null;
  }

  public void onDestroy() {
    super.onDestroy();
    _plot2_autoUpdateEvent_xjal.onDestroy();
    _autoCreatedDS_xjal.onDestroy();
    // Unregistering in Engine continuous part
    getEngine().unregisterActiveObjectWithEquations( this );
  }
}
```

Person.java

```
package insurgencySDABModel;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
```

```
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import java.awt.geom.Arc2D;

public class Person extends Agent
{

 // Statecharts
 public Statechart statechart = new Statechart( this, (short)4 );

 @Override
 public String getNameOf( Statechart _s ) {
  if(_s == this.statechart) return "statechart";
  return super.getNameOf( _s );
 }

 @Override
 public void executeActionOf( Statechart _s ) {
  if( _s == this.statechart ) {

// jump to supporters population
setXY( uniform(70,120), uniform(280,330));
setRotation( 0, true );
;
    enterState( Government_Supporter, true );
    return;
  }
  super.executeActionOf( _s );
 }

  // States of all statecharts

 public static final short Government_Supporter = 0;
 public static final short Dissident = 1;
 public static final short Insurgent = 2;
 public static final short Removed_Insurgent = 3;
 public static final short Active_Insurgent = 4;
 public static final short Inactive_Insurgent = 5;
 public static final short Active_Dissident = 6;
 public static final short Inactive_Dissident = 7;
 public static final short branch = 8;


 @Override
 public String getNameOfState( short _state ) {
  switch( _state ) {
   case Government_Supporter: return "Government_Supporter";
   case Dissident: return "Dissident";
   case Insurgent: return "Insurgent";
   case Removed_Insurgent: return "Removed_Insurgent";
   case Active_Insurgent: return "Active_Insurgent";
   case Inactive_Insurgent: return "Inactive_Insurgent";
   case Active_Dissident: return "Active_Dissident";
   case Inactive_Dissident: return "Inactive_Dissident";
   case branch: return "branch";
   default: return super.getNameOfState( _state );
  }
 }

 @Override
 public boolean stateContainsState( short compstate, short simpstate ) {
  if (compstate == Dissident && (simpstate == Inactive_Dissident ||
simpstate == Active_Dissident)) {
    return true;
  }
  if (compstate == Insurgent && (simpstate == Active_Insurgent ||
simpstate == Inactive_Insurgent)) {
    return true;
  }
  return super.stateContainsState( compstate, simpstate );
 }

 @Override
 public short getContainerStateOf( short _state ) {
  switch( _state ) {
   case Active_Insurgent: return Insurgent;
   case Inactive_Insurgent: return Insurgent;
   case Active_Dissident: return Dissident;
   case Inactive_Dissident: return Dissident;
   default: return super.getContainerStateOf( _state );
  }
 }

 @Override
 public void enterState( short _state, boolean _destination ) {
  switch( _state ) {
   case Government_Supporter: // (Simple state (not composite))
    statechart.setActiveState( Government_Supporter );
    {
((ABModel)getOwner()).Government_Supporters++;
;}
    transition.start();
    transition4.start();
    transition9.start();
    return;
   case Dissident: // (Composite state)
    {
((ABModel)getOwner()).Dissidents++;
;}
    transition10.start();
    transition11.start();
    if ( _destination ) {
     enterState( Active_Dissident, true );
    }
    return;
   case Insurgent: // (Composite state)
    {
((ABModel)getOwner()).Insurgents++;
//traceln(this.statechart.getActiveSimpleState());
;}
    transition3.start();
    if ( _destination ) {
     enterState( Active_Insurgent, true );
    }
    return;
   case Removed_Insurgent: // (Simple state (not composite))
    statechart.setActiveState( Removed_Insurgent );
    {
// update counters
((ABModel)getOwner()).Removed_Insurgents++;

// remove this person from agents population
((ABModel)getOwner()).remove_people(this);

// if this agent was displayed then select another agent to display
if (getPresentation().getPanel().getPresentable() == this) {
        getEngine().getPresentation().setPresentable(((Main)getEn
gine().getRoot()).aM.people.get(0));
}
;}
```

```
      return;
    case Active_Insurgent: // (Simple state (not composite))
      statechart.setActiveState( Active_Insurgent );
      transition7.start();
      transition2.start();
      return;
    case Inactive_Insurgent: // (Simple state (not composite))
      statechart.setActiveState( Inactive_Insurgent );
      transition8.start();
      return;
    case Active_Dissident: // (Simple state (not composite))
      statechart.setActiveState( Active_Dissident );
      transition5.start();
      transition1.start();
      return;
    case Inactive_Dissident: // (Simple state (not composite))
      statechart.setActiveState( Inactive_Dissident );
      transition6.start();
      return;
    case branch: // (Branch)
      if (
uniform() < ((ABModel)getOwner()).Propensity_to_be_Recruited
 ) { // transition12
        {
// move to dissident population
moveTo( uniform( 270, 320 ), this.getY() );
// change color
agentRectangle.setFillColor(darkOrange);
// update counters
((ABModel)getOwner()).ToDisidents++;
;}
        enterState( Dissident, true );
        return;
      }
      // transition13 (default)
        enterState( Government_Supporter, true );
      return;
    default:
      super.enterState( _state, _destination );
      return;
   }
  }

  @Override
  public void exitState( short _state, Transition _t, boolean _source,
Statechart _statechart) {
   switch( _state ) {
    case Government_Supporter: // (Simple state (not composite))
      if ( !_source || _t != transition) transition.cancel();
      if ( !_source || _t != transition4) transition4.cancel();
      if ( !_source || _t != transition9) transition9.cancel();
      {
((ABModel)getOwner()).Government_Supporters--;
;}
      return;
    case Dissident: // (Composite state)
      if ( _source ) exitInnerStates(_state, _statechart);
      if ( !_source || _t != transition10 ) transition10.cancel();
      if ( !_source || _t != transition11 ) transition11.cancel();
      {
((ABModel)getOwner()).Dissidents--;
;}
      return;
    case Insurgent: // (Composite state)
      if ( _source ) exitInnerStates(_state, _statechart);
      if ( !_source || _t != transition3 ) transition3.cancel();
      {
((ABModel)getOwner()).Insurgents--;
;}
      return;
    case Removed_Insurgent: // (Simple state (not composite))
      return;
    case Active_Insurgent: // (Simple state (not composite))
      if ( !_source || _t != transition7) transition7.cancel();
      if ( !_source || _t != transition2) transition2.cancel();
      return;
    case Inactive_Insurgent: // (Simple state (not composite))
      if ( !_source || _t != transition8) transition8.cancel();
      return;
    case Active_Dissident: // (Simple state (not composite))
      if ( !_source || _t != transition5) transition5.cancel();
      if ( !_source || _t != transition1) transition1.cancel();
      return;
    case Inactive_Dissident: // (Simple state (not composite))
      if ( !_source || _t != transition6) transition6.cancel();
      return;
    default:
      super.exitState( _state, _t, _source, _statechart);
      return;
   }
  }


  public TransitionRate transition10 = new TransitionRate( this );
  public TransitionRate transition11 = new TransitionRate( this );
  public TransitionRate transition3 = new TransitionRate( this );
  public TransitionRate transition7 = new TransitionRate( this );
  public TransitionRate transition8 = new TransitionRate( this );
  public TransitionRate transition2 = new TransitionRate( this );
  public TransitionRate transition5 = new TransitionRate( this );
  public TransitionRate transition6 = new TransitionRate( this );
  public TransitionRate transition1 = new TransitionRate( this );

  @Override
  public String getNameOf( TransitionRate _t ) {
   if ( _t == transition10 ) return "transition10";
   if ( _t == transition11 ) return "transition11";
   if ( _t == transition3 ) return "transition3";
   if ( _t == transition7 ) return "transition7";
   if ( _t == transition8 ) return "transition8";
   if ( _t == transition2 ) return "transition2";
   if ( _t == transition5 ) return "transition5";
   if ( _t == transition6 ) return "transition6";
   if ( _t == transition1 ) return "transition1";
   return super.getNameOf( _t );
  }

  @Override
  public Statechart getStatechartOf( TransitionRate _t ) {
   if ( _t == transition10 ) return statechart;
   if ( _t == transition11 ) return statechart;
   if ( _t == transition3 ) return statechart;
   if ( _t == transition7 ) return statechart;
   if ( _t == transition8 ) return statechart;
   if ( _t == transition2 ) return statechart;
   if ( _t == transition5 ) return statechart;
   if ( _t == transition6 ) return statechart;
   if ( _t == transition1 ) return statechart;
   return super.getStatechartOf( _t );
  }
```

```
 @Override
 public void executeActionOf( TransitionRate _t ) {
  if ( _t == transition10 ) {
   exitState( Dissident, _t, true, statechart );
   {
this.moveTo(uniform(70,120),this.getY());
this.agentRectangle.setFillColor(lime);
((ABModel)getOwner()).ToGovernment_Supporters++;
;}
    enterState( Government_Supporter, true );
   return;
  }
  if ( _t == transition11 ) {
   exitState( Dissident, _t, true, statechart );
   {
// move to insurgent population
moveTo(uniform(470,520),this.getY());
agentRectangle.setFillColor(orangeRed);
// update counters
((ABModel)getOwner()).ToInsurgents++;
;}
    enterState( Insurgent, true );
   return;
  }
  if ( _t == transition3 ) {
   exitState( Insurgent, _t, true, statechart );
   {
((ABModel)getOwner()).ToRemoved_Insurgents++;
;}
    enterState( Removed_Insurgent, true );
   return;
  }
  if ( _t == transition7 ) {
   exitState( Active_Insurgent, _t, true, statechart );
    enterState( Inactive_Insurgent, true );
   return;
  }
  if ( _t == transition8 ) {
   exitState( Inactive_Insurgent, _t, true, statechart );
    enterState( Active_Insurgent, true );
   return;
  }
  if ( _t == transition2 ) {
   {
send("ToDissidents", RANDOM);
;}
   transition2.start();
   return;
  }
  if ( _t == transition5 ) {
   exitState( Active_Dissident, _t, true, statechart );
    enterState( Inactive_Dissident, true );
   return;
  }
  if ( _t == transition6 ) {
   exitState( Inactive_Dissident, _t, true, statechart );
    enterState( Active_Dissident, true );
   return;
  }
  if ( _t == transition1 ) {
   {
send("ToDissidents", RANDOM);
;}
   transition1.start();
   return;
  }
  super.executeActionOf( _t );
 }
 @Override
 public double evaluateRateOf( TransitionRate _t ) {
  if ( _t == transition10 ) return
(1 / ((ABModel)getOwner()).Avg_Time_as_Dissident) *
((ABModel)getOwner()).Appeasement_Fraction
;
  if ( _t == transition11 ) return
(1 / ((ABModel)getOwner()).Avg_Time_as_Dissident) * (1 -
((ABModel)getOwner()).Appeasement_Fraction)
;
  if ( _t == transition3 ) return
((ABModel)getOwner()).Removal_Effectiveness /
((ABModel)getOwner()).Desired_Time_to_Remove_Insurgents
;
  if ( _t == transition7 ) return
1 - ((ABModel)getOwner()).Insurgents_Fraction
;
  if ( _t == transition8 ) return
((ABModel)getOwner()).Insurgents_Fraction
;
  if ( _t == transition2 ) return
((ABModel)getOwner()).Contact_Rate
;
  if ( _t == transition5 ) return
1 - ((ABModel)getOwner()).Dissidents_Fraction
;
  if ( _t == transition6 ) return
((ABModel)getOwner()).Dissidents_Fraction
;
  if ( _t == transition1 ) return
((ABModel)getOwner()).Contact_Rate
;
  return super.evaluateRateOf( _t );
 }

 public TransitionMessage transition = new TransitionMessage( this );
 public TransitionMessage transition4 = new TransitionMessage( this );
 public TransitionMessage transition9 = new TransitionMessage( this );

 @Override
 public String getNameOf( TransitionMessage _t ) {
  if ( _t == transition ) return "transition";
  if ( _t == transition4 ) return "transition4";
  if ( _t == transition9 ) return "transition9";
  return super.getNameOf( _t );
 }

 @Override
 public Statechart getStatechartOf( TransitionMessage _t ) {
  if ( _t == transition ) return statechart;
  if ( _t == transition4 ) return statechart;
  if ( _t == transition9 ) return statechart;
  return super.getStatechartOf( _t );
 }

 @Override
 public void executeActionOf( TransitionMessage _t, Object _msg ) {
  if ( _t == transition ) {
   exitState( Government_Supporter, _t, true, statechart );
    enterState( branch, true );
   return;
  }
  if ( _t == transition4 ) {
```

```
      exitState( Government_Supporter, _t, true, statechart );
      {
        String msg = (String) _msg;
// jump to dissident population
setXY( uniform( 270, 320 ), uniform( 280, 330 ) );
// change color
agentRectangle.setFillColor(darkOrange);
;}
        enterState( Dissident, true );
      return;
    }
    if ( _t == transition9 ) {
      exitState( Government_Supporter, _t, true, statechart );
      {
        String msg = (String) _msg;
// jump to insergent population
setXY( uniform( 470,520 ), uniform( 280,330 ) );
agentRectangle.setFillColor( orangeRed );
;}
        enterState( Insurgent, true );
      return;
    }
    super.executeActionOf( _t, _msg );
  }
  @Override
  public boolean testMessageOf( TransitionMessage _t, Object _msg ) {
    if ( _t == transition ) {
      if ( !(_msg instanceof String) )
        return false;
      String msg = (String) _msg;
      Object _g =
"ToDissidents"
;
      return msg.equals( _g );
    }
    if ( _t == transition4 ) {
      if ( !(_msg instanceof String) )
        return false;
      String msg = (String) _msg;
      Object _g =
"Dissident"
;
      return msg.equals( _g );
    }
    if ( _t == transition9 ) {
      if ( !(_msg instanceof String) )
        return false;
      String msg = (String) _msg;
      Object _g =
"Insurgent"
;
      return msg.equals( _g );
    }
    return super.testMessageOf( _t, _msg );
  }
  static final Font _button_Font = new Font("Dialog", 0, 11 );
  static final Font _button1_Font = _button_Font;
  static final Font _text2_Font = new Font("SansSerif", 1, 12 );
  static final Font _text3_Font = new Font("SansSerif", 0, 12 );
  static final Font _text4_Font = _text3_Font;
  static final Font _text5_Font = _text3_Font;
  static final Font _text6_Font = _text3_Font;
  static final Font _text1_Font = new Font("SansSerif", 0, 18 );
  static final Font _text7_Font = _text3_Font;
  static final Font _text_Font = new Font("SansSerif", 0, 32 );
  static final Font _text27_Font = _text1_Font;
  static final Font _text21_Font = _text1_Font;
  static final Font _text17_Font = _text1_Font;
  static final Font _text18_Font = _text1_Font;
  static final int _button = 1;
  static final int _button1 = 2;
  static final int _agentRectangle = 3;
  static final int _text2 = 4;
  static final int _text3 = 5;
  static final int _text4 = 6;
  static final int _text5 = 7;
  static final int _text6 = 8;
  static final int _oval = 9;
  static final int _text1 = 10;
  static final int _text7 = 11;
  static final int _group = 12;
  static final int _rectangle34 = 13;
  static final int _rectangle37 = 14;
  static final int _rectangle39 = 15;
  static final int _line66 = 16;
  static final int _text = 17;
  static final int line1 = 18;
  static final int _rectangle40 = 19;
  static final int _line67 = 20;
  static final int _text27 = 21;
  static final int _text21 = 22;
  static final int _line59 = 23;
  static final int _line54 = 24;
  static final int _line53 = 25;
  static final int _text17 = 26;
  static final int _text18 = 27;
  static final int _line57 = 28;
  static final int _line55 = 29;
  static final int _group1 = 30;

  /**
   * Top-level presentation group id
   */
  static final int _presentation = 0;

  /**
   * Top-level icon group id
   */
  static final int _icon = -1;


  @Override
  public boolean onShapeClick( int _shape, int index, double clickx,
double clicky ){
    switch( _shape ){
      case _agentRectangle:
        if (true) {

getEngine().getPresentation().setPresentable(this);
getPresentation().getPanel().setOffsets( 0, 0 );
        }
        break;
      case _rectangle34:
        if (true) {

get_ABModel().get_Main().sD.SDView.navigateTo();
        }
        break;
      case _rectangle37:
        if (true) {
```

```
get_ABModel().get_Main().MainView.navigateTo();
      }
      break;
    case _rectangle39:
      if (true) {

get_ABModel().ABView.navigateTo();
      }
      break;
    case _rectangle40:
      if (true) {

get_ABModel().get_Main().DescriptionView.navigateTo();
      }
      break;
    default: return super.onShapeClick( _shape, index, clickx, clicky );
              }
              return false;
  }

  @Override
  public void executeShapeControlAction( int _shape, int index ) {
    switch( _shape ) {
      case _button: {
if( this.getIndex() < get_ABModel().people.size()-1 ){
              getEngine().getPresentation().setPresentable(
get_ABModel().people.get( this.getIndex() +1 ) );
}
;}
      break;
    case _button1: {
if( this.getIndex() > 0 ){
              getEngine().getPresentation().setPresentable(
get_ABModel().people.get( this.getIndex()-1 ) );
}

;}
      break;
    default:
      super.executeShapeControlAction( _shape, index );
      break;
    }
  }


  @Override
  public String getNameOfShape( int _shape ) {
    switch( _shape ) {
      case line1: return "line1";
      default: return super.getNameOfShape( _shape );
    }
  }

  @Override
  public int getShapeType( int _shape ) {
    switch( _shape ) {
      case line1: return SHAPE_LINE;
      default: return super.getShapeType( _shape );
    }
  }

  @Override
  public boolean isShapePublic( int _shape ) {
    switch( _shape ) {
      case line1:
        return false;
    default:
      return true;
    }
  }

  @Override
  public double getShapeX( int _shape, int index ) {
    switch( _shape ) {
      case line1: return -490;
      default: return super.getShapeX( _shape, index );
    }
  }

  @Override
  public double getShapeY( int _shape, int index ) {
    switch( _shape ) {
      case line1: return -10;
      default: return super.getShapeY( _shape, index );
    }
  }

  @Override
  public Color getShapeLineColor( int _shape, int index ) {
    switch( _shape ) {
      // shapes
      case line1: return darkGray;
      // controls (text color)
      // charts (border color)
      default: return super.getShapeLineColor( _shape, index );
    }
  }

  @Override
  public int getShapeLineStyle( int _shape, int index ) {
    switch( _shape ) {
      case line1: return LINE_STYLE_SOLID;
      default: return super.getShapeLineStyle( _shape, index );
    }
  }

  @Override
  public double getShapeLineDx( int _shape, int index ) {
    switch( _shape ) {
      case line1: return 970;
      default: return super.getShapeLineDx( _shape, index );
    }
  }

  @Override
  public double getShapeLineDy( int _shape, int index ) {
    switch( _shape ) {
      case line1: return 0;
      default: return super.getShapeLineDy( _shape, index );
    }
  }


  ShapeButton button;
  ShapeButton button1;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _agentRectangle_SetDynamicParams( ShapeRectangle
shape ) {
    boolean _visible;
```

```
    _visible =
((getPresentation().getPanel().getPresentable() == this) ? false : true)
;
    shape.setVisible( _visible );
                if ( _visible ) {
                }
  }

  ShapeRectangle agentRectangle;
  ShapeText text2;
  ShapeText text3;
  ShapeText text4;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _text5_SetDynamicParams( ShapeText shape ) {
    boolean _visible;
    shape.setText(
"" + hashCode()
);
  }

  ShapeText text5;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _text6_SetDynamicParams( ShapeText shape ) {
    boolean _visible;
    shape.setText(
this.getNameOfState(this.statechart.getActiveSimpleState())
);
  }

  ShapeText text6;
  ShapeOval oval;
  ShapeText text1;
  ShapeText text7;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _group_SetDynamicParams( ShapeGroup shape ) {
    boolean _visible;

    shape.setX(
550 - this.getX()
);
    shape.setY(
260 - this.getY()
);
  }

  ShapeGroup group;
  ShapeRectangle rectangle34;
  ShapeRectangle rectangle37;
  ShapeRectangle rectangle39;
  ShapeLine line66;
  ShapeText text;
  ShapeRectangle rectangle40;
  ShapeLine line67;
  ShapeText text27;
  ShapeText text21;
  ShapeLine line59;
  ShapeLine line54;
  ShapeLine line53;
  ShapeText text17;
  ShapeText text18;
  ShapeLine line57;
  ShapeLine line55;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _group1_SetDynamicParams( ShapeGroup shape ) {
    boolean _visible;

    shape.setX(
520 - this.getX()
);
    shape.setY(
60 - this.getY()
);
  }

  ShapeGroup group1;

  // Static initialization of persistent elements
  {
    button = new ShapeButton(
                                        Person.this, false, -70, 50,
                                        130, 30,
          controlDefault, controlDefault,
          _button_Font,
                                        "Select next agent"
    ) {

      @Override
      public void action(){
        executeShapeControlAction( _button, 0 );
      }
    };
    button1 = new ShapeButton(
                                        Person.this, false, -70, 90,
                                        130, 30,
          controlDefault, controlDefault,
          _button1_Font,
                                        "Select prev agent"
    ) {

      @Override
      public void action(){
        executeShapeControlAction( _button1, 0 );
      }
    };
    agentRectangle = new ShapeRectangle(
      true,0, 0, 0.0,
          null, lime,
                                        2, 2,
                                        1, LINE_STYLE_SOLID
    ) {

      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
        _agentRectangle_SetDynamicParams( this );
        super.draw( panel, graphics, xform, publicOnly );
      }

      @Override
```

```
    public boolean onClick( double clickx, double clicky ) {
      return onShapeClick( _agentRectangle, 0, clickx, clicky );
    }
  };
  text2 = new ShapeText(
    false,-70, -30, 0.0,
    black,"Agent: ",
    _text2_Font, ALIGNMENT_LEFT
  );
  text3 = new ShapeText(
    false,-70, 0, 0.0,
    black,"id: ",
    _text3_Font, ALIGNMENT_LEFT
  );
  text4 = new ShapeText(
    false,-70, 20, 0.0,
    black,"current state: ",
    _text4_Font, ALIGNMENT_LEFT
  );
  text5 = new ShapeText(
    false,15, 0, 0.0,
    navy,"123",
    _text5_Font, ALIGNMENT_LEFT
  ) {

    @Override
    public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
      _text5_SetDynamicParams( this );
      super.draw( panel, graphics, xform, publicOnly );
    }
  };
  text6 = new ShapeText(
    false,15, 20, 0.0,
    navy,"123",
    _text6_Font, ALIGNMENT_LEFT
  ) {

    @Override
    public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
      _text6_SetDynamicParams( this );
      super.draw( panel, graphics, xform, publicOnly );
    }
  };
  oval = new ShapeOval(
    false,-76, 190, 0.0,
    null, yellow,
          11, 10,
          1, LINE_STYLE_SOLID
  );
  text1 = new ShapeText(
    false,-80, 180, 0.0,
    navy,"?",
    _text1_Font, ALIGNMENT_LEFT
  );
  text7 = new ShapeText(
    false,-55, 180, 0.0,
    black,"Agents states (Government_Supporter, \r\nDissident,
Insurgent and Removed_Insurgent) \r\ncorrespond to stocks
(Government_Supporters, \r\nDissidents, Insurgents and
Removed_Insurgents) of \r\nSystem Dynamics model.",
    _text7_Font, ALIGNMENT_LEFT
  );
  rectangle34 = new ShapeRectangle(
    false,-390, 10, 0.0,
      null, null,
                                    90, 30,
                                    1, LINE_STYLE_SOLID
) {

  @Override
  public boolean onClick( double clickx, double clicky ) {
    return onShapeClick( _rectangle34, 0, clickx, clicky );
  }
};
rectangle37 = new ShapeRectangle(
  false,-290, 10, 0.0,
      null, null,
                                    110, 30,
                                    1, LINE_STYLE_SOLID
) {

  @Override
  public boolean onClick( double clickx, double clicky ) {
    return onShapeClick( _rectangle37, 0, clickx, clicky );
  }
};
rectangle39 = new ShapeRectangle(
  false,-490, 10, 0.0,
      null, null,
                                    90, 30,
                                    1, LINE_STYLE_SOLID
) {

  @Override
  public boolean onClick( double clickx, double clicky ) {
    return onShapeClick( _rectangle39, 0, clickx, clicky );
  }
};
line66 = new ShapeLine(
                    false,-175, 10,
  black,
                    0,          30,
                    1, LINE_STYLE_SOLID
);
text = new ShapeText(
  false,-490, -50, 0.0,
  black,"SD+AB Insurgency Models",
  _text_Font, ALIGNMENT_LEFT
);
rectangle40 = new ShapeRectangle(
  false,-171, 10, 0.0,
      null, null,
                                    105, 30,
                                    1, LINE_STYLE_SOLID
) {

  @Override
  public boolean onClick( double clickx, double clicky ) {
    return onShapeClick( _rectangle40, 0, clickx, clicky );
  }
};
line67 = new ShapeLine(
                    false,-165, 35,
  royalBlue,
                    94,         0,
                    1, LINE_STYLE_SOLID
);
text27 = new ShapeText(
  false,-166, 15, 0.0,
  royalBlue,"Description",
```

```
    _text27_Font, ALIGNMENT_LEFT
  );
  text21 = new ShapeText(
    false,-485, 15, 0.0,
    royalBlue,"AB model",
    _text21_Font, ALIGNMENT_LEFT
  );
  line59 = new ShapeLine(
                    false,-484, 35,
    royalBlue,
                    80,          0,
                    1, LINE_STYLE_SOLID
  );
  line54 = new ShapeLine(
                    false,-285, 35,
    royalBlue,
                    100,         0,
                    1, LINE_STYLE_SOLID
  );
  line53 = new ShapeLine(
                    false,-395, 10,
    black,
                    0,           30,
                    1, LINE_STYLE_SOLID
  );
  text17 = new ShapeText(
    false,-285, 15, 0.0,
    royalBlue,"Comparison",
    _text17_Font, ALIGNMENT_LEFT
  );
  text18 = new ShapeText(
    false,-385, 15, 0.0,
    royalBlue,"SD model",
    _text18_Font, ALIGNMENT_LEFT
  );
  line57 = new ShapeLine(
                    false,-384, 35,
    royalBlue,
                    80,          0,
                    1, LINE_STYLE_SOLID
  );
  line55 = new ShapeLine(
                    false,-295, 10,
    black,
                    0,           30,
                    1, LINE_STYLE_SOLID
  );
 }
 ShapeGroup presentation;
 ShapeGroup icon;

 @Override
 public Object getPersistentShape( int _shape ) {
  switch(_shape){
    case _presentation: return presentation;
    case _icon: return icon;

    case _button: return button;
    case _button1: return button1;
    case _agentRectangle: return agentRectangle;
    case _text2: return text2;
    case _text3: return text3;
    case _text4: return text4;
    case _text5: return text5;
    case _text6: return text6;
    case _oval: return oval;
    case _text1: return text1;
    case _text7: return text7;
    case _group: return group;
    case _rectangle34: return rectangle34;
    case _rectangle37: return rectangle37;
    case _rectangle39: return rectangle39;
    case _line66: return line66;
    case _text: return text;
    case _rectangle40: return rectangle40;
    case _line67: return line67;
    case _text27: return text27;
    case _text21: return text21;
    case _line59: return line59;
    case _line54: return line54;
    case _line53: return line53;
    case _text17: return text17;
    case _text18: return text18;
    case _line57: return line57;
    case _line55: return line55;
    case _group1: return group1;
    default: return null;
   }
  }

  static final int[] _transition_pointsX = {200, 200, };
  static final int[] _transition_pointsY = {200, 220, };
  static final int[] _transition12_pointsX = {200, 200, };
  static final int[] _transition12_pointsY = {240, 260, };
  static final int[] _transition10_pointsX = {300, 300, };
  static final int[] _transition10_pointsY = {260, 200, };
  static final int[] _transition11_pointsX = {250, 250, };
  static final int[] _transition11_pointsY = {370, 400, };
  static final int[] _transition3_pointsX = {250, 250, };
  static final int[] _transition3_pointsY = {510, 540, };
  static final int[] _transition7_pointsX = {240, 240, };
  static final int[] _transition7_pointsY = {450, 470, };
  static final int[] _transition8_pointsX = {290, 290, };
  static final int[] _transition8_pointsY = {470, 450, };
  static final int[] _transition2_pointsX = {310, 310, 330, };
  static final int[] _transition2_pointsY = {420, 440, 440, };
  static final int[] _transition5_pointsX = {240, 240, };
  static final int[] _transition5_pointsY = {310, 330, };
  static final int[] _transition6_pointsX = {290, 290, };
  static final int[] _transition6_pointsY = {330, 310, };
  static final int[] _transition1_pointsX = {310, 310, 330, };
  static final int[] _transition1_pointsY = {280, 300, 300, };
  static final int[] _transition13_pointsX = {212, 250, 250, };
  static final int[] _transition13_pointsY = {230, 230, 200, };
  static final int[] _transition4_pointsX = {150, 120, 120, 150, };
  static final int[] _transition4_pointsY = {190, 190, 320, 320, };
  static final int[] _transition9_pointsX = {150, 100, 100, 150, };
  static final int[] _transition9_pointsY = {180, 180, 460, 460, };


  @Override
  public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
    if (!_publicOnly) {
      drawState( _panel, _g, 150, 170, 200, 30, 10, 10,
"Government_Supporter", lime, Government_Supporter, statechart );
    }
    if (!_publicOnly) {
      drawState( _panel, _g, 150, 260, 200, 110, 10, 10, "Dissident", null,
Dissident, statechart );
    }
```

```
    if (!_publicOnly) {
      drawState( _panel, _g, 150, 400, 200, 110, 10, 10, "Insurgent", null, Insurgent, statechart );
    }
    if (!_publicOnly) {
      drawState( _panel, _g, 150, 540, 200, 30, 10, 10, "Removed_Insurgent", silver, Removed_Insurgent, statechart );
    }
    if (!_publicOnly) {
      drawState( _panel, _g, 200, 280, 130, 30, 10, 10, "Active_Dissident", orangeRed, Active_Dissident, statechart );
    }
    if (!_publicOnly) {
      drawState( _panel, _g, 200, 330, 130, 30, 10, 10, "Inactive_Dissident", GOLD, Inactive_Dissident, statechart );
    }
    if (!_publicOnly) {
      drawState( _panel, _g, 200, 420, 130, 30, 10, 10, "Active_Insurgent", orangeRed, Active_Insurgent, statechart );
    }
    if (!_publicOnly) {
      drawState( _panel, _g, 200, 470, 130, 30, 10, 10, "Inactive_Insurgent", GOLD, Inactive_Insurgent, statechart );
    }
    if (!_publicOnly) {
      drawBranchState( _panel, _g, 200, 230, 10, 0, null, branch );
    }
    if (!_publicOnly) {
      drawStatechartEntryPoint( _panel, _g, 250, 140, 250, 170, 261, 140, "statechart", statechart );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _transition_pointsX, _transition_pointsY, 210, 200, null, transition );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _transition3_pointsX, _transition3_pointsY, 260, 510, null, transition3 );
    }
    if (!_publicOnly) {
      drawInitialStatePointer( _panel, _g, 170, 300, 200, 300 );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _transition5_pointsX, _transition5_pointsY, 250, 310, null, transition5 );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _transition6_pointsX, _transition6_pointsY, 300, 330, null, transition6 );
    }
    if (!_publicOnly) {
      drawInitialStatePointer( _panel, _g, 170, 440, 200, 440 );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _transition7_pointsX, _transition7_pointsY, 250, 450, null, transition7 );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _transition8_pointsX, _transition8_pointsY, 300, 470, null, transition8 );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _transition4_pointsX, _transition4_pointsY, 160, 190, null, transition4 );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _transition9_pointsX, _transition9_pointsY, 180, 180, null, transition9 );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _transition1_pointsX, _transition1_pointsY, 320, 280, null, transition1 );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _transition2_pointsX, _transition2_pointsY, 320, 420, null, transition2 );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _transition10_pointsX, _transition10_pointsY, 310, 260, null, transition10 );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _transition11_pointsX, _transition11_pointsY, 260, 370, null, transition11 );
    }
    if (!_publicOnly) {
      drawBranchExit(_panel, _g, _transition12_pointsX, _transition12_pointsY, 210, 240, null );
    }
    if (!_publicOnly) {
      drawBranchExit(_panel, _g, _transition13_pointsX, _transition13_pointsY, 222, 230, null );
    }
  }

  @Override
  public void onReceive( Object msg, Agent sender ) {

statechart.receiveMessage(msg);
  }

  /**
   * Constructor
   */
  public Person( Engine engine, ActiveObject owner, ActiveObjectCollection<? extends Person> collection ) {
    super( engine, owner, collection );
  }

  @Override
  public void create() {
    // Dynamic initialization of persistent elements
    {
                        group = new ShapeGroup( Person.this, true,550, 260, 0.0

                    ,text2
                    ,text3
                    ,text4
                    ,text5
                    ,text6
                    ,oval
                    ,text1
                    ,text7
                    ,button
                    ,button1
    ) {

      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform xform, boolean publicOnly ) {
        _group_SetDynamicParams( this );
        super.draw( panel, graphics, xform, publicOnly );
      }
    };
    }
    {
```

```
                        group1 = new ShapeGroup( Person.this,
true,520, 60, 0.0

                ,rectangle34
                ,rectangle37
                ,rectangle39
                ,line66
                ,text
                ,line1
                ,rectangle40
                ,line67
                ,text27
                ,text21
                ,line59
                ,line54
                ,line53
                ,text17
                ,text18
                ,line57
                ,line55
  ) {

    @Override
    public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
      _group1_SetDynamicParams( this );
      super.draw( panel, graphics, xform, publicOnly );
    }
  };
  }
  presentation = new ShapeGroup( Person.this, true, 0, 0, 0,
agentRectangle, group, group1 ) {
    @Override
    public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
      setX( getShapeX( 0, 0 ) );
      setY( getShapeY( 0, 0 ) );
      setRotation( getShapeRotation( 0, 0 ) );
      setVisible( isShapeVisible( 0 ) );
      super.draw( panel, graphics, xform, publicOnly );
    }
  };
  icon = new ShapeGroup( Person.this, true, 0, 0, 0
);
  // Agent properties setup
  if ( getEnvironment() != null && (getEnvironment().getSpaceType()
!= Environment.SPACE_CONTINUOUS_2D ||
getEnvironment().getGISMap() != null) ) {
    throw new RuntimeException( "Space type of an agent class 'Person'
doesn't match its environment. See 'Agent' page on the Properties View
of 'Person' in the AnyLogic" );
  }
  setVelocity(
100
);
            // Port connectors with non-replicated objects
  // Creating replicated embedded objects
  assignInitialConditions();
  onCreate();
 }

 @Override
 public void start() {
  statechart.start();
  onStartup();
 }

// User API ----------------------------------
 public ABModel get_ABModel() {
  ActiveObject owner = getOwner();
  if ( owner instanceof ABModel ) return (ABModel) owner;
  return null;
 }

 // Reaction on changes ----------------------
 public void onChange() {
  statechart.onChange();
 }

 public void onDestroy() {
  super.onDestroy();
  statechart.onDestroy();
 }


}
```

Main.java

```
package insurgencySDABModel;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
```

```
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import java.awt.geom.Arc2D;

public class Main extends ActiveObject
{
  // Parameters
  public int  Initial_Dissident;

  /**
   * Returns default value for parameter
<code>Initial_Dissident</code>.
   * <i>This method should not be called by user</i>
   */
  public
int  _Initial_Dissident_DefaultValue_xjal() {
    return 0;
  }

  public void set_Initial_Dissident(
int  Initial_Dissident ) {
    if (Initial_Dissident == this.Initial_Dissident) {
      return;
    }
    this.Initial_Dissident = Initial_Dissident;
    onChange_Initial_Dissident();
    onChange();
  }

  void onChange_Initial_Dissident() {
  }


  public int  Initial_Government_Supporters;

  /**
   * Returns default value for parameter
<code>Initial_Government_Supporters</code>.
   * <i>This method should not be called by user</i>
   */
  public
int  _Initial_Government_Supporters_DefaultValue_xjal() {
    return 0;
  }

  public void set_Initial_Government_Supporters(
int  Initial_Government_Supporters ) {
    if (Initial_Government_Supporters ==
this.Initial_Government_Supporters) {
      return;
    }
    this.Initial_Government_Supporters = Initial_Government_Supporters;
    onChange_Initial_Government_Supporters();
    onChange();
  }

  void onChange_Initial_Government_Supporters() {
  }

  public int  Initial_Insurgent;

  /**
   * Returns default value for parameter
<code>Initial_Insurgent</code>.
   * <i>This method should not be called by user</i>
   */
  public
int  _Initial_Insurgent_DefaultValue_xjal() {
    return 0;
  }

  public void set_Initial_Insurgent(
int  Initial_Insurgent ) {
    if (Initial_Insurgent == this.Initial_Insurgent) {
      return;
    }
    this.Initial_Insurgent = Initial_Insurgent;
    onChange_Initial_Insurgent();
    onChange();
  }

  void onChange_Initial_Insurgent() {
  }


  public double  People_Reaction_Delay_On_Anti_Regime_Messages;

  /**
   * Returns default value for parameter
<code>People_Reaction_Delay_On_Anti_Regime_Messages</code>.
   * <i>This method should not be called by user</i>
   */
  public
double
_People_Reaction_Delay_On_Anti_Regime_Messages_DefaultValue_xjal(
) {
    return 0.0;
  }

  public void set_People_Reaction_Delay_On_Anti_Regime_Messages(
double  People_Reaction_Delay_On_Anti_Regime_Messages ) {
    if (People_Reaction_Delay_On_Anti_Regime_Messages ==
this.People_Reaction_Delay_On_Anti_Regime_Messages) {
      return;
    }
    this.People_Reaction_Delay_On_Anti_Regime_Messages =
People_Reaction_Delay_On_Anti_Regime_Messages;
    onChange_People_Reaction_Delay_On_Anti_Regime_Messages();
    onChange();
  }

  void onChange_People_Reaction_Delay_On_Anti_Regime_Messages() {
      int index;
    aM.set_People_Reaction_Delay_On_Anti_Regime_Messages(
People_Reaction_Delay_On_Anti_Regime_Messages );
    sD.set_People_Reaction_Delay_On_Anti_Regime_Messages(
People_Reaction_Delay_On_Anti_Regime_Messages );

  }


  public double  Regime_Social_and_Economic_Efforts;

  /**
   * Returns default value for parameter
<code>Regime_Social_and_Economic_Efforts</code>.
   * <i>This method should not be called by user</i>
   */
  public
double  _Regime_Social_and_Economic_Efforts_DefaultValue_xjal() {
    return 0.0;
  }

  public void set_Regime_Social_and_Economic_Efforts(
```

```
double  Regime_Social_and_Economic_Efforts ) {
   if                (Regime_Social_and_Economic_Efforts                ==
this.Regime_Social_and_Economic_Efforts) {
     return;
   }
   this.Regime_Social_and_Economic_Efforts                                =
Regime_Social_and_Economic_Efforts;
   onChange_Regime_Social_and_Economic_Efforts();
   onChange();
 }
 void onChange_Regime_Social_and_Economic_Efforts() {
               int index;
   aM.set_Regime_Social_and_Economic_Efforts(
Regime_Social_and_Economic_Efforts );
   sD.set_Regime_Social_and_Economic_Efforts(
Regime_Social_and_Economic_Efforts );

 }


 public double  Contact_Rate;

 /**
  * Returns default value for parameter <code>Contact_Rate</code>.
  * <i>This method should not be called by user</i>
  */
 public
double  _Contact_Rate_DefaultValue_xjal() {
   return 0.0;
 }

 public void set_Contact_Rate(
double  Contact_Rate ) {
   if (Contact_Rate == this.Contact_Rate) {
     return;
   }
   this.Contact_Rate = Contact_Rate;
   onChange_Contact_Rate();
   onChange();
 }

 void onChange_Contact_Rate() {
               int index;
   aM.set_Contact_Rate( Contact_Rate );
   sD.set_Contact_Rate( Contact_Rate );

 }


 public int  People_Birth;

 /**
  * Returns default value for parameter <code>People_Birth</code>.
  * <i>This method should not be called by user</i>
  */
 public
int  _People_Birth_DefaultValue_xjal() {
   return 0;
 }

 public void set_People_Birth(
int  People_Birth ) {
   if (People_Birth == this.People_Birth) {
     return;
   }
   this.People_Birth = People_Birth;
   onChange_People_Birth();
   onChange();
 }

 void onChange_People_Birth() {
               int index;
   aM.set_People_Birth( People_Birth );
   sD.set_People_Birth( People_Birth );

 }
 public double  Avg_Time_as_Dissident;

 /**
  *        Returns        default        value        for        parameter
<code>Avg_Time_as_Dissident</code>.
  * <i>This method should not be called by user</i>
  */
 public
double  _Avg_Time_as_Dissident_DefaultValue_xjal() {
   return 0.0;
 }

 public void set_Avg_Time_as_Dissident(
double  Avg_Time_as_Dissident ) {
   if (Avg_Time_as_Dissident == this.Avg_Time_as_Dissident) {
     return;
   }
   this.Avg_Time_as_Dissident = Avg_Time_as_Dissident;
   onChange_Avg_Time_as_Dissident();
   onChange();
 }

 void onChange_Avg_Time_as_Dissident() {
 }


 public double  Desired_Time_to_Remove_Insurgents;

 /**
  *        Returns        default        value        for        parameter
<code>Desired_Time_to_Remove_Insurgents</code>.
  * <i>This method should not be called by user</i>
  */
 public
double  _Desired_Time_to_Remove_Insurgents_DefaultValue_xjal() {
   return 0.0;
 }

 public void set_Desired_Time_to_Remove_Insurgents(
double  Desired_Time_to_Remove_Insurgents ) {
   if                (Desired_Time_to_Remove_Insurgents                ==
this.Desired_Time_to_Remove_Insurgents) {
     return;
   }
   this.Desired_Time_to_Remove_Insurgents                                 =
Desired_Time_to_Remove_Insurgents;
   onChange_Desired_Time_to_Remove_Insurgents();
   onChange();
 }

 void onChange_Desired_Time_to_Remove_Insurgents() {
 }

 // Events

 public      EventTimeout      _plot3_autoUpdateEvent_xjal      =      new
EventTimeout(this);
```

```
 public EventTimeout _plot4_autoUpdateEvent_xjal = new EventTimeout(this);
 public EventTimeout _plot5_autoUpdateEvent_xjal = new EventTimeout(this);


 @Override
 public String getNameOf( EventTimeout _e ) {
  if ( _e == _plot3_autoUpdateEvent_xjal ) return "plot3 auto update event";
  if ( _e == _plot4_autoUpdateEvent_xjal ) return "plot4 auto update event";
  if ( _e == _plot5_autoUpdateEvent_xjal ) return "plot5 auto update event";
   return super.getNameOf( _e );
 }

 @Override
 public int getModeOf( EventTimeout _e ) {
  if ( _e == _plot3_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
  if ( _e == _plot4_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
  if ( _e == _plot5_autoUpdateEvent_xjal ) return EVENT_TIMEOUT_MODE_CYCLIC;
  return super.getModeOf( _e );
 }

 @Override
 public double getFirstOccurrenceTime( EventTimeout _e ) {
  if (
     _e == _plot3_autoUpdateEvent_xjal
     || _e == _plot4_autoUpdateEvent_xjal
     || _e == _plot5_autoUpdateEvent_xjal
   ) return getEngine().getStartTime();
  return super.getFirstOccurrenceTime( _e );
 }

 @Override
 public double evaluateTimeoutOf( EventTimeout _e ) {
  if ( _e == _plot3_autoUpdateEvent_xjal ) return 1;
  if ( _e == _plot4_autoUpdateEvent_xjal ) return 1;
  if ( _e == _plot5_autoUpdateEvent_xjal ) return 1;
  return super.evaluateTimeoutOf( _e );
 }

 @Override
 public void executeActionOf( EventTimeout _e ) {
  if ( _e == _plot3_autoUpdateEvent_xjal ) {
   plot3.updateData();
   return;
  }
  if ( _e == _plot4_autoUpdateEvent_xjal ) {
   plot4.updateData();
   return;
  }
  if ( _e == _plot5_autoUpdateEvent_xjal ) {
   plot5.updateData();
   return;
  }
  super.executeActionOf( _e );
 }
 // Embedded Objects

 public ABModel aM;
 public SDModel sD;

 public String getNameOf( ActiveObject ao ) {
  if ( ao == aM ) return "aM";
  if ( ao == sD ) return "sD";
  return null;
 }

 public String getNameOf( ActiveObjectCollection<?> aolist ) {
  return null;
 }

 /**
  * Creates an embedded object instance<br>
  * <i>This method should not be called by user</i>
  */
 private ABModel instantiate_aM_xjal() {ABModel object = new ABModel( getEngine(), this, null );

  return object;
 }

 /**
  * Setups parameters of an embedded object instance<br>
  * This method should not be called by user
  */
 private void setupParameters_aM_xjal(ABModel object ) {
  object.People_Birth =
People_Birth
;
  object.Contact_Rate =
Contact_Rate
;
  object.People_Reaction_Delay_On_Anti_Regime_Messages =
People_Reaction_Delay_On_Anti_Regime_Messages
;
  object.Regime_Social_and_Economic_Efforts =
Regime_Social_and_Economic_Efforts
;
  object.Avg_Time_as_Dissident = object._Avg_Time_as_Dissident_DefaultValue_xjal();
  object.Appeasement_Fraction = object._Appeasement_Fraction_DefaultValue_xjal();
  object.Desired_Time_to_Remove_Insurgents = object._Desired_Time_to_Remove_Insurgents_DefaultValue_xjal();
  object.Removal_Effectiveness = object._Removal_Effectiveness_DefaultValue_xjal();
 }

 /**
  * Setups an embedded object instance<br>
  * This method should not be called by user
  */
 private void create_aM_xjal(ABModel object ) {
  object.create();

 }
 /**
  * Creates an embedded object instance<br>
  * <i>This method should not be called by user</i>
  */
 private SDModel instantiate_sD_xjal() {SDModel object = new SDModel( getEngine(), this, null );

  return object;
 }

 /**
  * Setups parameters of an embedded object instance<br>
  * This method should not be called by user
```

```
   */
  private void setupParameters_sD_xjal(SDModel object ) {
    object.People_Birth =
People_Birth
;
    object.Contact_Rate =
Contact_Rate
;
    object.People_Reaction_Delay_On_Anti_Regime_Messages =
People_Reaction_Delay_On_Anti_Regime_Messages
;
    object.Regime_Social_and_Economic_Efforts =
Regime_Social_and_Economic_Efforts
;
    object.Avg_Time_as_Dissident                                    =
object._Avg_Time_as_Dissident_DefaultValue_xjal();
    object.Appeasement_Fraction                                     =
object._Appeasement_Fraction_DefaultValue_xjal();
    object.Desired_Time_to_Remove_Insurgents                        =
object._Desired_Time_to_Remove_Insurgents_DefaultValue_xjal();
    object.Removal_Effectiveness                                    =
object._Removal_Effectiveness_DefaultValue_xjal();
  }


  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_sD_xjal(SDModel object ) {
    object.create();


  }


  public DataSet _plot3_expression0_dataSet_xjal = new DataSet( 100 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
100 * (sD.Government_Supporters - aM.Government_Supporters) /
(Initial_Government_Supporters + Initial_Dissident + Initial_Insurgent)
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  public DataSet _plot3_expression1_dataSet_xjal = new DataSet( 100 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
100 * (sD.Dissidents - aM.Dissidents) / (Initial_Government_Supporters
+ Initial_Dissident + Initial_Insurgent)
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  public DataSet _plot3_expression2_dataSet_xjal = new DataSet( 100 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
100 * (sD.Insurgents - aM.Insurgents) / (Initial_Government_Supporters
+ Initial_Dissident + Initial_Insurgent)
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  public DataSet _plot3_expression3_dataSet_xjal = new DataSet( 100 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
100 * (sD.Removed_Insurgents - aM.Removed_Insurgents) /
(Initial_Government_Supporters + Initial_Dissident + Initial_Insurgent)
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  public DataSet _plot4_expression0_dataSet_xjal = new DataSet( 100 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
aM.Government_Supporters
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  public DataSet _plot4_expression1_dataSet_xjal = new DataSet( 100 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
aM.Dissidents
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  public DataSet _plot4_expression2_dataSet_xjal = new DataSet( 100 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
aM.Insurgents
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  public DataSet _plot4_expression3_dataSet_xjal = new DataSet( 100 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
```

```
aM.Removed_Insurgents
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  public DataSet _plot5_expression0_dataSet_xjal = new DataSet( 100 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
sD.Government_Supporters
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  public DataSet _plot5_expression1_dataSet_xjal = new DataSet( 100 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
sD.Dissidents
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  public DataSet _plot5_expression2_dataSet_xjal = new DataSet( 100 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
sD.Insurgents
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  public DataSet _plot5_expression3_dataSet_xjal = new DataSet( 100 )
{
    double _lastUpdateX = Double.NaN;
    @Override
    public void update() {
      if ( time() == _lastUpdateX ) { return; }
      double _a =
sD.Removed_Insurgents
;
      add( time(), _a );
      _lastUpdateX = time();
    }
  };
  // View areas
  public ViewArea MainView = new ViewArea( this, "MainView", 0, 0,
ViewArea.TOP_LEFT, ViewArea.NONE, 1.0, 100, 100 );
  public ViewArea DescriptionView = new ViewArea( this,
"DescriptionView", 0, -1500, ViewArea.TOP_LEFT, ViewArea.NONE, 1.0,
100, 100 );

  static final Font _text7_Font = new Font("SansSerif", 0, 11 );
  static final Font _text1_Font = new Font("SansSerif", 1, 12 );
  static final Font _text2_Font = _text1_Font;
  static final Font _text3_Font = _text1_Font;
  static final Font _text4_Font = _text7_Font;
  static final Font _text5_Font = _text1_Font;
  static final Font _text6_Font = new Font("SansSerif", 1, 11 );
  static final Font _text24_Font = new Font("SansSerif", 0, 18 );
  static final Font _text25_Font = _text24_Font;
  static final Font _text26_Font = _text24_Font;
  static final Font _text8_Font = new Font("SansSerif", 0, 32 );
  static final Font _text10_Font = _text1_Font;
  static final Font _text11_Font = _text7_Font;
  static final Font _text33_Font = _text24_Font;
  static final Font _text9_Font = _text1_Font;
  static final Font _text27_Font = _text24_Font;
  static final Font _text22_Font = _text24_Font;
  static final Font _text21_Font = _text24_Font;
  static final Font _text20_Font = _text24_Font;
  static final Font _text_Font = _text8_Font;
  static final int _text7 = 1;
  static final int _rectangle = 2;
  static final int text1 = 3;
  static final int text2 = 4;
  static final int text3 = 5;
  static final int _text4 = 6;
  static final int _text5 = 7;
  static final int line5 = 8;
  static final int line6 = 9;
  static final int line7 = 10;
  static final int _text6 = 11;
  static final int _text24 = 12;
  static final int _line61 = 13;
  static final int _line62 = 14;
  static final int _line57 = 15;
  static final int _rectangle37 = 16;
  static final int _rectangle38 = 17;
  static final int _text25 = 18;
  static final int _line63 = 19;
  static final int _text26 = 20;
  static final int _rectangle39 = 21;
  static final int _line64 = 22;
  static final int line2 = 23;
  static final int _text8 = 24;
  static final int _text10 = 25;
  static final int _text11 = 26;
  static final int _line76 = 27;
  static final int _text33 = 28;
  static final int _text9 = 29;
  static final int line8 = 30;
  static final int _line67 = 31;
  static final int _rectangle40 = 32;
  static final int _text27 = 33;
  static final int _line66 = 34;
  static final int _line60 = 35;
  static final int _line59 = 36;
  static final int _rectangle36 = 37;
  static final int _text22 = 38;
  static final int _line58 = 39;
  static final int _line56 = 40;
  static final int _text21 = 41;
  static final int _rectangle35 = 42;
  static final int _text20 = 43;
  static final int line1 = 44;
  static final int _text = 45;
  static final int _plot3 = 46;
  static final int _plot4 = 47;
```

```
 static final int _plot5 = 48;

 /**
  * Top-level presentation group id
  */
 static final int _presentation = 0;

 /**
  * Top-level icon group id
  */
 static final int _icon = -1;



 @Override
 public boolean onShapeClick( int _shape, int index, double clickx,
double clicky ){
   switch( _shape ){
     case _rectangle37:
       if (true) {

aM.ABView.navigateTo();
       }
       break;
     case _rectangle38:
       if (true) {

sD.SDView.navigateTo();
       }
       break;
     case _rectangle39:
       if (true) {

MainView.navigateTo();
       }
       break;
     case _rectangle40:
       if (true) {

DescriptionView.navigateTo();
       }
       break;
     case _rectangle36:
       if (true) {

aM.ABView.navigateTo();
       }
       break;
     case _rectangle35:
       if (true) {

sD.SDView.navigateTo();
       }
       break;
     default: return super.onShapeClick( _shape, index, clickx, clicky );
               }
              return false;
 }

 @Override
 public String getNameOfShape( int _shape ) {
  switch( _shape ) {
    case text1: return "text1";
    case text2: return "text2";
    case text3: return "text3";
    case line5: return "line5";
    case line6: return "line6";
    case line7: return "line7";
    case line2: return "line2";
    case line8: return "line8";
    case line1: return "line1";
    default: return super.getNameOfShape( _shape );
  }
 }

 @Override
 public int getShapeType( int _shape ) {
  switch( _shape ) {
    case text1: return SHAPE_TEXT;
    case text2: return SHAPE_TEXT;
    case text3: return SHAPE_TEXT;
    case line5: return SHAPE_LINE;
    case line6: return SHAPE_LINE;
    case line7: return SHAPE_LINE;
    case line2: return SHAPE_LINE;
    case line8: return SHAPE_LINE;
    case line1: return SHAPE_LINE;
    default: return super.getShapeType( _shape );
  }
 }

 @Override
 public double getShapeX( int _shape, int index ) {
  switch( _shape ) {
    case text1: return 520;
    case text2: return 50;
    case text3: return 520;
    case line5: return 770;
    case line6: return 580;
    case line7: return 110;
    case line2: return 30;
    case line8: return 250;
    case line1: return 30;
    default: return super.getShapeX( _shape, index );
  }
 }

 @Override
 public double getShapeY( int _shape, int index ) {
  switch( _shape ) {
    case text1: return 360;
    case text2: return 360;
    case text3: return 70;
    case line5: return 80;
    case line6: return 370;
    case line7: return 370;
    case line2: return -1450;
    case line8: return 160;
    case line1: return 50;
    default: return super.getShapeY( _shape, index );
  }
 }

 @Override
 public Color getShapeLineColor( int _shape, int index ) {
  switch( _shape ) {
    // shapes
    case line5: return gray;
    case line6: return slateGray;
    case line7: return slateGray;
    case line2: return slateGray;
    case line8: return slateGray;
```

```
    case line1: return slateGray;
    // controls (text color)
    // charts (border color)
    default: return super.getShapeLineColor( _shape, index );
  }
}

@Override
public Color getShapeFillColor( int _shape, int index ) {
  switch( _shape ) {
    case text1: return black;
    case text2: return black;
    case text3: return black;
    default: return super.getShapeFillColor( _shape, index );
  }
}

@Override
public int getShapeLineStyle( int _shape, int index ) {
  switch( _shape ) {
    case line5: return LINE_STYLE_SOLID;
    case line6: return LINE_STYLE_SOLID;
    case line7: return LINE_STYLE_SOLID;
    case line2: return LINE_STYLE_SOLID;
    case line8: return LINE_STYLE_SOLID;
    case line1: return LINE_STYLE_SOLID;
    default: return super.getShapeLineStyle( _shape, index );
  }
}

@Override
public double getShapeLineDx( int _shape, int index ) {
  switch( _shape ) {
    case line5: return 140;
    case line6: return 330;
    case line7: return 340;
    case line2: return 970;
    case line8: return 130;
    case line1: return 970;
    default: return super.getShapeLineDx( _shape, index );
  }
}

@Override
public double getShapeLineDy( int _shape, int index ) {
  switch( _shape ) {
    case line5: return 0;
    case line6: return 0;
    case line7: return 0;
    case line2: return 0;
    case line8: return 0;
    case line1: return 0;
    default: return super.getShapeLineDy( _shape, index );
  }
}

@Override
public Object getShapeText( int _shape, int index ) {
  switch( _shape ) {
    case text1: return "AB model";
    case text2: return "SD model";
    case text3: return "Differences between SD and AB models, %";
    default: return super.getShapeText( _shape, index );
  }
}

@Override
public Font getShapeFont( int _shape, int index ) {
  switch( _shape ) {
    case text1: return _text1_Font;
    case text2: return _text2_Font;
    case text3: return _text3_Font;
    default: return super.getShapeFont( _shape, index );
  }
}
@Override
public int getShapeTextAlignment( int _shape, int index ) {
  switch( _shape ) {
    case text1: return ALIGNMENT_LEFT;
    case text2: return ALIGNMENT_LEFT;
    case text3: return ALIGNMENT_LEFT;
    default: return super.getShapeTextAlignment( _shape, index );
  }
}

TimePlot plot3;
TimePlot plot4;
TimePlot plot5;
ShapeText text7;
ShapeRectangle rectangle;
ShapeText text4;
ShapeText text5;

/**
 * <i>This method should not be called by user</i>
 */
private void _text6_SetDynamicParams( ShapeText shape ) {
  boolean _visible;
  shape.setText(
Initial_Government_Supporters + "\n" +
Initial_Dissident + "\n" +
Initial_Insurgent + "\n" +
People_Reaction_Delay_On_Anti_Regime_Messages + "\n" +
Regime_Social_and_Economic_Efforts + "\n" +
Contact_Rate + "\n" +
People_Birth + "\n" +
Avg_Time_as_Dissident + "\n" +
Desired_Time_to_Remove_Insurgents

);
}

ShapeText text6;
ShapeText text24;
ShapeLine line61;
ShapeLine line62;
ShapeLine line57;
ShapeRectangle rectangle37;
ShapeRectangle rectangle38;
ShapeText text25;
ShapeLine line63;
ShapeText text26;
ShapeRectangle rectangle39;
ShapeLine line64;
ShapeText text8;
ShapeText text10;
ShapeText text11;
ShapeLine line76;
ShapeText text33;
ShapeText text9;
```

```
  ShapeLine line67;
  ShapeRectangle rectangle40;
  ShapeText text27;
  ShapeLine line66;
  ShapeLine line60;
  ShapeLine line59;
  ShapeRectangle rectangle36;
  ShapeText text22;
  ShapeLine line58;
  ShapeLine line56;
  ShapeText text21;
  ShapeRectangle rectangle35;
  ShapeText text20;
  ShapeText text;


  // Static initialization of persistent elements
  {
    text7 = new ShapeText(
        true,90, 170, 0.0,
        black,"Initial government supporters.....\r\nInitial dissidents.....\r\nInitial insurgents.....\r\nPeople reaction delay on anti regime messages.....\r\nRegime social and economic efforts.....\r\nContact rate.....\r\nPeople birth.....\r\nAverage time as dissident.....\r\nDesired time to remove insurgents.....\r\n",
        _text7_Font, ALIGNMENT_LEFT
    );
    rectangle = new ShapeRectangle(
      true,349, 161, 0.0,
          null, white,
                                        110, 130,
                                        1, LINE_STYLE_SOLID
    );
    text4 = new ShapeText(
        true,35, -1350, 0.0,
        black,"In the System Dynamics model all population groups are stored in four stocks: \r\nGovernment_Supporters, Dissidents, Insurgents and Removed_Insurgents.\r\nIncoming flow of births is a variable 'Birth' which contains a \r\ndynamically changing number of people entering Government_Supporters group per time unit. Flows \r\nbetween groups are Becoming_Dissident, Insurgent_Recruitment and Removing_Insurgents variables. A \r\nprocess of relaxation is Appeasement_Rate. Each of these variables contains an equation defining the\r\ncorresponding rate.\r\n\r\nMessages generated by active insurgents are stored in a Perceived_Intencity_of_Anti_Regime_Messages stock. \r\nThe stock represents a buffer between the insurgents and a social network.\r\n\r\n\r\n",
        _text4_Font, ALIGNMENT_LEFT
    );
    text5 = new ShapeText(
        true,35, -1380, 0.0,
        black,"System Dynamics model description",
        _text5_Font, ALIGNMENT_LEFT
    );
    text6 = new ShapeText(
        true,350, 170, 0.0,
        navy,"Parameters values\r\n123\r\n123\r\n123\r\n...",
        _text6_Font, ALIGNMENT_LEFT
    ) {


      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform xform, boolean publicOnly ) {
        _text6_SetDynamicParams( this );
        super.draw( panel, graphics, xform, publicOnly );
      }
    };
text24 = new ShapeText(
    true,235, -1425, 0.0,
    royalBlue,"Comparison",
    _text24_Font, ALIGNMENT_LEFT
);
line61 = new ShapeLine(
                        true,136, -1405,
    royalBlue,
                        80,         0,
                        1, LINE_STYLE_SOLID
);
line62 = new ShapeLine(
                        true,125, -1430,
    black,
                        0,          30,
                        1, LINE_STYLE_SOLID
);
line57 = new ShapeLine(
                        true,225, -1430,
    black,
                        0,          30,
                        1, LINE_STYLE_SOLID
);
rectangle37 = new ShapeRectangle(
  true,30, -1430, 0.0,
      null, null,
                                    90, 30,
                                    1, LINE_STYLE_SOLID
) {

  @Override
  public boolean onClick( double clickx, double clicky ) {
    return onShapeClick( _rectangle37, 0, clickx, clicky );
  }
};
rectangle38 = new ShapeRectangle(
  true,130, -1430, 0.0,
      null, null,
                                    90, 30,
                                    1, LINE_STYLE_SOLID
) {

  @Override
  public boolean onClick( double clickx, double clicky ) {
    return onShapeClick( _rectangle38, 0, clickx, clicky );
  }
};
text25 = new ShapeText(
    true,35, -1425, 0.0,
    royalBlue,"AB model",
    _text25_Font, ALIGNMENT_LEFT
);
line63 = new ShapeLine(
                        true,36, -1405,
    royalBlue,
                        80,         0,
                        1, LINE_STYLE_SOLID
);
text26 = new ShapeText(
    true,135, -1425, 0.0,
    royalBlue,"SD model",
    _text26_Font, ALIGNMENT_LEFT
);
rectangle39 = new ShapeRectangle(
  true,230, -1430, 0.0,
```

```
        null, null,
                                    110, 30,
                                    1, LINE_STYLE_SOLID
  ) {

    @Override
    public boolean onClick( double clickx, double clicky ) {
      return onShapeClick( _rectangle39, 0, clickx, clicky );
    }
  };
  line64 = new ShapeLine(
                        true,235, -1405,
     royalBlue,
                        100,        0,
                        1, LINE_STYLE_SOLID
  );
  text8 = new ShapeText(
     true,30, -1490, 0.0,
     slateGray,"SD+AB Insurgency Models",
     _text8_Font, ALIGNMENT_LEFT
  );
  text10 = new ShapeText(
     true,35, -1200, 0.0,
     black,"AB insurgency model description",
     _text10_Font, ALIGNMENT_LEFT
  );
  text11 = new ShapeText(
     true,35, -1170, 0.0,
     black,"In the Agent Based model people are modeled explicitly as agents (one agent per person).\r\nEach agent has four alternative states: Government_Supporter, Dissident, Insurgent and Removed_Insurgent. \r\nAgents communicate with each other by sending and receiving messages. E.g. insurgents and dissidents \r\nsend messages to government supporting citizens trying to recruit them. Each agent in Active_Dissident or \r\nin Active_Insurgent state sends messages with Contact_Rate. When an agent in \r\nGovernment_Supporter state receives this message he transitions to Dissident state with a probability of \r\nPropensity_to_be_Recruited. In other states the message is ignored.\r\nRelaxation process is a rate based transition from Dissident to Government_Supporter state. The rate is \r\ncalculated as a function of a probability of a dissident tend not to support opposition/being \r\nappeased (Appeasement_Fraction) and of an average time as dissident (Avg_Time_as_Dissident).\r\nDissidents become insurgents using a rate based transition. The rate is a function of an average time \r\nas dissident (Avg_Time_as_Dissident).\r\n\r\nWhen a government removes an agent from insurgents the agent is removed from the model.\r\n",
     _text11_Font, ALIGNMENT_LEFT
  );
  line76 = new ShapeLine(
                        false,345, -1430,
     black,
                        0,          30,
                        1, LINE_STYLE_SOLID
  );
  text33 = new ShapeText(
     true,354, -1425, 0.0,
     black,"Description",
     _text33_Font, ALIGNMENT_LEFT
  );
  text9 = new ShapeText(
     true,90, 150, 0.0,
     black,"Current model parameters",
     _text9_Font, ALIGNMENT_LEFT
  );
  line67 = new ShapeLine(
                        true,355, 95,
     royalBlue,
                        94,         0,
                        1, LINE_STYLE_SOLID
  );
  rectangle40 = new ShapeRectangle(
    true,350, 70, 0.0,
        null, null,
                                    105, 30,
                                    1, LINE_STYLE_SOLID
  ) {

    @Override
    public boolean onClick( double clickx, double clicky ) {
      return onShapeClick( _rectangle40, 0, clickx, clicky );
    }
  };
  text27 = new ShapeText(
     true,354, 75, 0.0,
     royalBlue,"Description",
     _text27_Font, ALIGNMENT_LEFT
  );
  line66 = new ShapeLine(
                        false,345, 70,
     black,
                        0,          30,
                        1, LINE_STYLE_SOLID
  );
  line60 = new ShapeLine(
                        true,36, 95,
     royalBlue,
                        80,         0,
                        1, LINE_STYLE_SOLID
  );
  line59 = new ShapeLine(
                        true,136, 95,
     royalBlue,
                        80,         0,
                        1, LINE_STYLE_SOLID
  );
  rectangle36 = new ShapeRectangle(
    true,30, 70, 0.0,
        null, null,
                                    90, 30,
                                    1, LINE_STYLE_SOLID
  ) {

    @Override
    public boolean onClick( double clickx, double clicky ) {
      return onShapeClick( _rectangle36, 0, clickx, clicky );
    }
  };
  text22 = new ShapeText(
     true,135, 75, 0.0,
     royalBlue,"SD model",
     _text22_Font, ALIGNMENT_LEFT
  );
  line58 = new ShapeLine(
                        true,125, 70,
     black,
                        0,          30,
                        1, LINE_STYLE_SOLID
  );
  line56 = new ShapeLine(
                        true,225, 70,
     black,
                        0,          30,
                        1, LINE_STYLE_SOLID
  );
  text21 = new ShapeText(
```

```
    true,35, 75, 0.0,
    royalBlue,"AB model",
    _text21_Font, ALIGNMENT_LEFT
  );
  rectangle35 = new ShapeRectangle(
    true,130, 70, 0.0,
      null, null,
                                    90, 30,
                                    1, LINE_STYLE_SOLID
  ) {

   @Override
   public boolean onClick( double clickx, double clicky ) {
    return onShapeClick( _rectangle35, 0, clickx, clicky );
   }
  };
  text20 = new ShapeText(
    true,235, 75, 0.0,
    black,"Comparison",
    _text20_Font, ALIGNMENT_LEFT
  );
  text = new ShapeText(
    true,30, 10, 0.0,
    slateGray,"SD+AB Insurgency Models",
    _text_Font, ALIGNMENT_LEFT
  );
}
ShapeGroup presentation;
ShapeGroup icon;

@Override
public Object getPersistentShape( int _shape ) {
 switch(_shape){
  case _presentation: return presentation;
  case _icon: return icon;
  case _plot3: return plot3;
  case _plot4: return plot4;
  case _plot5: return plot5;
  case _text7: return text7;
  case _rectangle: return rectangle;
  case _text4: return text4;
  case _text5: return text5;
  case _text6: return text6;
  case _text24: return text24;
  case _line61: return line61;
  case _line62: return line62;
  case _line57: return line57;
  case _rectangle37: return rectangle37;
  case _rectangle38: return rectangle38;
  case _text25: return text25;
  case _line63: return line63;
  case _text26: return text26;
  case _rectangle39: return rectangle39;
  case _line64: return line64;
  case _text8: return text8;
  case _text10: return text10;
  case _text11: return text11;
  case _line76: return line76;
  case _text33: return text33;
  case _text9: return text9;
  case _line67: return line67;
  case _rectangle40: return rectangle40;
  case _text27: return text27;
  case _line66: return line66;
  case _line60: return line60;
  case _line59: return line59;
  case _rectangle36: return rectangle36;
  case _text22: return text22;
  case _line58: return line58;
  case _line56: return line56;
  case _text21: return text21;
  case _rectangle35: return rectangle35;
  case _text20: return text20;
  case _text: return text;
  default: return null;
 }
}


@Override
public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
 if (!_publicOnly) {
  drawParameter( _panel, _g, -300, 130, 10, 0, "Initial_Dissident",
Initial_Dissident, false, false );
 }
 if (!_publicOnly) {
  drawParameter( _panel, _g, -300, 110, 10, 0,
"Initial_Government_Supporters", Initial_Government_Supporters, false,
false );
 }
 if (!_publicOnly) {
  drawParameter( _panel, _g, -300, 150, 10, 0, "Initial_Insurgent",
Initial_Insurgent, false, false );
 }
 if (!_publicOnly) {
  drawParameter( _panel, _g, -300, 180, 10, 0,
"People_Reaction_Delay_On_Anti_Regime_Messages",
People_Reaction_Delay_On_Anti_Regime_Messages, false, false );
 }
 if (!_publicOnly) {
  drawParameter( _panel, _g, -300, 200, 10, 0,
"Regime_Social_and_Economic_Efforts",
Regime_Social_and_Economic_Efforts, false, false );
 }
 if (!_publicOnly) {
  drawParameter( _panel, _g, -300, 220, 10, 0, "Contact_Rate",
Contact_Rate, false, false );
 }
 if (!_publicOnly) {
  drawParameter( _panel, _g, -300, 240, 10, 0, "People_Birth",
People_Birth, false, false );
 }
 if (!_publicOnly) {
  drawParameter( _panel, _g, -300, 260, 10, 0,
"Avg_Time_as_Dissident", Avg_Time_as_Dissident, false, false );
 }
 if (!_publicOnly) {
  drawParameter( _panel, _g, -300, 280, 10, 0,
"Desired_Time_to_Remove_Insurgents",
Desired_Time_to_Remove_Insurgents, false, false );
 }
 // Embedded object "aM"
 if (!_publicOnly) {
        drawEmbeddedObjectModel( _panel, _g, 2147483356 , -
2147483570 , -2147483646, 2147483640, "aM", this.aM );
 }
 // Embedded object "sD"
 if (!_publicOnly) {
        drawEmbeddedObjectModel( _panel, _g, 2147483356 , -
2147483590 , -2147483646, 2147483640, "sD", this.sD );
 }
}

@Override
```

```
 public boolean onClickModelAt( Panel panel, double x, double y, int
clickCount, boolean publicOnly ) {
  if( !publicOnly && modelElementContains(x, y, -300, 130) ) {
   panel.addInspect( -300, 130, this, "Initial_Dissident" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -300, 110) ) {
   panel.addInspect( -300, 110, this, "Initial_Government_Supporters"
);
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -300, 150) ) {
   panel.addInspect( -300, 150, this, "Initial_Insurgent" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -300, 180) ) {
   panel.addInspect( -300, 180, this,
"People_Reaction_Delay_On_Anti_Regime_Messages" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -300, 200) ) {
   panel.addInspect( -300, 200, this,
"Regime_Social_and_Economic_Efforts" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -300, 220) ) {
   panel.addInspect( -300, 220, this, "Contact_Rate" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -300, 240) ) {
   panel.addInspect( -300, 240, this, "People_Birth" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -300, 260) ) {
   panel.addInspect( -300, 260, this, "Avg_Time_as_Dissident" );
   return true;
  }
  if( !publicOnly && modelElementContains(x, y, -300, 280) ) {
   panel.addInspect( -300, 280, this,
"Desired_Time_to_Remove_Insurgents" );
   return true;
  }
  if ( aM.onClickIconAt( x - 2147483356, y - -2147483570, true ) ) {
   if ( clickCount == 2 ) {
    panel.browseEmbeddedObject( x, y, this, "aM" );
   } else {
    panel.addInspect( x, y, this, "aM" );
   }
   return true;
  }
  if ( sD.onClickIconAt( x - 2147483356, y - -2147483590, true ) ) {
   if ( clickCount == 2 ) {
    panel.browseEmbeddedObject( x, y, this, "sD" );
   } else {
    panel.addInspect( x, y, this, "sD" );
   }
   return true;
  }
  return false;
 }

 /**
  * Constructor
  */
 public Main( Engine engine, ActiveObject owner,
ActiveObjectCollection<? extends Main> collection ) {
  super( engine, owner, collection );
 }

 @Override
 public void create() {
  // Creating embedded object instances
  aM = instantiate_aM_xjal();
  sD = instantiate_sD_xjal();
  // Dynamic initialization of persistent elements
  {
  DataSet _item;
  List<DataSet> _items = new ArrayList<DataSet>( 4 );
  _items.add( _plot3_expression0_dataSet_xjal );
  _items.add( _plot3_expression1_dataSet_xjal );
  _items.add( _plot3_expression2_dataSet_xjal );
  _items.add( _plot3_expression3_dataSet_xjal );
  List<String> _titles = new ArrayList<String>( 4 );
  _titles.add( "Government supporters" );
  _titles.add( "Dissidents" );
  _titles.add( "Insurgents" );
  _titles.add( "Removed insurgents" );
  List<Chart2DPlot.Appearance> _appearances = new
ArrayList<Chart2DPlot.Appearance>( 4 );
  _appearances.add( new Chart2DPlot.Appearance( limeGreen, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
  _appearances.add( new Chart2DPlot.Appearance( gold, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
  _appearances.add( new Chart2DPlot.Appearance( orangeRed, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
  _appearances.add( new Chart2DPlot.Appearance( yellowGreen, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
  plot3 = new TimePlot(
                Main.this, true, 490, 80,
                440, 280,
      null, null,
      60, 20,
                360, 190, white, black, black,
      50, Chart.SOUTH,
100
      , null, Chart.SCALE_AUTO,
      0, 0, Chart.GRID_DEFAULT, Chart.GRID_DEFAULT,
      darkGray, darkGray, _items, _titles, _appearances
  );
  }
  {
  DataSet _item;
  List<DataSet> _items = new ArrayList<DataSet>( 4 );
  _items.add( _plot4_expression0_dataSet_xjal );
  _items.add( _plot4_expression1_dataSet_xjal );
  _items.add( _plot4_expression2_dataSet_xjal );
  _items.add( _plot4_expression3_dataSet_xjal );
  List<String> _titles = new ArrayList<String>( 4 );
  _titles.add( "Government supporters" );
  _titles.add( "Dissidents" );
  _titles.add( "Insurgents" );
  _titles.add( "Removed insurgents" );
  List<Chart2DPlot.Appearance> _appearances = new
ArrayList<Chart2DPlot.Appearance>( 4 );
  _appearances.add( new Chart2DPlot.Appearance( limeGreen, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
  _appearances.add( new Chart2DPlot.Appearance( gold, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
  _appearances.add( new Chart2DPlot.Appearance( orangeRed, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
  _appearances.add( new Chart2DPlot.Appearance( yellowGreen, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
  plot4 = new TimePlot(
                Main.this, true, 490, 370,
```

```
                                    440, 280,
        null, null,
        60, 20,
                                    360, 190, white, black, black,
        50, Chart.SOUTH,100
        , null, Chart.SCALE_AUTO,
        0, 0, Chart.GRID_DEFAULT, Chart.GRID_DEFAULT,
        darkGray, darkGray, _items, _titles, _appearances
    );
    }
    {
    DataSet _item;
    List<DataSet> _items = new ArrayList<DataSet>( 4 );
    _items.add( _plot5_expression0_dataSet_xjal );
    _items.add( _plot5_expression1_dataSet_xjal );
    _items.add( _plot5_expression2_dataSet_xjal );
    _items.add( _plot5_expression3_dataSet_xjal );
    List<String> _titles = new ArrayList<String>( 4 );
    _titles.add( "Government supporters" );
    _titles.add( "Dissidents" );
    _titles.add( "Insurgents" );
    _titles.add( "Removed insurgents" );
    List<Chart2DPlot.Appearance> _appearances = new
ArrayList<Chart2DPlot.Appearance>( 4 );
    _appearances.add( new Chart2DPlot.Appearance( limeGreen, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
    _appearances.add( new Chart2DPlot.Appearance( gold, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
    _appearances.add( new Chart2DPlot.Appearance( orangeRed, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
    _appearances.add( new Chart2DPlot.Appearance( yellowGreen, true,
Chart.INTERPOLATION_LINEAR, 1, Chart.POINT_NONE ) );
    plot5 = new TimePlot(
                            Main.this, true, 30, 370,
                                    440, 280,
        null, null,
        60, 20,
                                    360, 190, white, black, black,
        50, Chart.SOUTH,


100
        , null, Chart.SCALE_AUTO,
        0, 0, Chart.GRID_DEFAULT, Chart.GRID_DEFAULT,
        darkGray, darkGray, _items, _titles, _appearances
    );
    }
    presentation = new ShapeGroup( Main.this, true, 0, 0, 0, text7,
rectangle, text1, text2, text3, text4, text5, line5, line6, line7, text6,
text24, line61, line62, line57, rectangle37, rectangle38, text25, line63,
text26, rectangle39, line64, line2, text8, text10, text11, text33, text9,
line8, line67, rectangle40, text27, line60, line59, rectangle36, text22,
line58, line56, text21, rectangle35, text20, line1, text, plot3, plot4, plot5
);
    icon = new ShapeGroup( Main.this, true, 0, 0, 0
      ,line76
      ,line66                                );
    // Creating non-replicated embedded objects
    setupParameters_aM_xjal( aM );
    create_aM_xjal( aM );
    setupParameters_sD_xjal( sD );
    create_sD_xjal( sD );
                // Port connectors with non-replicated objects
    // Creating replicated embedded objects
    assignInitialConditions();
    onCreate();
  }

  @Override
  public void start() {
    _plot3_autoUpdateEvent_xjal.start();
    _plot4_autoUpdateEvent_xjal.start();
    _plot5_autoUpdateEvent_xjal.start();
    aM.start();
    sD.start();
    onStartup();
  }

  public List<Object> getEmbeddedObjects() {
    LinkedList<Object> list = new LinkedList<Object>();
    list.add( aM );
    list.add( sD );
    return list;
  }

  public void onDestroy() {
    super.onDestroy();
    _plot3_autoUpdateEvent_xjal.onDestroy();
    _plot4_autoUpdateEvent_xjal.onDestroy();
    _plot5_autoUpdateEvent_xjal.onDestroy();
    aM.onDestroy();
    sD.onDestroy();
  }


}
```

Simulation.java

```
package insurgencySDABModel;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
```

```
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import javax.swing.JApplet;
public class Simulation extends ExperimentSimulation<Main> {
 // Plain Variables
 public int  Initial_Government_Supporters = 2000 ;
 public int  Initial_Dissident = 450 ;
 public int  Initial_Insurgent = 250 ;
 public int People_Reaction_Delay_On_Anti_Regime_Messages = 10 ;
 public double  Regime_Social_and_Economic_Efforts = 0.12 ;
 public double  Contact_Rate = 20 ;
 public int People_Birth = 1 ;
 public double  Avg_Time_as_Dissident = 15.0 ;
 public double  Desired_Time_to_Remove_Insurgents = 15.0 ;


 @Override
 public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
   if (!_publicOnly) {
    drawPlainVariable( _panel, _g, -300, 150,
      10, 0,
     "Initial_Government_Supporters",   Initial_Government_Supporters,
false );
   }
   if (!_publicOnly) {
    drawPlainVariable( _panel, _g, -300, 180,
      10, 0,
     "Initial_Dissident", Initial_Dissident, false );
   }
   if (!_publicOnly) {
    drawPlainVariable( _panel, _g, -300, 210,
      10, 0,
     "Initial_Insurgent", Initial_Insurgent, false );
   }
   if (!_publicOnly) {
    drawPlainVariable( _panel, _g, -300, 300,
      10, 0,
     "People_Reaction_Delay_On_Anti_Regime_Messages",
People_Reaction_Delay_On_Anti_Regime_Messages, false );
   }
   if (!_publicOnly) {
    drawPlainVariable( _panel, _g, -300, 330,
      10, 0,
     "Regime_Social_and_Economic_Efforts",
Regime_Social_and_Economic_Efforts, false );
   }
   if (!_publicOnly) {
    drawPlainVariable( _panel, _g, -300, 360,
      10, 0,
     "Contact_Rate", Contact_Rate, false );
   }
   if (!_publicOnly) {
    drawPlainVariable( _panel, _g, -300, 390,
      10, 0,
     "People_Birth", People_Birth, false );
   }
   if (!_publicOnly) {
    drawPlainVariable( _panel, _g, -300, 420,
      10, 0,
     "Avg_Time_as_Dissident", Avg_Time_as_Dissident, false );
   }
   if (!_publicOnly) {
    drawPlainVariable( _panel, _g, -300, 450,
      10, 0,
     "Desired_Time_to_Remove_Insurgents",
Desired_Time_to_Remove_Insurgents, false );
   }
 }
 @Override
 public boolean onClickModelAt( Panel panel, double x, double y, int
clickCount, boolean publicOnly ) {
   if( !publicOnly && modelElementContains(x, y, -300, 150) ){
    panel.addInspect( -300, 150, this, "Initial_Government_Supporters"
);
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, -300, 180) ){
    panel.addInspect( -300, 180, this, "Initial_Dissident" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, -300, 210) ){
    panel.addInspect( -300, 210, this, "Initial_Insurgent" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, -300, 300) ){
    panel.addInspect( -300, 300, this,
"People_Reaction_Delay_On_Anti_Regime_Messages" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, -300, 330) ){
    panel.addInspect( -300, 330, this,
"Regime_Social_and_Economic_Efforts" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, -300, 360) ){
    panel.addInspect( -300, 360, this, "Contact_Rate" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, -300, 390) ){
    panel.addInspect( -300, 390, this, "People_Birth" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, -300, 420) ){
    panel.addInspect( -300, 420, this, "Avg_Time_as_Dissident" );
    return true;
   }
   if( !publicOnly && modelElementContains(x, y, -300, 450) ){
    panel.addInspect( -300, 450, this,
"Desired_Time_to_Remove_Insurgents" );
    return true;
   }
   return false;
 }
 static final Font _button_Font = new Font("Dialog", 0, 11 );
 static final Font _text_Font = new Font("SansSerif", 0, 32 );
 static final Font _text4_Font = new Font("SansSerif", 0, 12 );
 static final Font _text3_Font = _text4_Font;
 static final Font _text7_Font = new Font("SansSerif", 1, 12 );
 static final Font _text6_Font = _text7_Font;
 static final Font _text2_Font = _text4_Font;
 static final Font _text8_Font = _text7_Font;
 static final Font _text9_Font = _text7_Font;
 static final Font _text5_Font = _text4_Font;
 static final Font _text10_Font = _text7_Font;
 static final Font _text11_Font = _text4_Font;
 static final Font _text12_Font = _text7_Font;
```

```
static final Font _text13_Font = _text7_Font;
static final Font _text14_Font = _text7_Font;
static final Font _text15_Font = _text4_Font;
static final Font _text16_Font = _text4_Font;
static final Font _text17_Font = _text4_Font;
static final Font _text18_Font = _text4_Font;
static final Font _text19_Font = _text7_Font;
static final Font _text20_Font = _text7_Font;
static final Font _text21_Font = new Font("SansSerif", 0, 11 );
static final Font _text22_Font = _text7_Font;
static final Font _text1_Font = _text21_Font;
static final Font _text23_Font = new Font("SansSerif", 0, 9 );
static final Color _text23_Color = new Color( 0xFF2C5373, true );
static final int _button = 1;
static final int slider1 = 2;
static final int slider0 = 3;
static final int slider2 = 4;
static final int slider3 = 5;
static final int slider4 = 6;
static final int slider5 = 7;
static final int slider6 = 8;
static final int slider7 = 9;
static final int slider8 = 10;
static final int _text = 11;
static final int _text4 = 12;
static final int _text3 = 13;
static final int text7 = 14;
static final int text6 = 15;
static final int _text2 = 16;
static final int text8 = 17;
static final int text9 = 18;
static final int _text5 = 19;
static final int text10 = 20;
static final int _text11 = 21;
static final int line1 = 22;
static final int text12 = 23;
static final int text13 = 24;
static final int text14 = 25;
static final int _text15 = 26;
static final int _text16 = 27;
static final int line5 = 28;
static final int _text17 = 29;
static final int _text18 = 30;
static final int text19 = 31;
static final int text20 = 32;
static final int _text21 = 33;
static final int _text22 = 34;
static final int _text1 = 35;
static final int _image = 36;
static final int _text23 = 37;
static final int _polyline = 38;

/**
 * Top-level presentation group id
 */
static final int _presentation = 0;

/**
 * Top-level icon group id
 */
static final int _icon = -1;

static final double[] _polyline_pointsDX_xjal() {
  return new double[] { 0, 0, -10,  };
}

static final double[] _polyline_pointsDY_xjal() {
  return new double[] { 0, 20, 10,  };
}

@Override
public void executeShapeControlAction( int _shape, int index ) {
  switch( _shape ) {
    case _button: {
run();
getEngine().getPresentation().setPresentable(
((Main)getEngine().getRoot()).aM );
;}
      break;
    default:
      super.executeShapeControlAction( _shape, index );
      break;
  }
}

@Override
public void executeShapeControlAction( int _shape, int index, double
value ) {
  switch( _shape ) {
    case slider1:
      Initial_Dissident = (int) value;
      break;
    case slider0:
      Initial_Government_Supporters = (int) value;
      break;
    case slider2:
      Initial_Insurgent = (int) value;
      break;
    case slider3:
      Regime_Social_and_Economic_Efforts = value;
      break;
    case slider4:
      People_Reaction_Delay_On_Anti_Regime_Messages = (int) value;
      break;
    case slider5:
      Contact_Rate = value;
      break;
    case slider6:
      People_Birth = (int) value;
      break;
    case slider7:
      Desired_Time_to_Remove_Insurgents = value;
      break;
    case slider8:
      Avg_Time_as_Dissident = value;
      break;
    default:
      super.executeShapeControlAction( _shape, index, value );
      break;
  }
}

@Override
public int getShapeControlValueType( int _shape, int index ) {
  switch( _shape ) {
    case slider1: return ShapeControl.TYPE_INT;
    case slider0: return ShapeControl.TYPE_INT;
    case slider2: return ShapeControl.TYPE_INT;
    case slider3: return ShapeControl.TYPE_DOUBLE;
    case slider4: return ShapeControl.TYPE_INT;
    case slider5: return ShapeControl.TYPE_DOUBLE;
    case slider6: return ShapeControl.TYPE_INT;
```

```
    case slider7: return ShapeControl.TYPE_DOUBLE;
    case slider8: return ShapeControl.TYPE_DOUBLE;
    default: return super.getShapeControlValueType( _shape, index );
   }
  }

  @Override
  public double getShapeControlMinimum( int _shape, int index ) {
   switch( _shape ) {
    case slider1: return 300 ;
    case slider0: return 1190 ;
    case slider2: return 200 ;
    case slider3: return 0.001 ;
    case slider4: return 0.01 ;
    case slider5: return 5 ;
    case slider6: return 0 ;
    case slider7: return 10 ;
    case slider8: return 10 ;
    default: return super.getShapeControlMinimum( _shape, index );
   }
  }

  @Override
  public double getShapeControlMaximum( int _shape, int index ) {
   switch( _shape ) {
    case slider1: return 1000 ;
    case slider0: return 10000 ;
    case slider2: return 800 ;
    case slider3: return 1 ;
    case slider4: return 50 ;
    case slider5: return 50 ;
    case slider6: return 5 ;
    case slider7: return 30 ;
    case slider8: return 30 ;
    default: return super.getShapeControlMaximum( _shape, index );
   }
  }

  @Override
  public double getShapeControlDefaultValueDouble( int _shape, int index
) {
    switch(_shape) {
     case slider1: return Initial_Dissident ;
     case slider0: return Initial_Government_Supporters ;
     case slider2: return Initial_Insurgent ;
     case slider3: return Regime_Social_and_Economic_Efforts ;
     case slider4: return
People_Reaction_Delay_On_Anti_Regime_Messages ;
     case slider5: return Contact_Rate ;
     case slider6: return People_Birth ;
     case slider7: return Desired_Time_to_Remove_Insurgents ;
     case slider8: return Avg_Time_as_Dissident ;
     default:  return  super.getShapeControlDefaultValueDouble(  _shape,
index );
    }
   }


  @Override
  public String getNameOfShape( int _shape ) {
   switch( _shape ) {
    case text7: return "text7";
    case text6: return "text6";
    case text8: return "text8";
    case text9: return "text9";
    case text10: return "text10";
    case line1: return "line1";
    case text12: return "text12";
    case text13: return "text13";
    case text14: return "text14";
    case line5: return "line5";
    case text19: return "text19";
    case text20: return "text20";
    case slider1: return "slider1";
    case slider0: return "slider0";
    case slider2: return "slider2";
    case slider3: return "slider3";
    case slider4: return "slider4";
    case slider5: return "slider5";
    case slider6: return "slider6";
    case slider7: return "slider7";
    case slider8: return "slider8";
    default: return super.getNameOfShape( _shape );
   }
  }

  @Override
  public int getShapeType( int _shape ) {
   switch( _shape ) {
    case text7: return SHAPE_TEXT;
    case text6: return SHAPE_TEXT;
    case text8: return SHAPE_TEXT;
    case text9: return SHAPE_TEXT;
    case text10: return SHAPE_TEXT;
    case line1: return SHAPE_LINE;
    case text12: return SHAPE_TEXT;
    case text13: return SHAPE_TEXT;
    case text14: return SHAPE_TEXT;
    case line5: return SHAPE_LINE;
    case text19: return SHAPE_TEXT;
    case text20: return SHAPE_TEXT;
    case slider1: return SHAPE_SLIDER;
    case slider0: return SHAPE_SLIDER;
    case slider2: return SHAPE_SLIDER;
    case slider3: return SHAPE_SLIDER;
    case slider4: return SHAPE_SLIDER;
    case slider5: return SHAPE_SLIDER;
    case slider6: return SHAPE_SLIDER;
    case slider7: return SHAPE_SLIDER;
    case slider8: return SHAPE_SLIDER;
    default: return super.getShapeType( _shape );
   }
  }

  @Override
  public boolean isShapePublic( int _shape ) {
   switch( _shape ) {
    case text7:
    case text6:
    case text8:
    case text9:
    case text10:
    case text13:
    case text14:
    case text19:
    case text20:
    case slider1:
    case slider0:
    case slider2:
    case slider3:
    case slider4:
```

```
    case slider5:
    case slider6:
    case slider7:
    case slider8:
      return false;
    default:
      return true;
  }
 }

 @Override
 public double getShapeX( int _shape, int index ) {
  switch( _shape ) {
    case text7: return 346;
    case text6: return 346;
    case text8: return 346;
    case text9: return 350;
    case text10: return 350;
    case line1: return 30;
    case text12: return 45;
    case text13: return 350;
    case text14: return 350;
    case line5: return 120;
    case text19: return 350;
    case text20: return 350;
    case slider1: return 40;
    case slider0: return 40;
    case slider2: return 40;
    case slider3: return 40;
    case slider4: return 40;
    case slider5: return 40;
    case slider6: return 40;
    case slider7: return 40;
    case slider8: return 40;
    default: return super.getShapeX( _shape, index );
  }
 }

 @Override
 public double getShapeY( int _shape, int index ) {
  switch( _shape ) {
    case text7: return 170;
    case text6: return 130;
    case text8: return 210;
    case text9: return 279;
    case text10: return 319;
    case line1: return 50;
    case text12: return 90;
    case text13: return 359;
    case text14: return 399;
    case line5: return 100;
    case text19: return 439;
    case text20: return 479;
    case slider1: return 181;
    case slider0: return 141;
    case slider2: return 221;
    case slider3: return 331;
    case slider4: return 291;
    case slider5: return 371;
    case slider6: return 411;
    case slider7: return 491;
    case slider8: return 451;
    default: return super.getShapeY( _shape, index );
  }
 }

 @Override
 public Color getShapeLineColor( int _shape, int index ) {
  switch( _shape ) {
    // shapes
    case line1: return slateGray;
    case line5: return slateGray;
    // controls (text color)
    // charts (border color)
    default: return super.getShapeLineColor( _shape, index );
  }
 }

 @Override
 public Color getShapeFillColor( int _shape, int index ) {
  switch( _shape ) {
    case text7: return black;
    case text6: return black;
    case text8: return black;
    case text9: return black;
    case text10: return black;
    case text12: return black;
    case text13: return black;
    case text14: return black;
    case text19: return black;
    case text20: return black;
    case slider1: return transparent;
    case slider0: return transparent;
    case slider2: return transparent;
    case slider3: return transparent;
    case slider4: return transparent;
    case slider5: return transparent;
    case slider6: return transparent;
    case slider7: return transparent;
    case slider8: return transparent;
    default: return super.getShapeFillColor( _shape, index );
  }
 }

 @Override
 public int getShapeLineStyle( int _shape, int index ) {
  switch( _shape ) {
    case line1: return LINE_STYLE_SOLID;
    case line5: return LINE_STYLE_SOLID;
    default: return super.getShapeLineStyle( _shape, index );
  }
 }

 @Override
 public double getShapeLineDx( int _shape, int index ) {
  switch( _shape ) {
    case line1: return 970;
    case line5: return 230;
    default: return super.getShapeLineDx( _shape, index );
  }
 }

 @Override
 public double getShapeLineDy( int _shape, int index ) {
  switch( _shape ) {
    case line1: return 0;
    case line5: return 0;
    default: return super.getShapeLineDy( _shape, index );
  }
 }
```

```
 @Override
 public double getShapeWidth( int _shape, int index ) {
  switch( _shape ) {
   case slider1: return 316;
   case slider0: return 316;
   case slider2: return 316;
   case slider3: return 316;
   case slider4: return 316;
   case slider5: return 316;
   case slider6: return 316;
   case slider7: return 316;
   case slider8: return 316;
   default: return super.getShapeWidth( _shape, index );
  }
 }

 @Override
 public double getShapeHeight( int _shape, int index ) {
  switch( _shape ) {
   case slider1: return 30;
   case slider0: return 30;
   case slider2: return 30;
   case slider3: return 30;
   case slider4: return 30;
   case slider5: return 30;
   case slider6: return 30;
   case slider7: return 30;
   case slider8: return 30;
   default: return super.getShapeHeight( _shape, index );
  }
 }

 @Override
 public Object getShapeText( int _shape, int index ) {
  switch( _shape ) {
   case text7: return format(Initial_Dissident) ;
   case text6: return format(Initial_Government_Supporters) ;
   case text8: return format(Initial_Insurgent) ;
   case text9: return
format(People_Reaction_Delay_On_Anti_Regime_Messages) ;
   case text10: return
format(Regime_Social_and_Economic_Efforts) ;
   case text12: return "Parameters";
   case text13: return format(Contact_Rate) ;
   case text14: return format(People_Birth) ;
   case text19: return format(Avg_Time_as_Dissident) ;
   case text20: return format(Desired_Time_to_Remove_Insurgents) ;
   default: return super.getShapeText( _shape, index );
  }
 }


 @Override
 public Font getShapeFont( int _shape, int index ) {
  switch( _shape ) {
   case text7: return _text7_Font;
   case text6: return _text6_Font;
   case text8: return _text8_Font;
   case text9: return _text9_Font;
   case text10: return _text10_Font;
   case text12: return _text12_Font;
   case text13: return _text13_Font;
   case text14: return _text14_Font;
   case text19: return _text19_Font;
   case text20: return _text20_Font;
   default: return super.getShapeFont( _shape, index );
  }
 }

 @Override
 public int getShapeTextAlignment( int _shape, int index ) {
  switch( _shape ) {
   case text7: return ALIGNMENT_RIGHT;
   case text6: return ALIGNMENT_RIGHT;
   case text8: return ALIGNMENT_RIGHT;
   case text9: return ALIGNMENT_RIGHT;
   case text10: return ALIGNMENT_RIGHT;
   case text12: return ALIGNMENT_LEFT;
   case text13: return ALIGNMENT_RIGHT;
   case text14: return ALIGNMENT_RIGHT;
   case text19: return ALIGNMENT_RIGHT;
   case text20: return ALIGNMENT_RIGHT;
   default: return super.getShapeTextAlignment( _shape, index );
  }
 }

 @Override
 public boolean isShapeControlEnabled( int _shape, int index ) {
  switch( _shape ) {
   case slider1: return true ;
   case slider0: return true ;
   case slider2: return true ;
   case slider3: return true ;
   case slider4: return true ;
   case slider5: return true ;
   case slider6: return true ;
   case slider7: return true ;
   case slider8: return true ;
   default: return super.isShapeControlEnabled( _shape, index );
  }
 }

 @Override
 public boolean isShapeControlVertical( int _shape, int index ) {
  switch( _shape ) {
   case slider1: return false;
   case slider0: return false;
   case slider2: return false;
   case slider3: return false;
   case slider4: return false;
   case slider5: return false;
   case slider6: return false;
   case slider7: return false;
   case slider8: return false;
   default: return super.isShapeControlVertical( _shape, index );
  }
 }


 /**
  * <i>This method should not be called by user</i>
  */
 private void _button_SetDynamicParams( ShapeButton shape ) {
  boolean _visible;
  shape.setEnabled(getState() == IDLE );
 }

 ShapeButton button;
 ShapeText text;
```

```
  ShapeText text4;
  ShapeText text3;
  ShapeText text2;
  ShapeText text5;
  ShapeText text11;
  ShapeText text15;
  ShapeText text16;
  ShapeText text17;
  ShapeText text18;
  ShapeText text21;
  ShapeText text22;
  ShapeText text1;
  ShapeImage image;
  ShapeText text23;
  ShapePolyLine polyline;

  // Static initialization of persistent elements
  {
    button = new ShapeButton(
                                        Simulation.this, true, 50, 560,
                                        150, 30,
          controlDefault, controlDefault,
          _button_Font,
                                        "Run"
    ) {

      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
        _button_SetDynamicParams( this );
        super.draw( panel, graphics, xform, publicOnly );
      }

      @Override
      public void action(){
        executeShapeControlAction( _button, 0 );
      }
    };
    text = new ShapeText(
       true,30, 10, 0.0,
       slateGray,"SD+AB Insurgency Models",
       _text_Font, ALIGNMENT_LEFT
    );
    text4 = new ShapeText(
       false,46, 130, 0.0,
       black,"Initial government supporters",
       _text4_Font, ALIGNMENT_LEFT
    );
    text3 = new ShapeText(
       false,46, 170, 0.0,
       black,"Initial dissidents",
       _text3_Font, ALIGNMENT_LEFT
    );
    text2 = new ShapeText(
       false,46, 210, 0.0,
       black,"Initial insurgents",
       _text2_Font, ALIGNMENT_LEFT
    );
    text5 = new ShapeText(
       false,50, 279, 0.0,
       black,"People reaction delay on anti-regime messages",
       _text5_Font, ALIGNMENT_LEFT
    );
    text11 = new ShapeText(
       false,50, 319, 0.0,
       black,"Regime social and economic efforts",
       _text11_Font, ALIGNMENT_LEFT
    );
    text15 = new ShapeText(
       false,50, 399, 0.0,
       black,"People birth",
       _text15_Font, ALIGNMENT_LEFT
    );
    text16 = new ShapeText(
       false,50, 359, 0.0,
       black,"Contact rate",
       _text16_Font, ALIGNMENT_LEFT
    );
    text17 = new ShapeText(
       false,50, 439, 0.0,
       black,"Average time as dissident",
       _text17_Font, ALIGNMENT_LEFT
    );
    text18 = new ShapeText(
       false,50, 479, 0.0,
       black,"Desired time to remove insurgents",
       _text18_Font, ALIGNMENT_LEFT
    );
    text21 = new ShapeText(
       true,420, 120, 0.0,
       black,"This is a combined System Dynamics / Agent Based model of
insurgency in a society. The \r\nmodel contains the original System
Dynamics version and calculates the difference between \r\nthe mixed
model and the SD model. From the dynamics of insurgence viewpoint the
\r\npopulation is divided into four groups, namely
Government_Supporters, Dissidents, Insurgents and
\r\nRemoved_Insurgents. The model starts with Government_Supporters
= 2000, Dissidents = 450, \r\nInsurgents = 250. Dissidents and
Insurgents communicate with Government_Supporters recruiting
\r\npeople to become dissidents. This flow of new dissidents depends on
several \r\nfactors: number of active regime opponents (a fraction of
dissidents plus insurgents), \r\nfrequency of contacts (Contact_Rate) and
probability of recruiting (Propensity_to_be_Recruited). \r\nThere is a
process of dissident relaxation: not all the dissidents necessarily become
insurgents. \r\nA fraction of the dissident group is appeased by the
government and returns to supporting \r\npopulation. This returning flow
depends on an average time as dissident (Avg_Time_as_Dissident).
\r\nThe remaining part of the dissidents eventually becomes insurgents.
The government tries to reduce \r\nthe size of the insurgents group. As
a result of these efforts insurgents might be removed from \r\nthe
system (Removed_Insurgents). Strength of the efforts depends on
resources allocated and \r\nis proportional to the size of the insurgents
group. There is a feedback loop from dissidents and \r\ninsurgents
groups to the probability of recruiting mentioned at the very beginning of
this description. \r\nThe feedback is implemented as a flow of anti-
regime messages generated by a certain fraction \r\nof active dissidents
and active insurgents. This flow of messages with a certain delay reaches
a \r\nsocial network thus making the probability higher.\r\n\r\nRefer to
Description page of the model for more details (available at
runtime).\r\n\r\nThe model is based on a working paper "Using System
Dynamics to Model and Better \r\nUnderstand State Stability" Nazli
Choucri, Daniel Goldsmith, Stuart E. Madnick, Dinsha Mistree, \r\nJ.
Bradley Morrison, Michael D. Siegel\r\n",
       _text21_Font, ALIGNMENT_LEFT
    );
    text22 = new ShapeText(
       true,420, 90, 0.0,
       black,"Model description",
       _text22_Font, ALIGNMENT_LEFT
    );
    text1 = new ShapeText(
       true,230, 570, 0.0,
       slateGray,"Press \"Run\", the model starts in a while - please wait
while agents are created",
       _text1_Font, ALIGNMENT_LEFT
    );
    image = new ShapeImage(
                          Simulation.this, true, 420, 470, 0.0,
                                  116,                    40,
"/insurgencySDABModel/",
                                  new      String[]{"xjlogo.png",}
```

```
  );
  text23 = new ShapeText(
     true,460, 500, 0.0,
     _text23_Color,"This AnyLogic™ model is © 1992-2008 XJ Technologies. www.anylogic.com\r\n",
     _text23_Font, ALIGNMENT_LEFT
  );
  polyline = new ShapePolyLine(
                        true,220, 565,
      null, red,
      3, _polyline_pointsDX_xjal(), _polyline_pointsDY_xjal(),
      false, 1, LINE_STYLE_SOLID
  );
 }
 ShapeGroup presentation;
 ShapeGroup icon;

 @Override
 public Object getPersistentShape( int _shape ) {
  switch(_shape){
   case _presentation: return presentation;
   case _icon: return icon;

   case _button: return button;
   case _text: return text;
   case _text4: return text4;
   case _text3: return text3;
   case _text2: return text2;
   case _text5: return text5;
   case _text11: return text11;
   case _text15: return text15;
   case _text16: return text16;
   case _text17: return text17;
   case _text18: return text18;
   case _text21: return text21;
   case _text22: return text22;
   case _text1: return text1;
   case _image: return image;
   case _text23: return text23;
   case _polyline: return polyline;
   default: return null;
  }
 }


 @Override
 public int getWindowWidth() {
  return 1024;
 }

 @Override
 public int getWindowHeight() {
  return 681;
 }

 /**
  * Applet class to run experiment as java applet
  */
 public static class Applet extends JApplet {

  Simulation ex;

  @Override
  public void init() {
   ex = new Simulation();
   ex.setup( this );
  }

  @Override
  public void destroy() {
   ex.close();
  }

 }

 @Override
 public void initDefaultRandomNumberGenerator(Engine engine) {
  engine.getDefaultRandomGenerator().setSeed( 1 );
 }

 @Override
 public Main createRoot( Engine engine ) {
  // Create the root object
  return new Main( engine, null, null );
 }

 @Override
 public void setupRootParameters( Main root, boolean callOnChangeActions ) {
  int Initial_Dissident_xjal =
Initial_Dissident
;
  if (callOnChangeActions) {
   root.set_Initial_Dissident( Initial_Dissident_xjal );
  } else {
   root.Initial_Dissident = Initial_Dissident_xjal;
  }
  int Initial_Government_Supporters_xjal =
Initial_Government_Supporters
;
  if (callOnChangeActions) {
   root.set_Initial_Government_Supporters(
Initial_Government_Supporters_xjal );
  } else {
   root.Initial_Government_Supporters =
Initial_Government_Supporters_xjal;
  }
  int Initial_Insurgent_xjal =
Initial_Insurgent
;
  if (callOnChangeActions) {
   root.set_Initial_Insurgent( Initial_Insurgent_xjal );
  } else {
   root.Initial_Insurgent = Initial_Insurgent_xjal;
  }
  double People_Reaction_Delay_On_Anti_Regime_Messages_xjal =
People_Reaction_Delay_On_Anti_Regime_Messages
;
  if (callOnChangeActions) {
   root.set_People_Reaction_Delay_On_Anti_Regime_Messages(
People_Reaction_Delay_On_Anti_Regime_Messages_xjal );
  } else {
   root.People_Reaction_Delay_On_Anti_Regime_Messages =
People_Reaction_Delay_On_Anti_Regime_Messages_xjal;
  }
  double Regime_Social_and_Economic_Efforts_xjal =
Regime_Social_and_Economic_Efforts
;
  if (callOnChangeActions) {
   root.set_Regime_Social_and_Economic_Efforts(
Regime_Social_and_Economic_Efforts_xjal );
  } else {
```

```
    root.Regime_Social_and_Economic_Efforts =
Regime_Social_and_Economic_Efforts_xjal;
   }
   double Contact_Rate_xjal =
Contact_Rate
;
   if (callOnChangeActions) {
    root.set_Contact_Rate( Contact_Rate_xjal );
   } else {
    root.Contact_Rate = Contact_Rate_xjal;
   }
   int People_Birth_xjal = People_Birth ;
   if (callOnChangeActions) {
    root.set_People_Birth( People_Birth_xjal );
   } else {
    root.People_Birth = People_Birth_xjal;
   }
   double Avg_Time_as_Dissident_xjal =
Avg_Time_as_Dissident ;
   if (callOnChangeActions) {
    root.set_Avg_Time_as_Dissident( Avg_Time_as_Dissident_xjal );
   } else {
    root.Avg_Time_as_Dissident = Avg_Time_as_Dissident_xjal;
   }
   double Desired_Time_to_Remove_Insurgents_xjal =
Desired_Time_to_Remove_Insurgents ;
   if (callOnChangeActions) {
    root.set_Desired_Time_to_Remove_Insurgents(
Desired_Time_to_Remove_Insurgents_xjal );
   } else {
    root.Desired_Time_to_Remove_Insurgents =
Desired_Time_to_Remove_Insurgents_xjal;
   }
  }



  /**
   * Engine setup
   */
  @Override
  public void setupEngine(Engine engine) {
   engine.setATOL( 1.0E-5 );
   engine.setRTOL( 1.0E-5 );
   engine.setTTOL( 1.0E-5 );
   engine.setHTOL( 0.0010 );
   engine.setSolverODE( Engine.SOLVER_ODE_EULER );
   engine.setSolverNAE( Engine.SOLVER_NAE_MODIFIED_NEWTON );
   engine.setSolverDAE( Engine.SOLVER_DAE_RK45_NEWTON );
   engine.setVMethods( 16032 );

   engine.setStartTime( 0.0 );
   engine.setTimeUnit( TIME_UNIT_DAY );
   engine.setStopTime( 100.0 );
   engine.setRealTimeMode( true );
   engine.setRealTimeScale( 1.0 );
  }

  /**
   * Experiment setup
   */
  @Override
  public void setup( JApplet applet ) {
   setName( "InsurgencyUnitedModel : Simulation" );

   // Dynamic initialization of persistent elements
   presentation = new ShapeGroup( Simulation.this, true, 0, 0, 0, button,
slider1, slider0, slider2, slider3, slider4, slider5, slider6, slider7, slider8,
text, text4, text3, text7, text6, text2, text8, text9, text5, text10, text11,
line1, text12, text13, text14, text15, text16, line5, text17, text18, text19,
text20, text21, text22, text1, image, text23, polyline );
   icon = new ShapeGroup( Simulation.this, true, 0, 0, 0
);
   // Setup presentation
   Presentation _p = new Presentation( this, applet != null ?
Presentation.MODE_APPLET :
     Presentation.MODE_APPLICATION, applet );
   _p.start();

   Panel _panel = _p.getPanel();
   ToolBar _tb = _p.getToolBar();
   StatusBar _sb = _p.getStatusBar();

   _panel.setZoomEnabled( false );
   _panel.setPanningEnabled( false );
   _panel.setFrameManagementBalance( 2.0 );

   _sb.setSectionVisible( StatusBar.DATE, false );
   _sb.setSectionVisible( StatusBar.EPS, false );
   _sb.setSectionVisible( StatusBar.EXPERIMENT, false );
   _sb.setSectionVisible( StatusBar.FPS, false );
   _sb.setSectionVisible( StatusBar.MEMORY, true );
   _sb.setSectionVisible( StatusBar.SECONDS, true );
   _sb.setSectionVisible( StatusBar.SIMULATION, true );
   _sb.setSectionVisible( StatusBar.STATUS, true );
   _sb.setSectionVisible( StatusBar.STEP, false );
   _sb.setSectionVisible( StatusBar.TIME, true );
   _tb.setSectionVisible( ToolBar.ANIMATION, false );
   _tb.setSectionVisible( ToolBar.EXECUTION, true );
   _tb.setSectionVisible( ToolBar.FILE, false );
   _tb.setSectionVisible( ToolBar.NAVIGATION, true );
   _tb.setSectionVisible( ToolBar.TIME_SCALE, true );
   _tb.setSectionVisible( ToolBar.VIEW, false );
  }

}
```

Παράρτημα 8

UML – Διαγράμματα κλάσεων

Machine
+StartUpTimeout: double
+ConsistencyTimeout: double
+AcceptTimeout:double
+CandidateTimeout: double
+ElectionTimeout: double
+ConflictTimeout: double
+SynchroTimeout: double
+ResolveTimeoutHigh: double
+ResolveTimeoutLow: double
+ElectionTimeoutLow: double
+MTTR: double
+NoMasterTimeout: double
+MTTF: double
+NoOfMachines: int
+radius: int
+x: double
+y: double
+SynchroTimer: EventTimeout = new EventTimeout(this)
+protocol: Statechart = new Statechart(this, (short)7)
+Crashed: short=0
+Active: short=1
+StartUp: short=2
+Consistency: short=3
+Slave: short=4
+Candidate: short=5
+Master: short=6
+Conflict: short=7
+Accept: short=8
+NoMaster: short=9
+ElectionTimeout: TransitionTimeout = new TransitionTimeout(this)
+CandidateTimeout1: TransitionTimeout = new TransitionTimeout(this)
+ConflictTimeout1: TransitionTimeout = new TransitionTimeout(this)
+ResolveTimeout: TransitionTimeout = new TransitionTimeout(this)
+AcceptTimeout1: TransitionTimeout = new TransitionTimeout(this)
+ConsistencyTimeout1: TransitionTimeout = new TransitionTimeout(this)
+StartUpTimeout1: TransitionTimeout = new TransitionTimeout(this)
+NoMasterTimeout1: TransitionTimeout = new TransitionTimeout(this)
+Recovery: TransitionRate = new TransitionRate(this)
+Failure: TransitionRate = new TransitionRate(this)
+Off: TransitionMessage = new TransitionMessage(this)
+MASTERACK_1: TransitionMessage = new TransitionMessage(this)
+MASTERACK_2: TransitionMessage = new TransitionMessage(this)
+ACCEPT_: TransitionMessage = new TransitionMessage(this)
+QUIT_REFUSE_: TransitionMessage = new TransitionMessage(this)
+ELECTION_: TransitionMessage = new TransitionMessage(this)
+ELECTION_1: TransitionMessage = new TransitionMessage(this)
+ELECTION_2: TransitionMessage = new TransitionMessage(this)
+ELECTION_3: TransitionMessage = new TransitionMessage(this)
+CONFLICT_: TransitionMessage = new TransitionMessage(this)
+MASTERACK_: TransitionMessage = new TransitionMessage(this)
+QUIT_: TransitionMessage = new TransitionMessage(this)
+RESOLVE_MASTERREQ_: TransitionMessage = new TransitionMessage(this)
+MASTERUP_: TransitionMessage = new TransitionMessage(this)
+SYNCHROTICK_: TransitionMessage = new TransitionMessage(this)
+MASTERUP_1: TransitionMessage = new TransitionMessage(this)
+MASTERUP_MASTERACK_ELECTION_: TransitionMessage = new TransitionMessage(this)
+On: TransitionMessage = new TransitionMessage(this)
+statechartStatistics: DataSet = new DataSet(2000) { @Override public void update() { add(time),

```
_statechartStatistics_YValue(); } }
~_button1_Font: Font = new Font("Dialog", 0, 11)
~_button1: int=1
~_image: int=2
~_image1: int=3
~_group: int=4
~_oval: int=5
~_presentation: int=0
~_icon: int=-1
~button1: ShapeButton
~image: ShapeImage
~image1: ShapeImage
~group: ShapeGroup
~oval: ShapeOval
~presentation: ShapeGroup
~icon: ShapeGroup
~_Recovery_pointsX: int[*] = {500, 650, 650, 650}
~_Recovery_pointsY: int[*] = {530, 530, 520, 440}
~_Off_pointsX: int[*] = {490, 490}
~_Off_pointsY: int[*] = {440, 510}
~_Failure_pointsX: int[*] = {450, 450}
~_Failure_pointsY: int[*] = {440, 510}
~_MASTERACK__pointsX: int[*] = {710, 730, 730, 690, 690}
~_MASTERACK__pointsY: int[*] = {360, 360, 400, 400, 370}
~_MASTERACK_1_pointsX: int[*] = {270, 270}
~_MASTERACK_1_pointsY: int[*] = {100, 210}
~_MASTERACK_2_pointsX: int[*] = {440, 330}
~_MASTERACK_2_pointsY: int[*] = {80, 80}
~_ElectionTimeout_pointsX: int[*] = {330, 470, 470}
~_ElectionTimeout_pointsY: int[*] = {230, 280, 340}
~_ACCEPT__pointsX: int[*] = {440, 420, 420, 460, 460}
~_ACCEPT__pointsY: int[*] = {360, 360, 400, 400, 370}
~_QUIT_REFUSE__pointsX: int[*] = {450, 320}
~_QUIT_REFUSE__pointsY: int[*] = {340, 240}
~_ELECTION_1_pointsX: int[*] = {520, 540, 540, 500, 500}
~_ELECTION_1_pointsY: int[*] = {360, 360, 400, 400, 370}
~_CandidateTimeout1_pointsX: int[*] = {510, 632}
~_CandidateTimeout1_pointsY: int[*] = {340, 238}
~_CONFLICT__pointsX: int[*] = {640, 640}
~_CONFLICT__pointsY: int[*] = {240, 340}
~_Conflict_Timeout1_pointsX: int[*] = {690, 690}
~_Conflict_Timeout1_pointsY: int[*] = {340, 240}
~_QUIT__pointsX: int[*] = {630, 330}
~_QUIT__pointsY: int[*] = {220, 220}
~_ResolveTimeout_pointsX: int[*] = {660, 660}
~_ResolveTimeout_pointsY: int[*] = {240, 340}
~_ELECTION_2_pointsX: int[*] = {710, 740, 740, 700, 700}
~_ELECTION_2_pointsY: int[*] = {230, 230, 180, 180, 210}
~_RESOLVE_MASTERREQ__pointsX: int[*] = {710, 740, 740, 700, 700}
~_RESOLVE_MASTERREQ__pointsY: int[*] = {220, 220, 270, 270, 240}
~_MASTERUP_1_pointsX: int[*] = {240, 220, 220, 260, 260}
~_MASTERUP_1_pointsY: int[*] = {220, 220, 180, 180, 210}
~_SYNCHROTICK__pointsX: int[*] = {240, 220, 220, 260, 260}
~_SYNCHROTICK__pointsY: int[*] = {230, 230, 270, 270, 240}
~_ELECTION_3_pointsX: int[*] = {310, 310}
~_ELECTION_3_pointsY: int[*] = {240, 340}
~_MASTERUP__pointsX: int[*] = {330, 350, 350, 310, 310}
~_MASTERUP__pointsY: int[*] = {360, 360, 400, 400, 370}
~_AcceptTimeout1_pointsX: int[*] = {270, 270}
~_AcceptTimeout1_pointsY: int[*] = {340, 240}
~_ELECTION__pointsX: int[*] = {250, 230, 230, 270, 270}
~_ELECTION__pointsY: int[*] = {360, 360, 400, 400, 370}
~_Consistency_Timeout1_pointsX: int[*] = {310, 310}
~_Consistency_Timeout1_pointsY: int[*] = {100, 210}
~_StartUpTimeout1_pointsX: int[*] = {510, 630}
```

~_StartUpTimeout1_pointsY: int[*] = {80, 80} ~_MASTERUP_MASTERACK_ELECTION__pointsX: int[*] = {632, 328} ~_MASTERUP_MASTERACK_ELECTION__pointsY: int[*] = {98, 212} ~_NoMasterTimeout_pointsX: int[*] = {660, 660} ~_NoMasterTimeout_pointsY: int[*] = {100, 210} ~_On_pointsX: int[*] = {430, 300, 300} ~_On_pointsY: int[*] = {530, 530, 440} +port: Port = new Port(this) +SYNCHROTICK: int=0 +MASTERACK: int=1 +MASTERREQ: int=2 +MASTERUP: int=3 +ELECTION: int=4 +QUIT: int=5 +REFUSE: int=6 +ACCEPT: int=7 +RESOLVE: int=8 +CONFLICT: int=9 +SLAVEUP: int=10
<<create>>+Machine(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection) +_StartUpTimeout_DefaultValue_xjal(): double +set_StartUpTimeout(StartUpTimeout: double) ~onChange_StartUpTimeout() +_ConsistencyTimeout_DefaultValue_xjal(): double +set_ConsistencyTimeout(ConsistencyTimeout: double) ~onChange_ConsistencyTimeout() +_AcceptTimeout_DefaultValue_xjal(): double +set_AcceptTimeout(AcceptTimeout: double) ~onChange_AcceptTimeout() +_CandidateTimeout_DefaultValue_xjal(): double +set_CandidateTimeout(CandidateTimeout: double) ~onChange_CandidateTimeout() +_ElectionTimeoutHigh_DefaultValue_xjal(): double +set_ElectionTimeoutHigh(ElectionTimeoutHigh: double) ~onChange_ElectionTimeoutHigh() +_ConflictTimeout_DefaultValue_xjal(): double +set_ConflictTimeout(ConflictTimeout: double) ~onChange_ConflictTimeout() +_SynchroTimeout_DefaultValue_xjal(): double +set_SynchroTimeout(SynchroTimeout: double) ~onChange_SynchroTimeout() +_ResolveTimeoutHigh_DefaultValue_xjal(): double +set_ResolveTimeoutHigh(ResolveTimeoutHigh: double) ~onChange_ResolveTimeoutHigh() +_ResolveTimeoutLow_DefaultValue_xjal(): double +set_ResolveTimeoutLow(ResolveTimeoutLow: double) ~onChange_ResolveTimeoutLow() +_ElectionTimeoutLow_DefaultValue_xjal(): double +set_ElectionTimeoutLow(ElectionTimeoutLow: double) ~onChange_ElectionTimeoutLow() +_MTTR_DefaultValue_xjal(): double +set_MTTR(MTTR: double) ~onChange_MTTR() +_NoMasterTimeout_DefaultValue_xjal(): double +set_NoMasterTimeout(NoMasterTimeout: double) ~onChange_NoMasterTimeout() +_MTTF_DefaultValue_xjal(): double +set_MTTF(MTTF: double) ~onChange_MTTF() +_NoOfMachines_DefaultValue_xjal(): int +set_NoOfMachines(NoOfMachines: int) ~onChange_NoOfMachines() +_radius_DefaultValue_xjal(): int +set_radius(radius: int) ~onChange_radius()

+getNameOf(_e: EventTimeout): String
+getModeOf(_e: EventTimeout): int
+getFirstOccurrenceTime(_e: EventTimeout): double
+evaluateTimeoutOf(_e: EventTimeout): double
+executeActionOf(_e: EventTimeout)
+getNameOf(_s: StateChart): String
+executeActionOf(_s: StateChart)
+getNameOfState(_state: short): String
+stateContainsState(compstate: short, simpstate: short): boolean
+getContainerStateOf(_state: short): short
+enterState(_state: short, _destination: boolean)
+exitState(_state: short, _t: Transition, _source: boolean, _statechart: StateChart)
+getNameOf(_t: TransitionTimeout): String
+getStatechartOf(_t: TransitionTimeout): Statechart
+evaluateTimeoutOf(_t: TransitionTimeout): double
+executeActionOf(_t: TransitionTimeout)
+getNameOf(_t: TransitionRate): String
+getStatechartOf(_t: TransitionRate): Statechart
+evaluateRateOf(_t: TransitionRate): double
+executeActionOf(_t: TransitionRate)
+getNameOf(_t: TransitionMessage): String
+getStatechartOf(_t: TransitionMessage): Statechart
+executeActionOf(_t: TransitionMessage, _msg: Object)
+testMessageOf(_t: TransitionMessage, _msg: Object): boolean
~broadcast(id: int)
~sendTo(id: int, destination: Machine)
~linkExists(): boolean
-_statechartStatistics_YValue(): double
+onShapeClick(_shape: int, index: int, clickx:double, clicky: double): boolean
+executeShapeControlAction(_shape: int, index: int)
-_Image1_SetDynamicParams(shape: ShapeImage)
-_group_SetDynamicParams(shape: ShapeGroup)
-_oval_SetDynamicParams(shape: ShapeOval)
+getPersistentShape(_shape: int): Object
+drawModelElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean)
+onClickModelAt(panel: Panel, x: double, y: double, clickCount: int, publicOnly: boolean): boolean
+getNameOf(_port: Port): String
+executeOnReceiveActionOf(_port: Port, _msg: Object): boolean
+create()
+start()
+get_Main(): Main
+onChange()
+onDestroy()

Main

+MeanLatency: double
+LossRate: double
+NoOfMachines: int
+ConflictTimeout: double
+MTTR: double
+SynchroTimeout: double
+addDatasets: EventTimeout = new EventTimeout(this)
+_machinesStatesCharts_autoUpdateEvent_xjal: EventTimeout = new EventTimeout(this)
+machines: ActiveObjectArrayList = new ActiveObjectArrayList()
~_text_Font: Font = new Font("SansSerif", 1, 14)
~_text1_Font: Font = new Font("SansSerif", 0, 11)
~_text2_Font: Font = new Font("SansSerif", 0, 11)
~_text3_Font: Font = new Font("SansSerif", 0, 11)
~_text4_Font: Font = new Font("SansSerif", 0, 11)
~_text5_Font: Font = new Font("SansSerif", 0, 11)
~_text6_Font: Font = new Font("SansSerif", 0, 11)
~_text7_Font: Font = new Font("SansSerif", 0, 11)
~_text8_Font: Font = new Font("SansSerif", 0, 11)

```
~_text9_Font: Font = new Font("SansSerif", 0, 11)
~_Text_Font: Font = new Font("SansSerif", 1, 11)
~_Text1_Font: Font = new Font("SansSerif", 1, 11)
~_Text2_Font: Font = new Font("SansSerif", 0, 11)
~_Text3_Font: Font = new Font("SansSerif", 0, 11)
~_Text4_Font: Font = new Font("SansSerif", 0, 11)
~_Text5_Font: Font = new Font("SansSerif", 0, 11)
~_Text6_Font: Font = new Font("SansSerif", 0, 11)
~_Text7_Font: Font = new Font("SansSerif", 0, 11)
~_Text8_Font: Font = new Font("SansSerif", 0, 11)
~_Text9_Font: Font = new Font("SansSerif", 0, 11)
~_Text10_Font: Font = new Font("SansSerif", 1, 11)
~_Text11_Font: Font = new Font("SansSerif", 1, 11)
~_Text15_Font: Font = new Font("SansSerif", 1, 14)
~_Slider: int=1
~_Slider: int=2
~_Slider: int=3
~_Slider: int=4
~_Slider: int=5
~machines_presentation: int=6
~_line: int=7
~_group: int=8
~_text: int=9
~_rectangle1: int=10
~_text1: int=11
~_text2: int=12
~_rectangle2: int=13
~_rectangle3: int=14
~_text3: int=15
~_rectangle4: int=16
~_text4: int=17
~_text5: int=18
~_rectangle5: int=19
~_rectangle6: int=20
~_text6: int=21
~_rectangle7: int=22
~_text7: int=23
~_rectangle8: int=24
~_text8: int=25
~_text9: int=26
~_rectangle9: int=27
~_Text: int=28
~_Text1: int=29
~_Text2: int=30
~_Text3: int=31
~_Text4: int=32
~_Text5: int=33
~_Text6: int=34
~_Text7: int=35
~_Text8: int=36
~_Text9: int=37
~_text15: int=38
~_line1: int=39
~_line2: int=40
~_Text10: int=41
~_rectangle: int=42
~_rectangle10: int=43
~_Text11: int=44
~_machinesStatesCharts: int=45
~_presentation: int=0
~_icon: int=-1
~Slider: ShapeSlider
~Slider1: ShapeSlider
~Slider2: ShapeSlider
~Slider3: ShapeSlider
```

~Slider4: ShapeSlider
~_machinesStatesCharts: TimeColorChart
~line: ReplicatedShape
~group: ShapeGroup
~text: ShapeText
~rectangle1: ShapeRectangle
~text1: ShapeText
~text2: ShapeText
~rectangle2: ShapeRectangle
~rectangle3: ShapeRectangle
~text3: ShapeText
~rectangle4: ShapeRectangle
~text4: ShapeText
~text5: ShapeText
~rectangle5: ShapeRectangle
~rectangle6: ShapeRectangle
~text6: ShapeText
~rectangle7: ShapeRectangle
~text7: ShapeText
~rectangle8: ShapeRectangle
~text8: ShapeText
~text9: ShapeText
~rectangle9: ShapeRectangle
~Text: ShapeText
~Text1: ShapeText
~Text2: ShapeText
~Text3: ShapeText
~Text4: ShapeText
~Text5: ShapeText
~Text6: ShapeText
~Text7: ShapeText
~Text8: ShapeText
~Text9: ShapeText
~text15: ShapeText
~line1: ShapeLine
~line2: ShapeLine
~Text10: ShapeText
~rectangle: ShapeRectangle
~rectangle10: ShapeRectangle
~Text11: ShapeText
~presentation: ShapeGroup
~icon: ShapeGroup

<<create>>+Main(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection)
+_MeanLatency_DefaultValue_xjal(): double
+set_MeanLatency(MeanLatency: double)
~onChange_MeanLatency()
+_LossRate_DefaultValue_xjal(): double
+set_ LossRate(LossRate: double)
~onChange_LossRate()
+_NoOfMachines_DefaultValue_xjal(): int
+set_NoOfMachines(NoOfMachines: int)
~onChange_NoOfMachines()
+_ConflictTimeout_DefaultValue_xjal(): double
+set_ConflictTimeout(ConflictTimeout: double)
~onChange_ConflictTimeout()
+_MTTR_DefaultValue_xjal(): double
+set_MTTR(MTTR: double)
~onChange_MTTR()
+_SynchroTimeout_DefaultValue_xjal(): double
+set_SynchroTimeout(SynchroTimeout: double)
~onChange_SynchroTimeout()
+getNameOf(_e: EventTimeout): String
+getModeOf(_e: EventTimeout): int
+getFirstOccurrenceTime(_e: EventTimeout): double
+evaluateTimeoutOf(_e: EventTimeout): double

+executeActionOf(_e: EventTimeout)
+getNameOf(ao: ActiveObject): String
+getNameOf(aolist: ActiveObjectCollection): String
+add_machines(): Machine
+add_machines(StartUpTimeout: double, ConsistencyTimeout: double, AcceptTimeout: double, CandidateTimeout: double, ElectionTimeoutHigh: double, ConflictTimeout: double, SynchroTimeout: double, ResolveTimeoutHigh: double, ElectionTimeoutLow: double, MTTR: double, NoMasterTimeout: double, MTTF: double, NoOfMachines: int, radius: int): Machine
+remove_machines(object: Machine): boolean
-instantiate_machines_xjal(index: int): Machine
-setupParameters_machines_xjal(object: Machine, index: int)
-create_machines_xjal(object: Machine, index: int)
-instantiate_network_xjal(index: int): Network
-setupParameters_network_xjal(object: Network)
-create_network_xjal(object: Network)
+executeShapeControlAction(_shape: int, index: int, value: double)
+getShapeControlMinimum(_shape: int, index: int): double
+getShapeControlMaximum(_shape: int, index: int): double
+getShapeControlDefaultValueDouble(_shape: int, index: int): double
+getNameOfShape(_shape: int): String
+getShapeType(_shape: int): int
+getShapeReplicator(_shape: int): int
+getShapeX(_shape:int, index: int): double
+getShapeY(_shape:int, index: int): double
+getShapeEmbeddedObject(_shape: int): Object
-_Slider_SetDynamicParams(shape: ShapeSlider)
-_Slider1_SetDynamicParams(shape: ShapeSlider)
-_Slider2_SetDynamicParams(shape: ShapeSlider)
-_Slider3_SetDynamicParams(shape: ShapeSlider)
-_Slider4_SetDynamicParams(shape: ShapeSlider)
+_machinesStateCharts_ColorFromDouble(value: double): Color
-_line_SetDynamicParams(shape: ShapeLine, index: int)
+_line_createShapeWithStaticProperties(_index: int): ShapeLine
-_line_Replication(): int
-_Text_SetDynamicParams(shape: ShapeText)
-_Text1_SetDynamicParams(shape: ShapeText)
-_Text4_SetDynamicParams(shape: ShapeText)
-_Text7_SetDynamicParams(shape: ShapeText)
-_Text8_SetDynamicParams(shape: ShapeText)
+getPersistentShape(_shape: int): Object
+create()
+start()
+getEmbeddedObjects(): List
+onDestroy()

Network

+MeanLatency: double
+LossRate: double
~_text Font: Font = new Font("SansSerif", 1, 14)
~_oval_FillColor: Color = new Color(0xFF3366FF, true)
~_oval1_FillColor: Color = new Color(0xFF3366FF, true)
~_oval2_FillColor: Color = new Color(0xFF3366FF, true)
~_oval3_FillColor: Color = new Color(0xFF3366FF, true)
~_oval: int=1
~_oval1: int=2
~_oval2: int=3
~_oval3: int=4
~_text: int=5
~_presentation: int=0
~_icon: int=-1
~oval: ShapeOval
~oval1: ShapeOval
~oval2: ShapeOval
~oval3: ShapeOval

~text: ShapeText ~presentation: ShapeGroup ~icon: ShapeGroup +port: Port = new Port(this)
<<create>>+Network(engine: Engine, owner: ActiveObject, collection: ActiveObjectCollection) +_MeanLatency_DefaultValue_xjal(): double +set_MeanLatency(MeanLatency: double) ~onChange_MeanLatency() +_LossRate_DefaultValue_xjal(): double +set_LossRate(LossRate: double) ~onChange_LossRate() +create_TransmissionDelay(dt: double, msg: Object): TransmissionDelay +on_TransmissionDelay_xjal(msg: Object) +onShapeClick(_shape: int, index: int, clickx: double, clicky: double): boolean +getPersistentShape(_shape: int): Object +drawModelElements(_panel: Panel, _g: Graphics2D, _publicOnly: boolean) +onClickModelAt(panel: Panel, x: double, y: double, clickCount: int, publicOnly:Boolean): boolean +getNameOf(_p: Port): String +execute)nReceiveActionOf(_p: Port, _msg: Object): boolean +create() +start() +get_Main(): Main

Command
+id: int -serialVersionUID: long = 7222321206237672226L
<<create>>+Command(source: Machine, destination: Machine, id: int) +toString(): String

Simulation
~_button_Font: Font = new Font("Dialog", 0, 11) ~_text_Font: Font = new Font("SansSerif", 1, 28) ~_Text10 Font: Font = new Font("SansSerif", 0, 11) ~_text10 Font: Font = new Font("SansSerif", 0, 9) ~_button: int=1 ~_text: int=2 ~_Text10: int=3 ~image: int=4 ~text10: int=5 ~_presentation: int=0 ~-icon: int=-1 ~_image_filenames: String[*] = { "xjlogo.png" } ~button: ShapeButton ~text: ShapeText ~Text10: ShapeText ~presentation: ShapeGroup ~icon: ShapeGroup
+executeShapeControlAction(_shape: int, index: int) +getNameOfShape(_shape: int): String +getShapeType(_shape: int): int +getShapeX(_shape: int, index: int): double +getShapeY(_shape: int, index: int): double +getShapeFillColor(_shape: int, index: int): Color +getShapeWidth(_shape: int, index: int): double +getShapeHeight(_shape: int, index: int): double +getShapePackagePrefix(_shape: int): String +getShapeImageFileNames(_shape: int, index: int): String +getShapeText(_shape: int, index: int): Object +getShapeFont(_shape: int, index: int): Font +getShapeTextAlignment(_shape: int, index: int): int -_button_SetDynamicParams(shape: ShapeButton)

+getPersistentShape(_shape: int): Object
+getWindowWidth(): int
+getWindowHeight(): int
+initDefaultRandomNumberGenerator(engine: Engine)
+createRoot(engine: Engine): Main
+setupRootParameters(root: Main, callOnChangeActions: boolean)
+setupEngine(engine: Engine)
+setup(applet: JApplet)

Κώδικας Κεφαλαίου 10 – Έργο Leader Election Protocol

Main.java

```
package leader_election_protocol;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import java.awt.geom.Arc2D;

public class Main extends ActiveObject
{
 // Parameters

 public double  MeanLatency;

 /**
  * Returns default value for parameter <code>MeanLatency</code>.
  * <i>This method should not be called by user</i>
  */
 public double  _MeanLatency_DefaultValue_xjal() {
  return 1.0 ;
 }

 public void set_MeanLatency(
double  MeanLatency ) {
  if (MeanLatency == this.MeanLatency) {
   return;
  }
  this.MeanLatency = MeanLatency;
  onChange_MeanLatency();
  onChange();
 }

 void onChange_MeanLatency() {
            int index;
  network.set_MeanLatency( MeanLatency );

 }

 public double  LossRate;

 /**
  * Returns default value for parameter <code>LossRate</code>.
  * <i>This method should not be called by user</i>
  */
 public
double  _LossRate_DefaultValue_xjal() {
  return 0.0 ;
 }

 public void set_LossRate(
double  LossRate ) {
  if (LossRate == this.LossRate) {
   return;
  }
  this.LossRate = LossRate;
  onChange_LossRate();
  onChange();
 }

 void onChange_LossRate() {
            int index;
  network.set_LossRate( LossRate );

 }

 public int  NoOfMachines;

 /**
  * Returns default value for parameter <code>NoOfMachines</code>.
  * <i>This method should not be called by user</i>
  */
 public int  _NoOfMachines_DefaultValue_xjal() {
  return 19 ;
 }
```

```
  public void set_NoOfMachines(
int  NoOfMachines ) {
    if (NoOfMachines == this.NoOfMachines) {
      return;
    }
    this.NoOfMachines = NoOfMachines;
    onChange_NoOfMachines();
    onChange();
  }

  void onChange_NoOfMachines() {
                int index;
    index = 0;
    for ( Machine object : machines ) {
      object.set_NoOfMachines(NoOfMachines);
      index++;
    }

  }

  public double  ConflictTimeout;

  /**
   *       Returns       default       value       for       parameter
<code>ConflictTimeout</code>.
   * <i>This method should not be called by user</i>
   */
  public
double  _ConflictTimeout_DefaultValue_xjal() {
    return 5.0 ;
  }

  public void set_ConflictTimeout(
double  ConflictTimeout ) {
    if (ConflictTimeout == this.ConflictTimeout) {
      return;
    }
    this.ConflictTimeout = ConflictTimeout;
    onChange_ConflictTimeout();
    onChange();
  }

  void onChange_ConflictTimeout() {
                int index;
    index = 0;
    for ( Machine object : machines ) {
      object.set_ConflictTimeout(ConflictTimeout);
      index++;
    }
  }

  public double  MTTR;

  /**
   * Returns default value for parameter <code>MTTR</code>.
   * <i>This method should not be called by user</i>
   */
  public
double  _MTTR_DefaultValue_xjal() {
    return 100 ;
  }

  public void set_MTTR(
double  MTTR ) {
    if (MTTR == this.MTTR) {
      return;
    }
    this.MTTR = MTTR;
    onChange_MTTR();
    onChange();
  }

  void onChange_MTTR() {
                int index;
    index = 0;
    for ( Machine object : machines ) {
      object.set_MTTR(MTTR);
      index++;
    }

  }

  public double  SynchroTimeout;

  /**
   *       Returns       default       value       for       parameter
<code>SynchroTimeout</code>.
   * <i>This method should not be called by user</i>
   */
  public
double  _SynchroTimeout_DefaultValue_xjal() {
    return 3.0 ;
  }

  public void set_SynchroTimeout(
double  SynchroTimeout ) {
    if (SynchroTimeout == this.SynchroTimeout) {
      return;
    }
    this.SynchroTimeout = SynchroTimeout;
    onChange_SynchroTimeout();
    onChange();
  }

  void onChange_SynchroTimeout() {
                int index;
    index = 0;
    for ( Machine object : machines ) {
      object.set_SynchroTimeout(SynchroTimeout);
      index++;
    }

  }

  // Events

  public  EventTimeout  _machinesStatesCharts_autoUpdateEvent_xjal  =
new EventTimeout(this);


  @Override
  public String getNameOf( EventTimeout _e ) {
    if  (  _e  ==  _machinesStatesCharts_autoUpdateEvent_xjal  )  return
"machinesStatesCharts auto update event";
    return super.getNameOf( _e );
  }

  @Override
  public int getModeOf( EventTimeout _e ) {
    if  (  _e  ==  _machinesStatesCharts_autoUpdateEvent_xjal  )  return
EVENT_TIMEOUT_MODE_CYCLIC;
    return super.getModeOf( _e );
  }
```

```
 @Override
 public double getFirstOccurrenceTime( EventTimeout _e ) {
  if (
     _e == _machinesStatesCharts_autoUpdateEvent_xjal
   ) return getEngine().getStartTime();
  return super.getFirstOccurrenceTime( _e );
 }

 @Override
 public double evaluateTimeoutOf( EventTimeout _e ) {
  if ( _e == _machinesStatesCharts_autoUpdateEvent_xjal ) return
1
;
  return super.evaluateTimeoutOf( _e );
 }

 @Override
 public void executeActionOf( EventTimeout _e ) {
  if ( _e == _machinesStatesCharts_autoUpdateEvent_xjal ) {
   machinesStatesCharts.updateData();
   return;
  }
  super.executeActionOf( _e );
 }
 // Embedded Objects

 public Network network;

 public String getNameOf( ActiveObject ao ) {
  if ( ao == network ) return "network";
  return null;
 }

 public ActiveObjectArrayList<Machine> machines = new
ActiveObjectArrayList<Machine>();

 public String getNameOf( ActiveObjectCollection<?> aolist ) {
  if( aolist == machines ) return "machines";
  return null;
 }

 /**
  * This method creates and adds new embedded object in the
replicated embedded object collection machines<br>
  * @return newly created embedded object
  */
 public Machine add_machines() {
  int index = machines.size();
  Machine object = instantiate_machines_xjal( index );
  setupParameters_machines_xjal( object, index );
  create_machines_xjal( object, index );
  object.start();
  return object;
 }

 /**
  * This method creates and adds new embedded object in the
replicated embedded object collection machines<br>
  * This method uses given parameter values to setup created
embedded object<br>
  * Index of this new embedded object instance can be obtained
through calling <code>machines.size()</code> method
<strong>before</strong> this method is called
  * @param StartUpTimeout
  * @param ConsistencyTimeout
  * @param AcceptTimeout
  * @param CandidateTimeout
  * @param ElectionTimeoutHigh
  * @param ConflictTimeout
  * @param SynchroTimeout
  * @param ResolveTimeoutHigh
  * @param ResolveTimeoutLow
  * @param ElectionTimeoutLow
  * @param MTTR
  * @param NoMasterTimeout
  * @param MTTF
  * @param NoOfMachines
  * @param radius
  * @return newly created embedded object
  */
 public Machine add_machines( double StartUpTimeout, double
ConsistencyTimeout, double AcceptTimeout, double CandidateTimeout,
double ElectionTimeoutHigh, double ConflictTimeout, double
SynchroTimeout, double ResolveTimeoutHigh, double
ResolveTimeoutLow, double ElectionTimeoutLow, double MTTR, double
NoMasterTimeout, double MTTF, int NoOfMachines, int radius ) {
  int index = machines.size();
  Machine object = instantiate_machines_xjal( index );
  // Setup parameters
  object.StartUpTimeout = StartUpTimeout;
  object.ConsistencyTimeout = ConsistencyTimeout;
  object.AcceptTimeout = AcceptTimeout;
  object.CandidateTimeout = CandidateTimeout;
  object.ElectionTimeoutHigh = ElectionTimeoutHigh;
  object.ConflictTimeout = ConflictTimeout;
  object.SynchroTimeout = SynchroTimeout;
  object.ResolveTimeoutHigh = ResolveTimeoutHigh;
  object.ResolveTimeoutLow = ResolveTimeoutLow;
  object.ElectionTimeoutLow = ElectionTimeoutLow;
  object.MTTR = MTTR;
  object.NoMasterTimeout = NoMasterTimeout;
  object.MTTF = MTTF;
  object.NoOfMachines = NoOfMachines;
  object.radius = radius;
  // Finish embedded object creation
  create_machines_xjal( object, index );
  object.start();
  return object;
 }

 /**
  * This method removes the given embedded object from the replicated
embedded object collection machines<br>
  * The given object is destroyed, but not immediately in common case.
  * @param object the active object - element of replicated embedded
object machines - which should be removed
  * @return <code>true</code> if object was removed successfully,
<code>false</code> if it doesn't belong to machines
  */
 public boolean remove_machines( Machine object ) {
  if( !machines._remove( object ) ){
   return false;
  }
  object.setDestroyed();
  return true;
 }

 /**
  * Creates an embedded object instance and adds it to the end of
replicated embedded object list<br>
  * <i>This method should not be called by user</i>
  */
 private Machine instantiate_machines_xjal( final int index ) {Machine
object = new Machine( getEngine(), this, machines );
```

```
    machines._add(object);

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_machines_xjal(Machine object, final int index ) {
    object.StartUpTimeout = object._StartUpTimeout_DefaultValue_xjal();
    object.ConsistencyTimeout = object._ConsistencyTimeout_DefaultValue_xjal();
    object.AcceptTimeout = object._AcceptTimeout_DefaultValue_xjal();
    object.CandidateTimeout = object._CandidateTimeout_DefaultValue_xjal();
    object.ElectionTimeoutHigh = object._ElectionTimeoutHigh_DefaultValue_xjal();
    object.ConflictTimeout = ConflictTimeout;
    object.SynchroTimeout = SynchroTimeout;
    object.ResolveTimeoutHigh = object._ResolveTimeoutHigh_DefaultValue_xjal();
    object.ResolveTimeoutLow = object._ResolveTimeoutLow_DefaultValue_xjal();
    object.ElectionTimeoutLow = object._ElectionTimeoutLow_DefaultValue_xjal();
    object.MTTR = MTTR;
    object.NoMasterTimeout = object._NoMasterTimeout_DefaultValue_xjal();
    object.MTTF = object._MTTF_DefaultValue_xjal();
    object.NoOfMachines = NoOfMachines;
    object.radius = object._radius_DefaultValue_xjal();
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_machines_xjal(Machine object, final int index ) {
    object.create();

    // Port connections
    // connector:
    network.port.connect( object.port );
  }
  /**
   * Creates an embedded object instance<br>
   * <i>This method should not be called by user</i>
   */
  private Network instantiate_network_xjal() {Network object = new Network( getEngine(), this, null );

    return object;
  }

  /**
   * Setups parameters of an embedded object instance<br>
   * This method should not be called by user
   */
  private void setupParameters_network_xjal(Network object ) {
    object.MeanLatency = MeanLatency ;
    object.LossRate = LossRate ;
  }

  /**
   * Setups an embedded object instance<br>
   * This method should not be called by user
   */
  private void create_network_xjal(Network object ) {
    object.create();

  }

  // View areas
  public ViewArea MainView = new ViewArea( this, "MainView", 0, 0, ViewArea.TOP_LEFT, ViewArea.NONE, 1.0, 100, 100 );

  static final Font _text_Font = new Font("SansSerif", 1, 14 );
  static final Font _text1_Font = new Font("SansSerif", 0, 11 );
  static final Font _text2_Font = _text1_Font;
  static final Font _text3_Font = _text1_Font;
  static final Font _text4_Font = _text1_Font;
  static final Font _text5_Font = _text1_Font;
  static final Font _text6_Font = _text1_Font;
  static final Font _text7_Font = _text1_Font;
  static final Font _text8_Font = _text1_Font;
  static final Font _text9_Font = _text1_Font;
  static final Font _Text_Font = new Font("SansSerif", 1, 11 );
  static final Font _Text1_Font = _Text_Font;
  static final Font _Text2_Font = _text1_Font;
  static final Font _Text3_Font = _text1_Font;
  static final Font _Text4_Font = _Text_Font;
  static final Font _Text5_Font = _text1_Font;
  static final Font _Text6_Font = _text1_Font;
  static final Font _Text7_Font = _Text_Font;
  static final Font _Text8_Font = _Text_Font;
  static final Font _Text9_Font = _text1_Font;
  static final Font _text15_Font = _text_Font;
  static final Font _Text10_Font = _Text_Font;
  static final Font _Text11_Font = _Text_Font;
  static final int _Slider = 1;
  static final int _Slider1 = 2;
  static final int _Slider2 = 3;
  static final int _Slider3 = 4;
  static final int _Slider4 = 5;
  static final int machines_presentation = 6;
  static final int _line = 7;
  static final int _group = 8;
  static final int _text = 9;
  static final int _rectangle1 = 10;
  static final int _text1 = 11;
  static final int _text2 = 12;
  static final int _rectangle2 = 13;
  static final int _rectangle3 = 14;
  static final int _text3 = 15;
  static final int _rectangle4 = 16;
  static final int _text4 = 17;
  static final int _text5 = 18;
  static final int _rectangle5 = 19;
  static final int _rectangle6 = 20;
  static final int _text6 = 21;
  static final int _rectangle7 = 22;
  static final int _text7 = 23;
  static final int _rectangle8 = 24;
  static final int _text8 = 25;
```

```
  static final int _text9 = 26;
  static final int _rectangle9 = 27;
  static final int _Text = 28;
  static final int _Text1 = 29;
  static final int _Text2 = 30;
  static final int _Text3 = 31;
  static final int _Text4 = 32;
  static final int _Text5 = 33;
  static final int _Text6 = 34;
  static final int _Text7 = 35;
  static final int _Text8 = 36;
  static final int _Text9 = 37;
  static final int _text15 = 38;
  static final int _line1 = 39;
  static final int _line2 = 40;
  static final int _Text10 = 41;
  static final int _rectangle = 42;
  static final int _rectangle10 = 43;
  static final int _Text11 = 44;
  static final int _machinesStatesCharts = 45;

  /**
   * Top-level presentation group id
   */
  static final int _presentation = 0;

  /**
   * Top-level icon group id
   */
  static final int _icon = -1;



  @Override
  public void executeShapeControlAction( int _shape, int index, double
value ) {
    switch( _shape ) {
      case _Slider:
        set_MTTR( value );
        break;
      case _Slider1:
        set_LossRate( value );
        break;
      case _Slider2:
        set_ConflictTimeout( value );
        break;
      case _Slider3:
        set_SynchroTimeout( value );
        break;
      case _Slider4:
        set_MeanLatency( value );
        break;
      default:
        super.executeShapeControlAction( _shape, index, value );
        break;
    }
  }

  @Override
  public double getShapeControlMinimum( int _shape, int index ) {
    switch( _shape ) {
      case _Slider: return 20 ;
      case _Slider1: return 0.0 ;
      case _Slider2: return 1.0 ;
      case _Slider3: return 1.0 ;
      case _Slider4: return 0.2 ;
      default: return super.getShapeControlMinimum( _shape, index );
    }
  }

  @Override
  public double getShapeControlMaximum( int _shape, int index ) {
    switch( _shape ) {
      case _Slider: return 500 ;
      case _Slider1: return 1.0 ;
      case _Slider2: return 20.0 ;
      case _Slider3: return 7.0 ;
      case _Slider4: return 5.0 ;
      default: return super.getShapeControlMaximum( _shape, index );
    }
  }

  @Override
  public double getShapeControlDefaultValueDouble( int _shape, int index
) {
    switch(_shape) {
      case _Slider: return MTTR ;
      case _Slider1: return LossRate ;
      case _Slider2: return ConflictTimeout ;
      case _Slider3: return SynchroTimeout ;
      case _Slider4: return MeanLatency ;
      default:  return  super.getShapeControlDefaultValueDouble(  _shape,
index );
    }
  }


  @Override
  public String getNameOfShape( int _shape ) {
    switch( _shape ) {
      case machines_presentation: return "machines_presentation";
      default: return super.getNameOfShape( _shape );
    }
  }

  @Override
  public int getShapeType( int _shape ) {
    switch( _shape ) {
      case machines_presentation: return SHAPE_EMBEDDED_OBJECT;
      default: return super.getShapeType( _shape );
    }
  }

  @Override
  public int getShapeReplication( int _shape ) {
    switch( _shape ) {
      case machines_presentation: return
machines.size()
;
      default: return super.getShapeReplication( _shape );
    }
  }

  @Override
  public double getShapeX( int _shape, int index ) {
    switch( _shape ) {
      case machines_presentation: return 0;
      default: return super.getShapeX( _shape, index );
    }
  }

  @Override
  public double getShapeY( int _shape, int index ) {
```

```
    switch( _shape ) {
      case machines_presentation: return 0;
      default: return super.getShapeY( _shape, index );
    }
  }

  @Override
  public Object getShapeEmbeddedObject( int _shape ) {
    switch( _shape ) {
      case machines_presentation: return machines;
      default: return super.getShapeEmbeddedObject( _shape );
    }
  }



  /**
   * <i>This method should not be called by user</i>
   */
  private void _Slider_SetDynamicParams( ShapeSlider shape ) {
    boolean _visible;
    shape.setRange( getShapeControlMinimum( _Slider ),
getShapeControlMaximum( _Slider ) );
  }

  ShapeSlider Slider;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _Slider1_SetDynamicParams( ShapeSlider shape ) {
    boolean _visible;
    shape.setRange( getShapeControlMinimum( _Slider1 ),
getShapeControlMaximum( _Slider1 ) );
  }

  ShapeSlider Slider1;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _Slider2_SetDynamicParams( ShapeSlider shape ) {
    boolean _visible;
    shape.setRange( getShapeControlMinimum( _Slider2 ),
getShapeControlMaximum( _Slider2 ) );
  }

  ShapeSlider Slider2;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _Slider3_SetDynamicParams( ShapeSlider shape ) {
    boolean _visible;
    shape.setRange( getShapeControlMinimum( _Slider3 ),
getShapeControlMaximum( _Slider3 ) );
  }

  ShapeSlider Slider3;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _Slider4_SetDynamicParams( ShapeSlider shape ) {
    boolean _visible;
    shape.setRange( getShapeControlMinimum( _Slider4 ),
getShapeControlMaximum( _Slider4 ) );
  }

  ShapeSlider Slider4;
  public Color _machinesStatesCharts_ColorFromDouble( double value ) {
    if (
value == Machine.Consistency
      ) return blue;
    if (
value == Machine.StartUp
      ) return plum;
    if (
value == Machine.NoMaster
      ) return darkGoldenRod;
    if (
value == Machine.Slave
      ) return steelBlue;
    if (
value == Machine.Master
      ) return darkOrange;
    if (
value == Machine.Accept
      ) return lime;
    if (
value == Machine.Candidate
      ) return blueViolet;
    if (
value == Machine.Conflict
      ) return red;
    if (
value == Machine.Crashed
      ) return silver;
    return null;
  }

  TimeColorChart machinesStatesCharts;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _line_SetDynamicParams( ShapeLine shape, int index ) {
    boolean _visible;
    shape.setX(
machines.get( index ).x
);
    shape.setY(
machines.get( index ).y
);
    shape.setDx(
machines.get( index ).linkExists() ? machines.get( index ).MyMaster.x -
machines.get( index ).x : 0
);
    shape.setDy(
machines.get( index ).linkExists() ? machines.get( index ).MyMaster.y -
machines.get( index ).y : 0
);
    shape.setColor(
machines.get( index ).linkExists() ? slateBlue : null
);
  }



  /**
   * <i>This method should not be called by user</i>
   */
  public ShapeLine _line_createShapeWithStaticProperties( final int
_index ) {
    ShapeLine shape = new ShapeLine(
                              true,0, 0,
```

```
      slateBlue,
                              60,          0,
                              1, LINE_STYLE_SOLID
    );
    return shape;
  }

  /**
   * <i>This method should not be called by user</i>
   */
  private int _line_Replication() {
    return
NoOfMachines
;
  }

  ReplicatedShape<ShapeLine> line;
  ShapeGroup group;
  ShapeText text;
  ShapeRectangle rectangle1;
  ShapeText text1;
  ShapeText text2;
  ShapeRectangle rectangle2;
  ShapeRectangle rectangle3;
  ShapeText text3;
  ShapeRectangle rectangle4;
  ShapeText text4;
  ShapeText text5;
  ShapeRectangle rectangle5;
  ShapeRectangle rectangle6;
  ShapeText text6;
  ShapeRectangle rectangle7;
  ShapeText text7;
  ShapeRectangle rectangle8;
  ShapeText text8;
  ShapeText text9;
  ShapeRectangle rectangle9;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _Text_SetDynamicParams( ShapeText shape ) {
    boolean _visible;
    shape.setText(LossRate );
  }

  ShapeText Text;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _Text1_SetDynamicParams( ShapeText shape ) {
    boolean _visible;
    shape.setText(MeanLatency );
  }

  ShapeText Text1;
  ShapeText Text2;
  ShapeText Text3;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _Text4_SetDynamicParams( ShapeText shape ) {
    boolean _visible;
    shape.setText(ConflictTimeout );
  }

  ShapeText Text4;
  ShapeText Text5;
  ShapeText Text6;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _Text7_SetDynamicParams( ShapeText shape ) {
    boolean _visible;
    shape.setText(SynchroTimeout );
  }

  ShapeText Text7;

  /**
   * <i>This method should not be called by user</i>
   */
  private void _Text8_SetDynamicParams( ShapeText shape ) {
    boolean _visible;
    shape.setText(MTTR );
  }

  ShapeText Text8;
  ShapeText Text9;
  ShapeText text15;
  ShapeLine line1;
  ShapeLine line2;
  ShapeText Text10;
  ShapeRectangle rectangle;
  ShapeRectangle rectangle10;
  ShapeText Text11;

  // Static initialization of persistent elements
  {
    Slider = new ShapeSlider(
                                Main.this, true,230, 170,
                                        180, 30,
          transparent,
          false,          getShapeControlMinimum(          _Slider          ),
getShapeControlMaximum( _Slider ), ShapeControl.TYPE_DOUBLE
    ) {

      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
        _Slider_SetDynamicParams( this );
        super.draw( panel, graphics, xform, publicOnly );
      }

      @Override
      public void action(){
        executeShapeControlAction( _Slider, 0, value );
      }

      @Override
      public void setValueToDefault() {
                              setValue(              limit(              getMin(),
getShapeControlDefaultValueDouble( _Slider, 0 ), getMax() ) );
      }
    };
    Slider1 = new ShapeSlider(
                                Main.this, true,230, 50,
                                        180, 30,
```

```
        transparent,
        false, getShapeControlMinimum( _Slider1 ),
getShapeControlMaximum( _Slider1 ), ShapeControl.TYPE_DOUBLE
    ) {

      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
        _Slider1_SetDynamicParams( this );
        super.draw( panel, graphics, xform, publicOnly );
      }

      @Override
      public void action(){
        executeShapeControlAction( _Slider1, 0, value );
      }

      @Override
      public void setValueToDefault() {
                    setValue( limit( getMin(),
getShapeControlDefaultValueDouble( _Slider1, 0 ), getMax() ) );
      }
    };
    Slider2 = new ShapeSlider(
                        Main.this, true,230, 140,
                                180, 30,
        transparent,
        false, getShapeControlMinimum( _Slider2 ),
getShapeControlMaximum( _Slider2 ), ShapeControl.TYPE_DOUBLE
    ) {

      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
        _Slider2_SetDynamicParams( this );
        super.draw( panel, graphics, xform, publicOnly );
      }

      @Override
      public void action(){
        executeShapeControlAction( _Slider2, 0, value );
      }

      @Override
      public void setValueToDefault() {
                    setValue( limit( getMin(),
getShapeControlDefaultValueDouble( _Slider2, 0 ), getMax() ) );
      }
    };
    Slider3 = new ShapeSlider(
                        Main.this, true,230, 80,
                                180, 30,
        transparent,
        false, getShapeControlMinimum( _Slider3 ),
getShapeControlMaximum( _Slider3 ), ShapeControl.TYPE_DOUBLE
    ) {

      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
        _Slider3_SetDynamicParams( this );
        super.draw( panel, graphics, xform, publicOnly );
      }

      @Override
      public void action(){
        executeShapeControlAction( _Slider3, 0, value );
      }

      @Override
      public void setValueToDefault() {
                    setValue( limit( getMin(),
getShapeControlDefaultValueDouble( _Slider3, 0 ), getMax() ) );
      }
    };
    Slider4 = new ShapeSlider(
                        Main.this, true,230, 110,
                                180, 30,
        transparent,
        false, getShapeControlMinimum( _Slider4 ),
getShapeControlMaximum( _Slider4 ), ShapeControl.TYPE_DOUBLE
    ) {

      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
        _Slider4_SetDynamicParams( this );
        super.draw( panel, graphics, xform, publicOnly );
      }

      @Override
      public void action(){
        executeShapeControlAction( _Slider4, 0, value );
      }

      @Override
      public void setValueToDefault() {
                    setValue( limit( getMin(),
getShapeControlDefaultValueDouble( _Slider4, 0 ), getMax() ) );
      }
    };
    line = new ReplicatedShape<ShapeLine>() {
      @Override
      public int getReplication() {
        return _line_Replication();
      }

      @Override
      public ShapeLine createShapeWithStaticProperties( int index ) {
        return _line_createShapeWithStaticProperties( index );
      }

      @Override
      public void setShapeDynamicProperties( ShapeLine shape, int index )
{
        _line_SetDynamicParams( shape, index );
      }
    };
    text = new ShapeText(
      true,30, 260, 0.0,
      black,"Machine states",
      _text_Font, ALIGNMENT_LEFT
    );
    rectangle1 = new ShapeRectangle(
      true,210, 560, 0.0,
        null, blue,
                                20, 10,
                                1, LINE_STYLE_SOLID
    );
    text1 = new ShapeText(
      true,240, 560, 0.0,
      black,"Consistency",
      _text1_Font, ALIGNMENT_LEFT
    );
    text2 = new ShapeText(
```

```
    true,340, 560, 0.0,
    black,"StartUp",
    _text2_Font, ALIGNMENT_LEFT
  );
  rectangle2 = new ShapeRectangle(
    true,310, 560, 0.0,
      null, plum,
                                  20, 10,
                                  1, LINE_STYLE_SOLID
  );
  rectangle3 = new ShapeRectangle(
    true,390, 560, 0.0,
      null, darkGoldenRod,
                                  20, 10,
                                  1, LINE_STYLE_SOLID
  );
  text3 = new ShapeText(
    true,420, 560, 0.0,
    black,"NoMaster",
    _text3_Font, ALIGNMENT_LEFT
  );
  rectangle4 = new ShapeRectangle(
    true,130, 560, 0.0,
      null, steelBlue,
                                  20, 10,
                                  1, LINE_STYLE_SOLID
  );
  text4 = new ShapeText(
    true,160, 560, 0.0,
    black,"Slave",
    _text4_Font, ALIGNMENT_LEFT
  );
  text5 = new ShapeText(
    true,80, 560, 0.0,
    black,"Master",
    _text5_Font, ALIGNMENT_LEFT
  );
  rectangle5 = new ShapeRectangle(
    true,50, 560, 0.0,
      null, darkOrange,
                                  20, 10,
                                  1, LINE_STYLE_SOLID
  );
  rectangle6 = new ShapeRectangle(
    true,480, 561, 0.0,
      null, lime,
                                  20, 10,
                                  1, LINE_STYLE_SOLID
  );
  text6 = new ShapeText(
    true,510, 560, 0.0,
    black,"Accept",
    _text6_Font, ALIGNMENT_LEFT
  );
  rectangle7 = new ShapeRectangle(
    true,560, 560, 0.0,
      null, blueViolet,
                                  20, 10,
                                  1, LINE_STYLE_SOLID
  );
  text7 = new ShapeText(
    true,590, 560, 0.0,
    black,"Candidate",
    _text7_Font, ALIGNMENT_LEFT
  );
  rectangle8 = new ShapeRectangle(
    true,660, 560, 0.0,
      null, red,
                                  20, 10,
                                  1, LINE_STYLE_SOLID
  );
  text8 = new ShapeText(
    true,690, 560, 0.0,
    black,"Conflict",
    _text8_Font, ALIGNMENT_LEFT
  );
  text9 = new ShapeText(
    true,780, 560, 0.0,
    black,"Crashed",
    _text9_Font, ALIGNMENT_LEFT
  );
  rectangle9 = new ShapeRectangle(
    true,750, 560, 0.0,
      null, silver,
                                  20, 10,
                                  1, LINE_STYLE_SOLID
  );
  Text = new ShapeText(
    true,190, 60, 0.0,
    blue,"123",
    _Text_Font, ALIGNMENT_LEFT
  ) {

    @Override
    public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
      _Text_SetDynamicParams( this );
      super.draw( panel, graphics, xform, publicOnly );
    }
  };
  Text1 = new ShapeText(
    true,190, 120, 0.0,
    blue,"123",
    _Text1_Font, ALIGNMENT_LEFT
  ) {

    @Override
    public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
      _Text1_SetDynamicParams( this );
      super.draw( panel, graphics, xform, publicOnly );
    }
  };
  Text2 = new ShapeText(
    true,50, 120, 0.0,
    black,"Network mean latency:",
    _Text2_Font, ALIGNMENT_LEFT
  );
  Text3 = new ShapeText(
    true,50, 90, 0.0,
    black,"Synchro timeout:",
    _Text3_Font, ALIGNMENT_LEFT
  );
  Text4 = new ShapeText(
    true,190, 150, 0.0,
    blue,"123",
    _Text4_Font, ALIGNMENT_LEFT
  ) {

    @Override
```

```
    public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
      _Text4_SetDynamicParams( this );
      super.draw( panel, graphics, xform, publicOnly );
    }
  };
  Text5 = new ShapeText(
    true,50, 150, 0.0,
    black,"Conflict timeout:",
    _Text5_Font, ALIGNMENT_LEFT
  );
  Text6 = new ShapeText(
    true,50, 180, 0.0,
    black,"Mean recovery time:",
    _Text6_Font, ALIGNMENT_LEFT
  );
  Text7 = new ShapeText(
    true,190, 90, 0.0,
    blue,"123",
    _Text7_Font, ALIGNMENT_LEFT
  ) {

    @Override
    public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
      _Text7_SetDynamicParams( this );
      super.draw( panel, graphics, xform, publicOnly );
    }
  };
  Text8 = new ShapeText(
    true,190, 180, 0.0,
    blue,"123",
    _Text8_Font, ALIGNMENT_LEFT
  ) {

    @Override
    public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
      _Text8_SetDynamicParams( this );
      super.draw( panel, graphics, xform, publicOnly );
    }
  };
  Text9 = new ShapeText(
    true,50, 60, 0.0,
    black,"Network loss rate:",
    _Text9_Font, ALIGNMENT_LEFT
  );
  text15 = new ShapeText(
    true,30, 20, 0.0,
    black,"Parameters",
    _text15_Font, ALIGNMENT_LEFT
  );
  line1 = new ShapeLine(
                    true,30, 280,
    silver,
                    790,        0,
                    1, LINE_STYLE_SOLID
  );
  line2 = new ShapeLine(
                    true,30, 40,
    silver,
                    390,        0,
                    1, LINE_STYLE_SOLID
  );
  Text10 = new ShapeText(
    true,50, 210, 0.0,
    black,"Tip: click on a machine to view its statechart",
    _Text10_Font, ALIGNMENT_LEFT
  );
  rectangle = new ShapeRectangle(
    true,40, 210, 0.0,
      null, red,
                                    5, 5,
                                    1, LINE_STYLE_SOLID
  );
  rectangle10 = new ShapeRectangle(
    true,40, 225, 0.0,
      null, red,
                                    5, 5,
                                    1, LINE_STYLE_SOLID
  );
  Text11 = new ShapeText(
    true,50, 225, 0.0,
    black,"Tip: click a circle near the machine to force it to crash or
recover",
    _Text11_Font, ALIGNMENT_LEFT
  );
 }
 ShapeGroup presentation;
 ShapeGroup icon;

 @Override
 public Object getPersistentShape( int _shape ) {
  switch(_shape){
   case _presentation: return presentation;
   case _icon: return icon;

   case _Slider: return Slider;
   case _Slider1: return Slider1;
   case _Slider2: return Slider2;
   case _Slider3: return Slider3;
   case _Slider4: return Slider4;
   case _machinesStatesCharts: return machinesStatesCharts;
   case _line: return line;
   case _group: return group;
   case _text: return text;
   case _rectangle1: return rectangle1;
   case _text1: return text1;
   case _text2: return text2;
   case _rectangle2: return rectangle2;
   case _rectangle3: return rectangle3;
   case _text3: return text3;
   case _rectangle4: return rectangle4;
   case _text4: return text4;
   case _text5: return text5;
   case _rectangle5: return rectangle5;
   case _rectangle6: return rectangle6;
   case _text6: return text6;
   case _rectangle7: return rectangle7;
   case _text7: return text7;
   case _rectangle8: return rectangle8;
   case _text8: return text8;
   case _text9: return text9;
   case _rectangle9: return rectangle9;
   case _Text: return Text;
   case _Text1: return Text1;
   case _Text2: return Text2;
   case _Text3: return Text3;
   case _Text4: return Text4;
   case _Text5: return Text5;
   case _Text6: return Text6;
   case _Text7: return Text7;
```

```
    case _Text8: return Text8;
    case _Text9: return Text9;
    case _text15: return text15;
    case _line1: return line1;
    case _line2: return line2;
    case _Text10: return Text10;
    case _rectangle: return rectangle;
    case _rectangle10: return rectangle10;
    case _Text11: return Text11;
    default: return null;
   }
  }


  /**
   * Constructor
   */
  public Main( Engine engine, ActiveObject owner,
ActiveObjectCollection<? extends Main> collection ) {
    super( engine, owner, collection );
  }

  @Override
  public void create() {
    // Creating embedded object instances
    for ( int i = 0; i <
NoOfMachines
; i++ ) {
      instantiate_machines_xjal( i );
    }
    network = instantiate_network_xjal();
    // Dynamic initialization of persistent elements
    {
    DataSet _item;
    List<DataSet> _items = new ArrayList<DataSet>( 0 );
    List<String> _titles = new ArrayList<String>( 0 );
    machinesStatesCharts = new TimeColorChart(
                                Main.this, true, 10, 280,
                                        820, 270,
          null, null,
          40, 10,
                                        770, 240, whiteSmoke, null,
black,
          30, Chart.NONE,

1000
          , null, 0.8,
          Chart.GRID_DEFAULT,
          darkGray, darkGray, _items, _titles,
          new TimeColorChart.ColorMap() {
            @Override
            public Color colorFromDouble( double value ) {
              return _machinesStatesCharts_ColorFromDouble( value );
            }
          }
    );
    }
    {
                          group = new ShapeGroup( Main.this,
true,630, 140, 0.0

                  ,machines_presentation
                  ,line
    );
    }
    presentation = new ShapeGroup( Main.this, true, 0, 0, 0, Slider,
Slider1, Slider2, Slider3, Slider4, group, text, rectangle1, text1, text2,
rectangle2, rectangle3, text3, rectangle4, text4, text5, rectangle5,
rectangle6, text6, rectangle7, text7, rectangle8, text8, text9, rectangle9,
Text, Text1, Text2, Text3, Text4, Text5, Text6, Text7, Text8, Text9,
text15, line1, line2, Text10, rectangle, rectangle10, Text11,
machinesStatesCharts );
    icon = new ShapeGroup( Main.this, true, 0, 0, 0
);
    // Creating contents for replicated shapes
    line.createShapes();
    // Creating non-replicated embedded objects
    setupParameters_network_xjal( network );
    create_network_xjal( network );
                // Port connectors with non-replicated objects
    // Creating replicated embedded objects
    for ( int i = 0; i < machines.size(); i++ ) {
      setupParameters_machines_xjal( machines.get(i), i );
      create_machines_xjal( machines.get(i), i );
    }
    assignInitialConditions();
    Slider.setValueToDefault();
    Slider1.setValueToDefault();
    Slider2.setValueToDefault();
    Slider3.setValueToDefault();
    Slider4.setValueToDefault();
    onCreate();
  }

  @Override
  public void start() {
    _machinesStatesCharts_autoUpdateEvent_xjal.start();
    for (ActiveObject embeddedObject : machines){
      embeddedObject.start();
    }
    network.start();
    onStartup();
  }

  public void onStartup() {
    super.onStartup();

for ( Machine m : machines ){
            machinesStatesCharts.addDataSet( m.StatechartStateDS );
}
  }

  public List<Object> getEmbeddedObjects() {
    LinkedList<Object> list = new LinkedList<Object>();
    list.add( machines );
    list.add( network );
    return list;
  }

  public void onDestroy() {
    super.onDestroy();
    _machinesStatesCharts_autoUpdateEvent_xjal.onDestroy();
    for (ActiveObject embeddedObject : machines) {
      embeddedObject.onDestroy();
    }
    network.onDestroy();
  }


}
```

Machine.java

```
package leader_election_protocol;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import java.awt.geom.Arc2D;

public class Machine extends ActiveObject
{
 // Parameters

 public double StartUpTimeout;

 /**
  * Returns default value for parameter
<code>StartUpTimeout</code>.
  * <i>This method should not be called by user</i>
  */
 public double _StartUpTimeout_DefaultValue_xjal() {
  return 5.0 ;
 }

 public void set_StartUpTimeout(
double StartUpTimeout ) {
  if (StartUpTimeout == this.StartUpTimeout) {
   return;
  }
  this.StartUpTimeout = StartUpTimeout;
  onChange_StartUpTimeout();
  onChange();
 }

 void onChange_StartUpTimeout() {
 }


 public double ConsistencyTimeout;

 /**
  * Returns default value for parameter
<code>ConsistencyTimeout</code>.
  * <i>This method should not be called by user</i>
  */
 public double _ConsistencyTimeout_DefaultValue_xjal() {
  return 5.0 ;
 }

 public void set_ConsistencyTimeout(
double ConsistencyTimeout ) {
  if (ConsistencyTimeout == this.ConsistencyTimeout) {
   return;
  }
  this.ConsistencyTimeout = ConsistencyTimeout;
  onChange_ConsistencyTimeout();
  onChange();
 }

 void onChange_ConsistencyTimeout() {
 }


 public double AcceptTimeout;

 /**
  * Returns default value for parameter <code>AcceptTimeout</code>.
  * <i>This method should not be called by user</i>
  */
 public double _AcceptTimeout_DefaultValue_xjal() {
  return 5.0 ;
 }

 public void set_AcceptTimeout(
double AcceptTimeout ) {
  if (AcceptTimeout == this.AcceptTimeout) {
   return;
  }
  this.AcceptTimeout = AcceptTimeout;
  onChange_AcceptTimeout();
  onChange();
 }

 void onChange_AcceptTimeout() {
 }

 public double CandidateTimeout;

 /**
```

```
  * Returns default value for parameter
<code>CandidateTimeout</code>.
  * <i>This method should not be called by user</i>
  */
  public double _CandidateTimeout_DefaultValue_xjal() {
    return 5.0 ;
  }

  public void set_CandidateTimeout(
double CandidateTimeout ) {
    if (CandidateTimeout == this.CandidateTimeout) {
      return;
    }
    this.CandidateTimeout = CandidateTimeout;
    onChange_CandidateTimeout();
    onChange();
  }

  void onChange_CandidateTimeout() {
  }


  public double ElectionTimeoutHigh;

  /**
  * Returns default value for parameter
<code>ElectionTimeoutHigh</code>.
  * <i>This method should not be called by user</i>
  */
  public double _ElectionTimeoutHigh_DefaultValue_xjal() {
    return 100.0 ;
  }

  public void set_ElectionTimeoutHigh(
double ElectionTimeoutHigh ) {
    if (ElectionTimeoutHigh == this.ElectionTimeoutHigh) {
      return;
    }
    this.ElectionTimeoutHigh = ElectionTimeoutHigh;
    onChange_ElectionTimeoutHigh();
    onChange();
  }

  void onChange_ElectionTimeoutHigh() {
  }


  public double ConflictTimeout;

  /**
  * Returns default value for parameter
<code>ConflictTimeout</code>.
  * <i>This method should not be called by user</i>
  */
  public double _ConflictTimeout_DefaultValue_xjal() {
    return 5.0 ;
  }

  public void set_ConflictTimeout(
double ConflictTimeout ) {
    if (ConflictTimeout == this.ConflictTimeout) {
      return;
    }
    this.ConflictTimeout = ConflictTimeout;
    onChange_ConflictTimeout();
    onChange();
  }

  void onChange_ConflictTimeout() {
  }

  public double SynchroTimeout;

  /**
  * Returns default value for parameter
<code>SynchroTimeout</code>.
  * <i>This method should not be called by user</i>
  */
  public double _SynchroTimeout_DefaultValue_xjal() {
    return 3.0 ;
  }

  public void set_SynchroTimeout(
double SynchroTimeout ) {
    if (SynchroTimeout == this.SynchroTimeout) {
      return;
    }
    this.SynchroTimeout = SynchroTimeout;
    onChange_SynchroTimeout();
    onChange();
  }

  void onChange_SynchroTimeout() {
  }


  public double ResolveTimeoutHigh;

  /**
  * Returns default value for parameter
<code>ResolveTimeoutHigh</code>.
  * <i>This method should not be called by user</i>
  */
  public double _ResolveTimeoutHigh_DefaultValue_xjal() {
    return 35.0 ;
  }

  public void set_ResolveTimeoutHigh(
double ResolveTimeoutHigh ) {
    if (ResolveTimeoutHigh == this.ResolveTimeoutHigh) {
      return;
    }
    this.ResolveTimeoutHigh = ResolveTimeoutHigh;
    onChange_ResolveTimeoutHigh();
    onChange();
  }

  void onChange_ResolveTimeoutHigh() {
  }


  public double ResolveTimeoutLow;

  /**
  * Returns default value for parameter
<code>ResolveTimeoutLow</code>.
  * <i>This method should not be called by user</i>
  */
  public double _ResolveTimeoutLow_DefaultValue_xjal() {
    return 20.0 ;
  }
```

```
  public void set_ResolveTimeoutLow(
double  ResolveTimeoutLow ) {
    if (ResolveTimeoutLow == this.ResolveTimeoutLow) {
      return;
    }
    this.ResolveTimeoutLow = ResolveTimeoutLow;
    onChange_ResolveTimeoutLow();
    onChange();
  }

  void onChange_ResolveTimeoutLow() {
  }


  public double  ElectionTimeoutLow;

  /**
   *        Returns        default        value        for        parameter
<code>ElectionTimeoutLow</code>.
   * <i>This method should not be called by user</i>
   */
  public double  _ElectionTimeoutLow_DefaultValue_xjal() {
    return 5.0 ;
  }

  public void set_ElectionTimeoutLow(
double  ElectionTimeoutLow ) {
    if (ElectionTimeoutLow == this.ElectionTimeoutLow) {
      return;
    }
    this.ElectionTimeoutLow = ElectionTimeoutLow;
    onChange_ElectionTimeoutLow();
    onChange();
  }

  void onChange_ElectionTimeoutLow() {
  }


  public double  MTTR;

  /**
   * Returns default value for parameter <code>MTTR</code>.
   * <i>This method should not be called by user</i>
   */
  public double  _MTTR_DefaultValue_xjal() {
    return 100 ;
  }

  public void set_MTTR(
double  MTTR ) {
    if (MTTR == this.MTTR) {
      return;
    }
    this.MTTR = MTTR;
    onChange_MTTR();
    onChange();
  }

  void onChange_MTTR() {
  }

  public double  NoMasterTimeout;

  /**
   *        Returns        default        value        for        parameter
<code>NoMasterTimeout</code>.
   * <i>This method should not be called by user</i>
   */
  public double  _NoMasterTimeout_DefaultValue_xjal() {
    return 10.0 ;
  }

  public void set_NoMasterTimeout(
double  NoMasterTimeout ) {
    if (NoMasterTimeout == this.NoMasterTimeout) {
      return;
    }
    this.NoMasterTimeout = NoMasterTimeout;
    onChange_NoMasterTimeout();
    onChange();
  }

  void onChange_NoMasterTimeout() {
  }

  public double  MTTF;

  /**
   * Returns default value for parameter <code>MTTF</code>.
   * <i>This method should not be called by user</i>
   */
  public double  _MTTF_DefaultValue_xjal() {
    return 1000 ;
  }

  public void set_MTTF(
double  MTTF ) {
    if (MTTF == this.MTTF) {
      return;
    }
    this.MTTF = MTTF;
    onChange_MTTF();
    onChange();
  }

  void onChange_MTTF() {
  }

  public int  NoOfMachines;

  /**
   * Returns default value for parameter <code>NoOfMachines</code>.
   * <i>This method should not be called by user</i>
   */
  public int  _NoOfMachines_DefaultValue_xjal() {
    return 0;
  }

  public void set_NoOfMachines(
int  NoOfMachines ) {
    if (NoOfMachines == this.NoOfMachines) {
      return;
    }
    this.NoOfMachines = NoOfMachines;
    onChange_NoOfMachines();
    onChange();
  }

  void onChange_NoOfMachines() {
  }
```

```
 public int radius;

 /**
  * Returns default value for parameter <code>radius</code>.
  * <i>This method should not be called by user</i>
  */
 public int _radius_DefaultValue_xjal() {
  return 100 ;
 }

 public void set_radius(
int radius ) {
   if (radius == this.radius) {
     return;
   }
   this.radius = radius;
   onChange_radius();
   onChange();
 }

 void onChange_radius() {
 }

 // Plain Variables

 public Machine MyMaster;
 public Main main;
 public double x;
 public double y;

 // Collection Variables

 // Flow/Auxiliary Variables

 // Stock Variables

 // Events

 public EventTimeout SynchroTimer = new EventTimeout(this);

 @Override
 public String getNameOf( EventTimeout _e ) {
   if( _e == SynchroTimer ) return "SynchroTimer";
   return super.getNameOf( _e );
 }

 @Override
 public int getModeOf( EventTimeout _e ) {
  if ( _e == SynchroTimer ) return EVENT_TIMEOUT_MODE_CYCLIC;
  return super.getModeOf( _e );
 }

 @Override
 public double getFirstOccurrenceTime( EventTimeout _e ) {
  if ( _e == SynchroTimer ) return
0
;
   return super.getFirstOccurrenceTime( _e );
 }

 @Override
 public double evaluateTimeoutOf( EventTimeout _e ) {
  if( _e == SynchroTimer) return SynchroTimeout ;
  return super.evaluateTimeoutOf( _e );
 }

 @Override
 public void executeActionOf( EventTimeout _e ) {
  if ( _e == SynchroTimer ) {
broadcast( SYNCHROTICK ); ;
    return ;
  }
  super.executeActionOf( _e );
 }

 // Statecharts
 public Statechart protocol = new Statechart( this, (short)7 );

 @Override
 public String getNameOf( Statechart _s ) {
  if(_s == this.protocol) return "protocol";
  return super.getNameOf( _s );
 }

 @Override
 public void executeActionOf( Statechart _s ) {
  if( _s == this.protocol ) {
   enterState( Crashed, true );
   return;
  }
  super.executeActionOf( _s );
 }

 // States of all statecharts

 public static final short Crashed = 0;
 public static final short Active = 1;
 public static final short StartUp = 2;
 public static final short Consistency = 3;
 public static final short Slave = 4;
 public static final short Candidate = 5;
 public static final short Master = 6;
 public static final short Conflict = 7;
 public static final short Accept = 8;
 public static final short NoMaster = 9;


 @Override
 public String getNameOfState( short _state ) {
  switch( _state ) {
   case Crashed: return "Crashed";
   case Active: return "Active";
   case StartUp: return "StartUp";
   case Consistency: return "Consistency";
   case Slave: return "Slave";
   case Candidate: return "Candidate";
   case Master: return "Master";
   case Conflict: return "Conflict";
   case Accept: return "Accept";
   case NoMaster: return "NoMaster";
   default: return super.getNameOfState( _state );
  }
 }

 @Override
 public boolean stateContainsState( short compstate, short simpstate ) {
   if (compstate == Active && (simpstate == NoMaster || simpstate ==
Candidate || simpstate == Slave || simpstate == Accept || simpstate ==
Consistency || simpstate == StartUp || simpstate == Master ||
simpstate == Conflict)) {
     return true;
```

```
    }
    return super.stateContainsState( compstate, simpstate );
  }

  @Override
  public short getContainerStateOf( short _state ) {
    switch( _state ) {
      case StartUp: return Active;
      case Consistency: return Active;
      case Slave: return Active;
      case Candidate: return Active;
      case Master: return Active;
      case Conflict: return Active;
      case Accept: return Active;
      case NoMaster: return Active;
      default: return super.getContainerStateOf( _state );
    }
  }

  @Override
  public void enterState( short _state, boolean _destination ) {
    switch( _state ) {
      case Crashed: // (Simple state (not composite))
        protocol.setActiveState( Crashed );
        {
SynchroTimer.reset();
MyMaster = null;
StatechartStateDS.add( time(), Crashed );
;}
        Recovery.start();
        On.start();
        return;
      case Active: // (Composite state)
        Off.start();
        Failure.start();
        if ( _destination ) {
          enterState( StartUp, true );
        }
        return;
      case StartUp: // (Simple state (not composite))
        protocol.setActiveState( StartUp );
        {
broadcast( MASTERREQ );
StatechartStateDS.add( time(), StartUp );
;}
        MASTERACK_2.start();
        StartUpTimeout1.start();
        return;
      case Consistency: // (Simple state (not composite))
        protocol.setActiveState( Consistency );
        {
StatechartStateDS.add( time(), Consistency );
;}
        MASTERACK_1.start();
        ConsistencyTimeout1.start();
        return;
      case Slave: // (Simple state (not composite))
        protocol.setActiveState( Slave );
        {
StatechartStateDS.add( time(), Slave );
;}
        ElectionTimeout.start();
        MASTERUP_1.start();
        SYNCHROTICK_.start();
        ELECTION_3.start();
        return;
      case Candidate: // (Simple state (not composite))
        protocol.setActiveState( Candidate );
        {
StatechartStateDS.add( time(), Candidate );
;}
        ACCEPT_.start();
        QUIT_REFUSE_.start();
        ELECTION_1.start();
        CandidateTimeout1.start();
        return;
      case Master: // (Simple state (not composite))
        protocol.setActiveState( Master );
        {
MyMaster = Machine.this;
StatechartStateDS.add( time(), Master );
;}
        CONFLICT_.start();
        QUIT_.start();
        ResolveTimeout.start();
        ELECTION_2.start();
        RESOLVE_MASTERREQ_.start();
        return;
      case Conflict: // (Simple state (not composite))
        protocol.setActiveState( Conflict );
        {
broadcast( RESOLVE );
StatechartStateDS.add( time(), Conflict );
;}
        MASTERACK_.start();
        ConflictTimeout1.start();
        return;
      case Accept: // (Simple state (not composite))
        protocol.setActiveState( Accept );
        {
StatechartStateDS.add( time(), Accept );
;}
        MASTERUP_.start();
        AcceptTimeout1.start();
        ELECTION_.start();
        return;
      case NoMaster: // (Simple state (not composite))
        protocol.setActiveState( NoMaster );
        {
StatechartStateDS.add( time(), NoMaster );
;}
        MASTERUP_MASTERACK_ELECTION_.start();
        NoMasterTimeout1.start();
        return;
      default:
        super.enterState( _state, _destination );
        return;
    }
  }

  @Override
  public void exitState( short _state, Transition _t, boolean _source,
Statechart _statechart) {
    switch( _state ) {
      case Crashed: // (Simple state (not composite))
        if ( !_source || _t != Recovery) Recovery.cancel();
        if ( !_source || _t != On) On.cancel();
        return;
      case Active: // (Composite state)
        if ( _source ) exitInnerStates(_state, _statechart);
        if ( !_source || _t != Off ) Off.cancel();
```

```
      if ( !_source || _t != Failure ) Failure.cancel();
      return;
    case StartUp: // (Simple state (not composite))
      if ( !_source || _t != MASTERACK_2) MASTERACK_2.cancel();
      if ( !_source || _t != StartUpTimeout1) StartUpTimeout1.cancel();
      return;
    case Consistency: // (Simple state (not composite))
      if ( !_source || _t != MASTERACK_1) MASTERACK_1.cancel();
      if ( !_source || _t != ConsistencyTimeout1) ConsistencyTimeout1.cancel();
      return;
    case Slave: // (Simple state (not composite))
      if ( !_source || _t != ElectionTimeout) ElectionTimeout.cancel();
      if ( !_source || _t != MASTERUP_1) MASTERUP_1.cancel();
      if ( !_source || _t != SYNCHROTICK_) SYNCHROTICK_.cancel();
      if ( !_source || _t != ELECTION_3) ELECTION_3.cancel();
      return;
    case Candidate: // (Simple state (not composite))
      if ( !_source || _t != ACCEPT_) ACCEPT_.cancel();
      if ( !_source || _t != QUIT_REFUSE_) QUIT_REFUSE_.cancel();
      if ( !_source || _t != ELECTION_1) ELECTION_1.cancel();
      if ( !_source || _t != CandidateTimeout1) CandidateTimeout1.cancel();
      return;
    case Master: // (Simple state (not composite))
      if ( !_source || _t != CONFLICT_) CONFLICT_.cancel();
      if ( !_source || _t != QUIT_) QUIT_.cancel();
      if ( !_source || _t != ResolveTimeout) ResolveTimeout.cancel();
      if ( !_source || _t != ELECTION_2) ELECTION_2.cancel();
      if ( !_source || _t != RESOLVE_MASTERREQ_) RESOLVE_MASTERREQ_.cancel();
      return;
    case Conflict: // (Simple state (not composite))
      if ( !_source || _t != MASTERACK_) MASTERACK_.cancel();
      if ( !_source || _t != ConflictTimeout1) ConflictTimeout1.cancel();
      return;
    case Accept: // (Simple state (not composite))
      if ( !_source || _t != MASTERUP_) MASTERUP_.cancel();
      if ( !_source || _t != AcceptTimeout1) AcceptTimeout1.cancel();
      if ( !_source || _t != ELECTION_) ELECTION_.cancel();
      return;
    case NoMaster: // (Simple state (not composite))
      if ( !_source || _t != MASTERUP_MASTERACK_ELECTION_) MASTERUP_MASTERACK_ELECTION_.cancel();
      if ( !_source || _t != NoMasterTimeout1) NoMasterTimeout1.cancel();
      return;
    default:
      super.exitState( _state, _t, _source, _statechart);
      return;
   }
  }

   public TransitionTimeout ElectionTimeout = new TransitionTimeout( this );
  public TransitionTimeout CandidateTimeout1 = new TransitionTimeout( this );
  public TransitionTimeout ConflictTimeout1 = new TransitionTimeout( this );
  public TransitionTimeout ResolveTimeout = new TransitionTimeout( this );
  public TransitionTimeout AcceptTimeout1 = new TransitionTimeout( this );
  public TransitionTimeout ConsistencyTimeout1 = new TransitionTimeout( this );
  public TransitionTimeout StartUpTimeout1 = new TransitionTimeout( this );
  public TransitionTimeout NoMasterTimeout1 = new TransitionTimeout( this );

  @Override
  public String getNameOf( TransitionTimeout _t ) {
   if ( _t == ElectionTimeout ) return "ElectionTimeout";
   if ( _t == CandidateTimeout1 ) return "CandidateTimeout1";
   if ( _t == ConflictTimeout1 ) return "ConflictTimeout1";
   if ( _t == ResolveTimeout ) return "ResolveTimeout";
   if ( _t == AcceptTimeout1 ) return "AcceptTimeout1";
   if ( _t == ConsistencyTimeout1 ) return "ConsistencyTimeout1";
   if ( _t == StartUpTimeout1 ) return "StartUpTimeout1";
   if ( _t == NoMasterTimeout1 ) return "NoMasterTimeout1";
   return super.getNameOf( _t );
  }

  @Override
  public Statechart getStatechartOf( TransitionTimeout _t ) {
   if ( _t == ElectionTimeout ) return protocol;
   if ( _t == CandidateTimeout1 ) return protocol;
   if ( _t == ConflictTimeout1 ) return protocol;
   if ( _t == ResolveTimeout ) return protocol;
   if ( _t == AcceptTimeout1 ) return protocol;
   if ( _t == ConsistencyTimeout1 ) return protocol;
   if ( _t == StartUpTimeout1 ) return protocol;
   if ( _t == NoMasterTimeout1 ) return protocol;
   return super.getStatechartOf( _t );
  }

  @Override
  public void executeActionOf( TransitionTimeout _t ) {
   if ( _t == ElectionTimeout ) {
    exitState( Slave, _t, true, protocol );
    {
broadcast( ELECTION );
MyMaster = Machine.this;

;}
       enterState( Candidate, true );
    return;
   }
   if ( _t == CandidateTimeout1 ) {
    exitState( Candidate, _t, true, protocol );
    {
broadcast( MASTERUP );
SynchroTimer.restart( SynchroTimeout );

;}
       enterState( Master, true );
    return;
   }
   if ( _t == ConflictTimeout1 ) {
    exitState( Conflict, _t, true, protocol );
    {
broadcast( MASTERUP );

;}
       enterState( Master, true );
    return;
   }
   if ( _t == ResolveTimeout ) {
    exitState( Master, _t, true, protocol );
       enterState( Conflict, true );
    return;
   }
   if ( _t == AcceptTimeout1 ) {
    exitState( Accept, _t, true, protocol );
       enterState( Slave, true );
```

```
    return;
   }
   if ( _t == ConsistencyTimeout1 ) {
    exitState( Consistency, _t, true, protocol );
       enterState( Slave, true );
    return;
   }
   if ( _t == StartUpTimeout1 ) {
    exitState( StartUp, _t, true, protocol );
       enterState( NoMaster, true );
    return;
   }
   if ( _t == NoMasterTimeout1 ) {
    exitState( NoMaster, _t, true, protocol );
    {
broadcast( MASTERUP );
SynchroTimer.restart( SynchroTimeout );

;}
       enterState( Master, true );
    return;
   }
   super.executeActionOf( _t );
  }

  @Override
  public double evaluateTimeoutOf( TransitionTimeout _t ) {
   if ( _t == ElectionTimeout ) return
uniform( ElectionTimeoutLow, ElectionTimeoutHigh ) ;
   if ( _t == CandidateTimeout1 ) return CandidateTimeout ;
   if ( _t == ConflictTimeout1 ) return ConflictTimeout ;
   if ( _t == ResolveTimeout ) return
uniform( ResolveTimeoutLow, ResolveTimeoutHigh ) ;
   if ( _t == AcceptTimeout1 ) return AcceptTimeout ;
   if ( _t == ConsistencyTimeout1 ) return ConsistencyTimeout ;
   if ( _t == StartUpTimeout1 ) return StartUpTimeout ;
   if ( _t == NoMasterTimeout1 ) return NoMasterTimeout ;
   return super.evaluateTimeoutOf( _t );
  }


  public TransitionRate Recovery = new TransitionRate( this );
  public TransitionRate Failure = new TransitionRate( this );

  @Override
  public String getNameOf( TransitionRate _t ) {
   if ( _t == Recovery ) return "Recovery";
   if ( _t == Failure ) return "Failure";
   return super.getNameOf( _t );
  }

  @Override
  public Statechart getStatechartOf( TransitionRate _t ) {
   if ( _t == Recovery ) return protocol;
   if ( _t == Failure ) return protocol;
   return super.getStatechartOf( _t );
  }

  @Override
  public void executeActionOf( TransitionRate _t ) {
   if ( _t == Recovery ) {
    exitState( Crashed, _t, true, protocol );
       enterState( Active, true );
    return;
   }
   if ( _t == Failure ) {
    exitState( Active, _t, true, protocol );
       enterState( Crashed, true );
    return;
   }
   super.executeActionOf( _t );
  }
  @Override
  public double evaluateRateOf( TransitionRate _t ) {
   if ( _t == Recovery ) return 1/MTTR ;
   if ( _t == Failure ) return 1/MTTF ;
   return super.evaluateRateOf( _t );
  }


  public TransitionMessage Off = new TransitionMessage( this );
  public TransitionMessage MASTERACK_2 = new TransitionMessage( this
);
  public TransitionMessage MASTERACK_1 = new TransitionMessage( this
);
  public TransitionMessage ACCEPT_ = new TransitionMessage( this );
  public TransitionMessage QUIT_REFUSE_ = new TransitionMessage(
this );
  public TransitionMessage ELECTION_1 = new TransitionMessage( this
);
  public TransitionMessage CONFLICT_ = new TransitionMessage( this );
  public TransitionMessage MASTERACK_ = new TransitionMessage( this
);
  public TransitionMessage QUIT_ = new TransitionMessage( this );
  public TransitionMessage ELECTION_2 = new TransitionMessage( this
);
  public TransitionMessage RESOLVE_MASTERREQ_ = new
TransitionMessage( this );
  public TransitionMessage MASTERUP_1 = new TransitionMessage( this
);
  public TransitionMessage SYNCHROTICK_ = new TransitionMessage(
this );
  public TransitionMessage ELECTION_3 = new TransitionMessage( this
);
  public TransitionMessage MASTERUP_ = new TransitionMessage( this );
  public TransitionMessage ELECTION_ = new TransitionMessage( this );
  public TransitionMessage MASTERUP_MASTERACK_ELECTION_ = new
TransitionMessage( this );
  public TransitionMessage On = new TransitionMessage( this );


  @Override
  public String getNameOf( TransitionMessage _t ) {
   if ( _t == Off ) return "Off";
   if ( _t == MASTERACK_2 ) return "MASTERACK_2";
   if ( _t == MASTERACK_1 ) return "MASTERACK_1";
   if ( _t == ACCEPT_ ) return "ACCEPT_";
   if ( _t == QUIT_REFUSE_ ) return "QUIT_REFUSE_";
   if ( _t == ELECTION_1 ) return "ELECTION_1";
   if ( _t == CONFLICT_ ) return "CONFLICT_";
   if ( _t == MASTERACK_ ) return "MASTERACK_";
   if ( _t == QUIT_ ) return "QUIT_";
   if ( _t == ELECTION_2 ) return "ELECTION_2";
   if ( _t == RESOLVE_MASTERREQ_ ) return
"RESOLVE_MASTERREQ_";
   if ( _t == MASTERUP_1 ) return "MASTERUP_1";
   if ( _t == SYNCHROTICK_ ) return "SYNCHROTICK_";
   if ( _t == ELECTION_3 ) return "ELECTION_3";
   if ( _t == MASTERUP_ ) return "MASTERUP_";
   if ( _t == ELECTION_ ) return "ELECTION_";
   if ( _t == MASTERUP_MASTERACK_ELECTION_ ) return
"MASTERUP_MASTERACK_ELECTION_";
   if ( _t == On ) return "On";
   return super.getNameOf( _t );
  }
```

```
 @Override
 public Statechart getStatechartOf( TransitionMessage _t ) {
  if ( _t == Off ) return protocol;
  if ( _t == MASTERACK_2 ) return protocol;
  if ( _t == MASTERACK_1 ) return protocol;
  if ( _t == ACCEPT_ ) return protocol;
  if ( _t == QUIT_REFUSE_ ) return protocol;
  if ( _t == ELECTION_1 ) return protocol;
  if ( _t == CONFLICT_ ) return protocol;
  if ( _t == MASTERACK_ ) return protocol;
  if ( _t == QUIT_ ) return protocol;
  if ( _t == ELECTION_2 ) return protocol;
  if ( _t == RESOLVE_MASTERREQ_ ) return protocol;
  if ( _t == MASTERUP_1 ) return protocol;
  if ( _t == SYNCHROTICK_ ) return protocol;
  if ( _t == ELECTION_3 ) return protocol;
  if ( _t == MASTERUP_ ) return protocol;
  if ( _t == ELECTION_ ) return protocol;
  if ( _t == MASTERUP_MASTERACK_ELECTION_ ) return protocol;
  if ( _t == On ) return protocol;
  return super.getStatechartOf( _t );
 }

 @Override
 public void executeActionOf( TransitionMessage _t, Object _msg ) {
  if ( _t == Off ) {
   exitState( Active, _t, true, protocol );
      enterState( Crashed, true );
   return;
  }
  if ( _t == MASTERACK_2 ) {
   exitState( StartUp, _t, true, protocol );
   {
    Command msg = (Command) _msg;
MyMaster = ((Command)msg).source;

;}
      enterState( Consistency, true );
   return;
  }
  if ( _t == MASTERACK_1 ) {
   exitState( Consistency, _t, true, protocol );
   {
    Command msg = (Command) _msg;
if( ((Command)msg).source != MyMaster ){
 sendTo( CONFLICT, MyMaster );
}
MyMaster = ((Command)msg).source;
;}
      enterState( Slave, true );
   return;
  }
  if ( _t == ACCEPT_ ) {
   exitState( Candidate, _t, true, protocol );
      enterState( Candidate, true );
   return;
  }
  if ( _t == QUIT_REFUSE_ ) {
   exitState( Candidate, _t, true, protocol );
   {
    Command msg = (Command) _msg;
MyMaster = ((Command)msg).source;
;}
      enterState( Slave, true );
   return;
  }
  if ( _t == ELECTION_1 ) {
   exitState( Candidate, _t, true, protocol );
   {
    Command msg = (Command) _msg;
port.send( new Command( Machine.this, ((Command)msg).source,
REFUSE ) );
;}
      enterState( Candidate, true );
   return;
  }
  if ( _t == CONFLICT_ ) {
   exitState( Master, _t, true, protocol );
      enterState( Conflict, true );
   return;
  }
  if ( _t == MASTERACK_ ) {
   exitState( Conflict, _t, true, protocol );
   {
    Command msg = (Command) _msg;
port.send( new Command( Machine.this, ((Command)msg).source, QUIT
) );
;}
      enterState( Conflict, true );
   return;
  }
  if ( _t == QUIT_ ) {
   exitState( Master, _t, true, protocol );
   {
    Command msg = (Command) _msg;
SynchroTimer.reset();
MyMaster = ((Command)msg).source;
;}
      enterState( Slave, true );
   return;
  }
  if ( _t == ELECTION_2 ) {
   exitState( Master, _t, true, protocol );
   {
    Command msg = (Command) _msg;
port.send( new Command( Machine.this, ((Command)msg).source, QUIT
) );
;}
      enterState( Master, true );
   return;
  }
  if ( _t == RESOLVE_MASTERREQ_ ) {
   exitState( Master, _t, true, protocol );
   {
    Command msg = (Command) _msg;
broadcast( MASTERACK );
;}
      enterState( Master, true );
   return;
  }
  if ( _t == MASTERUP_1 ) {
   exitState( Slave, _t, true, protocol );
   {
    Command msg = (Command) _msg;
port.send( new Command( Machine.this, ((Command)msg).source,
SLAVEUP ) );
MyMaster = ((Command)msg).source;
;}
      enterState( Slave, true );
   return;
  }
  if ( _t == SYNCHROTICK_ ) {
```

```
    exitState( Slave, _t, true, protocol );
    {
     Command msg = (Command) _msg;
MyMaster = ((Command)msg).source;
;}
      enterState( Slave, true );
    return;
   }
   if ( _t == ELECTION_3 ) {
    exitState( Slave, _t, true, protocol );
    {
     Command msg = (Command) _msg;
port.send( new Command( Machine.this, ((Command)msg).source,
ACCEPT ) );
MyMaster = ((Command)msg).source;

;}
      enterState( Accept, true );
    return;
   }
   if ( _t == MASTERUP_ ) {
    exitState( Accept, _t, true, protocol );
    {
     Command msg = (Command) _msg;
port.send( new Command( Machine.this, ((Command)msg).source,
SLAVEUP ) );
MyMaster = ((Command)msg).source;
;}
      enterState( Accept, true );
    return;
   }
   if ( _t == ELECTION_ ) {
    exitState( Accept, _t, true, protocol );
    {
     Command msg = (Command) _msg;
port.send( new Command( Machine.this, ((Command)msg).source,
REFUSE ) );
;}
      enterState( Accept, true );
    return;
   }
   if ( _t == MASTERUP_MASTERACK_ELECTION_ ) {
    exitState( NoMaster, _t, true, protocol );
    {
     Command msg = (Command) _msg;
MyMaster = ((Command)msg).source;
;}
      enterState( Slave, true );
    return;
   }
   if ( _t == On ) {
    exitState( Crashed, _t, true, protocol );
      enterState( Active, true );
    return;
   }
   super.executeActionOf( _t, _msg );
  }
  @Override
  public boolean testMessageOf( TransitionMessage _t, Object _msg ) {
   if ( _t == Off ) {
    if ( !(_msg instanceof String) )
     return false;
    String msg = (String) _msg;
    Object _g =
"CRASH"
;
    return msg.equals( _g );
   }
   if ( _t == MASTERACK_2 ) {
    if ( !(_msg instanceof Command) )
     return false;
    Command msg = (Command) _msg;
    boolean _g =
msg.id == MASTERACK
;
    return _g;
   }
   if ( _t == MASTERACK_1 ) {
    if ( !(_msg instanceof Command) )
     return false;
    Command msg = (Command) _msg;
    boolean _g =
msg.id == MASTERACK
;
    return _g;
   }
   if ( _t == ACCEPT_ ) {
    if ( !(_msg instanceof Command) )
     return false;
    Command msg = (Command) _msg;
    boolean _g =
msg.id == ACCEPT
;
    return _g;
   }
   if ( _t == QUIT_REFUSE_ ) {
    if ( !(_msg instanceof Command) )
     return false;
    Command msg = (Command) _msg;
    boolean _g =
msg.id == QUIT || msg.id == REFUSE
;
    return _g;
   }
   if ( _t == ELECTION_1 ) {
    if ( !(_msg instanceof Command) )
     return false;
    Command msg = (Command) _msg;
    boolean _g =
msg.id == ELECTION
;
    return _g;
   }
   if ( _t == CONFLICT_ ) {
    if ( !(_msg instanceof Command) )
     return false;
    Command msg = (Command) _msg;
    boolean _g =
msg.id == CONFLICT
;
    return _g;
   }
   if ( _t == MASTERACK_ ) {
    if ( !(_msg instanceof Command) )
     return false;
    Command msg = (Command) _msg;
    boolean _g =
msg.id == MASTERACK
;
    return _g;
   }
```

```
  if ( _t == QUIT_ ) {
   if ( !(_msg instanceof Command) )
    return false;
   Command msg = (Command) _msg;
   boolean _g = msg.id == QUIT ;
   return _g;
  }
  if ( _t == ELECTION_2 ) {
   if ( !(_msg instanceof Command) )
    return false;
   Command msg = (Command) _msg;
   boolean _g = msg.id == ELECTION ;
   return _g;
  }
  if ( _t == RESOLVE_MASTERREQ_ ) {
   if ( !(_msg instanceof Command) )
    return false;
   Command msg = (Command) _msg;
   boolean _g = msg.id == RESOLVE || msg.id == MASTERREQ ;
   return _g;
  }
  if ( _t == MASTERUP_1 ) {
   if ( !(_msg instanceof Command) )
    return false;
   Command msg = (Command) _msg;
   boolean _g = msg.id == MASTERUP ;
   return _g;
  }
  if ( _t == SYNCHROTICK_ ) {
   if ( !(_msg instanceof Command) )
    return false;
   Command msg = (Command) _msg;
   boolean _g = msg.id == SYNCHROTICK ;
   return _g;
  }
  if ( _t == ELECTION_3 ) {
   if ( !(_msg instanceof Command) )
    return false;
   Command msg = (Command) _msg;
   boolean _g = msg.id == ELECTION ;
   return _g;
  }
  if ( _t == MASTERUP_ ) {
   if ( !(_msg instanceof Command) )
    return false;
   Command msg = (Command) _msg;
   boolean _g = msg.id == MASTERUP ;
   return _g;
  }
  if ( _t == ELECTION_ ) {
   if ( !(_msg instanceof Command) )
    return false;
   Command msg = (Command) _msg;
   boolean _g = msg.id == ELECTION ;
   return _g;
  }
  if ( _t == MASTERUP_MASTERACK_ELECTION_ ) {
   if ( !(_msg instanceof Command) )
    return false;
   Command msg = (Command) _msg;
   boolean _g =
msg.id == MASTERUP || msg.id == MASTERACK || msg.id ==
ELECTION ;
   return _g;
  }
  if ( _t == On ) {
   if ( !(_msg instanceof String) )
    return false;
   String msg = (String) _msg;
   Object _g = "RECOVER" ;
   return msg.equals( _g );
  }
  return super.testMessageOf( _t, _msg );
 }
 // Functions


void broadcast( int id ) {
port.send( new Command( Machine.this, null, id ) );
 }

 void sendTo( int id, Machine destination ) {
port.send( new Command( Machine.this, destination, id ) );
 }


boolean linkExists( ) {
return protocol.isStateActive( Slave ) && MyMaster != null;
 }
 // Analysis Data Elements
 public DataSet StatechartStateDS = new DataSet( 2000 ) {
  double _lastUpdateX = Double.NaN;
  @Override
  public void update() {
   if ( time() == _lastUpdateX ) { return; }
   add( time(), _StatechartStateDS_YValue() );
   _lastUpdateX = time();
  }
 };


 /**
  * <i>This method should not be called by user</i>
  */
 private double _StatechartStateDS_YValue() {
  return protocol.getActiveSimpleState() ;
 }


 // View areas
 public ViewArea MachineView = new ViewArea( this, "MachineView", 0,
0, ViewArea.TOP_LEFT, ViewArea.NONE, 1.0, 100, 100 );


 static final Font _button1_Font = new Font("Dialog", 0, 11 );
 static final int _button1 = 1;
 static final int _image = 2;
 static final int _image1 = 3;
 static final int _group = 4;
 static final int _oval = 5;


 /**
  * Top-level presentation group id
  */
 static final int _presentation = 0;


 /**
  * Top-level icon group id
  */
 static final int _icon = -1;


 @Override
 public boolean onShapeClick( int _shape, int index, double clickx,
double clicky ){
  switch( _shape ){
```

```
    case _image:
      if (true) {

getEngine().getPresentation().setPresentable( this );
      }
      break;
    case _image1:
      if (true) {

MachineView.navigateTo();
      }
      break;
    case _oval:
      if (true) {

protocol.fireEvent( protocol.isStateActive( Crashed ) ? "RECOVER" :
"CRASH" );
      }
      break;
    default: return super.onShapeClick( _shape, index, clickx, clicky );
              }
              return false;
 }

 @Override
 public void executeShapeControlAction( int _shape, int index ) {
  switch( _shape ) {
    case _button1: {
get_Main().MainView.navigateTo();
;}
      break;
    default:
      super.executeShapeControlAction( _shape, index );
      break;
  }
 }

 ShapeButton button1;
 ShapeImage image;

 /**
  * <i>This method should not be called by user</i>
  */
 private void _image1_SetDynamicParams( ShapeImage shape ) {
  boolean _visible;
  _visible =
getPresentation().getPanel().getPresentable() != this
;
  shape.setVisible( _visible );
              if ( _visible ) {
  shape.setIndex(
protocol.isStateActive( Master ) ? 1 : ( protocol.isStateActive( Crashed )
? 2 : 0 )
);
              }
 }

 ShapeImage image1;

 /**
  * <i>This method should not be called by user</i>
  */
 private void _group_SetDynamicParams( ShapeGroup shape ) {
  boolean _visible;

  shape.setX(
x
);
  shape.setY(
y
);
 }

 ShapeGroup group;

 /**
  * <i>This method should not be called by user</i>
  */
 private void _oval_SetDynamicParams( ShapeOval shape ) {
  boolean _visible;
  shape.setX(
( radius + 25 ) * cos( 2*Math.PI / NoOfMachines * getIndex() )
);
  shape.setY(
( radius + 25 ) * sin( 2*Math.PI / NoOfMachines * getIndex() )
);
  shape.setFillColor(
protocol.isStateActive( Crashed ) ? silver : steelBlue
);
 }

 ShapeOval oval;

 // Static initialization of persistent elements
 {
  button1 = new ShapeButton(
                                    Machine.this, false, 20, 20,
                                    81, 20,
      controlDefault, black,
      _button1_Font,
                                    "Main view"
  ) {

   @Override
   public void action(){
    executeShapeControlAction( _button1, 0 );
   }
  };
  image = new ShapeImage(
                        Machine.this, true, 940, 70, 0.0,
                                    32,                 32,
"/leader_election_protocol/",
                                    new
String[]{"Computer.png","ComputerDown.png","ComputerMaster.png",}

  ) {

   @Override
   public boolean onClick( double clickx, double clicky ) {
    return onShapeClick( _image, 0, clickx, clicky );
   }
  };
  image1 = new ShapeImage(
                        Machine.this, true, -17, -17, 0.0,
                                    32,                 32,
"/leader_election_protocol/",
                                    new
String[]{"Computer.png","ComputerMaster.png","ComputerDown.png",}

  ) {

   @Override
```

```
    public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
      _image1_SetDynamicParams( this );
      super.draw( panel, graphics, xform, publicOnly );
    }

    @Override
    public boolean onClick( double clickx, double clicky ) {
      return onShapeClick( _image1, 0, clickx, clicky );
    }
   };
   oval = new ShapeOval(
     true,-25, 0, 0.0,
     null, red,
                5, 5,
                1, LINE_STYLE_SOLID
   ) {

    @Override
    public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
      _oval_SetDynamicParams( this );
      super.draw( panel, graphics, xform, publicOnly );
    }

    @Override
    public boolean onClick( double clickx, double clicky ) {
      return onShapeClick( _oval, 0, clickx, clicky );
    }
   };
 }
 ShapeGroup presentation;
 ShapeGroup icon;

 @Override
 public Object getPersistentShape( int _shape ) {
  switch(_shape){
    case _presentation: return presentation;
    case _icon: return icon;

    case _button1: return button1;
    case _image: return image;
    case _image1: return image1;
    case _group: return group;
    case _oval: return oval;
    default: return null;
  }
 }

 static final int[] _Recovery_pointsX = {500, 650, 650, 650, };
 static final int[] _Recovery_pointsY = {530, 530, 520, 440, };
 static final int[] _Off_pointsX = {490, 490, };
 static final int[] _Off_pointsY = {440, 510, };
 static final int[] _Failure_pointsX = {450, 450, };
 static final int[] _Failure_pointsY = {440, 510, };
 static final int[] _MASTERACK_2_pointsX = {440, 330, };
 static final int[] _MASTERACK_2_pointsY = {80, 80, };
 static final int[] _MASTERACK_1_pointsX = {270, 270, };
 static final int[] _MASTERACK_1_pointsY = {100, 210, };
 static final int[] _ElectionTimeout_pointsX = {330, 470, 470, };
 static final int[] _ElectionTimeout_pointsY = {230, 280, 340, };
 static final int[] _ACCEPT__pointsX = {440, 420, 420, 460, 460, };
 static final int[] _ACCEPT__pointsY = {360, 360, 400, 400, 370, };
 static final int[] _QUIT_REFUSE__pointsX = {450, 320, };
 static final int[] _QUIT_REFUSE__pointsY = {340, 240, };
 static final int[] _ELECTION_1_pointsX = {520, 540, 540, 500, 500, };
 static final int[] _ELECTION_1_pointsY = {360, 360, 400, 400, 370, };
 static final int[] _CandidateTimeout1_pointsX = {510, 632, };
 static final int[] _CandidateTimeout1_pointsY = {340, 238, };
 static final int[] _CONFLICT__pointsX = {640, 640, };
 static final int[] _CONFLICT__pointsY = {240, 340, };
 static final int[] _MASTERACK__pointsX = {710, 730, 730, 690, 690, };
 static final int[] _MASTERACK__pointsY = {360, 360, 400, 400, 370, };
 static final int[] _ConflictTimeout1_pointsX = {690, 690, };
 static final int[] _ConflictTimeout1_pointsY = {340, 240, };
 static final int[] _QUIT__pointsX = {630, 330, };
 static final int[] _QUIT__pointsY = {220, 220, };
 static final int[] _ResolveTimeout_pointsX = {660, 660, };
 static final int[] _ResolveTimeout_pointsY = {240, 340, };
 static final int[] _ELECTION_2_pointsX = {710, 740, 740, 700, 700, };
 static final int[] _ELECTION_2_pointsY = {220, 220, 180, 180, 210, };
 static final int[] _RESOLVE_MASTERREQ__pointsX = {710, 740, 740,
700, 700, };
 static final int[] _RESOLVE_MASTERREQ__pointsY = {230, 230, 270,
270, 240, };
 static final int[] _MASTERUP_1_pointsX = {240, 220, 220, 260, 260, };
 static final int[] _MASTERUP_1_pointsY = {220, 220, 180, 180, 210, };
 static final int[] _SYNCHROTICK__pointsX = {240, 220, 220, 260, 260,
};
 static final int[] _SYNCHROTICK__pointsY = {230, 230, 270, 270, 240,
};
 static final int[] _ELECTION_3_pointsX = {310, 310, };
 static final int[] _ELECTION_3_pointsY = {240, 340, };
 static final int[] _MASTERUP__pointsX = {330, 350, 350, 310, 310, };
 static final int[] _MASTERUP__pointsY = {360, 360, 400, 400, 370, };
 static final int[] _AcceptTimeout1_pointsX = {270, 270, };
 static final int[] _AcceptTimeout1_pointsY = {340, 240, };
 static final int[] _ELECTION__pointsX = {250, 230, 230, 270, 270, };
 static final int[] _ELECTION__pointsY = {360, 360, 400, 400, 370, };
 static final int[] _ConsistencyTimeout1_pointsX = {310, 310, };
 static final int[] _ConsistencyTimeout1_pointsY = {100, 210, };
 static final int[] _StartUpTimeout1_pointsX = {510, 630, };
 static final int[] _StartUpTimeout1_pointsY = {80, 80, };
 static final int[] _MASTERUP_MASTERACK_ELECTION__pointsX = {632,
328, };
 static final int[] _MASTERUP_MASTERACK_ELECTION__pointsY = {98,
212, };
 static final int[] _NoMasterTimeout1_pointsX = {660, 660, };
 static final int[] _NoMasterTimeout1_pointsY = {100, 210, };
 static final int[] _On_pointsX = {430, 300, 300, };
 static final int[] _On_pointsY = {530, 530, 440, };


 @Override
 public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
   if (!_publicOnly) {
     drawEvent( _panel, _g, 160, 510, 10, 0, "SynchroTimer",
SynchroTimer );
   }
   if (!_publicOnly) {
     drawState( _panel, _g, 120, 20, 700, 420, 10, 13, "Active", null,
Active, protocol );
   }
   if (!_publicOnly) {
     drawState( _panel, _g, 630, 70, 80, 30, 10, 13, "NoMaster", GOLD,
NoMaster, protocol );
   }
   if (!_publicOnly) {
     drawState( _panel, _g, 630, 210, 80, 30, 20, 13, "Master", GOLD,
Master, protocol );
   }
   if (!_publicOnly) {
```

```
    drawState( _panel, _g, 440, 340, 80, 30, 10, 10, "Candidate", GOLD, Candidate, protocol );
  }
  if (!_publicOnly) {
    drawState( _panel, _g, 440, 70, 70, 30, 10, 13, "StartUp", GOLD, StartUp, protocol );
  }
  if (!_publicOnly) {
    drawState( _panel, _g, 240, 210, 90, 30, 20, 13, "Slave", GOLD, Slave, protocol );
  }
  if (!_publicOnly) {
    drawState( _panel, _g, 240, 70, 90, 30, 10, 13, "Consistency", GOLD, Consistency, protocol );
  }
  if (!_publicOnly) {
    drawState( _panel, _g, 630, 340, 80, 30, 20, 13, "Conflict", GOLD, Conflict, protocol );
  }
  if (!_publicOnly) {
    drawState( _panel, _g, 430, 510, 70, 30, 6, 12, "Crashed", GOLD, Crashed, protocol );
  }
  if (!_publicOnly) {
    drawState( _panel, _g, 250, 340, 80, 30, 10, 13, "Accept", GOLD, Accept, protocol );
  }
  if (!_publicOnly) {
    drawInitialStatePointer( _panel, _g, 450, 40, 450, 70 );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _MASTERACK__pointsX, _MASTERACK__pointsY, 670, 410, "MASTERACK_", MASTERACK_ );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _MASTERACK_1_pointsX, _MASTERACK_1_pointsY, 180, 120, "MASTERACK_1", MASTERACK_1 );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _Off_pointsX, _Off_pointsY, 500, 480, "Off", Off );
  }
  if (!_publicOnly) {
    drawStatechartEntryPoint( _panel, _g, 470, 580, 470, 540, 410, 570, "protocol", protocol );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _ACCEPT__pointsX, _ACCEPT__pointsY, 410, 410, "ACCEPT_", ACCEPT_ );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _CONFLICT__pointsX, _CONFLICT__pointsY, 560, 320, "CONFLICT_", CONFLICT_ );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _ElectionTimeout_pointsX, _ElectionTimeout_pointsY, 400, 250, "ElectionTimeout", ElectionTimeout );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _QUIT__pointsX, _QUIT__pointsY, 530, 210, "QUIT_", QUIT_ );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _MASTERUP__pointsX, _MASTERUP__pointsY, 300, 410, "MASTERUP_", MASTERUP_ );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _AcceptTimeout1_pointsX, _AcceptTimeout1_pointsY, 170, 320, "AcceptTimeout1", AcceptTimeout1 );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _QUIT_REFUSE__pointsX, _QUIT_REFUSE__pointsY, 350, 310, "QUIT_REFUSE_", QUIT_REFUSE_ );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _ResolveTimeout_pointsX, _ResolveTimeout_pointsY, 650, 300, "ResolveTimeout", ResolveTimeout );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _MASTERUP_MASTERACK_ELECTION__pointsX, _MASTERUP_MASTERACK_ELECTION__pointsY, 380, 170, "MASTERUP_MASTERACK_ELECTION_", MASTERUP_MASTERACK_ELECTION_ );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _ELECTION__pointsX, _ELECTION__pointsY, 160, 380, "ELECTION_", ELECTION_ );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _ELECTION_1_pointsX, _ELECTION_1_pointsY, 500, 410, "ELECTION_1", ELECTION_1 );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _MASTERACK_2_pointsX, _MASTERACK_2_pointsY, 340, 70, "MASTERACK_2", MASTERACK_2 );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _ConsistencyTimeout1_pointsX, _ConsistencyTimeout1_pointsY, 320, 120, "ConsistencyTimeout1", ConsistencyTimeout1 );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _StartUpTimeout1_pointsX, _StartUpTimeout1_pointsY, 520, 70, "StartUpTimeout1", StartUpTimeout1 );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _ELECTION_2_pointsX, _ELECTION_2_pointsY, 730, 170, "ELECTION_2", ELECTION_2 );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _NoMasterTimeout1_pointsX, _NoMasterTimeout1_pointsY, 670, 120, "NoMasterTimeout1", NoMasterTimeout1 );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _MASTERUP_1_pointsX, _MASTERUP_1_pointsY, 140, 190, "MASTERUP_1", MASTERUP_1 );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _SYNCHROTICK__pointsX, _SYNCHROTICK__pointsY, 130, 270, "SYNCHROTICK_", SYNCHROTICK_ );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _Recovery_pointsX, _Recovery_pointsY, 540, 540, "Recovery", Recovery );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _ELECTION_3_pointsX, _ELECTION_3_pointsY, 280, 290, "ELECTION_3", ELECTION_3 );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _Failure_pointsX, _Failure_pointsY, 400, 480, "Failure", Failure );
  }
  if (!_publicOnly) {
    drawTransition(_panel, _g, _On_pointsX, _On_pointsY, 380, 540, "On", On );
```

```
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _RESOLVE_MASTERREQ__pointsX,
_RESOLVE_MASTERREQ__pointsY, 700, 280, "RESOLVE_MASTERREQ_",
RESOLVE_MASTERREQ_ );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _CandidateTimeout1_pointsX,
_CandidateTimeout1_pointsY, 490, 280, "CandidateTimeout1",
CandidateTimeout1 );
    }
    if (!_publicOnly) {
      drawTransition(_panel, _g, _ConflictTimeout1_pointsX,
_ConflictTimeout1_pointsY, 680, 320, "ConflictTimeout1",
ConflictTimeout1 );
    }
  }

  @Override
  public boolean onClickModelAt( Panel panel, double x, double y, int
clickCount, boolean publicOnly ) {
    if( !publicOnly && modelElementContains(x, y, 160, 510) ) {
      panel.addInspect( 160, 510, this, "SynchroTimer" );
      return true;
    }
    return false;
  }

  // Ports

  public Port< Object , Object > port = new Port< Object, Object >( this
);

  public String getNameOf( Port<?, ?> _p ) {
    if( _p == this.port ) return "port";
    return null;
  }

  public boolean executeOnReceiveActionOf( Port<?, ?> _p, Object _msg
) {
    if( _p == this.port ) {
      Object msg = (Object) _msg;

protocol.fireEvent( msg );
    }
    return true;
  }

  /**
   * Constructor
   */
  public Machine( Engine engine, ActiveObject owner,
ActiveObjectCollection<? extends Machine> collection ) {
    super( engine, owner, collection );
  }

  @Override
  public void create() {
    // Assigning initial values for plain variables
    main =
(Main)getOwner()
;
    x =
radius * cos( 2*Math.PI / NoOfMachines * getIndex() )
;
    y =
radius * sin( 2*Math.PI / NoOfMachines * getIndex() )
;
    // Dynamic initialization of persistent elements
    {
                        group = new ShapeGroup( Machine.this,
true,0, 0, 0.0

                  ,image1
    ) {

      @Override
      public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
        _group_SetDynamicParams( this );
        super.draw( panel, graphics, xform, publicOnly );
      }
    };
    }
    presentation = new ShapeGroup( Machine.this, true, 0, 0, 0, button1,
group, oval );
    icon = new ShapeGroup( Machine.this, true, 0, 0, 0
      ,image            );
                // Port connectors with non-replicated objects
    // Creating replicated embedded objects
    assignInitialConditions();
    onCreate();
  }

  @Override
  public void start() {
    SynchroTimer.start();
    protocol.start();
    onStartup();
  }

// User API ----------------------------------
  public Main get_Main() {
    ActiveObject owner = getOwner();
    if ( owner instanceof Main ) return (Main) owner;
    return null;
  }

  // Reaction on changes -----------------------
  public void onChange() {
    protocol.onChange();
  }

  public void onDestroy() {
    super.onDestroy();
    SynchroTimer.onDestroy();
    protocol.onDestroy();
  }

// Additional class code
//command IDs
public static final int SYNCHROTICK = 0;
public static final int MASTERACK   = 1;
public static final int MASTERREQ   = 2;
public static final int MASTERUP    = 3;
public static final int ELECTION    = 4;
public static final int QUIT        = 5;
public static final int REFUSE      = 6;
public static final int ACCEPT      = 7;
public static final int RESOLVE     = 8;
public static final int CONFLICT    = 9;
public static final int SLAVEUP     = 10;
  // End of additional class code
}
```

Network.java

```
package leader_election_protocol;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import java.awt.geom.Arc2D;

public class Network extends ActiveObject
{
 /**
  * TransmissionDelay Dynamic Event Class
  */
 public static class TransmissionDelay extends DynamicEvent {
  public Object msg;

  /**
   * Constructor<br>
   * Use <code>create_TransmissionDelay</code> method
   */
  public TransmissionDelay( Network ao, double dt , Object msg ) {
   super( ao, dt );
   this.msg = msg;
  }

  public void execute() {
   super.execute(); // must be called!
   ((Network) getActiveObject()).on_TransmissionDelay_xjal( msg );
  }
 }

 // Parameters

 public double  MeanLatency;

 /**
  * Returns default value for parameter <code>MeanLatency</code>.
  * <i>This method should not be called by user</i>
  */
 public double  _MeanLatency_DefaultValue_xjal() {
  return 1 ;
 }

 public void set_MeanLatency(
double  MeanLatency ) {
  if (MeanLatency == this.MeanLatency) {
   return;
  }
  this.MeanLatency = MeanLatency;
  onChange_MeanLatency();
  onChange();
 }

 void onChange_MeanLatency() {
 }


 public double  LossRate;

 /**
  * Returns default value for parameter <code>LossRate</code>.
  * <i>This method should not be called by user</i>
  */
 public double  _LossRate_DefaultValue_xjal() {
  return 0.3 ;
 }

 public void set_LossRate(
double  LossRate ) {
  if (LossRate == this.LossRate) {
   return;
  }
  this.LossRate = LossRate;
  onChange_LossRate();
  onChange();
 }

 void onChange_LossRate() {
 }

 // Dynamic Events

 /**
  * Creates new dynamic event TransmissionDelay with given parameter
values and schedules in the timeout specified
  * @param dt the timeout
  * @param msg
  * @return created dynamic event instance
```

```
  */
  public TransmissionDelay create_TransmissionDelay(double dt , Object
msg ) {
    return new TransmissionDelay( this, dt , msg );
  }

  /**
   * <i>This method should not be called by user</i>
   */
  public void on_TransmissionDelay_xjal(Object msg ) {

// after the delay send the message back, drop the message
// based on the distribution (LossRate)
if( uniform() >= LossRate ){
  port.send( msg );
}
;
  }
  static final Font _text_Font = new Font("SansSerif", 1, 14 );
  static final Color _oval_FillColor = new Color( 0xFF3366FF, true );
  static final Color _oval1_FillColor = new Color( 0xFF3366FF, true );
  static final Color _oval2_FillColor = new Color( 0xFF3366FF, true );
  static final Color _oval3_FillColor = new Color( 0xFF3366FF, true );
  static final int _oval = 1;
  static final int _oval1 = 2;
  static final int _oval2 = 3;
  static final int _oval3 = 4;
  static final int _text = 5;

  /**
   * Top-level presentation group id
   */
  static final int _presentation = 0;

  /**
   * Top-level icon group id
   */
  static final int _icon = -1;


  @Override
  public boolean onShapeClick( int _shape, int index, double clickx,
double clicky ){
    switch( _shape ){
      case _oval:
        if (true) {

getEngine().getPresentation().setPresentable( this );
        }
        break;
      case _oval1:
        if (true) {

getEngine().getPresentation().setPresentable( this );
        }
        break;
      case _oval2:
        if (true) {

getEngine().getPresentation().setPresentable( this );
        }
        break;
      case _oval3:
        if (true) {

getEngine().getPresentation().setPresentable( this );
      }
      break;
    default: return super.onShapeClick( _shape, index, clickx, clicky );
            }
            return false;
}

ShapeOval oval;
ShapeOval oval1;
ShapeOval oval2;
ShapeOval oval3;
ShapeText text;

// Static initialization of persistent elements
{
  oval = new ShapeOval(
    true,118, 85, 0.0,
    null, _oval_FillColor,
                18, 15,
                1, LINE_STYLE_SOLID
  ) {

    @Override
    public boolean onClick( double clickx, double clicky ) {
      return onShapeClick( _oval, 0, clickx, clicky );
    }
  };
  oval1 = new ShapeOval(
    true,83, 78, 0.0,
    null, _oval1_FillColor,
                23, 18,
                1, LINE_STYLE_SOLID
  ) {

    @Override
    public boolean onClick( double clickx, double clicky ) {
      return onShapeClick( _oval1, 0, clickx, clicky );
    }
  };
  oval2 = new ShapeOval(
    true,94, 85, 0.0,
    null, _oval2_FillColor,
                16, 15,
                1, LINE_STYLE_SOLID
  ) {

    @Override
    public boolean onClick( double clickx, double clicky ) {
      return onShapeClick( _oval2, 0, clickx, clicky );
    }
  };
  oval3 = new ShapeOval(
    true,109, 78, 0.0,
    null, _oval3_FillColor,
                31, 18,
                1, LINE_STYLE_SOLID
  ) {

    @Override
    public boolean onClick( double clickx, double clicky ) {
      return onShapeClick( _oval3, 0, clickx, clicky );
    }
  };
  text = new ShapeText(
    true,10, 10, 0.0,
```

```
      black,"Network",
      _text_Font, ALIGNMENT_LEFT
    );
  }
  ShapeGroup presentation;
  ShapeGroup icon;

  @Override
  public Object getPersistentShape( int _shape ) {
    switch(_shape){
      case _presentation: return presentation;
      case _icon: return icon;

      case _oval: return oval;
      case _oval1: return oval1;
      case _oval2: return oval2;
      case _oval3: return oval3;
      case _text: return text;
      default: return null;
    }
  }


  @Override
  public void drawModelElements(Panel _panel, Graphics2D _g, boolean
_publicOnly ) {
      drawPort( _panel, _g, 100, 60, -5, -19, null, port );
    if (!_publicOnly) {
      drawDynamicEvent( _panel, _g, 200, 100, 20, 0,
"TransmissionDelay", TransmissionDelay.class );
    }
    if (!_publicOnly) {
      drawParameter( _panel, _g, 200, 70, 20, 0, "MeanLatency",
MeanLatency, false, false );
    }
    if (!_publicOnly) {
      drawParameter( _panel, _g, 200, 50, 20, 0, "LossRate", LossRate,
false, false );
    }
  }

  @Override
  public boolean onClickModelAt( Panel panel, double x, double y, int
clickCount, boolean publicOnly ) {
    if( !publicOnly && modelElementContains(x, y, 200, 70) ) {
      panel.addInspect( 200, 70, this, "MeanLatency" );
      return true;
    }
    if( !publicOnly && modelElementContains(x, y, 200, 50) ) {
      panel.addInspect( 200, 50, this, "LossRate" );
      return true;
    }
    if( modelElementContains(x, y, 100, 60) ) {
      panel.addInspect( 100, 60, this, "port" );
      return true;
    }
    return false;
  }

  // Ports

  public Port<
Object
,
Object > port = new Port< Object, Object >( this );

  public String getNameOf( Port<?, ?> _p ) {
    if( _p == this.port ) return "port";
    return null;
  }

  public boolean executeOnReceiveActionOf( Port<?, ?> _p, Object _msg
) {
    if( _p == this.port ) {
      Object msg = (Object) _msg;

new TransmissionDelay( this, exponential( 1/MeanLatency ), msg );
    }
    return true;
  }

  /**
   * Constructor
   */
  public Network( Engine engine, ActiveObject owner,
ActiveObjectCollection<? extends Network> collection ) {
    super( engine, owner, collection );
  }

  @Override
  public void create() {
    // Dynamic initialization of persistent elements
    presentation = new ShapeGroup( Network.this, true, 0, 0, 0, text );
    icon = new ShapeGroup( Network.this, true, 0, 0, 0
      ,oval
      ,oval1
      ,oval2
      ,oval3      );
                // Port connectors with non-replicated objects
    // Creating replicated embedded objects
    assignInitialConditions();
    onCreate();
  }

  @Override
  public void start() {
    onStartup();
  }


// User API ----------------------------------
  public Main get_Main() {
    ActiveObject owner = getOwner();
    if ( owner instanceof Main ) return (Main) owner;
    return null;
  }


}
```

Command.java

```
/**
 * Message class Command
 */
public class Command implements Cloneable, java.io.Serializable {

            /**
             * This number is here for model snapshot storing
purpose<br>
             * It needs to be changed when this class gets changed
```

```
		*/
		private static final long serialVersionUID = 7222321206237672226L;

		// Message parameters
		public Machine source;
		public Machine destination;
		public int id;

		/**
		 * Constructor with message parameters
		 */
		public Command( Machine source, Machine destination, int id) {
			this.source = source;
			this.destination = destination;
			this.id = id;
		}

		@Override
		public String toString() {
			return "Command[source=" + source + ";destination=" + destination + ";id=" + id + "]";
		}

}
```

Simulation.java

```
package leader_election_protocol;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;
import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;
import javax.swing.JApplet;

public class Simulation extends ExperimentSimulation<Main> {
 static final Font _button_Font = new Font("Dialog", 0, 11 );
 static final Font _text_Font = new Font("SansSerif", 1, 28 );
 static final Font _Text10_Font = new Font("SansSerif", 0, 11 );
 static final Font _text10_Font = new Font("SansSerif", 0, 9 );
 static final int _button = 1;
 static final int _text = 2;
 static final int _Text10 = 3;
 static final int image = 4;
 static final int text10 = 5;

 /**
  * Top-level presentation group id
  */
 static final int _presentation = 0;

 /**
  * Top-level icon group id
  */
 static final int _icon = -1;


 @Override
 public void executeShapeControlAction( int _shape, int index ) {
  switch( _shape ) {
   case _button: {
run();
getEngine().getPresentation().setPresentable( getEngine().getRoot() );
;}
    break;
   default:
    super.executeShapeControlAction( _shape, index );
    break;
  }
 }

 @Override
 public String getNameOfShape( int _shape ) {
  switch( _shape ) {
   case image: return "image";
   case text10: return "text10";
   default: return super.getNameOfShape( _shape );
  }
 }

 @Override
 public int getShapeType( int _shape ) {
  switch( _shape ) {
   case image: return SHAPE_IMAGE;
   case text10: return SHAPE_TEXT;
   default: return super.getShapeType( _shape );
  }
 }
```

```
@Override
public double getShapeX( int _shape, int index ) {
 switch( _shape ) {
  case image: return 30;
  case text10: return 70;
  default: return super.getShapeX( _shape, index );
 }
}

@Override
public double getShapeY( int _shape, int index ) {
 switch( _shape ) {
  case image: return 530;
  case text10: return 570;
  default: return super.getShapeY( _shape, index );
 }
}

@Override
public Color getShapeFillColor( int _shape, int index ) {
 switch( _shape ) {
  case text10: return navy;
  default: return super.getShapeFillColor( _shape, index );
 }
}

@Override
public double getShapeWidth( int _shape, int index ) {
 switch( _shape ) {
  case image: return 116;
  default: return super.getShapeWidth( _shape, index );
 }
}

@Override
public double getShapeHeight( int _shape, int index ) {
 switch( _shape ) {
  case image: return 40;
  default: return super.getShapeHeight( _shape, index );
 }
}


// Images files
static final String[] _image_filenames = {
  "xjlogo.png",
};

@Override
public String getShapePackagePrefix( int shape ) {
 switch( shape ) {
  case image: return "/leader_election_protocol/";
  default: return super.getShapePackagePrefix( shape );
 }
}

@Override
public String[] getShapeImageFileNames( int shape, int index ) {
 switch( shape ) {
  case image: return _image_filenames;
  default: return super.getShapeImageFileNames( shape, index );
 }
}
```

```
@Override
public Object getShapeText( int _shape, int index ) {
 switch( _shape ) {
  case text10: return "AnyLogic and this model is (c) XJ Technologies,
www.anylogic.com. All rights reserved.";
  default: return super.getShapeText( _shape, index );
 }
}

@Override
public Font getShapeFont( int _shape, int index ) {
 switch( _shape ) {
  case text10: return _text10_Font;
  default: return super.getShapeFont( _shape, index );
 }
}

@Override
public int getShapeTextAlignment( int _shape, int index ) {
 switch( _shape ) {
  case text10: return ALIGNMENT_LEFT;
  default: return super.getShapeTextAlignment( _shape, index );
 }
}

/**
 * <i>This method should not be called by user</i>
 */
private void _button_SetDynamicParams( ShapeButton shape ) {
 boolean _visible;
 shape.setEnabled(
getState() == IDLE
);
}

ShapeButton button;
ShapeText text;
ShapeText Text10;

// Static initialization of persistent elements
{
 button = new ShapeButton(
                                    Simulation.this, true, 40, 300,
                                    110, 30,
      controlDefault, black,
      _button_Font,
                                    "Run the model"
 ) {

  @Override
  public void draw( Panel panel, Graphics2D graphics, AffineTransform
xform, boolean publicOnly ) {
   _button_SetDynamicParams( this );
   super.draw( panel, graphics, xform, publicOnly );
  }

  @Override
  public void action(){
   executeShapeControlAction( _button, 0 );
  }
 };
 text = new ShapeText(
   true,40, 30, 0.0,
   black,"Leader Election Protocol",
   _text_Font, ALIGNMENT_LEFT
 );
```

```
  Text10 = new ShapeText(
      true,40, 80, 0.0,
      slateGray,"This is a model of a distributed Leader \r\nElection protocol. A number of machines are \r\nconnected with an unreliable network. The \r\nnewtork loses, duplicates and delays \r\nmessages randomly. The machines \r\nthemselves can crash and recover. The \r\nprotocol is used to maintain a single leader \r\n(master) known to all working machines at \r\nany time. This is acheived by using \r\nsynchronization and election algorithms \r\nbased on constant and random timeouts. \r\nEach machine executes the same algorithm \r\nspecified as a statechart. You can force a \r\nmachine (e.g. current master) to crach/repair \r\nand watch the elections on the color chart.",
      _Text10_Font, ALIGNMENT_LEFT
  );
 }
 ShapeGroup presentation;
 ShapeGroup icon;

 @Override
 public Object getPersistentShape( int _shape ) {
  switch(_shape){
   case _presentation: return presentation;
   case _icon: return icon;

   case _button: return button;
   case _text: return text;
   case _Text10: return Text10;
   default: return null;
  }
 }



 @Override
 public int getWindowWidth() {
  return 850;
 }

 @Override
 public int getWindowHeight() {
  return 600;
 }

 /**
  * Applet class to run experiment as java applet
  */
 public static class Applet extends JApplet {

  Simulation ex;

  @Override
  public void init() {
   ex = new Simulation();
   ex.setup( this );
  }

  @Override
  public void destroy() {
   ex.close();
  }

 }

 @Override
 public void initDefaultRandomNumberGenerator(Engine engine) {
  engine.getDefaultRandomGenerator().setSeed( 1 );
 }

 @Override
 public Main createRoot( Engine engine ) {
  // Create the root object
  return new Main( engine, null, null );
 }

 @Override
 public void setupRootParameters( Main root, boolean callOnChangeActions ) {
  double MeanLatency_xjal = root._MeanLatency_DefaultValue_xjal();
  if (callOnChangeActions) {
   root.set_MeanLatency( MeanLatency_xjal );
  } else {
   root.MeanLatency = MeanLatency_xjal;
  }
  double LossRate_xjal = root._LossRate_DefaultValue_xjal();
  if (callOnChangeActions) {
   root.set_LossRate( LossRate_xjal );
  } else {
   root.LossRate = LossRate_xjal;
  }
  int NoOfMachines_xjal = root._NoOfMachines_DefaultValue_xjal();
  if (callOnChangeActions) {
   root.set_NoOfMachines( NoOfMachines_xjal );
  } else {
   root.NoOfMachines = NoOfMachines_xjal;
  }
  double ConflictTimeout_xjal = root._ConflictTimeout_DefaultValue_xjal();
  if (callOnChangeActions) {
   root.set_ConflictTimeout( ConflictTimeout_xjal );
  } else {
   root.ConflictTimeout = ConflictTimeout_xjal;
  }
  double MTTR_xjal = root._MTTR_DefaultValue_xjal();
  if (callOnChangeActions) {
   root.set_MTTR( MTTR_xjal );
  } else {
   root.MTTR = MTTR_xjal;
  }
  double SynchroTimeout_xjal = root._SynchroTimeout_DefaultValue_xjal();
  if (callOnChangeActions) {
   root.set_SynchroTimeout( SynchroTimeout_xjal );
  } else {
   root.SynchroTimeout = SynchroTimeout_xjal;
  }
 }



 /**
  * Engine setup
  */
 @Override
 public void setupEngine(Engine engine) {
  engine.setATOL( 1.0E-5 );
  engine.setRTOL( 1.0E-5 );
  engine.setTTOL( 1.0E-5 );
  engine.setHTOL( 0.0010 );
  engine.setSolverODE( Engine.SOLVER_ODE_EULER );
  engine.setSolverNAE( Engine.SOLVER_NAE_MODIFIED_NEWTON );
  engine.setSolverDAE( Engine.SOLVER_DAE_RK45_NEWTON );
  engine.setVMethods( 16032 );

  engine.setStartTime( 0.0 );
  engine.setTimeUnit( TIME_UNIT_DAY );
  engine.setRealTimeMode( true );
```

```
    engine.setRealTimeScale( 64.0 );
  }

  /**
   * Experiment setup
   */
  @Override
  public void setup( JApplet applet ) {
    setName( "Leader Election Protocol : Simulation" );

    // Dynamic initialization of persistent elements
    presentation = new ShapeGroup( Simulation.this, true, 0, 0, 0, button,
text, Text10, image, text10 );
    icon = new ShapeGroup( Simulation.this, true, 0, 0, 0                );
    // Setup presentation
    Presentation _p = new Presentation( this, applet != null ?
Presentation.MODE_APPLET :
        Presentation.MODE_APPLICATION, applet );
    _p.start();

    Panel _panel = _p.getPanel();
    ToolBar _tb = _p.getToolBar();
    StatusBar _sb = _p.getStatusBar();
    _panel.setZoomEnabled( false );
    _panel.setPanningEnabled( false );
    _panel.setFrameManagementBalance( 2.0 );

    _sb.setSectionVisible( StatusBar.DATE, false );
    _sb.setSectionVisible( StatusBar.EPS, false );
    _sb.setSectionVisible( StatusBar.EXPERIMENT, false );
    _sb.setSectionVisible( StatusBar.FPS, false );
    _sb.setSectionVisible( StatusBar.MEMORY, true );
    _sb.setSectionVisible( StatusBar.SECONDS, true );
    _sb.setSectionVisible( StatusBar.SIMULATION, true );
    _sb.setSectionVisible( StatusBar.STATUS, true );
    _sb.setSectionVisible( StatusBar.STEP, false );
    _sb.setSectionVisible( StatusBar.TIME, false );
    _tb.setSectionVisible( ToolBar.ANIMATION, false );
    _tb.setSectionVisible( ToolBar.EXECUTION, true );
    _tb.setSectionVisible( ToolBar.FILE, false );
    _tb.setSectionVisible( ToolBar.NAVIGATION, true );
    _tb.setSectionVisible( ToolBar.TIME_SCALE, true );
    _tb.setSectionVisible( ToolBar.VIEW, false );
  }

}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] «Προσομοίωση», Καθ. Νικήτας Α. Ασημακόπουλος, Σημειώσεις από τα μαθήματα, Προπτυχιακό Τμήμα Πληροφορικής του Πανεπιστημίου Πειραιώς

[2] «Συστημική Ανάλυση», Καθ. Νικήτας Α. Ασημακόπουλος, Σημειώσεις από τα μαθήματα, Προπτυχιακό Τμήμα Πληροφορικής του Πανεπιστημίου Πειραιώς

[3] «Εικονικές Επιχειρήσεις», Καθ. Νικήτας Α. Ασημακόπουλος, Σημειώσεις από τα μαθήματα, Μεταπτυχιακό Πρόγραμμα Σπουδών «Πληροφορική», Τμήμα Πληροφορικής του Πανεπιστημίου Πειραιώς

[4] «Αντικειμενοστραφής Υβριδική Προσομοίωση με το σύστημα AnyLogic: Μεθοδολογία και εφαρμογές», Μαργαρίτα-Κλειώ Σταματιάδου, Πτυχιακή Εργασία, Τμήμα Πληροφορικής του Πανεπιστημίου Πειραιώς, Φεβ. 2009

[5] Ολοκληρωμένη βοήθεια σχετικά με το περιβάλλον ανάπτυξης του Λογισμικού. Ενσωματώνεται στο Λογισμικό κατά την εγκατάσταση του στους σταθμούς εργασίας. Σε απευθείας σύνδεση στον ιστότοπο: http://www.xjtek.com/anylogic/help/

[6] Συλλογή μοντέλων - παραδειγμάτων εφαρμογών του Λογισμικού, (η λίστα των εφαρμογών ανανεώνεται συνεχώς). Σε μορφή Java Applets και σε απευθείας σύνδεση στους ιστοτόπους:

http://www.xjtek.com/anylogic/demo_models/ και
http://www.xjtek.com/anylogic/demo_models_3d/

[7] Οπτικοποιημένο βοηθητικό υλικό, http://www.xjtek.com/anylogic/resources/videos/

[8] https://en.wikipedia.org/wiki/Anylogic

[9] Java Applet Παραδείγματος 1: http://www.xjtek.com/anylogic/demo_models/23/

[10] Java Applet Παραδείγματος 2: http://www.xjtek.com/anylogic/demo_models/24/

[11] Java Applet Παραδείγματος 3: http://www.xjtek.com/anylogic/demo_models/10/

[12] Java Applet Παραδείγματος 4: http://www.xjtek.com/anylogic/demo_models/11/

[13] Java Applet Παραδείγματος 5: http://www.xjtek.com/anylogic/demo_models/22/

[14] Java Applet Παραδείγματος 6: http://www.xjtek.com/anylogic/demo_models/17/

[15] Java Applet Παραδείγματος 7: http://www.xjtek.com/anylogic/demo_models/45/

[16] Java Applet Παραδείγματος 8: http://www.xjtek.com/anylogic/demo_models/53/