ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Διδακτικής Τεχνολογίας και Ψηφιακών Συστημάτων

ΣΧΕΔΙΑΣΗ ΣΥΣΤΗΜΑΤΟΣ ΑΝΑΓΝΩΡΙΣΗΣ CONTEXT ΣΕ ΟΙΚΙΑΚΟ, ΕΡΓΑΣΙΑΚΟ Ή ΔΗΜΟΣΙΟ ΠΕΡΙΒΑΛΛΟΝ

Ανδρέας Γεωργακόπουλος

Η εργασία υποβάλλεται για την μερική κάλυψη των απαιτήσεων με στόχο την απόκτηση του Μεταπτυχιακού Διπλώματος Σπουδών στην Διδακτική Τεχνολογίας και Ψηφιακά Συστήματα του Πανεπιστήμιο Πειραιώς

Abstract

The notion of context awareness combines the use of software agents, ontologies, middleware and of course ubiquitous, wireless, always-on network. Software agents can now been developed and deployed through well-known middleware platforms such as JADE or JADEX. JADE and JADEX are Java-based platforms which provide the necessary tools to create and manage easily and simply software agents. In addition, JADEX is based on the Belief-Desire-Intention (BDI) model and makes use of Agent Definition Files written in XML (eXtensible Markup Language) in order to enhance the degree of agent flexibility on achieving goals. Exchanged messages between agents in JADE or JADEX follow the FIPA-ACL (Agent Communication Language) standard which is created by the Foundation for Physical Agents (FIPA). Ontologies on the other hand, enhance the level of agent standardization by following a common pattern. A common tool for creating ontologies is the Protégé which is Java-based and has a user-friendly working environment.

The goal of this project is to investigate all the above research fields and propose a multi-agent communication model based on the JADEX platform and ontologies in order to achieve context aware, personalized applications on various physical environments. Pervasive computing, mobile application challenges and trends on intelligent services are explained as well.

Keywords

agent, jade, jadex, bdi model, ontology, protégé, intelligent services

Περίληψη

Η έννοια του ”context awareness” συνδυάζει τη χρήση πρακτόρων λογισμικού, οντολογιών, middleware και φυσικά ασύρματο δίκτυο με αδιάκοπτη λειτουργία οπουδήποτε. Οι πράκτορες μπορούν να σχεδιαστούν και να υλοποιηθούν με χρήση γνωστών πλατφορμών όπως οι JADE και JADEX. Τα JADE και JADEX είναι πλατφόρμες υλοποιημένες σε Java, οι οποίες προσφέρουν τα βασικά εργαλεία για την εύκολη δημιουργία και απλή διαχείριση πρακτόρων λογισμικού. Επιπροσθέτως, το JADEX βασίζεται στο μοντέλο Belief-Desire-Intention (BDI) και χρησιμοποιεί τα λεγόμενα Agent Definition Files (ADF) τα οποία αναπτύσσονται σε XML ώστε να προωθηθεί η βελτίωση του βαθμού ελαστικότητας στην επίτευξη των στόχων τους. Τα μηνύματα που ανταλλάσσονται στο JADE και JADEX ακολουθούν το FIPA-ACL (Agent Communication Language) πρότυπο το οποίο έχει δημιουργηθεί από το Foundation for Physical Agents (FIPA). Οι οντολογίες από την άλλη, αναβαθμίζουν το επίπεδο προτυποποίησης των πρακτόρων λογισμικού ώστε να ακολουθείται μία συγκεκριμένη δομή κατά την ανάπτυξή τους. Ένα κοινό και ευχρηστο εργαλείο για τη δημιουργία οντολογιών αποτελεί το Protege, το οποίο βασίζεται και αυτό στη Java και αποτελείται από ένα φιλικό προς το χρήστη γραφικό περιβάλλον.

Ο στόχος της παρούσας μελέτης είναι να αναλύσει τα προαναφερθέντα ερευνητικά πεδία και να προτείνει ένα πολύ-πρακτορικό μοντέλο επικοινωνίας βασιζόμενο στην πλατφόρμα JADEX και σε οντολογίες ώστε να επιτευχθούν εξατομικευμένες υπηρεσίες σε διάφορα φυσικά περιβάλλοντα. Τέλος, γίνεται αναφορά και στις προκλήσεις της βιομηχανίας των κινητών υπηρεσιών καθώς και οι πρόσφατες τάσεις στον τομέα των έξυπνων υπηρεσιών.

Λέξεις κλειδιά

πράκτορας λογισμικού, jade, jadex, bdi model, οντολογία, protégé, έξυπνες υπηρεσίες

Ευχαριστίες

Θερμές ευχαριστίες στον Αναπληρωτή Καθηγητή κ. Παναγιώτη Δεμέστιχα για την επίβλεψη και τη βοήθεια που παρείχε για την ολοκλήρωση της διπλωματικής. Ευχαριστίες επίσης οφείλονται στους συνεργάτες Υποψήφιους Διδάκτορες κ. Μάριο Λογοθέτη και κ. Νίκο Κουτσούρη.

Επίσης, ευχαριστίες προς το Διδακτικό και Ερευνητικό Προσωπικό του Τμήματος Διδακτικής Τεχνολογίας και Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς για τη γενικότερη καθοδήγηση και προσφορά τους κατά τη διάρκεια του Μεταπτυχιακού Προγράμματος Σπουδών.

Τέλος, εκφράζω την ευγνωμοσύνη μου στους συνεργάτες –συναδέλφους και σε όλους αυτούς που βοήθησαν στην επιτυχή ολοκλήρωση της παρούσας εργασίας και του μεταπτυχιακού προγράμματος γενικότερα.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

1. Εισαγωγή

1.1 Γενικά στοιχεία

Στόχος της παρούσας μελέτης είναι να αναλύσει και να προτείνει ένα πολύ-πρακτορικό μοντέλο επικοινωνίας (multi-agent communication model) βασιζόμενο στην πλατφόρμα JADEX και σε οντολογίες ώστε να επιτευχθούν εξατομικευμένες υπηρεσίες σε διάφορα φυσικά περιβάλλοντα.

Κάποια μειονεκτήματα που εμφανίζονται σε τέτοιου είδους τεχνολογίες, μπορεί να σχετίζονται με την περιορισμένη ακόμα προσφορά ειδικά σχεδιασμένων υπηρεσιών για κινητές συσκευές, την κατανάλωση ενέργειας αλλά και θέματα ασφαλείας. Διότι αν το σύστημα γνωρίζει και εκπέμπει την πληροφορία θέσης κάποιου στο ευρύτερο περιβάλλον και αυτήν την πληροφορία την προσπελάσει κάποιος άλλος κακόβουλος χρήστης, τότε θα έχουμε παραβίαση απορρήτου ευαίσθητων προσωπικών δεδομένων. Στον αντίποδα, τα πλεονεκτήματα των αρχιτεκτονικών βασιζόμενων σε πράκτορες λογισμικού είναι η δυνατότητα αυτόματης αναγνώρισης των χρηστών και των επιθυμιών τους, η ασύρματη πρόσβαση από οπουδήποτε και οποτεδήποτε καθώς και η ευελιξία.

Βασιζόμενοι στα πλεονεκτήματα αυτής της αρχιτεκτονικής μπορούμε να χρησιμοποιήσουμε στατικούς ή κινητούς πράκτορες (αναλόγως τη συσκευή που τους ενσωματώνει) είτε αυτόνομους ή κοινωνικούς πράκτορες (αναλόγως το επίπεδο διαλειτουργικότητας που έχει οριστεί). Στη μελέτη μας, οι πράκτορες λογισμικού που χρησιμοποιούνται αναλαμβάνουν να επικοινωνούν μεταξύ τους προς την επίτευξη ενός ευρύτερου στόχου του συστήματος που μπορεί να είναι για παράδειγμα η αναπαραγωγή ταινίας.

Για την επίτευξη της δημιουργίας τέτοιων πρακτόρων, χρησιμοποιείται η Java-based πλατφόρμα Jadex, η οποία αναλαμβάνει να αναγνωρίσει τους πράκτορες που εισέρχονται και εξέρχονται κάθε φορά από το σύστημα και να

εκτελέσει τις εντολές τους. Παράλληλα για την καλύτερη προτυποποίηση των μηνυμάτων που ανταλλάσσονται γίνεται χρήση οντολογίας. Για το σχεδιασμό μιας οντολογίας είναι ιδιαίτερο χρήσιμο το εργαλείο Protege το οποίο δίνει τη δυνατότητα μέσω ενός φιλικού προς τον χρήστη γραφικού περιβάλλοντος να ορίσει επαρκώς όλα εκείνα τα χαρακτηριστικά που χρειάζονται για την οντολογία. Με την οντολογία λοιπόν επιτυγχάνεται και ο έλεγχος ακεραιότητας της υπηρεσίας καθώς τα μηνύματα που ανταλλάσσονται μεταξύ των πρακτόρων λογισμικού δεν είναι τυχαία αλλά βασίζονται σε προκαθορισμένη δομή δεδομένων.

Η βασική ιδέα πάνω στην οποία στηριζόμαστε για την ανάπτυξη του συστήματος βασίζεται στο σενάριο επικοινωνίας μεταξύ ανθρώπου/χρήστη και έξυπνης συσκευής μέσω ενός εξυπηρετητή ο οποίος ελέγχει, δρομολογεί και καταγράφει τις κινήσεις του ανθρώπου/χρήστη. Σ’ αυτή την περίπτωση λοιπόν, οι οντότητες που εντοπίζονται είναι η κινητή συσκευή του χρήστη, ο εξυπηρετητής και η οικοσκευή με την οποία ενδιαφερόμαστε να επικοινωνήσουμε. Η κινητή συσκευή του χρήστη (δηλαδή το μήνυμα που στέλνει) αποτελεί το έναυσμα με το οποίο ο εξυπηρετητής λαμβάνει τις εκάστοτε αποφάσεις ώστε να ξεκινήσει η αναπαραγωγή της υπηρεσίας στην οικοσκευή που έχει ήδη αναγνωριστεί ως “standby” από το σύστημα (Jadex). Τα ανταλλασσόμενα μηνύματα συνήθως συνοδεύονται από ένα μήνυμα απάντησης ή επιβεβαίωσης ώστε να διασφαλίζεται η σωστή ροή της επικοινωνίας.

Το προαναφερθέν σενάριο μπορεί να επεκταθεί με την ενσωμάτωση στο μοντέλο μίας επιπλέον οντότητας (π.χ. δευτερεύον εξυπηρετητής) η οποία τοποθετείται σε επίπεδο αρχιτεκτονικής ανάμεσα στον βασικό εξυπηρετητή και την οικοσκευή. Το ανανεωμένο σενάριο στηρίζεται στην ιδέα ότι η κινητή συσκευή του χρήστη στέλνει μήνυμα στο βασικό εξυπηρετητή ο οποίος αποφασίζει να ενημερώσει τον βοηθητικό εξυπηρετητή (ο οποίος μπορεί να λειτουργεί ως π.χ. Tv Controller) και με τη σειρά του καταγράφει την εντολή ξεκινάει η αναπαραγωγή της υπηρεσίας στην πρώτη τηλεόραση. Έστω τώρα ότι επιθυμούμε να μεταφερθούμε στην τηλεόραση άλλου δωματίου, τότε το σύστημα

αναγνωρίζει την κίνησή μας και ο βασικός εξυπηρετητής αποφασίζει την αποστολή μηνύματος προς την δεύτερη τηλεόραση με τελικό αποτέλεσμα τη συνέχιση αναπαραγωγής της υπηρεσίας από το σημείο που σταμάτησε στην πρώτη τηλεόραση.

Τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του agent-based συστήματος στηρίζονται κατά κύριο λόγο στη γλώσσα Java και δευτερεύοντος στην Xml. Η πλατφόρμα πάνω στην οποία υλοποιούνται οι πράκτορες λογισμικού είναι η Jadex. Επιπροσθέτως για την καλύτερη οπτική παρουσίαση των πρακτόρων χρησιμοποιείται το Java Swing package, ενώ στην αναπαραγωγή αρχείων ήχου και εικόνας συμβάλλει το Java Media Framework το οποίο παρέχει τις κατάλληλες κλάσεις και μεθόδους προς την επίτευξη αυτού του στόχου. Τέλος, οι οντολογίες δημιουργήθηκαν με τη βοήθεια του ειδικού εργαλείου Protege και με το κατάλληλο plug-in (Jadex Beanynizer) έγινε εξαγωγή των σχεδιασμένων κλάσσεων και μεθόδων σε μορφή κώδικα Java.

Τα βασικά συμπεράσματα που προκύπτουν από τη συγκεκριμένη μελέτη μπορούν να συνοψισθούν στα ακόλουθα:

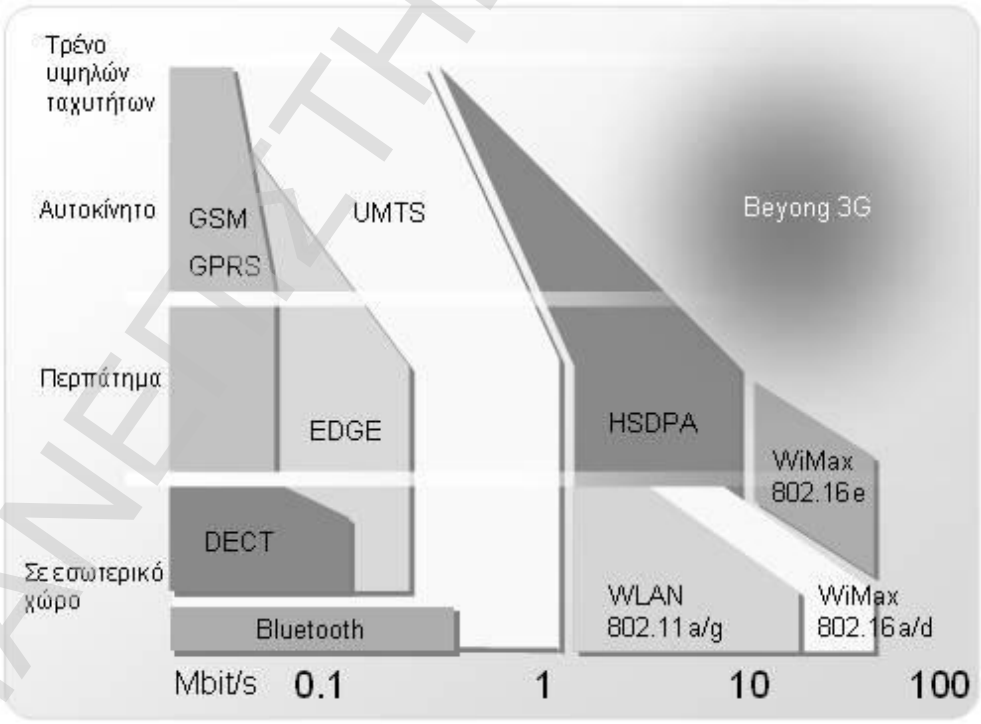
- Οι πράκτορες λογισμικού μπορούν να ενσωματωθούν σε οποιαδήποτε ηλεκτρονική συσκευή (κινητή ή σταθερή) με αποτέλεσμα να εξασφαλίζεται η διαλειτουργικότητα.
- Οι πράκτορες αναλαμβάνουν την συμπεριφορά και το σκοπό της εκάστοτε συσκευής σύμφωνα με ένα προκαθορισμένο πλάνο ενεργειών το οποίο συντάσσεται στη γλώσσα Xml.
- Η ακεραιότητα των μηνυμάτων που ανταλλάσσουν οι πράκτορες του συστήματος, διασφαλίζεται με τη χρήση οντολογιών οι οποίες χτίζονται πάνω σε προκαθορισμένες δομές δεδομένων αναλόγως με τις ανάγκες του εκάστοτε πράκτορα αλλά και του συστήματος γενικότερα.
- Οι οντολογίες μπορούν να σχεδιαστούν με τη χρήση του ειδικού εργαλείου Protege το οποίο βασίζεται στη Java και προορίζεται για αυτό τον σκοπό.

1.2 Εντοπισμός βασικών αναγκών

Η ανάγκη αναζήτησης πληροφοριών οπουδήποτε, οποτεδήποτε φαίνεται να γίνεται όλο και πιο σημαντική με την πάροδο του χρόνου. Αυτό οδηγεί στη δημιουργία νέων ασύρματων υπηρεσιών οι οποίες θα μπορούν να εκτελούνται από τις κινητές συσκευές του εμπορίου. Για την αποτελεσματικότερη χρήση τέτοιων υπηρεσιών είναι αναγκαία η χρήση δικτύων νέας γενεάς όπως τα κυψελοειδή WCDMA, GPRS ή τα ασύρματα μέσω των προτύπων IEEE 802.11g/n, Bluetooth 2.0 κτλ. [1].

Ένα βασικό πλεονέκτημα των ασύρματων υπηρεσιών αποτελεί το γεγονός ότι μπορούν να χρησιμοποιηθούν από οπουδήποτε χωρίς να υπάρχει ο περιορισμός του χώρου (π.χ. γραφείο, σπίτι κτλ.), αν και σύμφωνα με το σχήμα που ακολουθεί φαίνεται να προκύπτουν κάποιοι περιορισμοί προς το παρόν.

Σχήμα 1.1: Μέγιστοι ρυθμοί μετάδοσης δεδομένων εν κινήσει (2006)



Όπως λοιπόν φαίνεται και στο σχήμα υπάρχουν περιορισμοί στον μέγιστο ρυθμό μετάδοσης δεδομένων αναλόγως την ταχύτητα και το κινούμενο μέσο που βρισκόμαστε. Επιπροσθέτως έχει προκύψει ότι το ασύρματο δίκτυο (κυψελοειδές ή μη) δεν είναι πάντα αξιόπιστο λόγω αλλοίωσης σήματος από κτίρια και άλλες κατασκευές που συναντάμε στο εξωτερικό περιβάλλον [16]. Επίσης, η κατανάλωση ενέργειας είναι ένας παράγοντας που μας απασχολεί καθώς από μετρήσεις που έχουν γίνει σε ασύρματο δίκτυο υπό το πρότυπο 802.11 μπορεί να καταναλωθεί το 5 με 10% της μπαταρίας της κινητής συσκευής σε μισή ώρα, αναλόγως πάντα τη χρήση και τη συσκευή [36].

Η ασφάλεια είναι άλλο ένα θέμα που μας απασχολεί σε τέτοιου είδους δίκτυα πρόσβασης καθώς σε context-aware συστήματα πρέπει να μεταδώσουμε πληροφορίες θέσης σε μία ευρύτερη περιοχή (για αναγνώριση της κινητής συσκευής από το δίκτυο –δέκτες) με αποτέλεσμα να υπάρχει κίνδυνος υποκλοπής ευαίσθητων προσωπικών δεδομένων [39]. Για παράδειγμα τι θα συνέβαινε αν:

- Το άτομο Α επιτρέπει την πρόσβαση στα αρχεία του υπολογιστή του από συναδέλφους μόνο όταν βρίσκεται ο ίδιος στο χώρο του γραφείο του. Αυτό θα οδηγούσε σε διαρροή πληροφορίας καθώς κάποιος θα μπορούσε να εκμεταλλευτεί το γεγονός της απουσίας του ατόμου Α από το σπίτι και να προχωρήσει σε διάρρηξη.
- Το άτομο Β επιτρέπει την πρόσβαση στα αρχεία του υπολογιστή του από συναδέλφους μόνο όταν βρίσκεται σε έναν συγκεκριμένο χώρο. Τότε το σύστημα το οποίο γνωρίζει ότι ο Β βρίσκεται εκεί, το μεταδίδει εμμέσως στους υπόλοιπους αφήνοντας ελεύθερη πρόσβαση στα αρχεία για τους συναδέλφους. Κάτι τέτοιο ίσως οδηγούσε σε παραβίαση ιδιωτικής ζωής.

Επομένως, εκτός από τη χρησιμότητα και τη δικτυακή υποδομή αλλά και το σχεδιασμό των ασύρματων υπηρεσιών πρέπει να αντιμετωπισθούν και ζητήματα ασφάλειας και ιδιωτικότητας.

1.3 Προκλήσεις

Οι προκλήσεις που καλούμαστε να αντιμετωπίσουμε κατά τη διάρκεια του σχεδιασμού ασύρματων εφαρμογών μπορούν να κατηγοριοποιηθούν στα εξής:

- *Context awareness:* Με αυτό τον όρο εννοούμε την ικανότητα εντοπισμού διαφόρων παραμέτρων και χαρακτηριστικών του περιβάλλοντος στο οποίο κινείται ο χρήστης μία δεδομένη στιγμή. Μετά πρέπει να διασφαλίσουμε την ομαλή ροή δεδομένων καθώς και την υψηλότερη δυνατή ποιότητα υπηρεσιών (Quality of Service –QoS). Σε αντίθετη περίπτωση ο χρήστης θα αναγκαστεί να διακόψει τη χρήση ή να μεταφερθεί σε άλλη ανταγωνιστική υπηρεσία.
- *Προσβασιμότητα:* Οι ασύρματες υπηρεσίες πρέπει να είναι διαθέσιμες χωρίς σημαντικές παρεμβολές –αλλοιώσεις όσον αφορά το κομμάτι της μετάδοσης δεδομένων.
- *Συσκευές:* Άλλος ένας περιορισμός των κινητών υπηρεσιών είναι η ίδια η κινητή συσκευή (PDA) που διαθέτει ο εκάστοτε χρήστης καθώς όταν σχεδιάζονται τέτοιου είδους υπηρεσίες πρέπει να λαμβάνεται υπόψη ο μικρός χώρος απεικόνισης πληροφορίας (οθόνη), η περιορισμένη επεξεργαστική ισχύς και η περιορισμένη διαθέσιμη μνήμη σε σύγκριση με τους επιτραπέζιους ή φορητούς υπολογιστές (laptop).

Κατά τη διάρκεια του 3^{ου} τριμήνου του 2008, τα βασικά χαρακτηριστικά δείγματος κινητών συσκευών (Personal Digital Assistants –PDAs) της αγοράς έχουν ως εξής:

- Glofiish models με 2.8 inches οθόνη και 320x240 ανάλυση, CPU στα 500MHz και 192Mbytes μνήμη.
- HTC models με οθόνες από 3 ως 5 inches, CPU από 200 ως 600MHz και μνήμη από 192 ως 256Mbytes.

- HP iPAQ models με 3 inches οθόνη και 240x240 ανάλυση, CPU στα 500MHz και μνήμη 192Mbytes.
- Mio models με παρόμοια χαρακτηριστικά και ενσωματωμένο GPS.
- Blackberry models με παρόμοια χαρακτηριστικά και ιδιότητες κινητού τηλεφώνου και PDA.

1.4 Τάσεις στις έξυπνες υπηρεσίες

Η έξυπνη δικτύωση βρίσκεται σε ανοδική πορεία και πολλά σενάρια εκτιμούν ότι σε 10 έτη από σήμερα, θα χρησιμοποιούμε έξυπνες υπηρεσίες μέσω πρακτόρων λογισμικού σε κάθε έκφανση της καθημερινότητας (Siemens R&D, 2005). Επιπροσθέτως, ο ολικός πληθυσμός της γης αναμένεται να αγγίξει τα 8 δις. και σύμφωνα με τον Παγκόσμιο Οργανισμό Υγείας (World Health Organization), ο αριθμός των ηλικιωμένων παγκοσμίως θα διπλασιαστεί ξεπερνώντας το 1 δις μέχρι το 2020. Μέχρι το 2050 το 1/3 του παγκοσμίου πληθυσμού θα είναι άτομα ηλικίας άνω των 65 ετών. Ως αποτέλεσμα της βελτίωσης του βιοτικού επίπεδου, περίπου 250,000 λίτρα πετρελαίου καταναλώνονται σήμερα (2008) κάθε δευτερόλεπτο [4, 6]. Επομένως, κρίνεται απαραίτητη η έξυπνη διαχείριση των διαθεσίμων ενεργειακών πόρων προς την επίτευξη μεγιστοποίησης της αποτελεσματικότητας [35]. Επίσης, ως το 2011, αναμένεται η ολοκλήρωση της προτυποποίησης των κινητών δικτύων τέταρτης γενεάς (4G) και ως το 2014 θα υπάρχουν οι πρωτότυπες ασύρματες συσκευές 4G. Ο στόχος των δικτύων 4G είναι η ικανότητα επίτευξης ρυθμών μετάδοσης δεδομένων στο 1Gbit/s. Μέχρι τότε, αυτόνομοι, proactive και κοινωνικοί πράκτορες λογισμικού θα έχουν τη δυνατότητα να αναλαμβάνουν έξυπνες υπηρεσίες όπως:

- Χειρισμός οπτικοακουστικού περιεχομένου όπως on demand TV προγράμματα, μουσική κ.ά.
- Ρύθμιση φωτισμού και εξωφύλλων.
- Χειρισμός διαφόρων οικιακών συσκευών. Για παράδειγμα μπορούμε να ελέγχουμε το πλυντήριο ρούχων μέσω του PDA, ή να ελέγχουμε τα αποθέματα τροφίμων στο ψυγείο και να γίνεται αυτόματη παραγγελία στο διασυνδεδεμένο super-market.
- Έλεγχος παραμέτρων περιβάλλοντος όπως θέρμανση, ψύξη, σχετική υγρασία, ποιότητα αέρα κτλ.
- Έλεγχος συστημάτων φύλαξης χώρου και συναγερμού

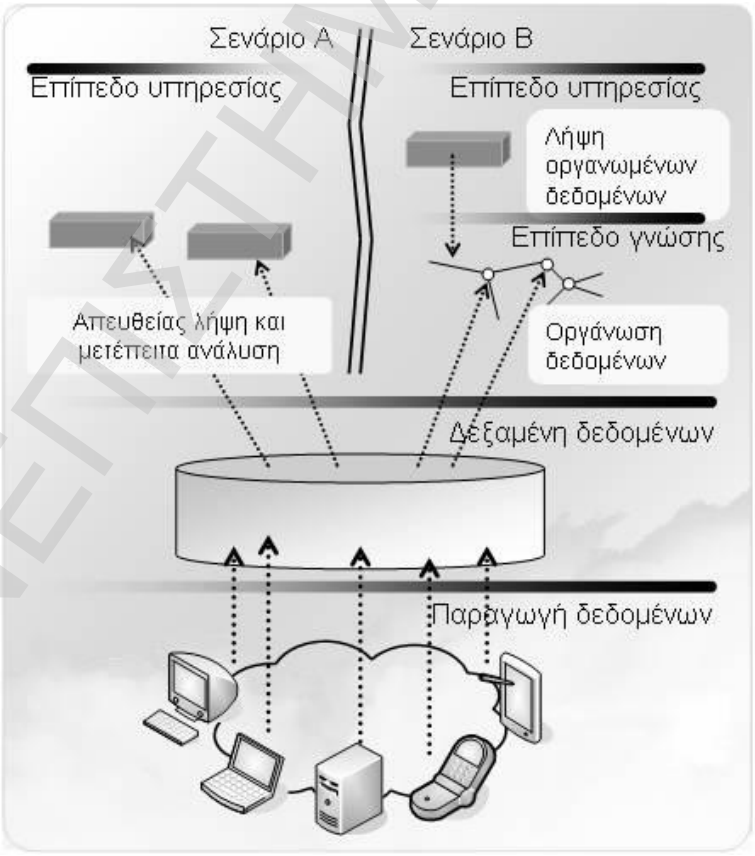
- Τηλεδιάσκεψη, πρόσβαση στο διαδίκτυο, έλεγχος μηνυμάτων ή απλή τηλεφωνική κλήση μέσω ενός τηλεπικοινωνιακού πίνακα.
- Πληροφορίες κίνησης στους δρόμους, βέλτιστη δρομολόγηση προς τον προορισμό, καιρικές συνθήκες κ.ά.
- Τηλε-ιατρική /24ωρη παρακολούθηση στο σπίτι και ενημέρωση του θεράποντα ιατρού.

Στο σχήμα που ακολουθεί εμφανίζεται ένα δείγμα της κονσόλας ελέγχου υπηρεσιών σε ένα έξυπνα δικτυωμένο σπίτι. Καθώς εισερχόμαστε στο σπίτι εμφανίζεται μήνυμα καλωσορίσματος με την ώρα και τη μέρα. Στην άνω δεξιά γωνία εμφανίζονται συνοπτικά νέα για την κίνηση στους δρόμους. Σε εξατομικευμένο επίπεδο θα μπορούσαν να εμφανίζονται μόνο οι άξονες που μας ενδιαφέρουν και όχι όλοι οι βασικοί άξονες της πόλης. Στην κάτω αριστερή γωνία εμφανίζονται οι παρούσες καιρικές συνθήκες, η εσωτερική θερμοκρασία και οι επιλογές ελέγχου των παραμέτρων περιβάλλοντος. Στην κάτω δεξιά γωνία εμφανίζεται ένα μπουκέτο υπηρεσιών όπως διαχείριση ενέργειας, διαχείριση συσκευών, προβολή μηνυμάτων, ασφάλεια, έλεγχος πολυμεσικού περιεχομένου, επιλογές τηλεφωνικών κλήσεων κτλ.

Σχήμα 1.2: Γραφική κονσόλα πρόσβασης έξυπνων υπηρεσιών

Μία άλλη τάση δείχνει την ανάγκη πρόσβασης σε φιλτραρισμένα και οργανωμένα δεδομένα αντί για την κλασσική πρόσβαση σε ακατέργαστα στοιχεία που βρίσκονται σε μία κοινή δεξαμενή δεδομένων. Το ακόλουθο σχήμα εξηγεί δύο σενάρια για τους τρόπους σύμφωνα με τους οποίους μπορούν να προσπελαστούν. Το σενάριο Α είναι ένα απλό σενάριο στο οποίο μια υπηρεσία λαμβάνει τα δεδομένα απευθείας από την κοινή δεξαμενή και μετά πρέπει να αναλάβει η ίδια την ανάλυση των ακατέργαστων δεδομένων ώστε να τα μετατρέψει σε χρήσιμη για το χρήστη πληροφορία. Από την άλλη, το σενάριο Β, χρησιμοποιεί ένα ενδιάμεσο επίπεδο οργάνωσης ανάμεσα στην υπηρεσία και την κοινή δεξαμενή δεδομένων, σύμφωνα με το οποίο οργανώνονται τα ακατέργαστα στοιχεία σε συναφείς ομάδες και μετά στέλνει μόνο τα απαραίτητα, δομημένα πλέον, δεδομένα στην υπηρεσία. Έτσι επιτυγχάνονται ταχύτεροι υπολογισμοί στον πυρήνα της υπηρεσίας αφού ενεργεί χωρίς να χρειαστεί να αναλύσει τα δοθέντα δεδομένα [37].

Σχήμα 1.3: Ακατέργαστα δεδομένα vs. Οργανωμένη πληροφορία



1.5 Ευχρηστία και εξατομίκευση έξυπνων υπηρεσιών

Οι προαναφερόμενες προκλήσεις μας οδηγούν σε εξατομικευμένες, ασύρματες υπηρεσίες με έμφαση στην ευχρηστία, την αξιόπιστη προσβασιμότητα μέσω ασύρματων δικτύων και στην καλύτερη δυνατή ποιότητα υπηρεσιών.

Με τον όρο ευχρηστία νοείται η ικανότητα μιας υπηρεσίας να είναι εύκολη προς χρήση και ο οιοσδήποτε χρήστης να προσπελαύνει τα διάφορα μέρη της υπηρεσίας εύκολα και γρήγορα με όσο λιγότερα clicks είναι δυνατόν. Παραταύτα, το γραφικό περιβάλλον της εκάστοτε υπηρεσίας πρέπει να είναι έτσι σχεδιασμένο ώστε να προσφέρει μία μοναδική εμπειρία στο χρήστη. Προς την επίτευξη λοιπόν υψηλού βαθμού ευχρηστίας οι σχεδιαστές πρέπει να λαμβάνουν υπόψη τους θεωρίες παρακίνησης και αναγνωρισμένες μεθόδους αξιολόγησης ευχρηστίας. Επίσης, στην εποχή του Web 2.0/3.0 γίνεται κατανοητή η μετακίνηση ενδιαφέροντος από την απλή αντίληψη της απόδοσης στην ευρύτερη αντιλαμβανόμενη γενική εμπειρία χρήσης [8].

Με τον όρο εξατομίκευση νοείται η ικανότητα του συστήματος να δημιουργεί και να διατηρεί προσωπικά προφίλ των χρηστών μέσω των οποίων μπορεί το σύστημα να αντλήσει χρήσιμες πληροφορίες γι’ αυτούς σύμφωνα με τους εκάστοτε όρους χρήσης προσωπικών δεδομένων στους οποίους συμφωνούν οι συμβαλλόμενοι. Έτσι, υπάρχει η δυνατότητα παρουσίασης χρήσιμων προς το χρήστη πληροφοριών προς την κάλυψη των αναγκών του [16].

Για να πετύχουμε τη μεγιστοποίηση της εμπειρίας του χρήστη η έξυπνη υπηρεσία που σχεδιάζουμε θα πρέπει να είναι χρήσιμη, επιθυμητή, πολύτιμη, αξιόπιστη, προσπελάσιμη και φυσικά εύχρηστη με προσεγμένο περιεχόμενο και λειτουργικότητα που να προσεγγίζει τις ανάγκες της πλειοψηφίας των χρηστών μετά από σχετική έρευνα αγοράς.

Επιπροσθέτως, οι context-aware υπηρεσίες θα πρέπει [38]:

- Να έχουν χαλαρή διασύνδεση μεταξύ των μερών τους (loosely-coupled) λόγω της δυναμικού χαρακτήρα της πληροφορίας.
- Να υπάρχει δυνατότητα επέκτασης σύμφωνα με τις αυξανόμενες ανάγκες.
- Να είναι αόρατες (δηλαδή το middleware να χρησιμοποιεί κάποιου είδους φιλτράρισμα ώστε να αποκρύπτονται οι κρίσιμες λειτουργίες του συστήματος από άλλες εφαρμογές).

2. Middleware

2.1 Γενικά χαρακτηριστικά του middleware

Ο σχεδιασμός και η δημιουργία συστημάτων πρακτόρων λογισμικού γίνεται σε middleware πλατφόρμες όπως η JADE (Java Agent Development Framework) και η JADEX (πλατφόρμα συνδυασμού JADE με XML). Η XML είναι πολύ σημαντική διότι κάνει πιο εύκολη την οργάνωση και πιο γρήγορη την αναζήτηση της δομημένης πλέον πληροφορίας μετατρέποντας το Διαδίκτυο σε δίκτυο γνώσης (knowledge network) [16]. Ειδικοί του χώρου υποστηρίζουν ότι το διαδίκτυο θα μετατραπεί την επόμενη δεκαετία σε μία τεράστια πηγή γνώσης. Η XML (EXtensible Markup Language) λοιπόν είναι μία γλώσσα σήμανσης όπως η HTML η οποία σχεδιάστηκε για την περιγραφή δεδομένων. Ο χρήστης έχει την δυνατότητα να ορίζει τις δικές του ετικέτες (tags), άρα αυτό σημαίνει ότι οι ετικέτες δεν είναι προκαθορισμένες από τη γλώσσα αυτή καθ' αυτή. Βέβαια για τη δημιουργία και την εκτέλεση πρακτόρων στο JADEX υπάρχουν συγκεκριμένες ετικέτες που αναγνωρίζονται από το σύστημα, αλλά αυτές ορίζονται στην πλατφόρμα του JADEX. Έτσι με την HTML, έχουμε παρουσίαση πληροφορίας, ενώ με την XML έχουμε περιγραφή της πληροφορίας. Γι' αυτό το λόγο η XML όπως ορίζεται από το World Wide Web Consortium, δομεί και αποθηκεύει πληροφορίες.

Με τον όρο middleware περιγράφονται όλες εκείνες οι βιβλιοθήκες υψηλού επιπέδου οι οποίες επιτρέπουν την ευκολότερη και αποδοτικότερη δημιουργία εφαρμογών, προσφέροντας υπηρεσίες που είναι χρήσιμες για μια συλλογή εφαρμογών και όχι μόνο για μία εφαρμογή, όπως π.χ. επικοινωνίες, προσπέλαση δεδομένων, κωδικοποίηση, έλεγχος πόρων. Μερικά από τα πλεονεκτήματα του middleware μπορούν να συνοψισθούν στα εξής [5]:

- Οι εφαρμογές που δημιουργούνται είναι ανεξάρτητες από το λειτουργικό σύστημα, την γλώσσα προγραμματισμού και το μέρος στο οποίο είναι αποθηκευμένα τα κατανεμημένα αντικείμενα.
- Μπορούν να εφαρμοστούν τα αντικειμενοστραφή (object-oriented) και client-server μοντέλα.
- Οι εφαρμογές αποτελούνται από διάφορα μέρη τα οποία έχουν την ικανότητα να επικοινωνούν μεταξύ τους.

2.2 Πράκτορες λογισμικού

Οι πράκτορες λογισμικού είναι έξυπνες εφαρμογές οι οποίες αναγνωρίζουν τις ανάγκες και τις προτιμήσεις των χρηστών τους με αποτέλεσμα να τους υποστηρίζουν στις καθημερινές τους εργασίες ή ακόμη και να δρουν εκ μέρους των χρηστών ως προσωπικοί εκπρόσωποι αυτών (‘accomplish tasks on behalf of its user [19]’). Ο αγγλικός όρος του πράκτορα είναι agent και προέρχεται από το Λατινικό *agere* (to do): δηλ. μία συμφωνία δράσης εκ μέρους κάποιου. Οι πράκτορες λογισμικού χαρακτηρίζονται από ευέλικτη συμπεριφορά επειδή είναι [9]:

- αυτόνομοι (autonomous)
- proactive και
- κοινωνικοί (social)

Μάλιστα σύμφωνα με ερευνητές, στο κοντινό μέλλον, οι οικιακές συσκευές, τα προσωπικά ρομπότ κ.ά. θα είναι δικτυωμένα. Το κλειδί λοιπόν για μία παγκοσμιοποιημένη δικτύωση και αναγνώριση τέτοιων συσκευών είναι οι πράκτορες λογισμικού. Μέσω των πρακτόρων επιτυγχάνεται διασφάλιση της ανταλλαγής των πληροφοριών, διαλειτουργικότητα και διαθεσιμότητα οπουδήποτε, οποτεδήποτε (Siemens R&D, 2005).

Ιστορικό πρακτόρων

Η έννοια του πράκτορα πηγάζει από τη δεκαετία του ’70 στα πλαίσια έρευνας για Κατανεμημένη Τεχνητή Νοημοσύνη (Distributed Artificial Intelligence). Στο σχετικό μοντέλο του Hewitt γίνεται αναφορά στη θεωρία ενός διαδραστικού (interactive) και self-contained αντικειμένου. Αυτό το αντικείμενο εμφανιζόταν σε μία συγκεκριμένη κατάσταση μέσω της οποίας μπορούσε να λαμβάνει και να απαντά σε μηνύματα άλλων τέτοιων αντικειμένων. Έκτοτε, διάφορα έργα έλαβαν χώρα σχετικά με πράκτορες όπως τα Strand 1 και Strand 2 για έξυπνους πράκτορες. Το Strand 1 εξετάζει θεωρητικά, αρχιτεκτονικά και γλωσσικά θέματα, ενώ το Strand 2 εστιάζει στη διαφοροποίηση των τύπων των πρακτόρων.

Κατηγοριοποίηση πρακτόρων

- *σε επίπεδο κινητικότητας*
  - στατικοί πράκτορες: όταν ο πράκτορας βρίσκεται σε σταθερή συσκευή.
  - κινητοί πράκτορες: όταν οι πράκτορες βρίσκονται σε κινητές συσκευές (π.χ. PDA).

- *σε επίπεδο αντίδρασης*
  - Οι αυτόνομοι πράκτορες διατηρούν ένα επίπεδο ελέγχου στις πράξεις τους ενώ είναι ικανοί να λαμβάνουν και αποφάσεις εκ μέρους του χρήστη. Αυτή η κατηγορία πρακτόρων λογισμικού δείχνει ότι οι αυτόνομοι πράκτορες έχουν τη δυνατότητα μέσω των αποφάσεων που λαμβάνουν να πράττουν αναλόγως ώστε να επιτευχθεί κάποιος εσωτερικός στόχος (goal) σύμφωνα με το αντιλαμβανόμενο περιβάλλον του πράκτορα.
  - Οι proactive πράκτορες αντιδρούν σε διάφορα εξωτερικά γεγονότα (π.χ. remote method call).
  - Οι κοινωνικοί πράκτορες έχουν τη δυνατότητα να επικοινωνούν με άλλους πράκτορες ώστε να επιτευχθεί ο σκοπός του συστήματος. Γι' αυτό το λόγο χρησιμοποιούνται σε context-aware συστήματα [3].

- *σε επίπεδο ρόλων*
  - π.χ. World Wide Web (WWW) πράκτορες οι οποίοι βασίζονται σε μηχανές αναζήτησης του διαδικτύου. Βασική λειτουργία τους αποτελεί η διαχείριση του τεράστιου όγκου δεδομένων που παρέχονται από τέτοιου είδους υπηρεσίες (αναζήτησης).
  - Intranet πράκτορες: διαχειρίζονται τα δεδομένα που διακινούνται στο εσωτερικό, εταιρικό LAN.

Βασικά χαρακτηριστικά

Τα βασικά χαρακτηριστικά περιγράφονται από τον τομέα Έρευνας και Ανάπτυξης της Siemens Corp. και μπορούν να συνοψισθούν στα εξής:

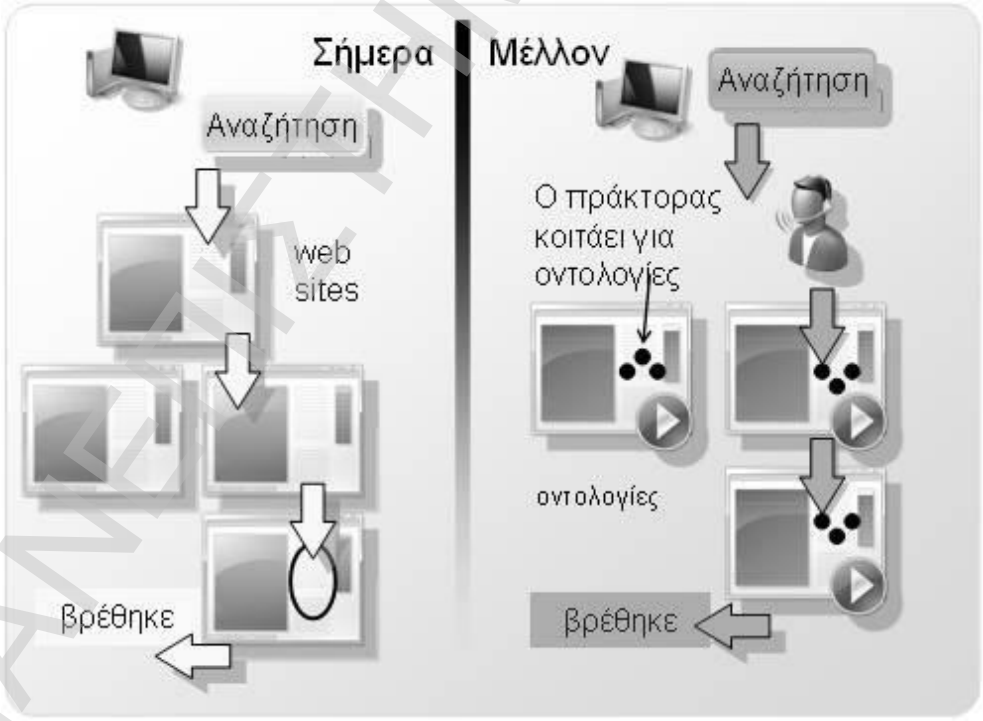
- Ευελιξία: Οι πράκτορες πρέπει να μπορούν να λειτουργούν χωρίς προβλήματα σε διάφορες πλατφόρμες, δίκτυα και λειτουργικά συστήματα και να έχουν και την ικανότητα να αυτό-διορθώνονται χωρίς να επεμβαίνουν οι χρήστες τους.
- Διαφάνεια: Αν κρίνεται απαραίτητο, οι πράκτορες πρέπει να είναι απόλυτα διαφανείς για τους χρήστες τους και να υπάρχει και η δυνατότητα τεκμηρίωσης των πράξεων τους(που ήταν, τι έκαναν, με ποιους επικοινώνησαν).
- Ικανότητα να αντιδρούν: Πρέπει να μπορούν να αντιμετωπίζουν σφάλματα, ελλιπείς πηγές, εξυπηρετητές χαμηλής απόδοσης και ελλιπή δεδομένα. Επίσης, πρέπει να αντιδρούν στις αλλαγές του περιβάλλοντος σε εύλογο χρονικό διάστημα.
- Αυτονομία: Πρέπει να πράττουν αυτόνομα στην έναρξη και διεκπεραίωση των αποστολών τους, ασχέτως αν πρόκειται για ωριαία, εβδομαδιαία κτλ. γεγονότα. Επομένως, έχουν δυνατότητες επιλογής εργασίας, προτεραιοποίησης και λήψης αποφάσεων χωρίς ανθρώπινη παρέμβαση.
- Εστίαση στο χρήστη: Ο πράκτορας πρέπει να ενεργεί με βάση τα ενδιαφέροντα του χρήστη, χωρίς παρακάμψεις από αυτά.
- Ικανότητα μάθησης και υιοθέτησης γνώσεων: Ένας έξυπνος πράκτορας θα πρέπει να τροποποιεί τη συμπεριφορά του αναλόγως τις προγενέστερες εμπειρίες του, δηλαδή πρέπει να διαισθάνεται το περιβάλλον και να αναδιαμορφώνεται ανάλογα. Αυτό μπορεί να επιτευχθεί με εναλλακτικούς αλγορίθμους. Από την άλλη μεριά, η μάθηση μπορεί να επιτευχθεί μέσω δοκιμής (by trial and error) ή μέσω παραδειγματισμού και γενίκευσης (by example and generalization).
- Επικοινωνία: Οι πράκτορες πρέπει να επικοινωνούν μεταξύ τους και αν κρίνεται σκόπιμο και με τους χρήστες. Μ’ αυτό τον τρόπο προωθείται η συνεργασία στην εκτέλεση των εργασιών.

Ενδεικτικές εφαρμογές πρακτόρων λογισμικού

Οι πράκτορες μπορούν να χρησιμοποιηθούν σε διάφορες καθημερινές εργασίες, από απλή πλοήγηση και αναζήτηση πληροφορίας στο διαδίκτυο μέχρι εφαρμογές υγείας όπως προγραμματισμός ασθενών και η πρόσβαση σε στοιχεία ιατρικών φακέλων σε ένα νοσηλευτικό ίδρυμα [17].

Η επόμενη εικόνα απεικονίζει ένα παράδειγμα αναζήτησης στο διαδίκτυο με τη χρήση πρακτόρων λογισμικού. Σήμερα το διαδίκτυο αποτελεί κυρίως από μη δομημένο περιεχόμενο. Στο μέλλον λοιπόν προβλέπεται οι σελίδες να σημειώνονται αναλόγως με τη σχετικότητά τους. Οι πράκτορες θα αναγνωρίζουν αυτόματα τα συγκεκριμένα tags των οποίων η δομή θα ορίζεται από οντολογίες. Έτσι μέσα από συγκεκριμένες βάσεις δεδομένων που θα περιέχουν αυτά τα tags, η αναζήτηση θα είναι πιο αξιόπιστη και πιο γρήγορη.

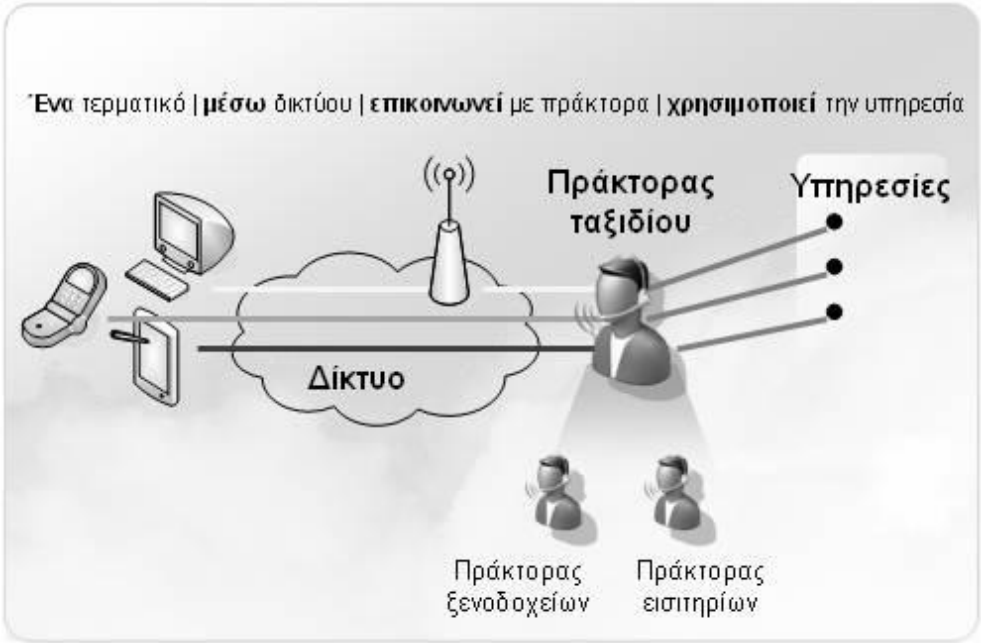
Σχήμα 2.1: Αναζήτηση περιεχομένου με και χωρίς πράκτορες



Άλλο χαρακτηριστικό παράδειγμα είναι η χρήση ενός πράκτορα ταξιδίου (σχήμα 2.2) το οποίο ασχολείται και υπηρεσίες ταξιδίων και είναι πολυπρακτορικό σύστημα (δηλ. επικοινωνεί και με άλλους πράκτορες για την έκβαση του τελικού αποτελέσματος-στόχου). Ο βασικός στόχος του είναι ο σχεδιασμός ταξιδίου του χρήστη. Ο πράκτορας έχει τη δυνατότητα να αναζητήσει αεροπορικές συνδέσεις, να κλείσει δωμάτια ξενοδοχείων, εστιατόρια κτλ. σύμφωνα με τις προτιμήσεις που έχει προεπιλέξει ο εκάστοτε χρήστης.

Επομένως ο χρήστης χρησιμοποιεί μία κινητή συσκευή ή υπολογιστή, γενικά ένα τερματικό, το οποίο μέσω του δικτύου επικοινωνεί με τον βασικό πράκτορα ταξιδίου ο οποίος χρησιμοποιεί τις σχετικές υπηρεσίες. Μετέπειτα, ο βασικός πράκτορας ταξιδίου αναλαμβάνει να επικοινωνήσει με τους πράκτορες ξενοδοχείων (για κλείσιμο δωματίων), με τους πράκτορες εισιτηρίων (για αναζήτηση αεροπορικών και άλλων εισιτηρίων) κτλ.

Σχήμα 2.2: Ενδεικτικός πράκτορας ταξιδίων



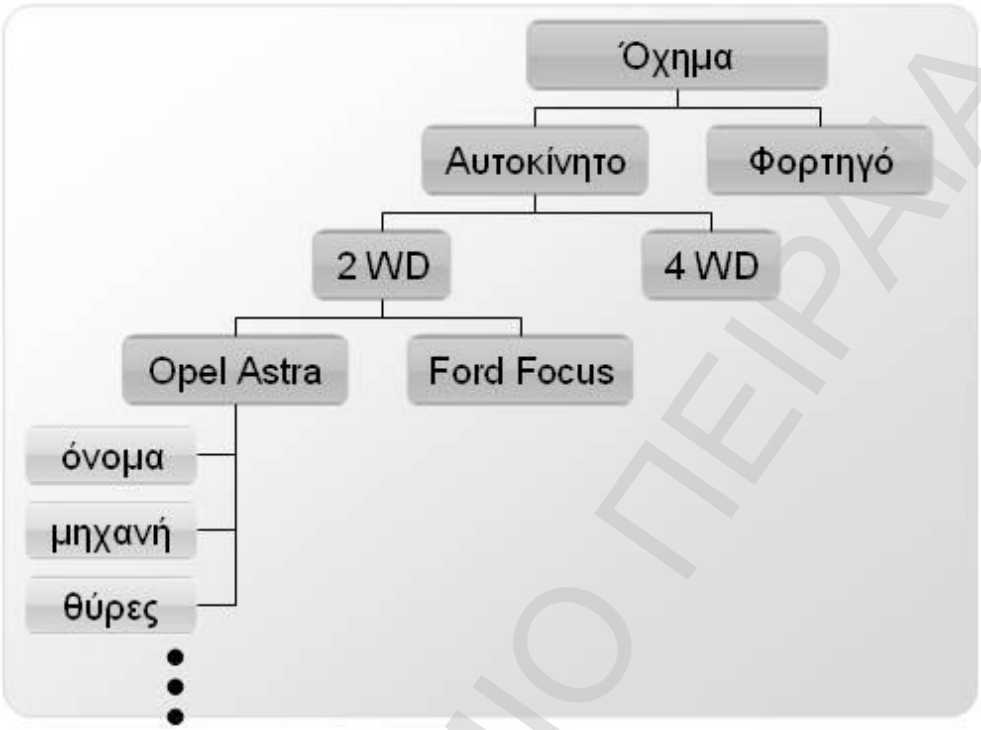
2.3 Οντολογίες

Στην επιστήμη των υπολογιστών ως οντολογία ορίζεται το μοντέλο δεδομένων το οποίο αντιπροσωπεύει μία ομάδα ομοειδών στοιχείων και τη συσχέτιση αυτών των στοιχείων. Αποτελεί βολικό τρόπο για την ομαδοποίηση γνώσης για διάφορους λόγους [7]. Οι οντολογίες μπορούν να εφαρμοστούν στην τεχνητή νοημοσύνη, τον σημασιολογικό ιστό, την ανάπτυξη λογισμικού και άλλα πεδία. Οι οντολογίες χρησιμοποιούν:

- Κλάσεις
- Αντικείμενα
- Παραμέτρους
- Σχέσεις
- Γεγονότα
- Εκφάνσεις των αντικειμένων

Παρακάτω βλέπουμε μία ενδεικτική οντολογία για κατηγοριοποίηση και δόμηση των οχημάτων. Η γονική κλάση είναι το Όχημα, ενώ το Αυτοκίνητο και Φορτηγό αποτελούν υποκλάσεις της βασικής γονικής κλάσης των Οχημάτων. Τα Αυτοκίνητα μπορούν με τη σειρά τους να χωριστούν σε αυτά με κίνηση σε 2 Τροχούς και αυτά με κίνηση σε 4 Τροχούς. Τα αυτοκίνητα με κίνηση σε 2 Τροχούς περιλαμβάνουν για παράδειγμα τα μοντέλα του Opel Astra και του Ford Focus. Τα προαναφερθέντα μοντέλα αυτοκινήτων περιλαμβάνουν τις εξής παραμέτρους: όνομα, μηχανή, θύρες κ.ά.

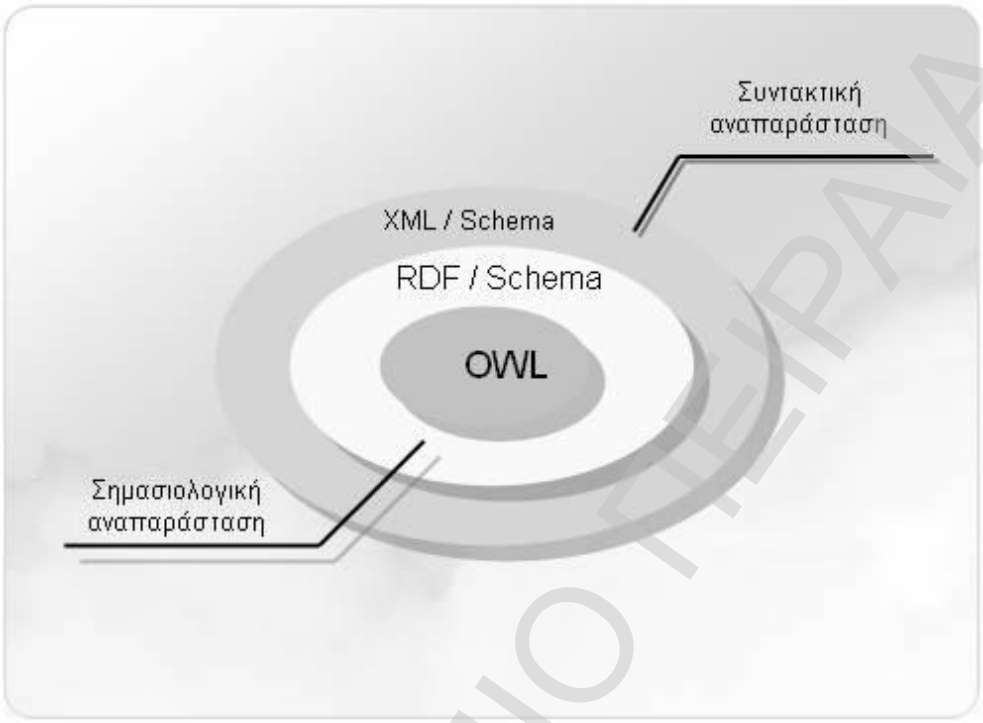
Σχήμα 2.3: Οντολογία Οχημάτων



Για μα δημιουργήσουμε οντολογίες πρέπει να χρησιμοποιηθούν συγκεκριμένες γλώσσες ορισμού οντολογιών. Η γλώσσα αυτή είναι μία επίσημη γλώσσα που χρησιμοποιείται για την κωδικοποίηση της οντολογίας. Υπάρχουν διάφορες γλώσσες οντολογίας για την περιγραφή αντικειμένων, σχέσεων και παραμέτρων. Παράδειγμα τέτοιας γλώσσας αποτελεί η Ontology Web Language (OWL) σύμφωνα με την οποία δηλώνονται οι οντολογίες.

Η OWL προορίζεται για χρήση στο Παγκόσμιο Ιστό και όλα τα στοιχεία της (π.χ. κλάσεις κτλ.) ορίζονται ως RDF (Resource Description Framework) πηγές και αναγνωρίζονται από URIs (Uniform Resource Identifiers).

Σχήμα 2.4: Ορισμός RDF και OWL επιπέδων



Όπως φαίνεται και στην παραπάνω εικόνα, RDF και OWL τα οποία αποτελούν τη σημασιολογική απεικόνιση μιας οντολογίας χρησιμοποιούν τη σύνταξη της XML [14].

Σημαντικότητα οντολογιών

Οι οντολογίες αποδεικνύονται ιδιαίτερα σημαντικές διότι:

- Υπάρχει η ανάγκη κατηγοριοποίησης της γνώσης με ακρίβεια και ευκολία για τους χρήστες.
- Υπάρχει η ανάγκη προτυποποίησης της βάσης γνώσης ώστε να υπάρχει η δυνατότητα επαναχρησιμοποίησης σε διάφορες εφαρμογές.
- Υπάρχει και η ανάγκη ανάπτυξης αντικειμενοστραφών εφαρμογών λογισμικού προς κατανόηση των οντολογιακών αρχών [15].

Απαιτήσεις οντολογιών

Μια οντολογία για να είναι αποτελεσματική πρέπει να καλύπτει μια σειρά απαιτήσεων όπως:

- **Χαλαρή διασύνδεση μερών:** Μια οντολογία δεν είναι απαραίτητο να έχει κοινά στοιχεία με μια άλλη οντολογία (π.χ. άλλη λογική, άλλες κλάσεις, αντικείμενα, σχέσεις αντικειμένων κτλ.)
- **Ανεξάρτητη:** Για να πετύχουμε την επαναχρησιμοποίηση των αρθρωμάτων, πρέπει να διασφαλίσουμε ότι δεν απαιτείται η πρόσβαση σε άλλα αρθρώματα για την εκτέλεση κάποιου ερωτήματος.
- **Ακεραιότητα:** Πρέπει να υπάρχουν τέτοιοι μηχανισμοί ώστε να ελέγχεται η αλλαγή σε σχετικά αρθρώματα άλλων συστημάτων ώστε να διασφαλιστεί η ορθότητα της οντολογίας [34].

2.4 JADE

Βασικό ιστορικό του JADE

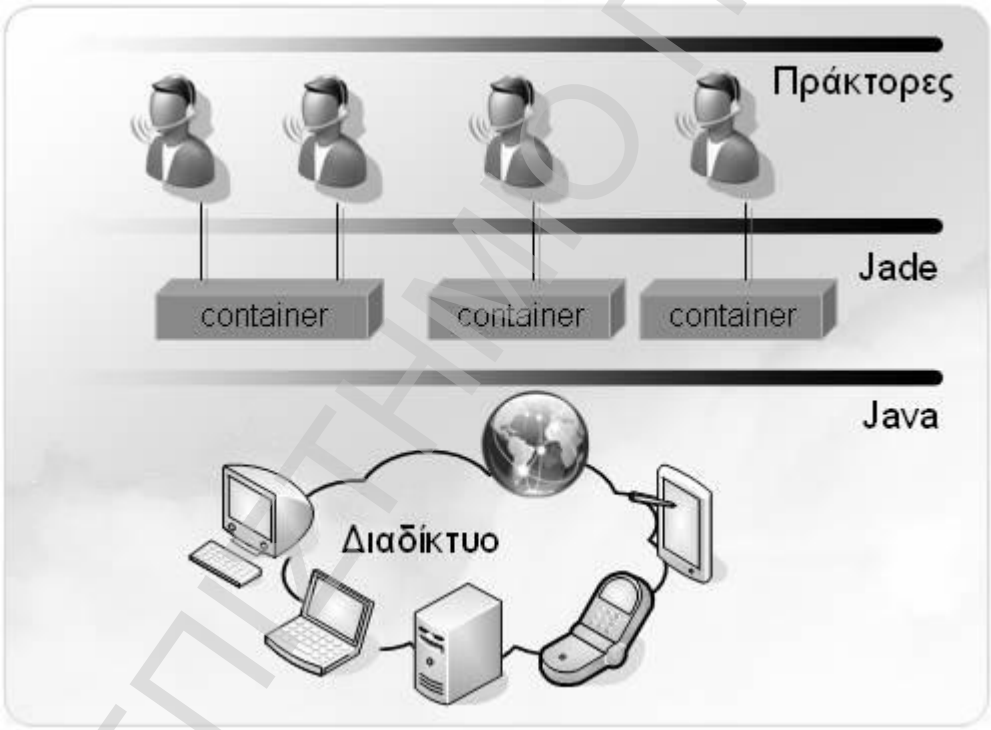
Οι πρώτες εφαρμογές της πλατφόρμας Jade έγιναν την περίοδο 1998-1999 από τα εργαστήρια της Telecom Italia. Αυτό που τους παρακίνησε στο να προχωρήσουν σε κάτι τέτοιο, ήταν το όραμα παροχής υπηρεσιών στους δημιουργούς εφαρμογών με λιγοστές ή καθόλου γνώσεις των προτύπων της FIPA. Γύρω στο 2000, το JADE έγινε λογισμικό ανοικτού κώδικα (open source). Τα αρχικά του σημαίνουν “Java Agent Development Environment”.

Πλαίσιο και αρχιτεκτονική του JADE

Το πλαίσιο του JADE υποστηρίζει την ανάπτυξη ολοκληρωμένων εφαρμογών βασισμένων σε πράκτορες λογισμικού παρέχοντας το κατάλληλο runtime περιβάλλον καθώς και το γραφικό περιβάλλον για την απλούστερη χρήση της πλατφόρμας. Το JADE είναι δωρεάν προς χρήση λογισμικό (freeware) με ευέλικτη υποδομή η οποία επιτρέπει την εύκολη επέκτασή του με αρθρώματα επεκτάσεων και έχει αναπτυχθεί ολοκληρωτικά στη γλώσσα Java. Μέσω απλών διεπαφών γίνεται ευκολότερη η δημιουργία πολυπρακτορικών συστημάτων (multi-agent systems) υπό τις οδηγίες της FIPA. Η πλατφόρμα JADE είναι ανεξάρτητη από λειτουργικά συστήματα και μπορεί να κατανεμηθεί σε διάφορους ηλεκτρονικούς υπολογιστές με διαφορετικά λειτουργικά συστήματα και ο έλεγχος του συστήματος μπορεί να επιτευχθεί μέσω απομακρυσμένων γραφικών διεπαφών (remote graphical interface). Επίσης μέσω των εργαλείων που παρέχονται στο πακέτο, ο εντοπισμός σφαλμάτων και η υλοποίηση γίνονται ευκολότερα.

Επιπροσθέτως το JADE αποτελείται από διάφορους πράκτορες. Οι πρακτορές χρησιμοποιούν και αλληλεπιδρούν με τα containers, τα οποία είναι Java διαδικασίες που παρέχουν το JADE runtime και όλες τις υπηρεσίες που χρειάζονται για την φιλοξενία και εκτέλεση πρακτόρων. Το βασικό (main) container είναι ένα ειδικό container το οποίο ξεκινάει μαζί με την πλατφόρμα. Όλα τα άλλα containers πρέπει να ενταχθούν στο βασικό container πραγματοποιώντας εγγραφή.

Σχήμα 2.5: Αρχιτεκτονική του JADE



Το παραπάνω σχήμα απεικονίζει την αρχιτεκτονική του JADE. Στο κάτω επίπεδο βλέπουμε τα διάφορα τερματικά (Η/Υ, κινητά τηλέφωνα, PDAs κτλ.) τα οποία χρησιμοποιούν τη Java τεχνολογία για την παροχή περιεχομένου μέσω κάποιου δικτύου (π.χ. Internet κτλ.). Στο επόμενο επίπεδο έχουμε το binding της JADE πλατφόρμας η οποία χρησιμοποιεί τα containers. Στο ανώτατο επίπεδο υπάρχουν οι πράκτορες οι οποίοι προσκολλούν στα containers τα οποία δημιουργήθηκαν στο επίπεδο του JADE.

2.5 Foundation for Intelligent Physical Agents (FIPA)

Από το 1996 το Ίδρυμα Έξυπνων Φυσικών Πρακτόρων (Foundation for Intelligent Physical Agents –FIPA) είναι ένας μη-κερδοσκοπικός και διεθνής οργανισμός με έδρα τη Γενεύη της Ελβετίας. Οι προτεραιότητες του περιλαμβάνουν την προώθηση των εφαρμογών και υπηρεσιών βασισμένων σε πράκτορες λογισμικού. Για την επίτευξη του ανωτέρω στόχου, παρέχονται πρότυπα και οδηγίες ώστε να επιτευχθεί η χρησιμότητα των εφαρμογών που βασίζονται σε πράκτορες. Αυτά τα πρότυπα παρέχονται δωρεάν στο κοινό. Η πλατφόρμα JADE χρησιμοποιεί ήδη τα πρότυπα της FIPA για την ανταλλαγή μηνυμάτων (δηλ. την επικοινωνία μεταξύ πρακτόρων) και τη γενικότερη λειτουργία των πρακτόρων λογισμικού. Οι βασικές αρχές που διέπουν τη FIPA μπορούν να συνοψισθούν στα εξής:

- Οι πράκτορες μπορούν να δημιουργήσουν μία νέα βάση στη λύση παλαιοτέρων αλλά και νέων προβλημάτων.
- Κάποιες τεχνολογίες πρακτόρων έχουν φθάσει σε ικανοποιητικό βαθμό ωριμότητας.
- Κάποιες άλλες τεχνολογίες πρέπει να προτυποποιηθούν ώστε να αποδειχθούν χρήσιμες.
- Η υποδομή και η γλώσσα των πρακτόρων χρειάζεται περαιτέρω προτυποποίηση.

Η επικοινωνία των πρακτόρων εμπεριέχει την έννοια του μηνύματος FIPA-ACL (Agent Communication Language –Γλώσσα Επικοινωνίας Πρακτόρων). Αυτή η έννοια αντιπροσωπεύει τις απαραίτητες πράξεις επικοινωνίας (communicative acts) όπως ***inform, request, agree, not understood,*** και ***refuse*** σύμφωνα με το μοντέλο BDI (Belief, Desires, Intention). Αν το λαμβανόμενο μήνυμα δεν μπορεί να επεξεργασθεί από το σύστημα τότε το σύστημα πρέπει να απαντήσει τουλαχιστόν με ένα ***not understood*** μήνυμα σύμφωνα με τους κανονισμούς της FIPA. Κάποιοι παράμετροι των FIPA-ACL μηνυμάτων φαίνονται παρακάτω:

Πίνακας 2.1: Παράμετροι FIPA-ACL μηνυμάτων	
performative	Type of the communicative act of the message
sender	Participant in communication
receiver	Participant in communication
reply-to	Which agent to send subsequent messages to within a conversation thread
content	Content of message
language	Description of Content
encoding	Description of Content
ontology	Reference to an ontology to give meaning to symbols in the message content
protocol	Control of conversation
conversation-id	Control of conversation
reply-with	An expression to be used by a responding agent to identify the message
in-reply-to	Reference to an earlier action to which the message is a reply
reply-by	Timestamp

Πηγή: Foundation for Intelligent Physical Agents (2002).

Ακολουθεί δείγμα ενός FIPA-ACL μηνύματος (*request*):

Σχήμα 2.6: Παράδειγμα FIPA-ACL μηνύματος

Πίνακας 2.2: Επικοινωνιακές ενέργειες FIPA	
accept-proposal	The action of accepting a previously submitted proposal to perform an action
agree	The action of agreeing to perform some action
cancel	The action of one agent informing another agent that the first agent no longer has the intention that the second agent performs some action
call-for-proposal cfp	The action of calling for proposals to perform a given action
confirm	The sender informs the receiver that a given request is true
disconfirm	The sender informs that a given request is false
failure	The action of telling another agent that an action failed
inform	The sender informs the receiver that a request is true
inform-if	An action to inform the recipient whether or not a request is true
inform-ref	An action enabling the sender to inform the receiver of some object e.g. a name
not-understood	When the message received is not valid according to the FIPA specification
propagate	The sender intends that the receiver treat the embedded message as sent directly to the receiver, and wants the receiver to identify the agents denoted by the given descriptor and send the received propagate message to them
propose	The action of submitting a proposal to perform a certain action
proxy	The sender wants the receiver to select target agents denoted by a given description and to send an embedded message to them
query-if	The action of asking another agent whether or not a given request is true
query-ref	The action of asking another agent for the object referred to by an expression
refuse	The action of refusing to perform a given action, and explaining the reason why
reject-proposal	The action of rejecting a proposal
request	The sender requests the receiver to perform an action
request-when	The sender wants the receiver to perform some action when a given expression becomes true
request-whenever	The sender wants the receiver to perform some action everytime a given expression becomes true
subscribe	The act of requesting a persistent intention to notify the sender of the value of a reference

Πηγή: Foundation for Intelligent Physical Agents (2002).

2.6 JADEX

Μοντέλο Jadex BDI

Το Jadex χρησιμοποιεί το μοντέλο BDI εμφανίζοντας διάφορα beliefs, goals και plans τα οποία μπορούν να δημιουργηθούν και να ελεγχθούν από κάποιον πράκτορα λογισμικού. Στο Jadex έχουμε τη δυνατότητα δημιουργίας έξυπνων πρακτόρων λογισμικού βασιζόμενοι στην XML και τη Java. Επιπροσθέτως, οι πράκτορες του Jadex έχουν αυξημένες δυνατότητες συγκρινόμενοι με τους Jade πράκτορες επειδή στο Jadex χρησιμοποιούνται beliefs, goals και plans τα οποία μπορούν να ελεγχθούν μέσω του πράκτορα, έτσι αντί ο προγραμματιστής να ζητήσει απευθείας από τον πράκτορα να εκτελέσει κάτι, έχει τη δυνατότητα να συμπεριλάβει αφηρημένους στόχους για τον πράκτορα και να δώσει κάποιο βαθμό ευελιξίας –ελαστικότητας στο τρόπο με τον οποίο θα επιτευχθούν οι στόχοι. Αυτός είναι και ο βασικός λόγος που έγινε η μετάβαση από απλούς Jade πράκτορες λογισμικού σε Jadex πράκτορες.

Το BDI (Belief Desire Intention) μοντέλο παρουσιάστηκε πρώτα από τον Bratman προς τα τέλη της δεκαετίας του ’80 ως μοντέλο περιγραφής πρακτόρων. Οι belief, desire και intention είναι τρεις έννοιες οι οποίες λειτουργούν ως πνευματικές συμπεριφορές.

- Τα beliefs εμπεριέχουν πληροφοριακές συμπεριφορές
- Τα desires εμπεριέχουν συμπεριφορές παρακίνησης
- Τα intentions εμπεριέχουν συμβουλευτικές συμπεριφορές πρακτόρων [22].

Στο Jadex το μοντέλο BDI αντιπροσωπεύεται από τα beliefs, goals και plans. Για να είμαστε πιο συγκεκριμένοι τα beliefs μπορούν οποιοδήποτε αντικείμενο της Java. Τα goals αντιπροσωπεύουν την παρακίνηση προς την επίτευξη κάποιας ενέργειας. Για την επίτευξη των στόχων, ο εκάστοτε πράκτορας πρέπει να εκτελέσει κάποια πλάνα τα οποία έχουν αναπτυχθεί σε Java. Τα beliefs, goals και plans δημιουργούνται από τον προγραμματιστή και καθορίζουν τη συμπεριφορά των πρακτόρων λογισμικού.

Beliefs (πληροφοριακές συμπεριφορές)

Πρόκειται για μία απλή σχετικά διαδικασία στο Jadex. Μια beliefbase περιέχει αλφαριθμητικά τα οποία χρησιμοποιούμε για την αναγνώριση συγκεκριμένων belief όπως χρησιμοποιούμε τα ονόματα πινάκων σε μια βάσης δεδομένων. Αυτά τα αναγνωριστικά είναι συνδεδεμένα με τιμές των beliefs οι οποίες αποκαλούνται facts και αυτές μπορούν να είναι Java αντικείμενα. Απλά τέτοια facts φαίνονται στο παράδειγμα που ακολουθεί. Επίσης, η beliefbase ελέγχει αν τα αντικείμενα που έχουν αποθηκευτεί είναι σωστά πληκτρολογημένα [22].

Ακολουθεί παράδειγμα “beliefs” το οποίο τυπώνει σε συγκεκριμένο σημείο του πράκτορα το μήνυμα "Agent CommA is up and running." :

```
<beliefs>
	<belief name="msg" class="String" exported="true">
		<fact>"Agent CommA is up and running."</fact>
	</belief>
</beliefs>
```

Goals (συμπεριφορές παρακίνησης)

Τα goals στο Jadex έχουν ιδιαίτερα σημαντικό ρόλο στη διαδικασία εκτέλεσης των πρακτόρων λογισμικού. Ο εκάστοτε πράκτορας ενεργεί αναλόγως μέχρι να επιτύχει κάποιο ‘επιθυμητό’ ή ‘μη επιθυμητό πλέον’ σημείο. Όταν υιοθετηθεί ο στόχος, μετατρέπεται σε επιλογή η οποία προστίθεται στη δομή επιθυμιών του πράκτορα λογισμικού. Στην περίπτωση ενός άκυρου στόχου, τότε έχουμε παύση λειτουργίας μέχρι την επανεκτίμησή του. Οι στόχοι μπορούν να χωρισθούν σε:

- performgoal: χρησιμοποιείται για συγκεκριμένες δραστηριότητες
- achievegoal: χρησιμοποιείται για να πετύχουμε μια κατάσταση
- querygoal: χρησιμοποιείται για τη λήψη πληροφοριών
- maintaingoal: παρακολουθεί μια κατάσταση και αν παρατηρηθεί απόκλιση από αυτή τότε επιστρέφουμε στην αρχική, σωστή κατάσταση
- metagoal: όταν ένας στόχος ή ένα γεγονός εκτελείται και ακολουθούν περισσότερα από ένα plans τα οποία ταιριάζουν, τότε δημιουργείται το κατάλληλο metagoal του goal / event.

Ακολουθεί κάποιο παράδειγμα ενός “performgoal” :
Ο στόχος μπορεί να ξανακάνει κάποιες δραστηριότητες (retry=true) και δεν επιτρέπει να απορριφθεί κάποιο plan (exclude=never). Το σύμβολο “&” τοποθετείται στη θέση του χαρακτήρα “&”

```
<performgoal name="patrol" retry="true" exclude="never">
    <contextcondition>
          !$beliefbase.is_loading &amp;&amp; !$beliefbase.daytime
    </contextcondition>
</performgoal>
```

Plans (συμβουλευτικές συμπεριφορές)

Τα plans περιγράφουν ενέργειες στις οποίες πρέπει να προβεί ο εκάστοτε πράκτορας λογισμικού προς την επίτευξη του στόχου. Ο προγραμματιστής ορίζει το head και το body του plan. Στην κεφαλίδα ορίζονται οι όροι σύμφωνα με τους οποίους ξεκινάει να εκτελείται το plan ενώ στο σώμα ορίζονται οι ενέργειες που πρέπει να γίνουν για την ολοκλήρωση του στόχου. Το σώμα του πλάνου συντάσσεται σε Java.

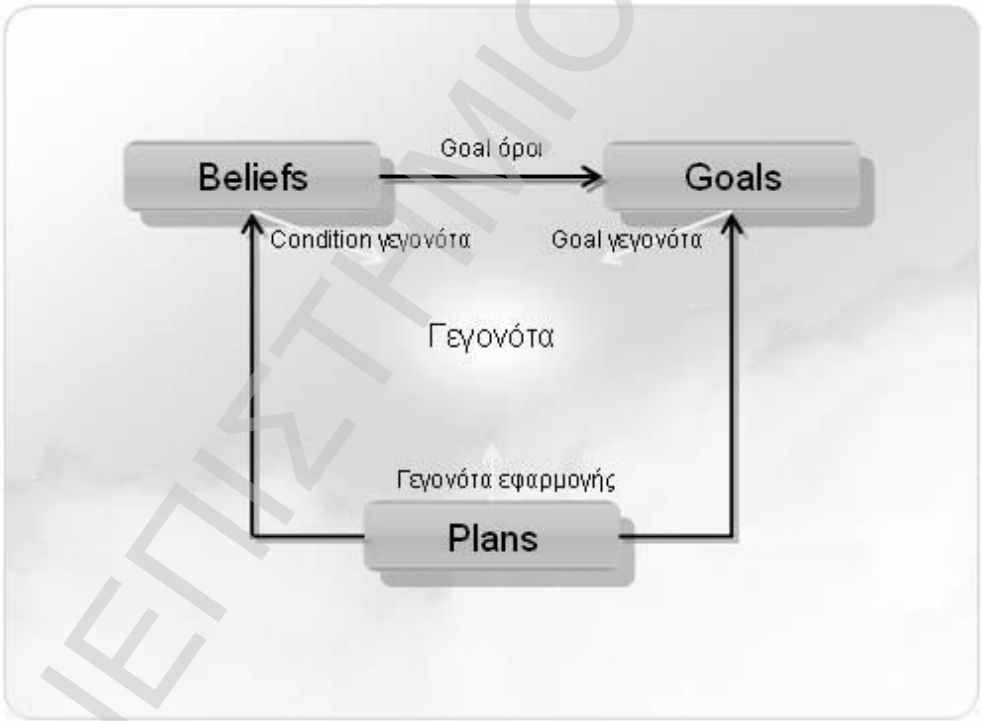
Ακολουθεί δείγμα των “plans” :
Το XML plan επικοινωνεί με το CommPlanA το οποίο βασίζεται σε Java και χρησιμοποιεί δύο γεγονότα μηνυμάτων (request_msg, request_msg2).

```
<plans>
	<plan name="myplanA">
		<body class="CommPlanA"/>
		<trigger>
			<messageevent ref="request_msg"/>
			<messageevent ref="request_msg2"/>
		</trigger>
	</plan>
</plans>
```

Αρχιτεκτονική του Jadex

Το παρακάτω σχήμα απεικονίζει τη βασική αρχιτεκτονική του Jadex στην οποία φαίνεται η διάδραση των beliefs, goals και plans για την εκτέλεση γεγονότων. Τα plans είναι απλές Java κλάσεις οι οποίες μπορούν να προσπελάσουν και να τροποποιήσουν τα beliefs του πράκτορα. Επίσης πρέπει να δημιουργήσουμε και ένα Agent Definition File (ADF) σε XML ώστε να ορίσουμε τα αρχικά beliefs, goals και plans του πράκτορα. Μετά η μηχανή του Jadex διαβάζει το ADF και εκτελεί τον πράκτορα σύμφωνα με τους προκαθορισμένους στόχους.

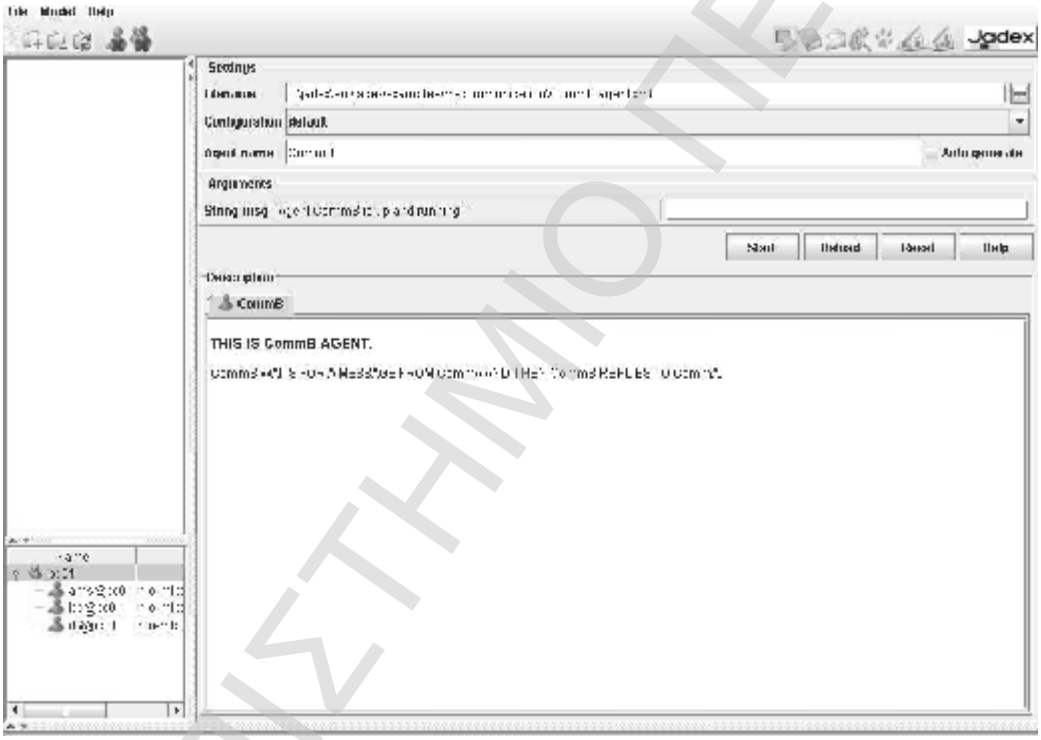
Σχήμα 2.7: Βασική αρχιτεκτονική του Jadex



Περιβάλλον εργασίας του Jadex

Το περιβάλλον εργασίας του Jadex αποτελείται από ένα σχετικά απλό γραφικό περιβάλλον όπως φαίνεται παρακάτω. Από τη συγκεκριμένη κονσόλα δίδεται η δυνατότητα στο χρήστη να ξεκινά νέους πράκτορες και να παρακολουθεί τους πράκτορες που έχουν ήδη ξεκινήσει να εκτελούνται.

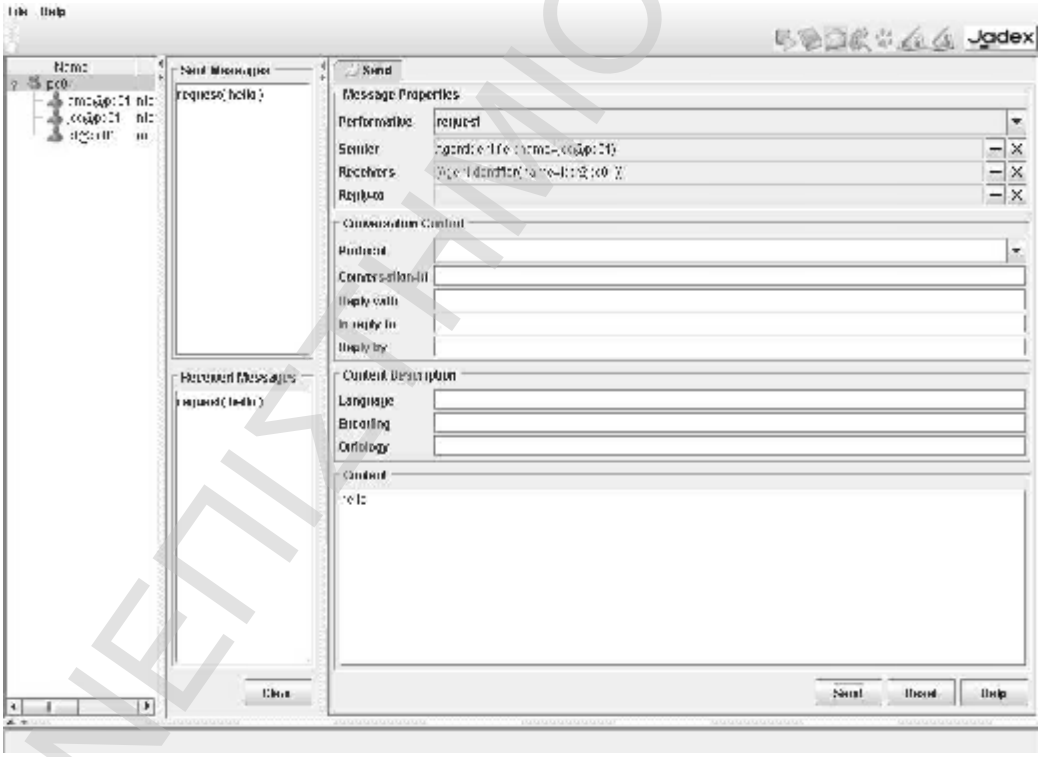
Σχήμα 2.8: Περιβάλλον εργασίας του Jadex



Κέντρο διαλόγου του Jadex

Μέσω του κέντρου διαλόγου του Jadex μπορούμε να στείλουμε μηνύματα σε άλλους πράκτορες. Επίσης υπάρχει η δυνατότητα καθορισμού του τύπου του μηνύματος (π.χ. request, query κτλ.), τον αποστολέα πράκτορα, τον παραλήπτη πράκτορα, το πρωτόκολλο επικοινωνίας, τον ξεχωριστό κωδικό της συνομιλίας, την κωδικοποίηση και φυσικά το περιεχόμενο του μηνύματος. Αν οι παράμετροι έχουν ορισθεί σωστά τότε εγγράφεται το μήνυμα στο 'Sent Messages' textarea και η απάντηση εμφανίζεται στο 'Received Messages' textarea. Η δομή του μηνύματος ακολουθεί τα προαναφερθέντα πρότυπα του FIPA.

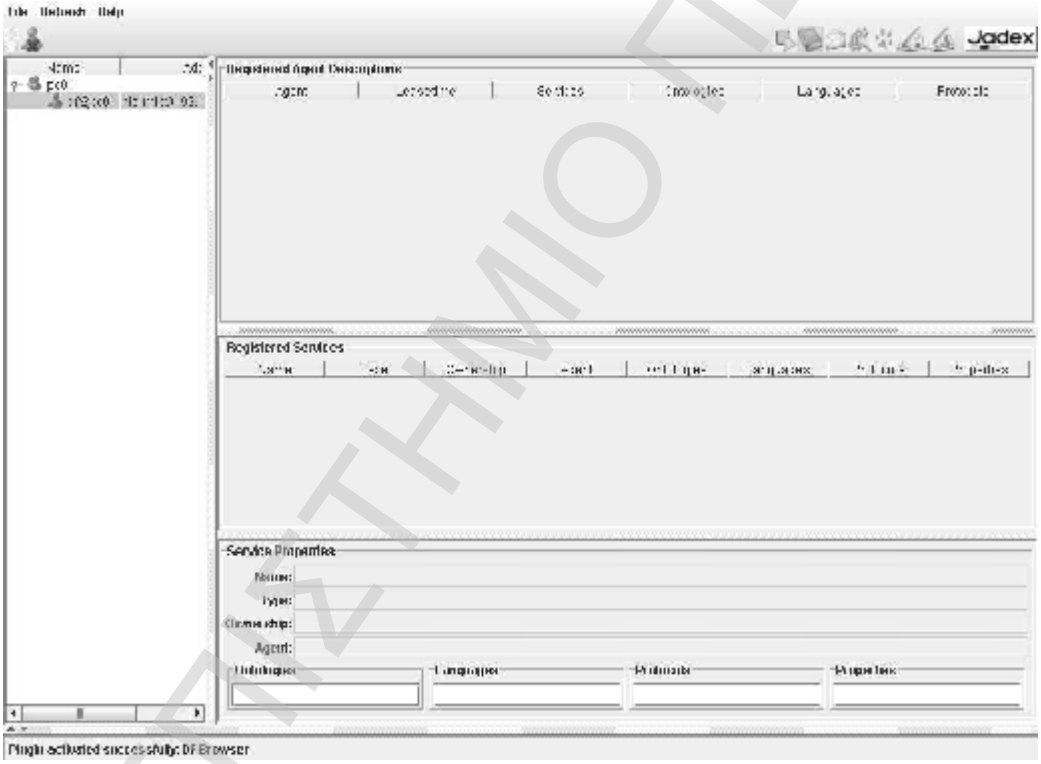
Σχήμα 2.9: Κέντρο διαλόγου του Jadex



Jadex DF Browser

Για την παρακολούθηση κάποιου πράκτορα ο όποιος έχει ήδη ξεκινήσει να εκτελείται μπορούμε να χρησιμοποιήσουμε και το ‘DF Browser’. Από εκεί μπορούμε να δούμε τις υπηρεσίες και τους πράκτορες που εκτελούνται, ενώ το περιβάλλον απεικόνισης μπορεί να ανανεώνεται αυτόματα ή χειροκίνητα.

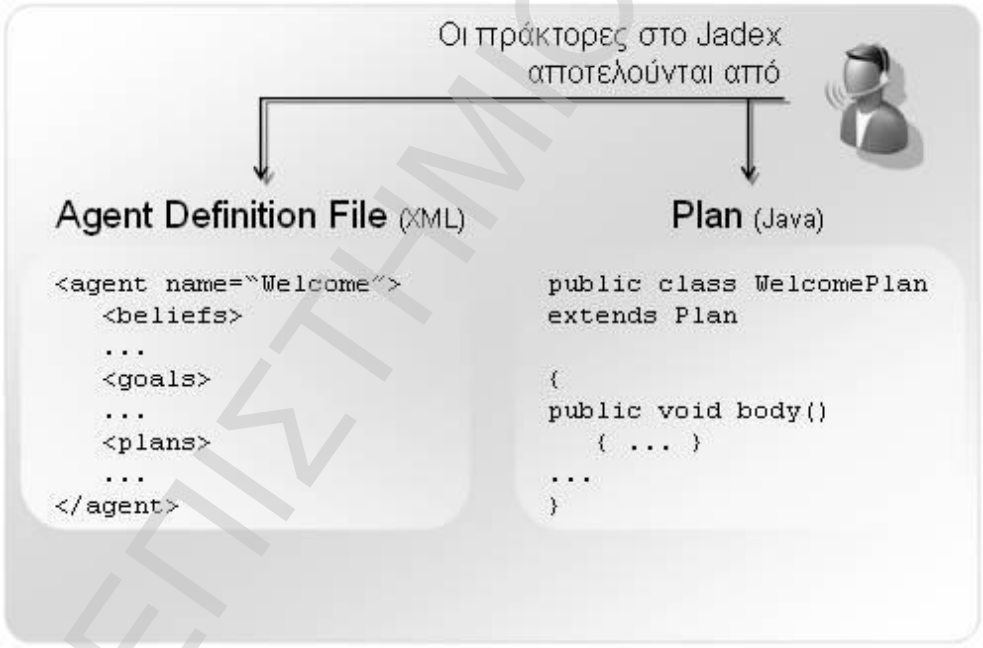
Σχήμα 2.10: Jadex DF Browser



2.7 Πράκτορες στο JADEX

Οι πράκτορες στο Jadex χρησιμοποιούν ‘Agent Definition Files’ (ADFs) και ‘Plans’. Τα ADFs συντάσσονται σε XML ενώ τα Plans σε Java. Για να δημιουργήσουμε και να εκτελέσουμε κάποιον πράκτορα λογισμικού επιτυχώς πρέπει να χρησιμοποιήσουμε σωστά και τα δύο. Σε περίπτωση σφάλματος σε κάποιο από τα δύο, ο πράκτορας δεν είναι σε θέση να ξεκινήσει τη διαδικασία εκτέλεσης. Όταν φορτώνεται το ADF, δημιουργούνται Java αντικείμενα για τα διάφορα XML elements που ορίζονται στο ADF (π.χ. beliefs, goals, plans).

Σχήμα 2.11: Πράκτορες στο Jadex



Δομή ADF

Για να θεωρείται έγκυρο και σωστό συντακτικά το ADF πρέπει να ακολουθεί μια προκαθορισμένη δομή. Αυτή η δομή λοιπόν αρχικά περιλαμβάνει το agent name και σημαντικες πληροφορίες για το xml schema. Ακολουθεί η ετικέτα imports στην οποία μπορούμε να προσθέσουμε επιπλέον πακέτα που χρειάζονται για την ομαλή εκτέλεση του πράκτορα. Η ετικέτα capabilities χρησιμοποιείται για τη ρύθμιση των λειτουργιών του πράκτορα. Τα beliefs, goals και plans αποτελούν και τον πυρήνα του πράκτορα. Η ετικέτα events περιλαμβάνει κάποια γνωστά για τον πράκτορα γεγονότα. Η ετικέτα expressions επιτρέπει τον καθορισμό κάποιων πιθανών όρων, οι οποίοι μπορούν να χρησιμοποιηθούν ως προκαθορισμένοι όροι από τα plans. Η ετικέτα properties με τη σειρά της χρησιμεύει για ρυθμίσεις όπως debugging και logging ενώ η τελική configurations ετικέτα περιλαμβάνει ρυθμίσεις όπως initial beliefs ή/και end beliefs.

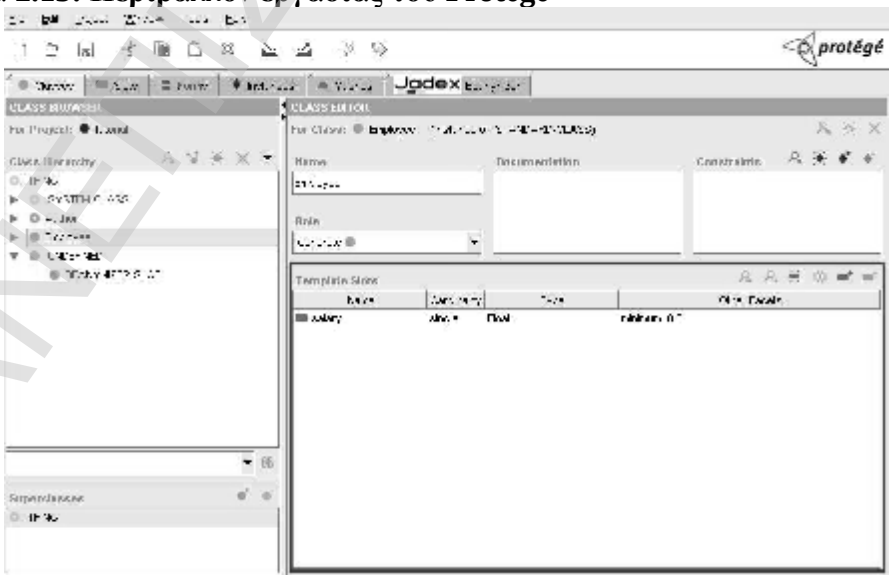
Σχήμα 2.12: Δομή ADF (με χρήση XML)

2.8 Protégé

Ένα κοινό εργαλείο για τη δημιουργία οντολογιών είναι το σύστημα Protege το οποίο περιγράφεται συνοπτικά παρακάτω. Έχει δημιουργηθεί και βασίζεται στη Java, ενώ το γραφικό περιβάλλον έχει δημιουργηθεί με τη χρήση του πακέτου γραφικών Swing της Java. Μέσω του απλού γραφικού περιβάλλοντος η δημιουργία πρωτοτύπων και εφαρμογών είναι ευκολότερη και ταχύτερη. Οι οντολογίες στο Protege μπορούν να εξαχθούν σε διάφορες μορφές όπως RDF/Schema, OWL, XML Schema ή ακόμα και σε Java (μέσω της επέκτασης Jadex beanynizer). Ο χρήστης μπορεί να δημιουργήσει και να διαχειριστεί:

- Κλάσεις (astract, concrete)
- Slots (κάτι σαν αντικείμενα της Java)
- Φόρμες
- Εκφάνσεις των αντικείμενων (τιμές των αντικειμένων)
- Ερωτήματα (για λήψη των τιμών των αντικείμενων)
- Εξαγωγή σε απλή Java ή Jade Java μέσω του Jadex Beanynizer plug-in

Σχήμα 2.13: Περιβάλλον εργασίας του Protégé



3. Υλοποίηση σε πραγματικές συνθήκες

3.1 Γενικά

Όπως αναφέρεται και στην εισαγωγή της μελέτης ο κύριος στόχος εντοπίζεται στη δημιουργία ενός συστήματος context-aware το οποίο θα έχει τη δυνατότητα να αναγνωρίζει τους χρήστες που εισέρχονται στην εμβέλεια ελέγχου του και θα εκτελεί τις ανάλογες υπηρεσίες. Στις επόμενες παραγράφους μελετούνται συνοπτικά τα κυριότερα ασύρματα και ενσύρματα πρότυπα επικοινωνίας που βρίσκονται αυτή τη στιγμή (2008) σε λειτουργία καθώς και οι απαιτούμενες τεχνολογίες αισθητήρων και συσκευών που θεωρούνται αναγκαίες για την υλοποίηση και λειτουργία του συστήματος σε πραγματικό περιβάλλον είτε αυτό είναι οικιακό, εργασιακό ή δημόσιο.

3.2 Σύγχρονα επικοινωνιακά πρότυπα

Για την υλοποίηση του εν λόγω συστήματος το ενδιαφέρον μας εστιάζεται στα επικοινωνιακά πρότυπα τα οποία χρησιμοποιούνται σε LANs/ WLANs καθώς και σε δίκτυα μικρής εμβέλειας Personal Area Networks (π.χ. περ. 10m). Αυτό είναι αναγκαίο επειδή ο κάθε αισθητήρας πρέπει να ελέγχει μόνο το δωμάτιο στο οποίο βρίσκεται –επομένως ενδιαφερόμαστε για τεχνολογίες που εφαρμόζονται σε PANs, σε αντίθεση με τον server ο οποίος πρέπει να συντονίζει όλες τις συσκευές του χώρου γενικότερα –άρα ενδιαφερόμαστε για τεχνολογίες που εφαρμόζονται σε WLANs. Ο πίνακας που ακολουθεί περιλαμβάνει μερικά από τα πιο διαδεδομένα πρότυπα:

Πίνακας 3.1: Σύγχρονα επικοινωνιακά πρότυπα (ενδεικτικός πίνακας)

Πρότυπο	Υλοποίηση	Συχν.(GHz)	Μεγ. bitrate (Mbit/s)	Εμβέλεια εσωτερική (m)
802.11a	1999	5	54	35
802.11b	1999	2.4	11	38
802.11g	2003	2.4	54	38
802.11n	2008	2.4 / 5	248	70
802.11y	2008	3.7	N/A	50
Bluetooth 1.0	1999	2.45	0.72	10
Bluetooth 1.1	2001	2.45	0.72	10
Bluetooth 1.2	2002	2.45	0.72	10
Bluetooth 2.0	2004	2.45	2.1	10
Bluetooth 2.1	2007	2.45	3	10
Bluetooth 3.0	Υπό μελέτη	6-9	100	10
Infrared	‘90s		16	1
ZigBee 1.0 802.15.4	2004	2.4	0.25	10-70
ZigBee 2007	2007	2.4	0.25	10-70
WUSB Certified	2006	3.1-10.6	480	3
WUSB Cypress	2007	2.4	1	10-50
WirelessHD	2008	3.1-10.6	480	10
Ethernet	1980	Wired	Variable	Wired
RFID Passive	‘90s	N/A	N/A	0.1-10-180

Πηγές: Επεξεργασμένα στοιχεία από Bluetooth Special Interest Group, IEEE Working Group for WLAN Standards, IEEE 802.15 WPAN Task Group 4, USB Implementers Forum, Cypress Semiconductor Corporation

ZigBee [IEEE 802.15.4]

Η έναρξη σχεδιασμού της τεχνολογίας ZigBee έγινε το 1998. Τότε δημιουργήθηκε η ανάγκη ενός νέου προτύπου που θα κάλυπτε το κενό του Bluetooth και του WiFi με ένα ad-hoc ασύρματο δίκτυο με δυνατότητες αυτο-οργάνωσης. Το νέο πρότυπο υπό την ονομασία 802.15.4 οριστικοποιήθηκε το 2003. Έκτοτε έχουν υπάρξει αρκετές βελτιώσεις, ενώ η λίστα των εταιριών που το υποστηρίζουν έχει αυξηθεί σημαντικά. Το ZigBee δημιουργεί ένα mesh ασύρματο δίκτυο χαμηλής κατανάλωσης ενέργειας. Παράλληλα το χαμηλό κόστος υλοποίησης επιτρέπει την ενσωμάτωσή του σε ασύρματες εφαρμογές, ενώ η χαμηλή κατανάλωση επιτρέπει την επιμήκυνση του χρόνου ζωής των μπαταριών. Αξιοσημείωτη είναι και η ικανότητα αξιόπιστης επικοινωνίας σε μεγαλύτερη ακτίνα από το Bluetooth. Οι χρήσεις για τις οποίες προορίζεται η τεχνολογία ZigBee ποικίλουν από οικιακές μέχρι βιομηχανικές. Ενδεικτικά αναφέρονται οι εξής [44]:

- Οικιακή χρήση και ψυχαγωγία:
  - έλεγχος εσωτερικού κλίματος –φωτισμός
  - συστήματα ασφάλειας
  - αναπαραγωγή ταινιών-μουσικής
  - αισθητήρες κίνησης, πυρόσβεσης, πρόσβασης, έξυπνων συσκευών
- Κινητές υπηρεσίες:
  - κινητές πληρωμές
  - κινητός έλεγχος και ασφάλεια
  - κινητές υπηρεσίες υγείας
  - κινητή υποστήριξη κ.ά.
- Εργασιακή χρήση:
  - ενεργειακός έλεγχος
  - έλεγχος πρόσβασης, ασφάλεια
- Βιομηχανική χρήση:
  - διαχείριση περιβάλλοντος
  - διαχείριση διαδικασιών παραγωγής
  - έλεγχος αποθεμάτων και κινήσεων κτλ.

Bluetooth [IEEE 802.15.1]

Το Bluetooth αποτελεί ένα πρωτόκολλο ασύρματης επικοινωνίας μικρής εμβέλειας, γι' αυτό το λόγο εντάσσεται στην κατηγορία των Personal Area Networks. Δίνει τη δυνατότητα μετάδοσης δεδομένων μεταξύ κινητών και σταθερών συσκευών, προσπερνώντας τα εμπόδια συγχρονισμού των εμπλεκομένων συσκευών. Επιπροσθέτως, έχει σχεδιαστεί να καταναλώνει χαμηλή ενέργεια και αναλόγως την κλάση μπορεί να εκπέμπει και να αναγνωρίζει άλλες συσκευές σε ακτίνες 1, 10 ή 100 μέτρων. Πιο συγκεκριμένα, η κλάση 1 εκπέμπει στα 100m με κατανάλωση 100mW, η κλάση 2 εκπέμπει στα 10m με κατανάλωση περίπου 2.5mW και τέλος η κλάση 3 στο 1m με κατανάλωση 1mW. Για να επικοινωνήσουν οι συσκευές εντός της προαναφερόμενης ακτίνας λειτουργίας, δεν είναι αναγκαίο να βρίσκονται σε οπτική ευθεία. Κάθε συσκευή κατά τη σύνδεση μπορεί να κληθεί να μεταδώσει στοιχεία όπως: ονομασία/αναγνωριστικό συσκευής, κλάση συσκευής και τεχνικές ιδιότητες (π.χ. κατασκευαστής, μοντέλο κτλ.). Ενδεικτικά αναφέρονται μερικές από τις ακόλουθες χρήσεις [45]:

- Ασύρματη επικοινωνία κινητών τηλεφώνων με περιφερειακές συσκευές π.χ. handfree, εκτυπωτές κτλ.
- Ασύρματη σύνδεση ηλεκτρονικών υπολογιστών με περιορισμένες απαιτήσεις σε bandwidth.
- Σε επικοινωνίες όπου πριν χρησιμοποιούνταν οι υπέρυθρες.
- Αντικατάσταση των κλασικών σειριακών επικοινωνιών μεταξύ GPS, ιατρικό εξοπλισμό, αναγνώστες γραμμικού κώδικα (barcode) κτλ.

Βασικές διαφορές με το WiFi είναι ότι στο WiFi χρειαζόμαστε ακριβότερο εξοπλισμό και περισσότερη κατανάλωση ενέργειας, αλλά από την άλλη στο WiFi υπάρχει η δυνατότητα σύνδεσης σε μεγαλύτερες αποστάσεις και με αυξημένους ρυθμούς μετάδοσης δεδομένων. Γι αυτό το λόγο το Bluetooth χρησιμοποιείται κυρίως για την αντικατάσταση προγενέστερων καλωδιωμένων μικρών συσκευών και εφαρμογών, σε αντίθεση με το WiFi που αντικαθιστά τα ενσύρματα LANs.

Wireless LAN [802.11a/b/g/n/y]

Το 802.11 περιλαμβάνει μία ομάδα προτύπων για τα ασύρματα δίκτυα υπολογιστών (WLANs) που αναπτύθηκαν από την IEEE και εκπέμπουν ελεύθερα στα 2.4 και 5GHz. Μέσω αυτών των προτύπων υπάρχει η δυνατότητα υλοποίησης εσωτερικών δικτύων (LANs) χωρίς τη χρήση καλωδίων, απελευθερώνοντας έτσι την ανάπτυξη των δικτύων, καθώς δεν χρειάζεται να επενδύονται χρήματα σε εγκαταστάσεις και συντηρήσεις καλωδίων. Επιπροσθέτως, κρίνεται ιδιαίτερα βολικό και σε ανοικτούς χώρους περιορισμένης ακτίνας (μερικές δεκάδες μέτρα) καθώς και σε κτίρια όπου δεν είναι δυνατή η χρήση καλωδίου. Μερικοί από τους βασικούς στόχους του WLAN(Wi-Fi) είναι οι εξής [46]:

- Διασφάλιση συμβατότητας και συλειτουργίας των εμπλεκόμενων συσκευών
- Απαλοιφή καλωδιώσεων, πριζών κτλ.
- Ευκολότερη και ταχύτερη πρόσβασγ σε πληροφοριές οπουδήποτε, οποτεδήποτε

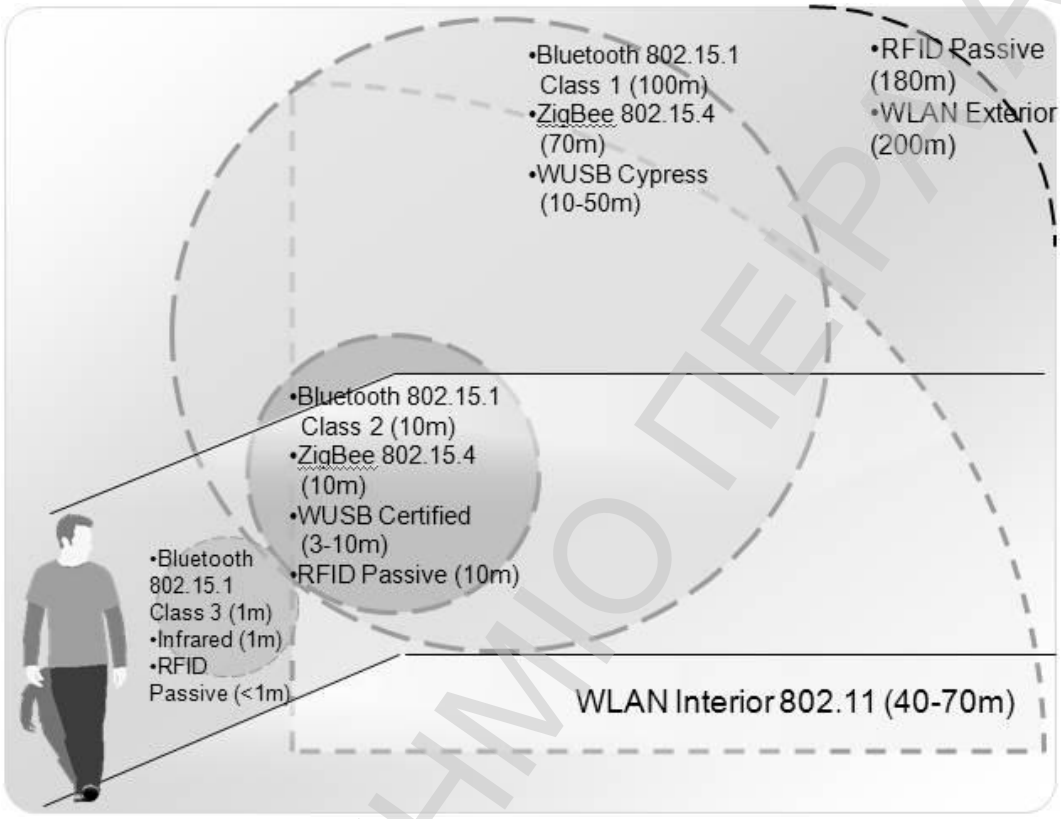
Wireless USB Certified

Το Wireless USB Certified είναι ένα ασύρματο πρωτόκολλο επικοινωνιών μικρής σχετικά εμβέλειας το οποίο έχει δημιουργηθεί από το Wireless USB Promoter Group. Ο όρος ‘Certified’ χρησιμοποιείται για το διαχωρισμό του από ομοειδή ανταγωνιστικά πρωτόκολλα όπως το Wireless USB της Cypress Semiconductor. Η προτυποποίηση των βασικών χαρακτηριστικών ολοκληρώθηκε το 2005. Ήδη χρησιμοποιείται σε μερικές από τις παρακάτω εφαρμογές [47]:

- Κονσόλες παιχνιδιού
- Περιφερειακές συσκευές Η/Υ (εκτυπωτές, σαρωτές κτλ.)
- Ψηφιακές κάμερες
- Σκληροί και flash δίσκοι
- Video-streaming (μέχρι 480Mbit/s)

Ακολουθεί σχήμα στο οποίο απεικονίζονται ενδεικτικές αποστάσεις εμβέλειας και εφαρμογές των προαναφερθέτων επικοινωνιακών προτύπων.

Σχήμα 3.1: Ασύρματα πρότυπα επικοινωνιών και εμβέλεια δράσης σε εσωτερικούς και εξωτερικούς χώρους



Τα παραπάνω πρότυπα ασύρματων επικοινωνιών αποτελούν υποψήφιες τεχνολογίες για την υλοποίηση του συστήματος υπό πραγματικές συνθήκες. Επομένως, αναλόγως την ακτίνα που θέλουμε να καλύπτει κάθε αισθητήρας αλλά και ο βασικός εξυπηρετητής, επιλέγουμε την κατάλληλη τεχνολογία ή συνδυασμό αυτών. Παράλληλα, λαμβάνεται υπόψη το κόστος υλοποίησης, οι δυνατότητες του συστήματος, οι περιορισμοί, η ασφάλεια και ότι άλλο χρήζει ιδιαίτερης προσοχής.

3.3 Προτεινόμενη αρχιτεκτονική συστήματος

Για τη λύση του προβλήματος της συγκεκριμένης μελέτης, πρέπει να ληφθούν υπόψη τα χαρακτηριστικά των συσκευών που εμπλέκονται στο σύστημα καθώς και οι διαθέσιμες μέχρι σήμερα τεχνολογίες ασύρματων και ενσύρματων επικοινωνιών. Για τη δημιουργία ενός τέτοιου συστήματος λοιπόν οι ομάδες συσκευών που χρειάζονται για τη βασική υλοποίηση προσδιορίζεται στις 4. Αυτές θα μπορούσαν να είναι:

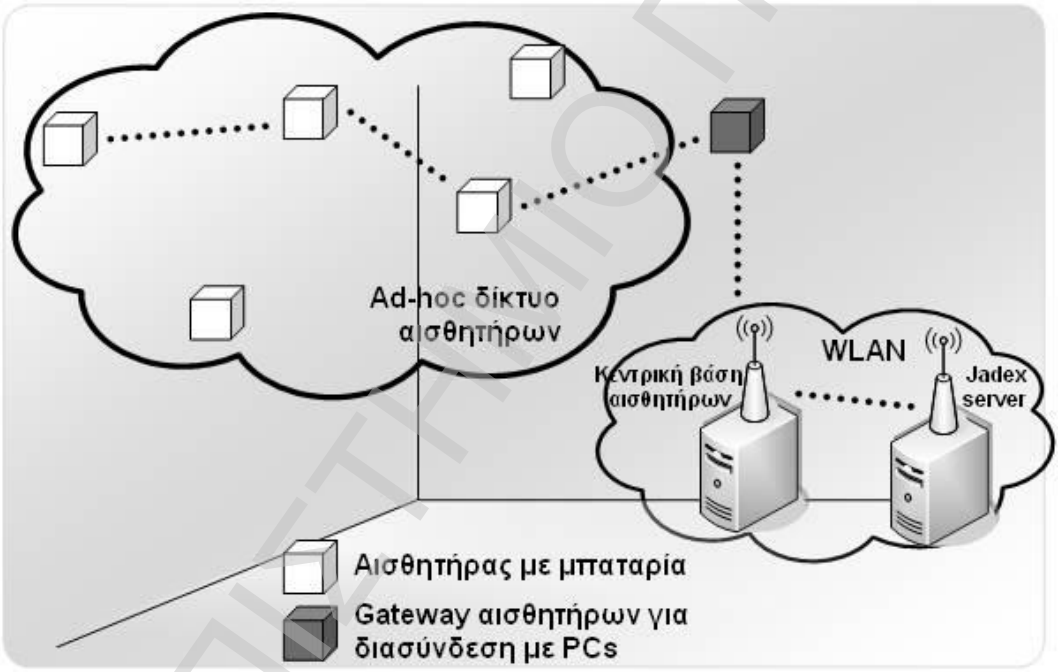
1. Κινητή συσκευή (π.χ. PDA)
2. Ad-hoc ασύρματο δίκτυο αισθητήρων με:
    a. Αισθητήρας δωματίου για ανίχνευση δεδομένων
    b. Gateway αισθητήρων
    c. Κεντρικός σταθμός βάσης αισθητήρων με δυνατότητα συλλογής, επεξεργασίας και οπτικοποίησης αριθμητικών δεδομένων
3. Βασικός εξυπηρετητής που φιλοξενεί και τη Jadex πλατφόρμα
4. Συσκευή υπηρεσίας (π.χ. τηλεόραση)

Η κινητή συσκευή χρησιμοποιείται επειδή με κάποιο τρόπο θα πρέπει ο χρήστης να γίνεται αντιληπτός μόλις εισέρχεται σε ένα χώρο όπου υπάρχει ο αισθητήρας, αποφεύγοντας προς το παρόν βιομετρικές μεθόδους. Βασική λειτουργία του αισθητήρα είναι να λαμβάνει το εκάστοτε αναγνωριστικό της συσκευής όταν αυτή εισέρχεται εντός εμβέλειας. Όταν υπάρχουν πολλοί αισθητήρες στο χώρο τότε υπάρχει η δυνατότητα προώθησης των δεδομένων (multi-hop δρομολόγηση σε ad-hoc ασύρματα δίκτυα) από αισθητήρα σε αισθητήρα μέχρι το gateway αισθητήρων [27]. Τα δεδομένα που συλλέγονται από το δίκτυο ασύρματων αισθητήρων συνήθως σώζονται σε αριθμητική μορφή σε κάποιο κεντρικό σταθμό βάσης. Μέσω ειδικών προγραμμάτων ανάλυσης μπορούμε να τα εξάγουμε και να τα οπτικοποιήσουμε. Άλλωστε υπό μελέτη βρίσκεται και η προτυποποίηση διαλειτουργικών διεπαφών από το Open

Geospatial Consortium (OGC) οι οποίες θα επιτρέπουν τον απευθείας και σε πραγματικό χρόνο έλεγχο των συλλεχθέντων δεδομένων μέσω του διαδικτύου.

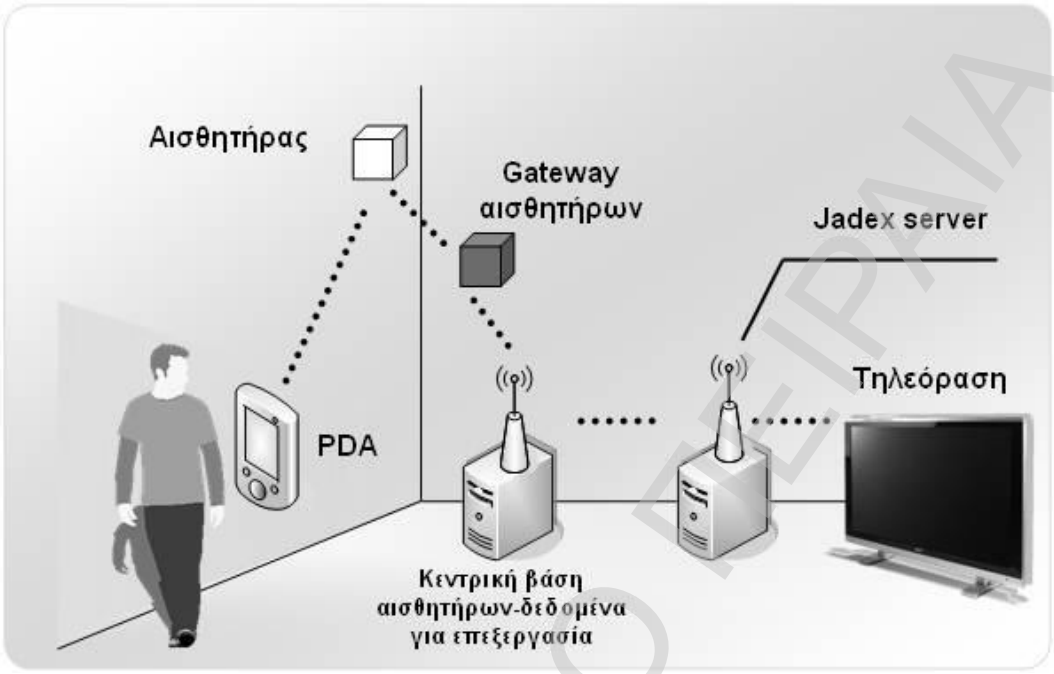
Το gateway των αισθητήρων αναλαμβάνει τη διεπαφή του δικτύου αισθητήρων με άλλα δίκτυα και στέλνει τα δεδομένα του PDA και του αισθητήρα στην κεντρική βάση διαχείρισης των αισθητήρων η οποία κεντρική βάση έχει τη δυνατότητα επεξεργασίας των στοιχείων. Αμέσως μετά ενημερώνει τον βασικό εξυπηρετητή ο οποίος θα αναλάβει την αυθεντικοποίηση του χρήστη [27].

Σχήμα 3.2: Ενδεικτικό δίκτυο αισθητήρων



Εν συνεχεία, ο βασικός εξυπηρετητής πρωτίστως φιλοξενεί την Jadex πλατφόρμα και βρίσκεται μονίμως σε κατάσταση αναμονής νέων εναυσμάτων από τους αισθητήρες ώστε να εκτελεστούν οι εκάστοτε υπηρεσίες (π.χ. αναπαραγωγή ταινίας). Τέλος η συσκευή υπηρεσίας αναλαμβάνει την εκτέλεση της υπηρεσίας (π.χ. αν πρόκειται για ταινία, τότε η συσκευή υπηρεσίας θα είναι η τηλεόραση). Το σχήμα που ακολουθεί απεικονίζει τη βασική αυτή αρχιτεκτονική.

Σχήμα 3.3: Προτεινόμενη αρχιτεκτονική συστήματος



3.4 Προτεινόμενοι τρόποι επικοινωνίας

Στην αγορά κυκλοφορούν αρκετά μοντέλα αισθητήρων τα οποία χρησιμοποιούν διάφορα επικοινωνιακά πρότυπα για τη μετάδοση δεδομένων. Αρκετά διαδεδομένα είναι οι αισθητήρες με Bluetooth 2.0 και αισθητήρες με 802.15.4/ZigBee [27]. Μάλιστα τα εν λόγω μοντέλα μπορούν να ενσωματώνουν μνήμη από μερικά kilobytes μέχρι μερικά megabytes. Ενδεικτικά αναφέρονται μοντέλα όπως SunSpot (ανάπτυξη από Sun Microsystems), BTnode (ανάπτυξη από ETH Zurich) [8] κ.ά. Επιπροσθέτως, ένα ενδεικτικό gateway αισθητήρων για την διεπαφή με την κεντρική βάση μπορεί να ενσωματώνει την οικογένεια προτύπων 802.11 ή WirelessUSB ή απλό Ethernet και από αυτό εξαρτάται και ο τρόπος με τον οποίο θα επιτυγχάνεται η σύνδεση με τον Jadex server. Ενδεικτικό μοντέλο gateway αισθητήρων είναι Stargate με επεξεργαστή Intel PXA255 [Intel Corp.].

3.5 Σενάριο υλοποίησης

Αρχικά εισερχόμαστε στο χώρο με το PDA το οποίο ενσωματώνει τεχνολογίες όπως Bluetooth ή ZigBee και WLAN. Είναι ενεργοποιημένο και ρυθμισμένο με συγκεκριμένο μοναδικό αναγνωριστικό για τον συγκεκριμένο χώρο ελέγχου. Επίσης έχει προεγκατεστημένο έναν Jadex πράκτορα λογισμικού για την επικοινωνία με τον Jadex Server. Στο δωμάτιο υπάρχει ένας αισθητήρας με εμβέλεια π.χ. 5-10m αναλόγως τις διαστάσεις του δωματίου. Ο αισθητήρας ενσωματώνει την τεχνολογία Bluetooth ή ZigBee και υπάρχει δυνατότητα προγραμματισμού αυτών με τις γλώσσες C ή Java (Squawk Java Virtual Machine). Λαμβάνει το ερέθισμα από το PDA μέσω ασύρματης ζεύξης μόλις το PDA βρεθεί στην περιοχή ελέγχου του συγκεκριμένου αισθητήρα.

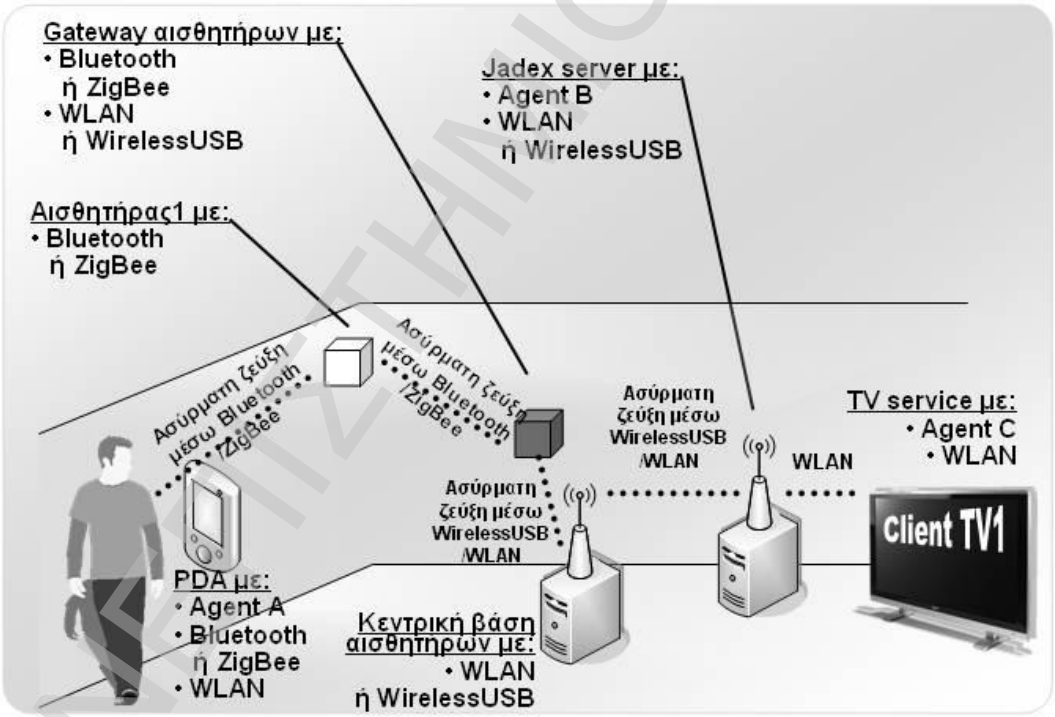
Για την επικοινωνία του αισθητήρα (ή δικτύου αισθητήρων) με την κεντρική βάση ελέγχου των αισθητήρων χρειάζεται κάποιο gateway αισθητήρων το οποίο χρησιμοποιεί τις τεχνολογίες Bluetooth ή ZigBee για τη διάδραση με τους αισθητήρες καθώς και WLAN ή WirelessUSB ή απλή Ethernet διεπαφή για τη ζεύξη του gateway με τη βάση. Στη βάση των αισθητήρων υπάρχει η δυνατότητα συλλογής, επεξεργασίας και οπτικοποίησης αριθμητικών δεδομένων. Μετά από επεξεργασία αποστέλλεται προς τον Jadex server μέσω WirelessUSB ή WLAN ένα ζεύγος τιμών {ID χρήστη- ID αισθητήρα}.

Στην περίπτωση που δεν είναι δυνατή η αποστολή ζεύγους τιμών λόγω προγραμματιστικών περιορισμών των αισθητήρων θα μπορούσαμε να ρυθμίσουμε κάθε αισθητήρα να κάνει ‘append’ στο τέλος του ID χρήστη που έχει λάβει από το PDA, ένα δικό του νούμερο. Ο server θα παραλάβει ένα μήνυμα της μορφής π.χ. ‘100555’ , όπου το 100 είναι το ID χρήστη και το 555 είναι το νούμερο που έκανε ‘append’ ο συγκεκριμένος αισθητήρας. Ο Jadex server θα διαβάσει τα τρία πρώτα νούμερα (100) και θα τα αντιστοιχήσει στο χρήστη, ενώ τα τρία τελευταία θα τα αντιστοιχήσει στον αισθητήρα.

Με την επίτευξης της αποστολής του ζεύγους τιμών προς τον Jadex server, ο Jadex server αναγνωρίζει τον χρήστη μέσω του πίνακα ελέγχου των ID χρηστών και την τοποθεσία μέσω του πίνακα ελέγχου των ID αισθητήρων. Επιπροσθέτως, ο Jadex server ενσωματώνει την πλατφόρμα Jadex και έχει προεγκατεστημένο έναν άλλο πράκτορα λογισμικού Jadex. Γι' αυτό το λόγο πρέπει να βρίσκεται μόνιμα σε κατάσταση αναμονής.

Μόλις ολοκληρωθεί και η διαδικασία αναγνώρισης του χρήστη-τοποθεσίας, ο πράκτορας που ενσωματώνει ο server αναλαμβάνει μέσω WLAN να στείλει μήνυμα προς τον πράκτορα της τηλεόρασης (ή οποιασδήποτε άλλης υπηρεσίας), προς εκτέλεση των εκάστοτε υπηρεσιών (π.χ. αναπαραγωγή ταινίας).

Σχήμα 3.4: Προτεινόμενη υλοποίηση συστήματος –επίπεδο υποδομής



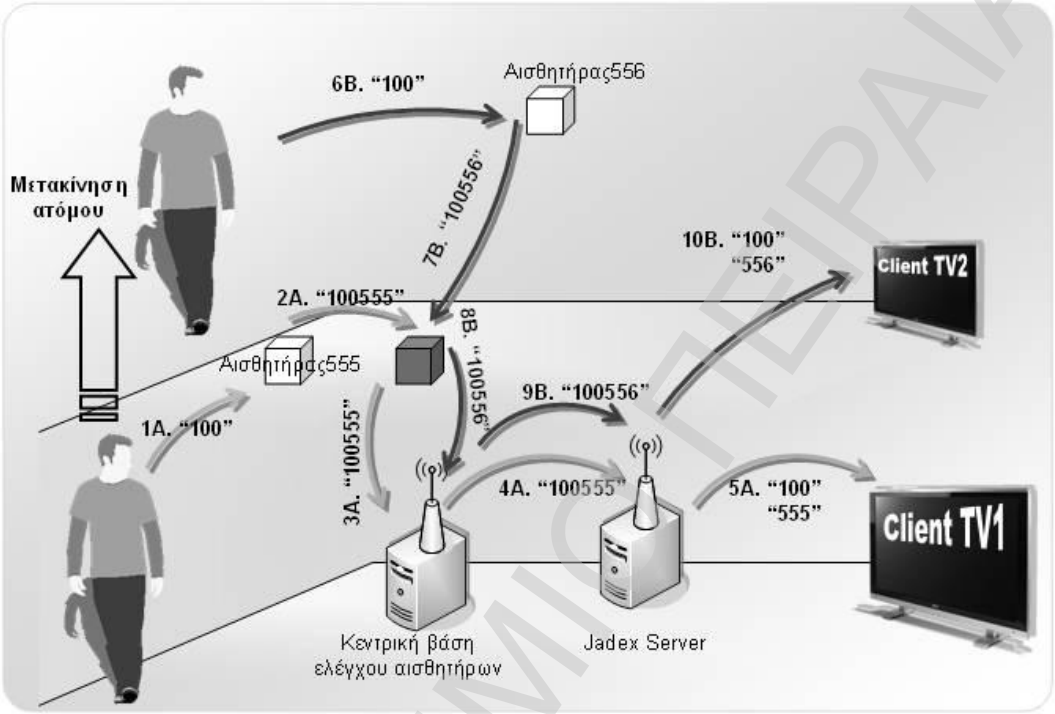
Το παραπάνω σχήμα απεικονίζει την προτεινόμενη υλοποίηση του συστήματος λαμβάνοντας υπόψη την αρχιτεκτονική, τους 'παίκτες' του συστήματος, τα διαθέσιμα interfaces των συσκευών, τις ζεύξεις μεταξύ των

συσκευών καθώς και τους πράκτορες λογισμικού (agents) που ενσωματώνουν οι συσκευές.

Όσον αφορά τους πράκτορες λογισμικού ο πράκτορας Α που ενσωματώνεται στο PDA, αποτελεί ουσιαστικά και το έναυσμα για την εκτέλεση οποιασδήποτε υπηρεσίας, αφού εάν ο χρήστης δεν εισέλθει σε κάποια περιοχή ελέγχου δεν θα ξεκινήσει η ανταλλαγή μηνυμάτων για εκτέλεση υπηρεσιών. Μέσω λοιπόν των αισθητήρων λαμβάνεται το ID που εκπέμπει το PDA του χρήστη, η κεντρική βάση ελέγχου των αισθητήρων επεξεργάζεται τα στοιχεία και στέλνει μέσω WLAN/WirelessUSB το ζεύγος τιμών {ID χρήστη- ID αισθητήρα} προς τον πράκτορα Β του Jadex server. Ο πράκτορας του server ελέγχει τους πίνακες χρηστών και αισθητήρων και αν τα δεδομένα είναι όπως πρέπει, αποστέλλει μέσω WLAN/WirelessUSB μήνυμα εκκίνησης προς τον πράκτορα C (π.χ. της τηλεόρασης), ο οποίος και αυτός βρίσκεται σε κατάσταση αναμονής. Ο πράκτορας C αναλαμβάνει την εκτέλεση της υπηρεσίας (π.χ. αναπαραγωγή ταινίας).

Στην περίπτωση που ο χρήστης αποφασίσει σε άλλο δωμάτιο με άλλη συσκευή τηλεόρασης, το PDA εκπέμπει πάλι το ίδιο ID το οποίο όμως λαμβάνεται αυτή τη φορά από κάποιον άλλο αισθητήρα. Η πληροφορία αυτή μεταφέρεται μέσω του ad-hoc ασύρματου δικτύου αισθητήρων προς την κεντρική βάση αισθητήρων όπου και επεξεργάζεται. Αυτή τη φορά στον πράκτορα Β του server στέλνεται ένα νέο ζεύγος τιμών {ID χρήστη- ID αισθητήρα} με αποτέλεσμα να αναγνωρίζει την αλλαγή τοποθεσίας και να δίνει εντολή στο πράκτορα της πρώτης τηλεόρασης να σταματήσει την αναπαραγωγή, λαμβάνει το χρόνο παύσης σε μια μεταβλητή, στέλνει μήνυμα εκκίνησης (μέσω WirelessUSB/WLAN) στον πράκτορα της δεύτερης τηλεόρασης περνώντας παράλληλα και την τιμή της μεταβλητής του χρόνου παύσης από την πρώτη τηλεόραση, ώστε η εκκίνηση να πραγματοποιηθεί από το σημείο παύσης και όχι από την αρχή.

Σχήμα 3.5: Προτεινόμενη υλοποίηση συστήματος –επίπεδο μηνυμάτων



Το παραπάνω σχήμα εξηγεί τα ενδεικτικά μηνύματα που μπορούν να ανταλλάσσονται μεταξύ των διαφόρων συσκευών του συστήματος. Ο χρήστης εισέρχεται στο πρώτο δωμάτιο και ο αισθητήρας555 λαμβάνει το μήνυμα ‘100’. Μετά προγραμματίζεται έτσι ώστε να προσθέτει στο τέλος το κωδικό του και δημιουργεί ένα μήνυμα της μορφής ‘100555’ το οποίο αποστέλλει στο gateway αισθητήρων. Το gateway επικοινωνεί με τη βάση ελέγχου αισθητήρων και η βάση στέλνει στον Jadex server το μήνυμα ‘100555’. Ο server χωρίζει το πρώτο μέρος του μηνύματος από το δεύτερο και αναγνωρίζει τον χρήστη και τον αισθητήρα αναγνώρισης. Ο πράκτορας του server στέλνει μήνυμα έναρξης στον πράκτορα της τηλεόρασης και ξεκινάει η αναπαραγωγή ταινίας.

Έστω ότι ο χρήστης μετακινείται στο δεύτερο δωμάτιο. Ο αισθητήρας556 εντάσσει στην περιοχή εμβέλειάς του το χρήστη ‘100’, προσθέτει τον κωδικό του ‘100556’ και στέλνει το μήνυμα στο gateway. Γίνεται καταγραφή στη βάση ελέγχου και ο server λαμβάνει το μήνυμα ‘100556’. Ο server χωρίζει πάλι το πρώτο από το δεύτερο μέρος και μέσω του πράκτορα στέλνει μήνυμα προς τη δεύτερη τηλεόραση που ανήκει στην περιοχή ευθύνης του αισθητήρα 556 ώστε να συνεχιστεί η αναπαραγωγή της ταινίας στη δεύτερη τηλεόραση.

4. Ανάπτυξη ενδεικτικού επικοινωνιακού μοντέλου στο JADEX

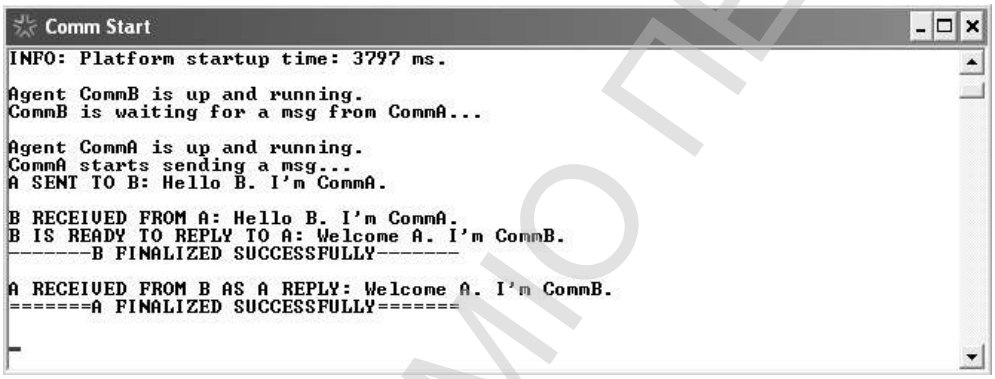
4.1 Βασικό επικοινωνιακό μοντέλο

Στο Jadex, οι πράκτορες συντάσσονται σε XML και εκτελούνται με τη χρήση μιας κλάσης Java (η οποία στο παράδειγμά μας κληρονομεί ιδιότητες από την γονική κλάση Plan). Στο σχήμα 4.1, αρχικά ξεκινάει ο πράκτορας B ο οποίος μας πληροφορεί για την έναρξη με ένα καλωσορίσματος όπως βλέπουμε στο μικρό εικονίδιο 'Welcome'.

Έπειτα, αναμένει για κάποιο μήνυμα από τον A. Τότε, ο πράκτορας A ξεκινάει, αναγνωρίζει την κατάσταση αναμονής του πράκτορα B, και στέλνει ένα μήνυμα στο B. Ο B λαμβάνει το μήνυμα και απαντάει πίσω στον A ότι το έλαβε.. Τέλος, ο πράκτορας A λαμβάνει την απάντηση του B και η διαδικασία τερματίζεται. Τα αποτελέσματα αυτής της απλοποιημένης αλλά βασικής επικοινωνιακής διαδικασίας μεταξύ δύο πρακτόρων λογισμικού του Jadex εμφανίζονται στο σχήμα 4.1 το οποίο ακολουθεί.

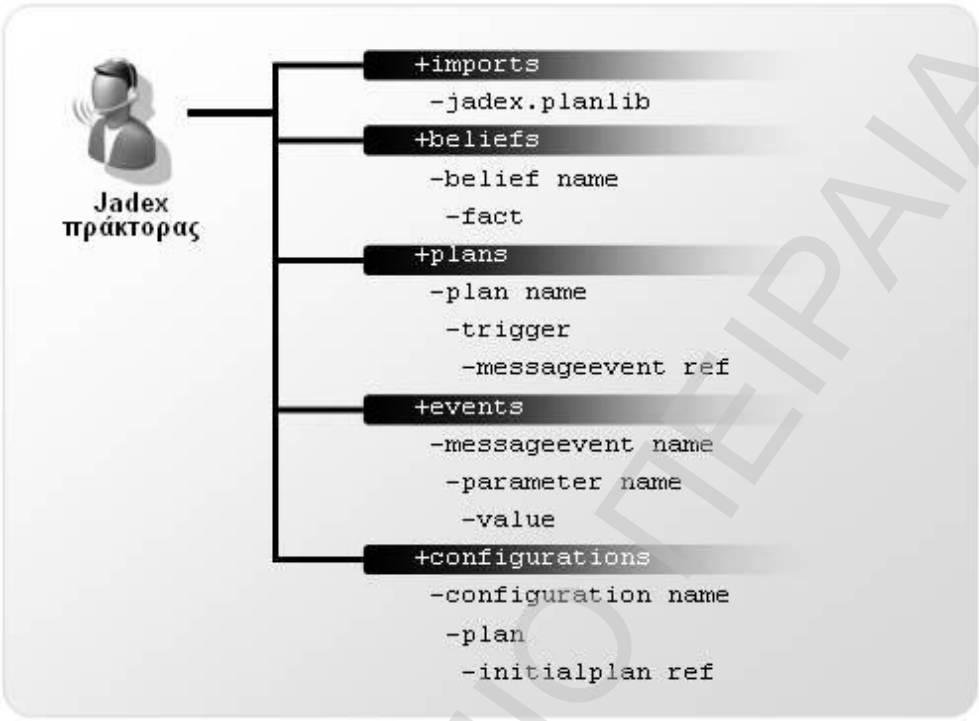
Σχήμα 4.1: Πράκτορες βασικού επικοινωνιακού μοντέλου Jadex

Αποτελέσματα επικοινωνιακής διαδικασίας:



Οι πράκτορες Α και Β, χρησιμοποιούν beliefs ώστε να παρουσιάσουν τα αρχικά Welcome μηνύματα στην κονσόλα χρησιμοποιώντας ένα JOptionPane. Στο Jadex τα beliefs μπορούν να είναι οποιοδήποτε αντικείμενο της Java. Τα plans από την άλλη χρησιμοποιούνται για να ξεκινήσουν την κλάση Plan η οποία υλοποιείται σε Java. Επιπροσθέτως, στο παρόν παράδειγμα ορίζονται και κάποια messageevents ώστε να αποκτήσουν οι πράκτορες πρόσβαση στα request_msg και request_msg2 αιτήματα. Τέλος, στο configuration tag του xml αρχείου των πρακτόρων, καθορίζεται το initialplan ώστε να ξεκινήσει η διαδικασία εκτέλεσης σύμφωνα με τη ροή που προαναφέρθηκε. Στο σχήμα που ακολουθεί απεικονίζεται το XML schema του δικού μας πράκτορα λογισμικού (Comm agent).

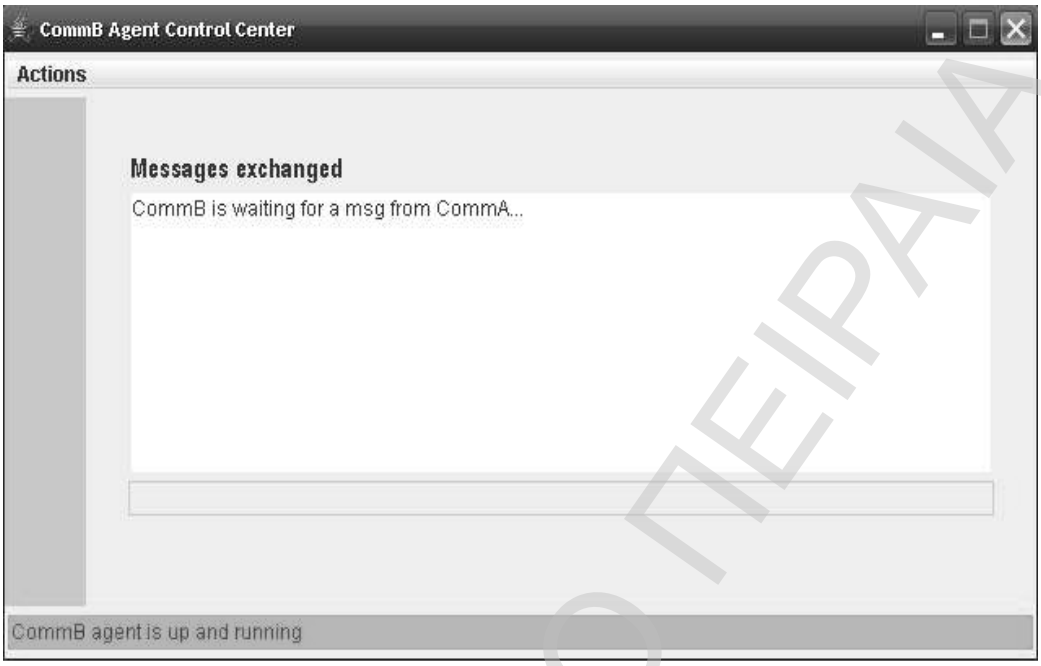
Σχήμα 4.2: XML schema του πράκτορα Comm



4.2 Πρωταρχικό γραφικό περιβάλλον επικοινωνιακού μοντέλου

Μετά το βασικό επικοινωνιακό μοντέλο και την δημιουργία της βάσης των πρακτόρων λογισμικού, γίνεται προσπάθεια να γίνει η διαδικασία λίγο πιο φιλική προς το χρήστη μέσω της ανάπτυξης ενός γραφικού περιβάλλοντος στο οποίο θα εμφανίζονται οι ενέργειες των πρακτόρων. Επομένως, τη στιγμή που ξεκινάει ο πράκτορας Β εμφανίζεται ένα κέντρο ελέγχου όπως το παρακάτω:

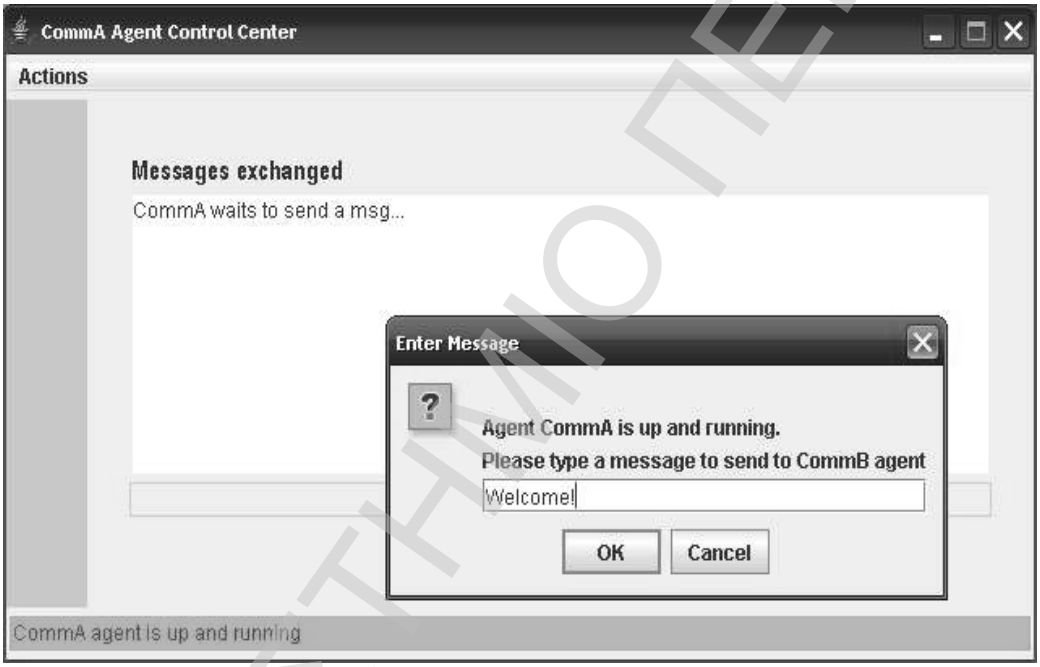
Σχήμα 4.3: Βασικό κέντρο ελέγχου του πράκτορα CommB



Σε αυτό το παράθυρο μπορούμε να δούμε τα μηνύματα που θα ανταλλάξουν οι πράκτορες CommB & CommA, στο textarea υπό του τίτλου ‘Messages Exchanged’. Επίσης μέσω του μενού ‘Actions’ μπορούν να ελεγχθούν διάφορες επιπρόσθετες λειτουργίες όπως π.χ. Κλείσιμο παραθύρου κτλ. Για την ώρα όμως εμφανίζεται ένα μήνυμα αναμονής. Το επόμενο βήμα είναι να ξεκινήσει ο πράκτορας CommA.

Με την εκκίνηση του πράκτορα CommA εμφανίζεται το παράθυρο του σχήματος που ακολουθεί. Ομοιάζει με το προηγούμενο στην εμφάνιση, μόνο που από εδώ υπάρχει και η δυνατότητα πληκτρολόγησης του δικού μας μηνύματος το οποίο θα αποσταλεί στο πράκτορα CommB, ο οποίος ήδη περιμένει για κάποιο μήνυμα.

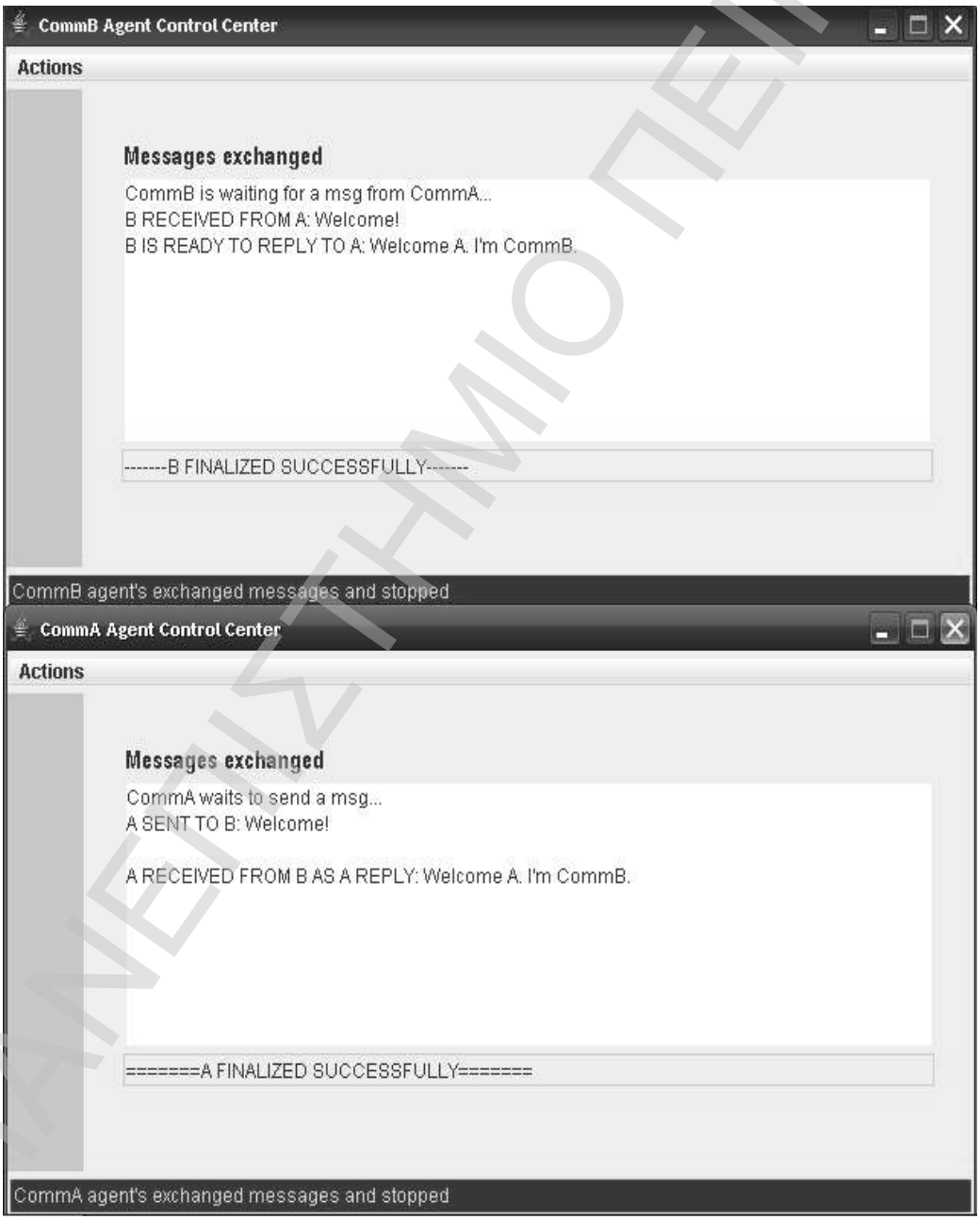
Σχήμα 4.4: Βασικό κέντρο ελέγχου του πράκτορα CommA



Από αυτό το παράθυρο μπορούμε να δούμε τα μηνύματα που θα ανταλλάξουμε με τον πράκτορα CommB. Επίσης στο μενού 'Actions' περιλαμβάνονται λειτουργίες όπως 'Κλείσιμο παραθύρου' κτλ. Όταν πληκτρολογούμε το μήνυμα μας στο ειδικό πεδίο εισαγωγής κειμένου και επιλέγουμε 'OK', τότε το μήνυμα μορφοποιείται σε ένα φάκελο σύμφωνα με τα προαναφερθέντα πρότυπα της FIPA και όλο αυτό αποστέλλεται στον CommB. Ο CommB λαμβάνει το μήνυμα και απαντάει στον A ώστε να έχουμε επιτυχή ολοκλήρωση της διαδικασίας.

Τέλος, η διαδικασία τερματίζεται όταν όλοι οι πράκτορες έχουν ανταλλάξει τα απαραίτητα μηνύματα σύμφωνα με το πλάνο εκτέλεσης. Και τα δύο κέντρα ελέγχου τυπώνουν τα αποτελέσματα όπως φαίνεται και στο σχήμα που ακολουθεί. Ας θυμηθούμε ότι το αρχικό που έπρεπε να στείλει ο CommA προς τον CommB ήταν 'Welcome!'.

Σχήμα 4.5: Αποτελέσματα βασικού επικοινωνιακού μοντέλου



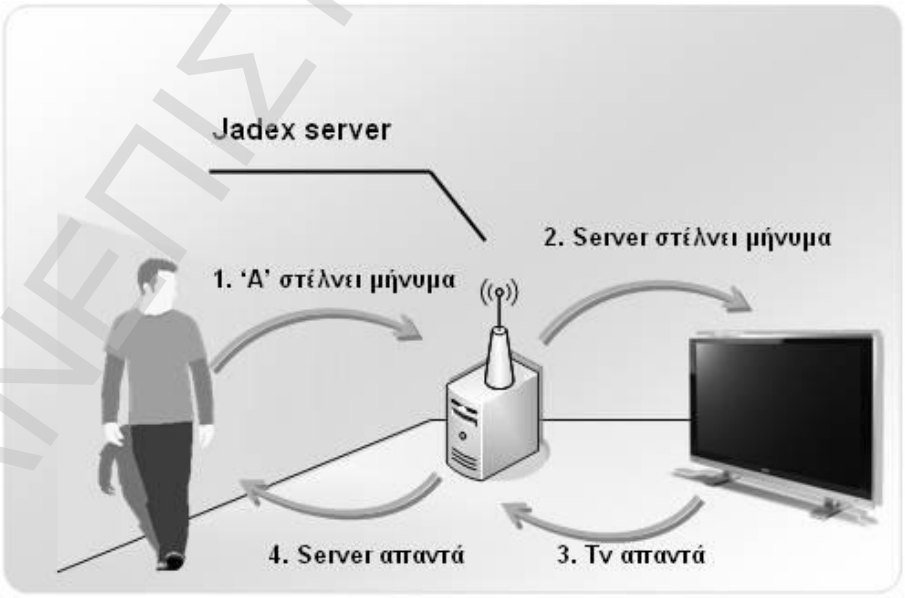
4.3 Επέκταση επικοινωνιακού μοντέλου

Σε ένα πολυπρακτορικό περιβάλλον συνήθως περισσότεροι από δύο πράκτορες αλληλεπιδρούν προς την εκτέλεση κάποιας υπηρεσίας. Με αυτό το σενάριο λοιπόν ασχολείται και η μελέτη.

Σενάριο "Α"

Υποθέτουμε ότι εισερχόμαστε στην οικία μας όπου υπάρχει ένας πράκτορας λογισμικού ο οποίος ενεργεί ως communication server και αναμένει για μηνύματα. Επίσης, η συσκευή της τηλεόρασης φιλοξενεί κάποιον άλλο πράκτορα ο οποίος αναμένει για μηνύματα από τον βασικό εξυπηρετητή (communication server) της οικίας. Τη στιγμή που εισερχόμαστε στο χώρο εμβέλειας, η κινητή μας συσκευή (π.χ. PDA) στέλνει μήνυμα στο server ο οποίος με τη σειρά του ενημερώνει τον πράκτορα της τηλεόρασης ότι εισήλθαμε στο χώρο. Τότε η τηλεόραση απαντά πίσω στον server και αυτός με τη σειρά του ενημερώνει την κινητή μας συσκευή ότι η τηλεόραση έλαβε την εντολή και είναι έτοιμη προς χρήση. Η γραφική αναπαράσταση της ανωτέρω διαδικασίας φαίνεται και στο επόμενο σχήμα.

Σχήμα 4.6: Σενάριο "Α" –Γραφική αναπαράσταση



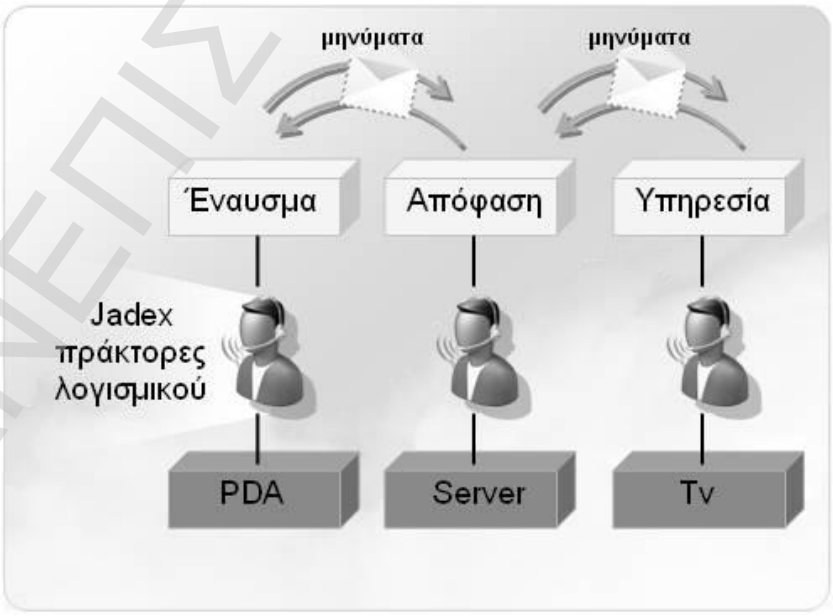
Αρχιτεκτονική

Η αρχιτεκτονική του εν λόγω σεναρίου βασίζεται στην ιδέα χρήσης τριών βασικών ομάδων οντοτήτων. Σύμφωνα με αυτή την παραδοχή λοιπόν οι ομάδες αυτές είναι:

- Ομάδα 1 –Κινητή συσκευή (π.χ. PDA, κινητό τηλέφωνο κτλ.)
- Ομάδα 2 –Βασικός εξυπηρετητής
- Ομάδα 3 –Οικιακή συσκευή/υπηρεσία (π.χ. τηλεόραση, ψυγείο, μουσική κτλ.)

Η ομάδα οντοτήτων 1 στέλνει και λαμβάνει μηνύματα μόνο από την ομάδα οντοτήτων 2 και η ομάδα οντοτήτων 2 αναλαμβάνει την αποστολή και λήψη μηνυμάτων στην ομάδα οντοτήτων 3. Επιπροσθέτως, η ομάδα οντοτήτων 1 ευθύνεται για τα εναύσματα τα οποία ενεργοποιούν την ομάδα οντοτήτων 2 η οποία με τη σειρά της αποφασίζει ποια υπηρεσία πρέπει να ξεκινήσει από την ομάδα 3. Όλα αυτά τα μηνύματα ανταλλάσσονται μέσω του middleware (π.χ. πλατφόρμα Jadex), οπότε οι ομάδες οντοτήτων πρέπει να ενσωματώνουν Jadex πράκτορες λογισμικού ώστε να είναι ικανές να επικοινωνούν μεταξύ τους.

Σχήμα 4.7: Σενάριο "Α" –Αρχιτεκτονική



Σενάριο "Α" –Βασική αποστολή

Η βασική αποστολή του εν λόγω σεναρίου είναι η παροχή πολυπρακτορικού πλαισίου επικοινωνίας προς την επίτευξη πρόσβασης σε συγκεκριμένες υπηρεσίες. Κατ' επέκταση, διάφορες κινητές συσκευές (ομάδα οντοτήτων 1) μπορούν να αποκτήσουν σε διάφορες υπηρεσίες (ομάδα οντοτήτων 3) μέσω ενός εξυπηρετητή (ομάδα οντοτήτων 2), όπως φαίνεται και στη γραφική αναπαράσταση του παρακάτω σχήματος.

Σχήμα 4.8: Βασική αποστολή του Σεναρίου "Α"

Σενάριο "Α" –Εκτέλεση

Σύμφωνα με το σενάριο χρησιμοποιούνται τρεις πράκτορες λογισμικού. Αυτοί είναι (κατά σειρά εμφάνισης –εκτέλεσης):

I/ Από την ομάδα οντοτήτων 2 > CommServer πράκτορας

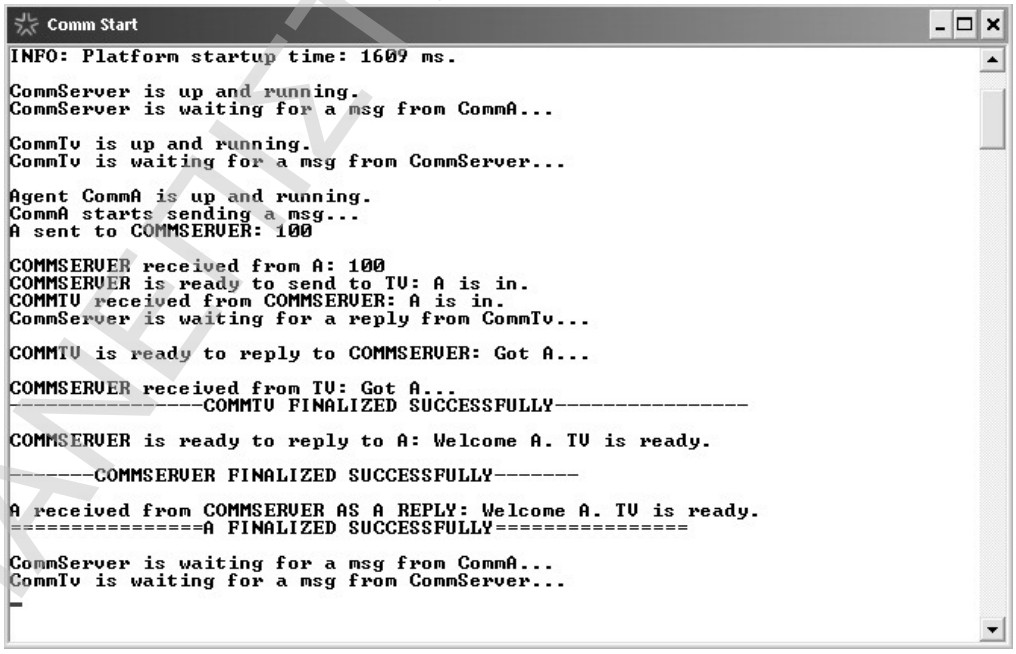
II/ Από την ομάδα οντοτήτων 3 > CommTv πράκτορας (υπηρεσία)

III/ Από την ομάδα οντοτήτων 1 > CommA (έναυσμα)

Οι CommServer και CommTv λειτουργούν συνεχώς ώστε να μπορούν να λαμβάνουν και να στέλνουν μηνύματα στον CommA.

1. Επομένως, ο CommA στέλνει μήνυμα προς CommServer (πχ.100)
2. Ο CommServer λαμβάνει το μήνυμα και ενημερώνει τον CommTv (πχ. A is in)
3. Ο CommTv απαντά στον CommServer (πχ.Got A) και αναμένει για νέο μήνυμα.
4. Ο CommServer απαντά στον CommA (πχ. Welcome A. TV is ready) και αναμένει για νέο μήνυμα.
5. Ο CommA λαμβάνει την επιβεβαίωση από τον CommServer και τερματίζει.

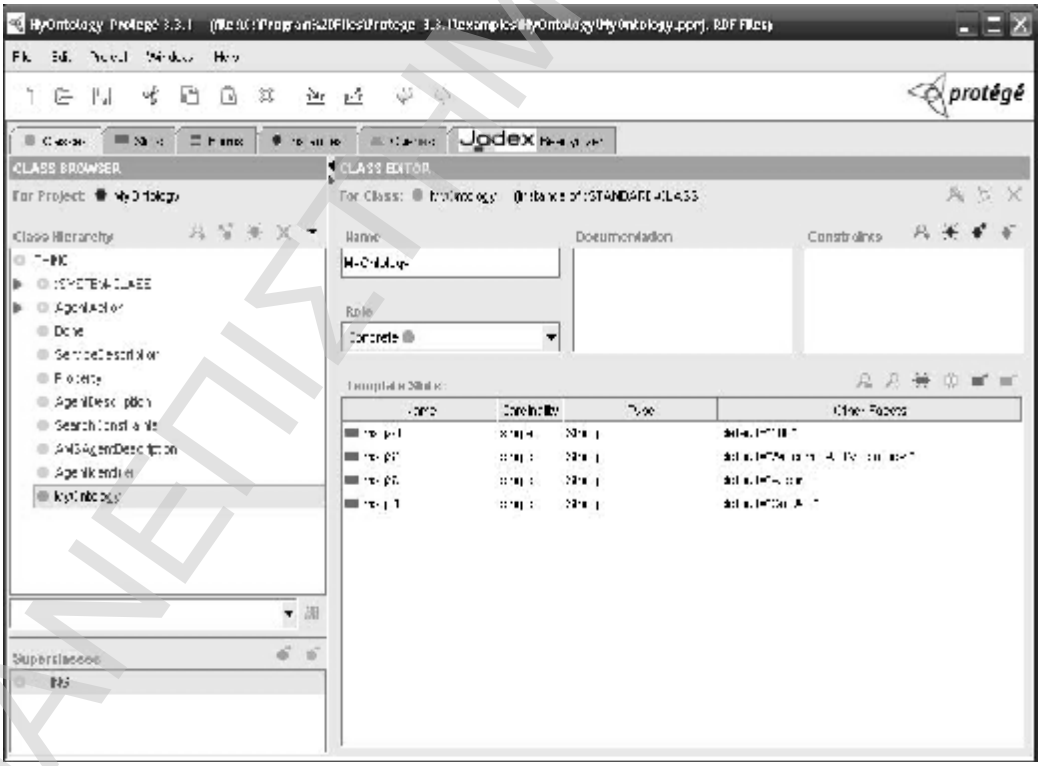
Σχήμα 4.9: Σενάριο "Α" –Αποτελέσματα εκτέλεσης



Σενάριο “Α” –Οντολογία

Οι Jadex πράκτορες που αναπτύχθηκαν για τις ανάγκες του σεναρίου κάνουν χρήση της οντολογίας MyOntology η οποία σχεδιάστηκε στο εργαλείο σχεδιασμού οντολογιών Protege και εξάχθηκε με τη βοήθεια του Jadex Beanynizer plugin. Αυτή η οντολογία έχει για μεταβλητές (slots) τέσσερα μηνύματα τύπου String. Αυτά είναι τα msgA1, msgS1, msgS2 και msgT1. Ως αποτέλεσμα, οι πράκτορες CommA, CommServer και CommTv προσπελαύνουν την οντολογία ώστε να κάνουν λήψη των δεδομένων που έχουν αποθηκευτεί στις προαναφερθείσες μεταβλητές. Το σχήμα που ακολουθεί απεικονίζει τη δημιουργία του MyOntology όπως φαίνεται στο εργαλείο σχεδιασμού οντολογιών Protege.

Σχήμα 4.10: Οντολογία σεναρίου

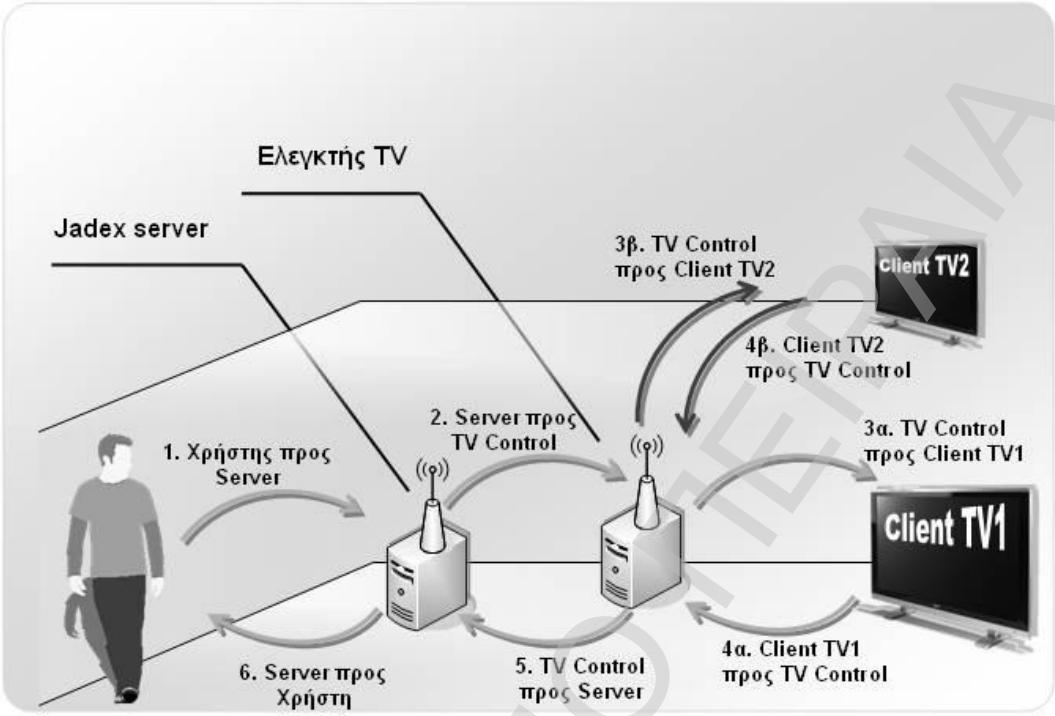


Σενάριο “Β”

Το σενάριο “Β” λειτουργεί ως επέκταση του προαναφερθέντος σεναρίου “Α”. Η επέκταση λαμβάνει υπόψη το γεγονός ότι στην οικία μας υπάρχει ένας εγκατεστημένος βασικός εξυπηρετητής ο οποίος αναμένει για μηνύματα. Επιπροσθέτως, υπάρχει ένας δευτερεύον εξυπηρετητής ο οποίος ενεργεί ως ελεγκτής των τηλεοράσεων (TV Controller). Πλέον, υπάρχουν δύο συσκευές τηλεοράσεων σε δύο διαφορετικούς χώρους της οικίας οι οποίες αναμένουν για μηνύματα από τον ελεγκτή τηλεοράσεων. Επομένως, τη στιγμή που το σύστημα αναγνωρίσει ότι εισήλθαμε στην οικία στέλνει μήνυμα στον βασικό εξυπηρετητή και αμέσως με τη σειρά του αυτός ενημερώνει τον ελεγκτή τηλεοράσεων. Ο ελεγκτής στέλνει μήνυμα στο Client TV1 και η πρώτη τηλεόραση (π.χ. στο σαλόνι) ξεκινάει την αναπαραγωγή κάποιας ταινίας. Η τηλεόραση επιβεβαιώνει ότι ξεκίνησε στον ελεγκτή και τέλος ο βασικός πλέον εξυπηρετητής μας ενημερώνει ότι η τηλεόραση έλαβε το μήνυμα και ξεκίνησε την αναπαραγωγή.

Μετά από λίγο, υποθέτουμε ότι μεταφερόμαστε στον δεύτερο όροφο της οικίας μας, ή σε άλλο δωμάτιο όπου υπάρχει η Client TV2. Ο βασικός εξυπηρετητής ενημερώνει τον ελεγκτή τηλεοράσεων και τότε ο ελεγκτής δίνει εντολή τερματισμού αναπαραγωγής από την πρώτη τηλεόραση και μεταφορά της εικόνας στη δεύτερη τηλεόραση από το σημείο που την αφήσαμε (όχι από την αρχή πάλι). Η γραφική αναπαράσταση της ανωτέρω διαδικασίας εκτέλεσης του σεναρίου ‘Β’ ακολουθεί.

Σχήμα 4.11: Σενάριο "Β" –Γραφική αναπαράσταση



Αρχιτεκτονική

Η αρχιτεκτονική του σεναρίου 'Β' μοιάζει με αυτή του σεναρίου 'Α' στην ιδέα ότι υπάρχουν πάλι τρεις βασικές ομάδες οντοτήτων, μόνο που εδώ η δεύτερη ομάδα χωρίζεται σε δύο υποομάδες ως ακολούθως:

- Ομάδα 1 –Κινητή συσκευή (π.χ. PDA, κινητό τηλέφωνο κτλ.)
- Ομάδα 2 –Εξυπηρετητές
  - Ομάδα 2α –Βασικός εξυπηρετητής
  - Ομάδα 2β –Δευτερεύον εξυπηρετητής (ελεγκτής τηλεοράσεων)
- Ομάδα 3 –Οικιακή συσκευή/υπηρεσία (π.χ. τηλεόραση, ψυγείο, μουσική κτλ.)

Η μικρή διαφορά λοιπόν της ομάδας οντοτήτων 2 δείχνει ότι υπάρχει ο βασικός Jadex εξυπηρετητής και ο ελεγκτής συσκευών τηλεόρασης (προς καλύτερη οργάνωση των εντολών). Επιπροσθέτως, λαμβάνεται υπόψη το γεγονός ότι η ομάδα οντοτήτων 1 λαμβάνει και στέλνει μηνύματα στην ομάδα οντοτήτων 2α η οποία με τη σειρά της επικοινωνεί με την ομάδα 2β. Η ομάδα 2β αναλαμβάνει τη λήψη και αποστολή μηνυμάτων από την ομάδα 3.

Τέλος, η ομάδα 1 ευθύνεται για τα εναύσματα τα οποία ενεργοποιούν την ομάδα 2α η οποία με τη σειρά της λαμβάνει τις αποφάσεις για το ποια υπηρεσία της ομάδας 3 θα εκτελεστεί και πράττει αναλόγως. Όλα τα μηνύματα ανταλλάσσονται μέσω της Jadex πλατφόρμας, επομένως όλες οι ομάδες οντοτήτων πρέπει να ενσωματώνουν Jadex πράκτορες λογισμικού για να είναι σε θέση να επικοινωνούν μεταξύ τους.

Σενάριο "Β" –Εκτέλεση

Χρησιμοποιούνται τέσσερεις πράκτορες λογισμικού. Αυτοί είναι οι ακόλουθοι:

I/ Από ομάδα οντοτήτων 2α > CommServer πράκτορας

II/ Από ομάδα οντοτήτων 2β > CommTV πράκτορας (ελεγκτής TV)

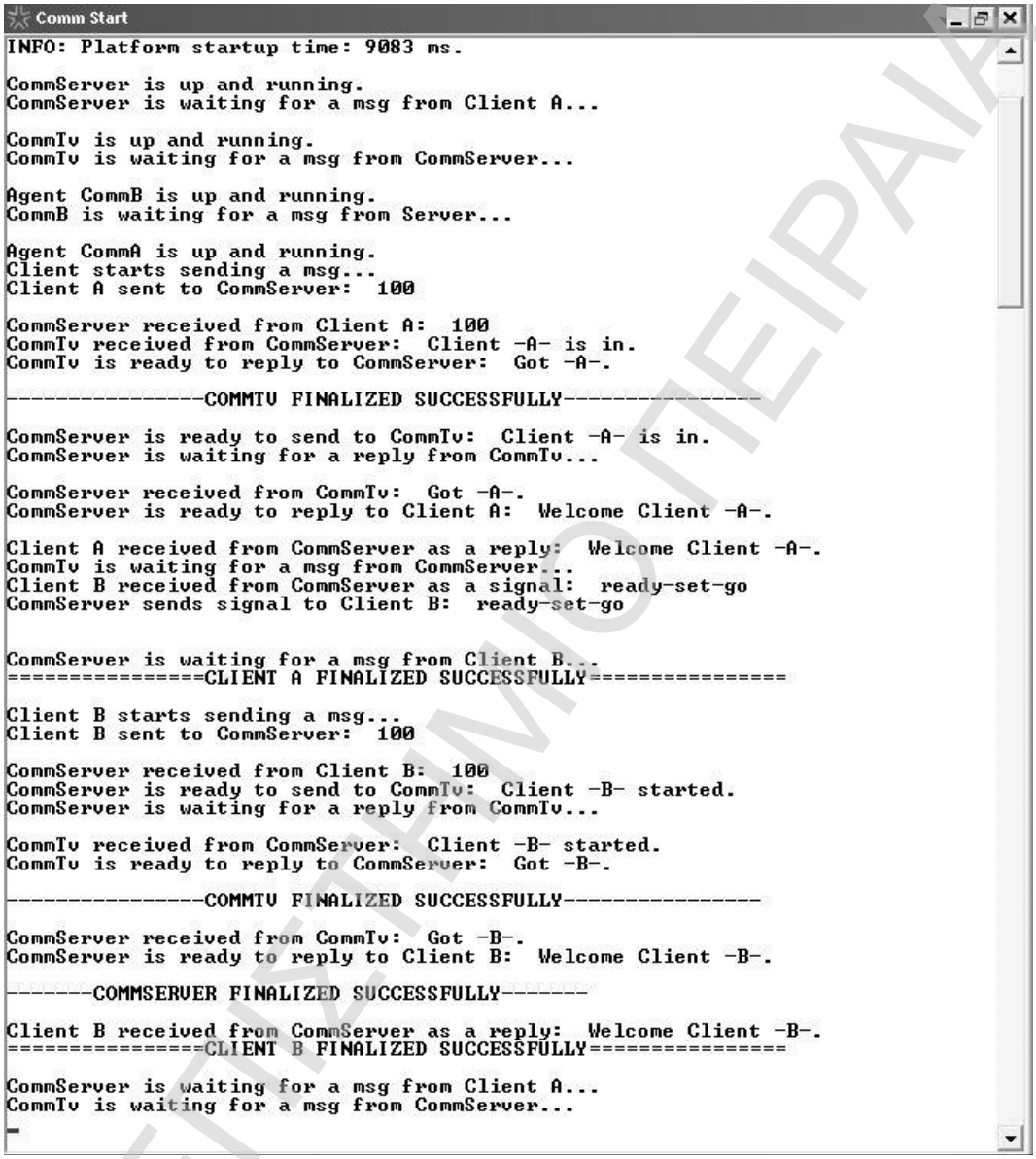
III/ Από ομάδα οντοτήτων 3 > (ΝΕΟΣ) CommB πράκτορας (Client TV2)

IV/ Από ομάδα οντοτήτων 1 > (ΝΕΟΣ) CommA πράκτορας (Client TV1)

Οι CommServer και CommTv βρισκονταί συνεχώς σε λειτουργία ώστε να καλύπτουν τις ανάγκες ανταλλαγής μηνυμάτων των χρηστών.

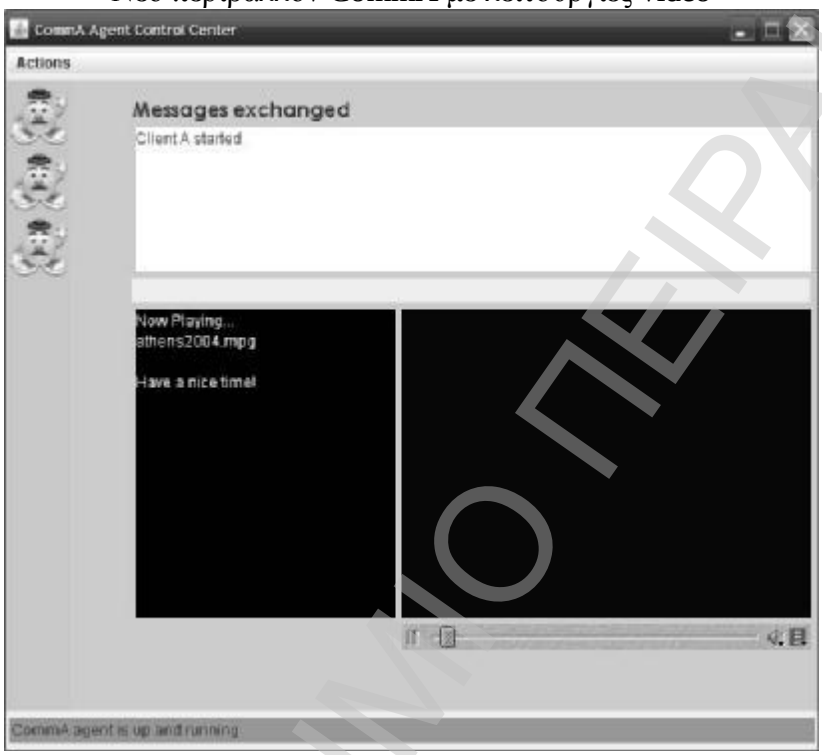
1. Επομένως, ο νέος CommA στέλνει μήνυμα στον CommServer και ξεκινάει την αναπαραγωγή του video (e.g. 100)
2. Ο CommServer λαμβάνει το μήνυμα και ενημερώνει το CommTv (e.g. A is in)
3. Ο CommTv απαντά στον CommServer (e.g. Got A) και αναμένει για νέα μηνύματα.
4. Ο CommServer απαντά στον CommA (e.g. Welcome A. TV is ready) και περιμένει για νέα μηνύματα από τους CommA ή CommB.
5. Ο CommA λαμβάνει την επιβεβαίωση του CommServer και τερματίζει.
6. Μετά, ο CommB (ο οποίος βρίσκεται σε κατάσταση αναμονής), λαμβάνει "Go" ως μήνυμα από τον CommServer και συνεχίζει την αναπαραγωγή του video από το σημείο που το σταμάτησε ο CommA.
7. Ο CommB ενημερώνει τον CommServer ότι έχει ήδη ξεκινήσει.
8. Ο CommServer ενημερώνει το CommTv Controller και ο CommTv απαντά στον CommServer.
9. Ο CommServer απαντά στον πράκτορα CommB και αναμένει για νέα μηνύματα από τους CommA ή CommB.
10. Το αρχείο video τερματίζει.

Σχήμα 4.12: Σενάριο "B" –Αποτελέσματα

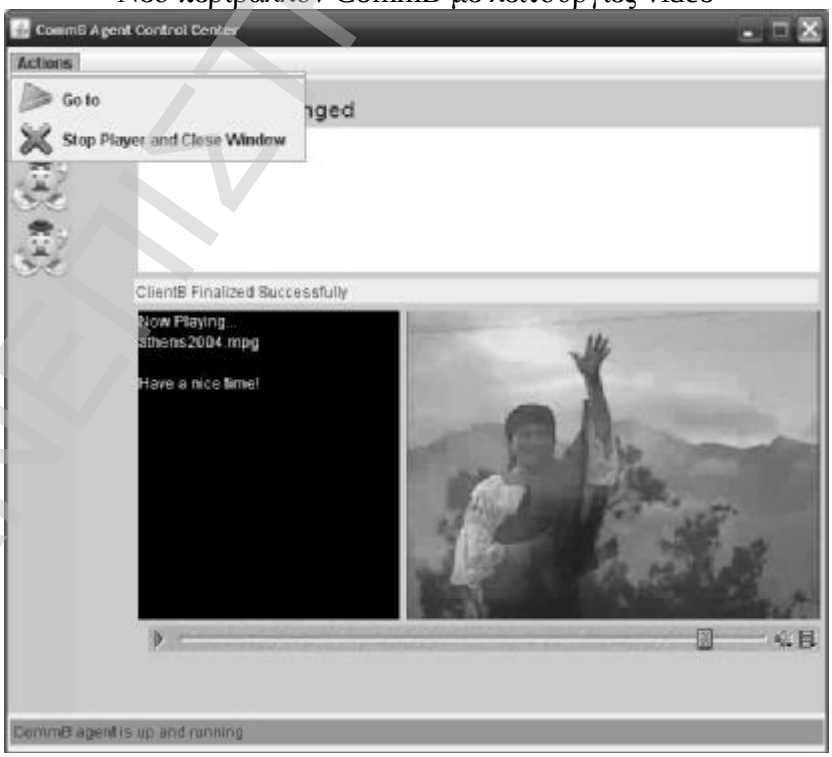


Σχήμα 4.13: Γραφικό περιβάλλον πρακτόρων σεναρίου "Β"

Νέο περιβάλλον CommA με λειτουργίες video



Νέο περιβάλλον CommB με λειτουργίες video



5. Συμπεράσματα

5.1 Ανασκόπηση

Το Jadex είναι μια πλατφόρμα middleware η οποία επιτρέπει την επικοινωνία μεταξύ πρακτόρων. Οι πράκτορες μπορούν να ενσωματωθούν και να εκτελεστούν σε διάφορες συσκευές ώστε αυτές οι συσκευές να έχουν την ικανότητα αλληλεπίδρασης. Επομένως, το σύστημά μας αναπτύχθηκε έχοντας υπόψη το ακόλουθο πλάνο ροής:

1. Ένας βασικός εξυπηρετητής αναμένει για μηνύματα από συγκεκριμένους πράκτορες λογισμικού και απαντά σε αυτά τα μηνύματα. Ο εξυπηρετητής έχει την ικανότητα επικοινωνίας και συνεργασίας με οικιακές συσκευές αλλά και με τις κινητές συσκευές των χρηστών.

2. Οι οικιακές συσκευές έχουν προεγκατεστημένους πράκτορες λογισμικού οι οποίοι αναμένουν για μηνύματα από τον εξυπηρετητή.

3. Οι κινητές συσκευές των χρηστών έχουν προεγκατεστημένους πράκτορες που επιτρέπουν την επικοινωνία με τον εξυπηρετητή.

Επίσης, στην πρόταση υλοποίησης συμπεριλαμβάνονται ενδεικτικές τεχνολογίες ασύρματης επικοινωνίας οι οποίες καλύπτουν τις ανάγκες αλληλεπίδρασης των εμπλεκομένων συσκευών του συστήματος (π.χ. PDA, αισθητήρες, εξυπηρετητές, κέντρα ελέγχου αισθητήρων, τερματικές συσκευές παροχής υπηρεσιών κτλ.). Ως εκ τούτου κρίθηκε απαραίτητη η συνοπτική ανάλυση τεχνολογικών χαρακτηριστικών του Bluetooth, του WLAN, του Wireless USB, της τεχνολογίας ZigBee κ.ά., ώστε η μελέτη να προτείνει ένα

ολοκληρωμένο μοντέλο υλοποίησης και σε συνδυασμό με την ενδεικτική εφαρμογή στην πλατφόρμα Jadex να υπάρχει η δυνατότητα οπτικοποίησης των αποτελεσμάτων της διαδικασίας εκτέλεσης έξυπνων υπηρεσιών μέσω πρακτόρων λογισμικού.

5.2 Εργαλεία

Κατά τη διαδικασία της μελέτης και ανάπτυξης του συστήματος χρησιμοποιήθηκαν διάφορα εργαλεία όπως:

i/ Java v.1.6 or greater (2007Q2)
ii/ Java Media Framework 2.1.1e or greater
iii/ Jade v.3.5 (released 25-June-07)
iv/ Jadex v.0.96 (released 15-June-07)
v/ Protégé v.3.3.1 (2007Q3)

Συντομογραφίες

ADF	Agent Definition File
BDI	Belief-Desire-Intention
CPU	Central Processing Unit
ETH	Eidgenossische Technische Hochschule (Zurich)
FIPA	Foundation for Intelligent Physical Agents
GPRS	General Packet Radio Service
GPS	Global Positioning System
HTML	Hyper-Text Markup Language
ID	Identification
IEEE	Institute of Electrical and Electronics Engineers
IrDA	Infrared Data Association
JADE	Java Agent Development Environment
JADEX	Java Agent Development Environment and XML
mW	milliWatt
OGC	Open Geospatial Consortium
OWL	Ontology Web Language
PAN	Personal Area Network
PDA	Personal Digital Agent
QoS	Quality of Service
R&D	Research and Development
RDF	Resource Description Framework
RFID	Radio-Frequency Identification
URI	Uniform Resource Identifier
USB	Universal Serial Bus
W3C	World Wide Web Consortium
WCDMA	Wideband Code Division Multiple Access
WHO	World Health Organization
WiFi	Wireless Fidelity
WLAN	Wireless Local Area Network
XML	eXtensible Markup Language

Πηγές

1. Bellavista, P., Corradi, A., Foschini, L., "Context-aware handoff middleware for transparent service continuity in wireless networks", *Pervasive and mobile computing,* nr. 3, 2007, pp.439-466.

2. Bellifemine, F., Caire, G., Greenwood, D., *Developing multi-agent systems with JADE,* Wiley and Sons, 2007.

3. Bellifemine, F., Caire, G., Poggi, A., Rimassa G., "JADE", *Exp,* vol. 3, nr. 3, 2003, pp.6-19.

4. Chevron Corporation –Renewables Project

5. De Turck, F., Vanhastel, S., Volckaert, B., Demeester, P., "A generic middleware-based platform for scalable cluster computing", *Future Generation Computer Systems,* nr. 18, 2002, pp. 549-560.

6. Electricite de France (EDF) Research and Development

7. Eriksson, H., "The semantic-document approach to combining documents and ontologies", *International Journal of Human-Computer Studies,* nr. 65, pp. 624–639, 2007.

8. ETH –Swiss Federal Institute of Technology Zurich

9. Framling, K., Ala-Risku, T., Karkkainen, M., Holmstrom, J., "Agent-based model for managing composite product information", *Computers in Industry,* nr. 57, 2006, pp. 72-81.

10. Friedewald M., Da Costa, O., Punie, Y., Alahuhta, P., Heinonen, S., "Perspectives of ambient intelligence in the home environment", *Telematics and Informatics,* nr. 22, 2005, pp. 221–238.

11. Google Incorporated

12. HTML Specification –World Wide Web Consortium

13. Java Agent Development Environment –Telecom Italia Labs

14. Jovanovic, J., Gasevic, D., "Achieving knowledge interoperability: An XML/XSLT approach", *Expert Systems with Applications,* nr. 29, pp. 535–553, 2005.

15. Kingston, J., "Multi-perspective ontologies: Resolving common ontology development problems", *Expert Systems with Applications,* nr. 34, pp. 541–550, 2008.

16. Lee, W., "Deploying personalized mobile services in an agent-based environment", *Expert Systems with Applications,* nr. 32, 2007, pp. 1194-1207.

17. Moreno, A., Garbay, C., "Software agents in health care", *Artificial Intelligence in Medicine,* nr.27, 2003, pp. 229-232.

18. Nijholt, A., "Google home: Experience, support and re-experience of social home activities", *Information Sciences,* 2007.

19. Nwana, H., "Software agents", *Knowledge Engineering Review,* vol. 11, nr. 3, pp.1-40, 1996.

20. Pokahr, A., Braubach, L., *Jadex Tool Guide,* 2007.

21. Pokahr, A., Braubach, L., *Jadex Tutorial,* 2007.

22. Pokahr, A., Braubach, L., *Jadex User Guide,* 2007.

23. Siemens Aktiengesellschaft, "Intelligent Networking", *Pictures of the Future,* 2005.

24. Siemens Aktiengesellschaft, "Invisible Helpers", *Pictures of the Future,* 2001.

25. Siemens Aktiengesellschaft, "The Thinking Web", *Pictures of the Future,* 2002.

26. Siemens Research and Development Division

27. Wikipedia.org –ontology, software agents, middleware, bdi reasoning, wireless networking

28. World Health Organization

29. XML Specification –World Wide Web Consortium

30. FIPA Specification SC00061, FIPA ACL Message Structure Specification.

31. FIPA Specification SC00037, FIPA Communicative Act Library Specification.

32. Jadex Project Official Website: http://vsis-www.informatik.uni-hamburg.de/projects/jadex/

33. Miguel, S., "Ontology of systems and software engineering", *Advanced Engineering Informatics*, nr. 21, 2007, pp.117–118.

34. Stuckenschmidt, H., Klein, M., "Reasoning and change management in modular ontologies", *Data & Knowledge Engineering,* nr. 63, 2007, pp.200-223.

35. NYSE Euronext Group/ NYMEX

36. Charlet D., Issarny, V., Chibout, R., "Energy-efficient middleware-layer multi-radio networking: An assessment in the area of service discovery", *Computer Networks*, nr. 52, 2008, pp.4–24.

37. Castelli, G., Mamei, M., Zambonelli, F., "Engineering contextual knowledge for autonomic pervasive services", *Information and Software Technology*, nr. 50, 2008, pp.36–50.

38. Qin, W., Shi, Y., Suo, Y., "Ontology-Based Context-Aware Middleware for Smart Spaces", *TSINGHUA Science and Technology*, vol. 12, nr. 6, December 2007, pp.707-713.

39. Hengartnera, U., Steenkisteb, P., "Avoiding privacy violations caused by context-sensitive services", *Pervasive and Mobile Computing*, nr. 2, 2006, pp. 427–452.

40. Toninelli, A., Corradi, A., Montanari, R., "Semantic-based discovery to support mobile context-aware service access", *Computer Communications*, January 2008.

41. Sachs, E., Vendetti, J., "Getting Started with Protégé", Stanford Medical Informatics, June 2006.

42. Java version 6 Documentation, Sun Microsystems, 2008.

43. Java Media Framework Documentation, Sun Microsystems, 2008.

44. ZigBee Alliance –ZigBee Technology documents

45. Bluetooth Special Interest Group –Bluetooth Technology documents

46. IEEE 802.11 WIRELESS LOCAL AREA NETWORKS –The Working Group for WLAN Standards –Documentation

47. The Wireless Universal (Certified) Serial Bus Promoter Group –WUSB Technology Documents

48. Η εργασία έχει δομηθεί σύμφωνα με υποδείξεις από τον 'ΟΔΗΓΟ ΕΚΠΟΝΗΣΗΣ ΔΙΠΛΩΜΑΤΙΚΩΝ ΕΡΓΑΣΙΩΝ για το Πρόγραμμα Μεταπτυχιακών Σπουδών στη Διδακτική Τεχνολογίας και Ψηφιακά Συστήματα' Έκδοση 0.9^{A} , Δεκέμβριος 2005 του Λέκτορα κ. Ιωάννη Παραβάντη.

ΠΑΡΑΡΤΗΜΑ Α

Παράρτημα Α

Α.1 Ροή εκτέλεσης της εφαρμογής

Βήμα 1
Λήψη και εγκατάσταση της Java στον Η/Υ από : www.java.com/getjava/
Για την αναπαραγωγή πολυμεσικού περιεχομένου μέσω Java πρέπει να γίνει επιπρόσθετη λήψη και εγκατάσταση του Java Media Framework από: http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/download.html

Βήμα 2
Λήψη και εγκατάσταση της πλατφόρμας Jadex.
Λεπτομερής κώδικας και αρχεία εγκατάστασης βρίσκονται στη διεύθυνση: http://vsis-www.informatik.uni-hamburg.de/projects/jadex/

Βήμα 3
Μόλις η πλατφόρμα εγκατασταθεί επιτυχώς ανοίγουμε ένα παράθυρο command prompt και πληκτρολογούμε την ακόλουθη για την έναρξη της πλατφόρμας:
java jadex.adapter.standalone.Platform

Βήμα 4
Ο ηλεκτρονικός φάκελος που δίδεται με την εργασία πρέπει να επικολληθεί στο φάκελο examples του Jadex. Κάνουμε compile τα .java αρχεία του φακέλου μας και φορτώνουμε τους πράκτορες στο Jadex. Αυτό το πετυχαίνουμε επιλέγοντας Browse στο Jadex.

Βήμα 5
Ξεκινάμε τους πράκτορες με τη σειρά που αναφέρεται στην παρούσα μελέτη και βλέπουμε τα αποτελέσματα στην οθόνη τα οποία πρέπει να ομοιάζουν με αυτά που έχουν τυπωθεί στις εικόνες της μελέτης.

Για την εκκίνηση κάποιου πράκτορα χωρίς τη χρήση του γραφικού περιβάλλοντος του Jadex αρκεί να γράψουμε την ακόλουθη εντολή στο command prompt:
java jadex.adapter.standalone.Platform -nogui
"CommServer:jadex/examples/mycommunication/CommServer.agent.xml(default)"

A.2 Κώδικας εφαρμογής με σχόλια

Γραφικό περιβάλλον πράκτορα CommA

Ευθύνεται για την κατασκευή των γραφικών του πράκτορα

```
package jadex.examples.mycommunicationwgui;

import java.awt.*;
import java.io.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.MalformedURLException;
import java.net.URL;
import java.io.IOException;
import java.io.File;
import javax.media.CannotRealizeException;
import javax.media.Manager;
import javax.media.NoPlayerException;
import javax.media.IncompatibleTimeBaseException;
import javax.media.Player;
import javax.media.Time;
import javax.swing.JPanel;
import jadex.adapter.fipa.SFipa;
import jadex.adapter.fipa.*;
import jadex.adapter.fipa.AgentIdentifier;


public class CommApanel extends JPanel implements WindowListener,
ActionListener, MouseListener {

	JLabel l01,l02,myback,jl1,jl2,jl3;
	Label st2;
	JTextArea responseArea,st1;
	JTextField responseArea2, responseArea3;
	JButton b1;
	Image image1,x,plus,diamonds,imageback;
	JMenuBar mb;
	JFrame mywindow;
	ImageIcon iback;
	Component video,controls,compo;
	URL mediaURL;
	Player mediaPlayer;
	double d,then;
	String ichoose,when,where,full;
	JFileChooser chooser;

	//=============================================================
=====
	public CommApanel()
	{
		image1 =
Toolkit.getDefaultToolkit().getImage("jadex/examples/mycommunicationwgui/
images/arrow.png");
		//image1 =
Toolkit.getDefaultToolkit().getImage("images/arrow.png");
		ImageIcon ic1 = new ImageIcon(image1);
```

```
            x =
Toolkit.getDefaultToolkit().getImage("jadex/examples/mycommunicationwgui/
images/cancel.png");
            //x =
Toolkit.getDefaultToolkit().getImage("images/cancel.png");
            ImageIcon ix = new ImageIcon(x);

            plus =
Toolkit.getDefaultToolkit().getImage("jadex/examples/mycommunicationwgui/
images/wave.png");
            ImageIcon iplus = new ImageIcon(plus);

            diamonds =
Toolkit.getDefaultToolkit().getImage("jadex/examples/mycommunicationwgui/
images/diamonds.png");
            ImageIcon idiamonds = new ImageIcon(diamonds);

            imageback =
Toolkit.getDefaultToolkit().getImage("images/magic02.png");
            iback = new ImageIcon(imageback);

            //-----------------------------
            JMenu ch1 = new JMenu("Actions");
            ch1.addSeparator();

            JMenuItem closew = new JMenuItem("Stop Player and Close
Window",ix);
            closew.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        //System.exit(0);
                        mywindow.dispose();
                        mediaPlayer.stop();
                  }
            });


            JMenuItem leave = new JMenuItem("Go to",ic1);
            leave.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        //new play

                        /*try {
                        BufferedWriter lineout = new
BufferedWriter(new OutputStreamWriter(System.out));
                        lineout.write("1",0,1);
                        lineout.newLine();
                        lineout.close();
                        lineout.flush();

                        }catch (Exception ee)
{  }//e.printStackTrace();*/

                  }
            });

            ch1.add(leave);
            ch1.add(closew);

            //-----------------------------


            mb = new JMenuBar();
            mb.setPreferredSize(new Dimension(630,16));
            mb.add(ch1);
```

```
		//----------------------------

		b1 = new JButton("Send Now");
		b1.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				//.....
			}
		});

		l02 = new JLabel("");
		l02.setPreferredSize(new Dimension(530,2));

		l01 = new JLabel("Messages exchanged");
		l01.setPreferredSize(new Dimension(530,20));
		l01.setFont(new Font("Century Gothic", Font.BOLD, 16));
		l01.setBackground(new Color(255,255,153));

		responseArea = new JTextArea(7,48);
		responseArea.setLineWrap(true);
		responseArea.setEditable(false);

		responseArea2 = new JTextField(48);
		responseArea2.setEditable(false);

		//responseArea3 = new JTextField("Hello B. I'm CommA
graphical.",48);
		//responseArea3.setEditable(true);

		st1 = new JTextArea();
		st1.setPreferredSize(new Dimension(203,240));
		st1.setBackground(new Color(0,0,0));
		st1.setForeground(new Color(255,255,255));
		st1.setLineWrap(true);
		st1.setEditable(false);

		st2 = new Label("CommA agent is up and running");
		st2.setPreferredSize(new Dimension(630,20));
		st2.setBackground(new Color(0,255,0));

		myback = new JLabel("",iback,0);
		jl1 = new JLabel(iplus); jl2 = new JLabel(iplus); jl3 = new
JLabel(iplus);
		//myback.setPreferredSize(new Dimension(530,120));
		//-------------------------------------------------------
---------starts v

		File myfile=new File("multi");
		chooser = new JFileChooser();
		int returnVal = chooser.showOpenDialog(compo);
		if(returnVal == JFileChooser.APPROVE_OPTION) {
			ichoose=chooser.getSelectedFile().getName();
			where=myfile.getAbsolutePath()+"\\";
			full="file:"+where+ichoose;
			System.out.println("Your choise: "+where+ichoose);
			AuxClass.ert3=full; AuxClass.ert1=ichoose;
			st1.setText("Now Playing...\n"+ichoose+"\n\nHave a
nice time!");
		}


		Manager.setHint( Manager.LIGHTWEIGHT_RENDERER, true );
		try {

			mediaURL= null;
```

```
                    mediaURL = new URL(full);
                    mediaPlayer =
Manager.createRealizedPlayer( mediaURL );
                    video = mediaPlayer.getVisualComponent();
                    video.setPreferredSize(new Dimension(320,240));
                    controls = mediaPlayer.getControlPanelComponent();
                    controls.setPreferredSize(new Dimension(320,20));
                    mediaPlayer.start();


      //mediaPlayer.setTimeBase(Manager.getSystemTimeBase());
                    double now =
mediaPlayer.getTimeBase().getTime().getSeconds();//find the time now
                    //double then = now + 9.0;
                    //mediaPlayer.setStopTime(new Time(5.0));
                    //mediaPlayer.setMediaTime(new Time(then)); //set
media time
                    //System.out.println(then);
                    //mediaPlayer.setRate((float)1.0); //set rate
                    //mediaPlayer.syncStart(new Time(then)); //start the
clock

      //System.out.println(mediaPlayer.getMediaTime().toString());
            }//closes try
            catch ( NoPlayerException
noPlayerException ){System.err.println( "No media player found" );} //
end catch
            catch ( CannotRealizeException
cannotRealizeException ){System.err.println( "Could not realize media
player" );} // end catch
            catch ( IOException iOException ){System.err.println( "Error
reading from the source" );} // end catch
            //catch ( IncompatibleTimeBaseException
incException ){System.err.println( "Incompatibility Error reading from
the source" );}


            //-------------------------------------------------------
-----------closes v


      } //closes constructor

      //----------------------------------------

      public void createMap() {

            mywindow = new JFrame ();

            JPanel NorthernLine=new JPanel();
            //NorthernLine.setBackground(new Color(255,255,153));
            NorthernLine.add(mb);

            JPanel PiccadillyLine=new JPanel();
            PiccadillyLine.setBackground(new Color(200,200,200));
            PiccadillyLine.setPreferredSize(new Dimension(50,0));
            PiccadillyLine.setLocation(0,0);
            PiccadillyLine.add(jl1);
            PiccadillyLine.add(jl2);
            PiccadillyLine.add(jl3);
            //PiccadillyLine.add(bbb);

            JPanel CentralLine=new JPanel();
            CentralLine.setBackground(new Color(135,231,252));
            CentralLine.setLayout(new FlowLayout(FlowLayout.RIGHT));
```

```
                CentralLine.add(l02);
                CentralLine.add(l01);
                CentralLine.add(responseArea);
                CentralLine.add(responseArea2);
                //CentralLine.add(myback);
                CentralLine.add(st1);
                if ( video != null )CentralLine.add(video);
                if ( controls != null )CentralLine.add(controls);

                JPanel DistrictLine=new JPanel();
                //DistrictLine.setBackground(new Color(255,255,153));
                DistrictLine.add(st2);

                //-----------------------------------------

                Container mycontainer=mywindow.getContentPane();
                mycontainer.setLayout(new BorderLayout(0,0)); //
(horizontal,vertical)
                mycontainer.add(NorthernLine,BorderLayout.NORTH);
                mycontainer.add(PiccadillyLine,BorderLayout.WEST);
                mycontainer.add(CentralLine,BorderLayout.CENTER);
                mycontainer.add(DistrictLine,BorderLayout.SOUTH);

                Dimension windowSize = new Dimension(640, 580);
                //Dimension windowSize=window.getToolkit().getScreenSize();

                mywindow.setBounds(windowSize.width/4,windowSize.height/4,
windowSize.width/1,windowSize.height/1); //position, size
                mywindow.setResizable(false);
                mywindow.setTitle("CommA Agent Control Center");
                mywindow.setVisible(true);
                mywindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        } //closes Map
        //-----------------------------------------

        public void actionPerformed(ActionEvent e) {}
        public void windowActivated(WindowEvent e) {}
        public void windowClosed(WindowEvent e) {}
        public void windowClosing(WindowEvent e) { System.exit(0); }
        public void windowDeactivated(WindowEvent e) {}
        public void windowDeiconified(WindowEvent e) {}
        public void windowIconified(WindowEvent e) {}
        public void windowOpened(WindowEvent e) {}
        public void mouseClicked (MouseEvent e) {}
        public void mouseEntered (MouseEvent e) {}
        public void mouseExited (MouseEvent e) {}
        public void mousePressed (MouseEvent e) {}
        public void mouseReleased (MouseEvent e) {}

        /*public static void main(String [] args) {
                CommApanel wow = new CommApanel();
                wow.createMap();

        }*/
}//closes class
```

Γραφικό περιβάλλον πράκτορα CommB

Ευθύνεται για την κατασκευή των γραφικών του πράκτορα

```
package jadex.examples.mycommunicationwgui;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.MalformedURLException;
import java.net.URL;
import java.io.IOException;
import javax.media.CannotRealizeException;
import javax.media.Manager;
import javax.media.NoPlayerException;
import javax.media.IncompatibleTimeBaseException;
import javax.media.Player;
import javax.media.Time;
import javax.swing.JPanel;

public class CommBpanel extends JPanel implements WindowListener,
ActionListener, MouseListener {

	JLabel l01,l02,myback,jl1,jl2,jl3;
	Label st2;
	JTextArea responseArea,st1;
	JTextField responseArea2, responseArea3;
	JButton b1;
	Image image1,x,plus,diamonds,imageback;
	JMenuBar mb;
	JFrame mywindow;
	ImageIcon iback;
	Component video,controls;
	URL mediaURL;
	Player mediaPlayer2;
	CommApanel aa;
	double whatd2, ddd;


	//============================================================
=====
	public CommBpanel(double whatd )
	{

		image1                                                    =
Toolkit.getDefaultToolkit().getImage("jadex/examples/mycommunicationwgui/
images/arrow.png");
		//image1                                                  =
Toolkit.getDefaultToolkit().getImage("images/arrow.png");
		ImageIcon ic1 = new ImageIcon(image1);

		x                                                         =
Toolkit.getDefaultToolkit().getImage("jadex/examples/mycommunicationwgui/
images/cancel.png");
		//x                                                       =
Toolkit.getDefaultToolkit().getImage("images/cancel.png");
		ImageIcon ix = new ImageIcon(x);

		plus                                                      =
Toolkit.getDefaultToolkit().getImage("jadex/examples/mycommunicationwgui/
images/wave.png");
		ImageIcon iplus = new ImageIcon(plus);
```

```
            diamonds                                                    =
Toolkit.getDefaultToolkit().getImage("jadex/examples/mycommunicationwgui/
images/diamonds.png");
            ImageIcon idiamonds = new ImageIcon(diamonds);

            imageback                                                   =
Toolkit.getDefaultToolkit().getImage("images/magic02.png");
            iback = new ImageIcon(imageback);

            //-----------------------------
            JMenu ch1 = new JMenu("Actions");
            ch1.addSeparator();

            JMenuItem  closew  =  new  JMenuItem("Stop  Player  and  Close
Window",ix);
            closew.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        //System.exit(0);
                        mywindow.dispose();
                        mediaPlayer2.stop();
                  }
            });

            JMenuItem leave2 = new JMenuItem("Go to",ic1);
            leave2.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        //open here the new agent class
                  }
            });

            ch1.add(leave2);
            ch1.add(closew);

            //-----------------------------


            mb = new JMenuBar();
            mb.setPreferredSize(new Dimension(630,16));
            mb.add(ch1);


            //-----------------------------

            b1 = new JButton("Send Now");
            b1.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        //.....
                  }
            });

            l02 = new JLabel("");
            l02.setPreferredSize(new Dimension(530,2));

            l01 = new JLabel("Messages exchanged");
            l01.setPreferredSize(new Dimension(530,20));
            l01.setFont(new Font("Century Gothic", Font.BOLD, 16));
            l01.setBackground(new Color(255,255,153));

            responseArea = new JTextArea(7,48);
            responseArea.setLineWrap(true);
            responseArea.setEditable(false);

            responseArea2 = new JTextField(48);
            responseArea2.setEditable(false);
```

```
            //responseArea3  =  new  JTextField("Hello  B.  I'm  CommA
graphical.",48);
            //responseArea3.setEditable(true);

            st1 = new JTextArea();
            st1.setPreferredSize(new Dimension(203,240));
            st1.setBackground(new Color(0,0,0));
            st1.setForeground(new Color(255,255,255));
            st1.setLineWrap(true);
            st1.setEditable(false);

            st2 = new Label("CommB agent is up and running");
            st2.setPreferredSize(new Dimension(630,20));
            st2.setBackground(new Color(0,255,0));

            myback = new JLabel("",iback,0);
            jl1 = new JLabel(iplus); jl2 = new JLabel(iplus); jl3 = new
JLabel(iplus);
            //myback.setPreferredSize(new Dimension(530,120));
            //-------------------------------------------------------
---------starts v
            Manager.setHint( Manager.LIGHTWEIGHT_RENDERER, true );
            try {
                  mediaURL= null;
                  mediaURL = new URL(AuxClass.ert3);
                  st1.setText("Now
Playing...\n"+AuxClass.ert1+"\n\nHave a nice time!");
                  mediaPlayer2                                        =
Manager.createRealizedPlayer( mediaURL );
                  video = mediaPlayer2.getVisualComponent();
                  video.setPreferredSize(new Dimension(520,240));
                  controls = mediaPlayer2.getControlPanelComponent();
                  controls.setPreferredSize(new Dimension(520,20));
                  mediaPlayer2.start();

      //mediaPlayer2.setTimeBase(Manager.getSystemTimeBase());
                  //double                        now                   =
mediaPlayer2.getTimeBase().getTime().getSeconds();//find the time now
                  //double then = now + 9.0;//5 secs later
                  whatd2=whatd;
                  //System.out.println("kkkkkkkkkkkkkk       "+whatd2+"
"+whatd);
                  mediaPlayer2.setMediaTime(new   Time(whatd2));   //set
media time
                  //System.out.println(then);
                  //mediaPlayer2.setRate((float)1.0); //set rate
                  //mediaPlayer2.syncStart(new Time(then)); //start the
clock
                  //mediaPlayer2.setStopTime(new Time(then));

      //System.out.println(mediaPlayer2.getMediaTime().toString());
            }//closes try
            catch                  (                  NoPlayerException
noPlayerException ){System.err.println( "No media player found" );} //
end catch
            catch               (               CannotRealizeException
cannotRealizeException  ){System.err.println(  "Could  not  realize  media
player" );} // end catch
            catch ( IOException iOException ){System.err.println( "Error
reading from the source" );} // end catch
            //catch           (           IncompatibleTimeBaseException
incException  ){System.err.println(  "Incompatibility  Error  reading  from
the source" );}
            //-------------------------------------------------------
-----------closes v
```

```
	} //closes constructor

	//-----------------------------------------

	public void createMap() {

		mywindow = new JFrame ();

		JPanel NorthernLine=new JPanel();
		//NorthernLine.setBackground(new Color(255,255,153));
		NorthernLine.add(mb);

		JPanel PiccadillyLine=new JPanel();
		PiccadillyLine.setBackground(new Color(200,200,200));
		PiccadillyLine.setPreferredSize(new Dimension(50,0));
		PiccadillyLine.setLocation(0,0);
		PiccadillyLine.add(jl1);
		PiccadillyLine.add(jl2);
		PiccadillyLine.add(jl3);
		//PiccadillyLine.add(bbb);

		JPanel CentralLine=new JPanel();
		CentralLine.setBackground(new Color(135,231,252));
		CentralLine.setLayout(new FlowLayout(FlowLayout.RIGHT));
		CentralLine.add(l02);
		CentralLine.add(l01);
		CentralLine.add(responseArea);
		CentralLine.add(responseArea2);
		CentralLine.add(st1);
		//CentralLine.add(myback);
		if ( video != null )CentralLine.add(video);
		if ( controls != null )CentralLine.add(controls);

		JPanel DistrictLine=new JPanel();
		//DistrictLine.setBackground(new Color(255,255,153));
		DistrictLine.add(st2);

		//-----------------------------------------

		Container mycontainer=mywindow.getContentPane();
		mycontainer.setLayout(new	BorderLayout(0,0));	//
(horizontal,vertical)
		mycontainer.add(NorthernLine,BorderLayout.NORTH);
		mycontainer.add(PiccadillyLine,BorderLayout.WEST);
		mycontainer.add(CentralLine,BorderLayout.CENTER);
		mycontainer.add(DistrictLine,BorderLayout.SOUTH);

		Dimension windowSize = new Dimension(640, 580);
		//Dimension windowSize=window.getToolkit().getScreenSize();

		mywindow.setBounds(windowSize.width/4,windowSize.height/4,
windowSize.width/1,windowSize.height/1); //position, size
		mywindow.setResizable(false);
		mywindow.setTitle("CommB Agent Control Center");
		mywindow.setVisible(true);
		mywindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

	} //closes Map
	//-----------------------------------------

	public void actionPerformed(ActionEvent e) {}
	public void windowActivated(WindowEvent e) {}
	public void windowClosed(WindowEvent e) {}
```

```
        public void windowClosing(WindowEvent e) { System.exit(0); }
        public void windowDeactivated(WindowEvent e) {}
        public void windowDeiconified(WindowEvent e) {}
        public void windowIconified(WindowEvent e) {}
        public void windowOpened(WindowEvent e) {}
        public void mouseClicked (MouseEvent e) {}
        public void mouseEntered (MouseEvent e) {}
        public void mouseExited (MouseEvent e) {}
        public void mousePressed (MouseEvent e) {}
        public void mouseReleased (MouseEvent e) {}

        /*           public static void main(String [] args) {
        CommBpanel wow = new CommBpanel();
        wow.createMap();

        }*/
}//closes class
```

CommA Plan (για Jadex)

Java πλάνο του CommA xml πράκτορα λογισμικού

```
package jadex.examples.mycommunicationwgui;

import jadex.runtime.*;
import jadex.adapter.fipa.SFipa;
import javax.swing.JOptionPane;
import jadex.adapter.fipa.*;
import jadex.adapter.fipa.AgentIdentifier;
import java.io.*;
import java.awt.Color;
import javax.media.Manager;

public class CommPlanA extends Plan
{
	MyOntology ont100;
	CommApanel now01;
	String str1,str3,sth;
	String str301,mysel,what;
	double now5;

	public void body()
	{

		ont100=new MyOntology(); //ontology class

		str1="\n"+getBeliefbase().getBelief("msg").getFact();
		System.out.println(str1);

		//mysel = JOptionPane.showInputDialog(null, str1+"\nPlease
select 100 or 200", "Enter Message", JOptionPane.QUESTION_MESSAGE);
		mysel="100";
		now01 = new CommApanel();
		now01.createMap();
		now01.responseArea.append("Client A started");


		//try {
		//System.out.print("\nPlease select 100 or 200 > ");
		//BufferedReader linein = new BufferedReader(new
InputStreamReader(System.in));
		//mysel = linein.readLine(); }catch (Exception e)
{ e.printStackTrace(); }
		//JOptionPane.showMessageDialog(null, str1, "Welcome!!",
JOptionPane.INFORMATION_MESSAGE);
		System.out.println("Client starts sending a msg...");
		AgentIdentifier ag1 = new AgentIdentifier("CommServer",
true);
		AMSCreateAgent ac1 = new AMSCreateAgent();
		ac1.setAgentIdentifier(ag1);
		ac1.setStart(true);

		//We create a message event in order to send a message to
CommServer agent, which waits for it.
		IMessageEvent me = createMessageEvent("request_msg");
		if (mysel.equals("100") == true) { ont100.msgA1 = "100"; }
		//else if (mysel.equals("200") == true) { ont100.msgA1 =
"200"; }
		else { ont100.msgA1 = "100"; }
```

```
            str301=ont100.msgA1.toString() ; //msg from ontology
            //     String                                          str12[]
={"192.168.100.33:9876","pc01:9876","pc01:9877","pc01:9875"};
            me.getParameterSet(SFipa.RECEIVERS).addValue(ag1);
            me.setContent(str301);
            sth = (String)me.getContent();
            sendMessage(me);
            System.out.println("Client    A    sent    to    CommServer:
"+sth+"\n");

            //while(true){
            //CommA agent waits for CommServer's reply message.
            IMessageEvent   re   =   waitForMessageEvent("request_msg2");
//pauses here.
            str3 = (String)re.getContent();
            System.out.println("Client A received from CommServer as a
reply:  "+str3);

            what=ont100.selection;
            do{
                  now5                                                =
now01.mediaPlayer.getMediaTime().getSeconds();//find the time now
                  ont100.takeme(now5);
                  AuxClass.ert2=ont100.thatis();
                  now01.mediaPlayer.stop();
                  now01.mywindow.dispose();
                  killAgent();
                  //break;
            }while (what.equals("0")==false);
            //{
            System.out.println("================CLIENT    A    FINALIZED
SUCCESSFULLY================\n");
            now01.responseArea2.setText("ClientA               Finalized
Successfully");
            //}

      }//closes body
}//closes class
```

CommB Plan (για Jadex)

Java πλάνο για τον CommB xml πράκτορα

```
package jadex.examples.mycommunicationwgui;

import jadex.runtime.*;
import jadex.adapter.fipa.SFipa;
import javax.swing.JOptionPane;
import jadex.adapter.fipa.*;
import jadex.adapter.fipa.AgentIdentifier;
import java.io.*;
import java.awt.Color;

public class CommPlanB extends Plan
{
	MyOntology ont1;
	CommApanel now01;
	CommBpanel now02;
	String str1,str2,str3,sth;
	String str301,mysel;
	double dd;

	public void body()
	{

		ont1=new MyOntology(); //ontology class

		str1="\n"+getBeliefbase().getBelief("msg").getFact();
		System.out.println(str1);


		System.out.println("CommB is waiting for a msg from
Server...");
		IMessageEvent re20 = waitForMessageEvent("request_msg20");
//pauses here.
		str2 = (String)re20.getContent();
		System.out.println("Client B received from CommServer as a
signal:  "+str2);


		//mysel = JOptionPane.showInputDialog(null, str1+"\nPlease
select 100 or 200", "Enter Message", JOptionPane.QUESTION_MESSAGE);
		try	{Thread.currentThread().sleep(1000);}	catch
(InterruptedException ie) {}
		mysel="100";
		Double fgh = new Double(str2);
		now02 = new CommBpanel(fgh);
		now02.createMap();
		now02.responseArea.append("Client B started");


		//try {
		//System.out.print("\nPlease select 100 or 200 > ");
		//BufferedReader	linein	=	new	BufferedReader(new
InputStreamReader(System.in));
		//mysel = linein.readLine();	}catch	(Exception	e)
{ e.printStackTrace(); }
		//JOptionPane.showMessageDialog(null,	str1,	"Welcome!!",
JOptionPane.INFORMATION_MESSAGE);
		System.out.println("Client B starts sending a msg...");
		AgentIdentifier ag1 = new AgentIdentifier("CommServer",
true);
```

```
            //We create a message event in order to send a message to
CommServer agent, which waits for it.
            IMessageEvent me = createMessageEvent("request_msg");
            if (mysel.equals("100") == true) { ont1.msgA1 = "100"; }
            else { ont1.msgA1 = "100"; }

            str301=ont1.msgA1.toString() ; //msg from ontology
            //     String                                          str12[]
={"192.168.100.33:9876","pc01:9876","pc01:9877","pc01:9875"};
            me.getParameterSet(SFipa.RECEIVERS).addValue(ag1);
            me.setContent(str301);
            sth = (String)me.getContent();
            sendMessage(me);
            System.out.println("Client    B    sent    to    CommServer:
"+sth+"\n");

            //while(true){
            //CommA agent waits for CommServer's reply message.
            IMessageEvent   re   =   waitForMessageEvent("request_msg2");
//pauses here.
            str3 = (String)re.getContent();
            System.out.println("Client B received from CommServer as a
reply:  "+str3);

            //try      {Thread.currentThread().sleep(1000);}      catch
(InterruptedException ie) {}
            System.out.println("================CLIENT    B    FINALIZED
SUCCESSFULLY===============\n");
            now02.responseArea2.setText("ClientB              Finalized
Successfully");
            //}
            killAgent();
      }//closes body
}//closes class
```

CommServer Plan (για Jadex)

Java πλάνο του CommServer xml πράκτορα λογισμικού

```
package jadex.examples.mycommunicationwgui;

import jadex.runtime.*;
import java.io.*;
import jadex.adapter.fipa.SFipa;
import javax.swing.JOptionPane;
import jadex.adapter.fipa.*;
import jadex.adapter.fipa.AgentIdentifier;

public class CommPlanServer extends Plan
{
	MyOntology ont1;
	BufferedReader linein;
	String str1,str3,sth,str12,sth12;
	String str101,str102,str20,mysel;
	double ert;

	public void body()
	{
		ont1=new MyOntology(); //ontology class

		str1="\n"+getBeliefbase().getBelief("msg").getFact();
		System.out.println(str1);

		//Environment env =
(Environment)getBeliefbase().getBelief("env").getFact();
		//str4="\n"+getBeliefbase().getBelief("env").getFact();
		//JOptionPane.showMessageDialog(null, str1, "Welcome!!",
JOptionPane.INFORMATION_MESSAGE);

		AgentIdentifier ag1 = new AgentIdentifier("CommTv", true);
		AgentIdentifier ag2 = new AgentIdentifier("CommA", true);
		AgentIdentifier ag3 = new AgentIdentifier("CommB", true);

		/*AMSCreateAgent ac1 = new AMSCreateAgent();
		ac1.setAgentIdentifier(ag1);
		ac1.setStart(true);
		System.out.println(ag1.getPlatformName());
		System.out.println(ag1.getLocalName());
		AMSCreateAgent ac2 = new AMSCreateAgent();
		ac2.setAgentIdentifier(ag2);
		ac2.setStart(true);
		System.out.println(ac2.toString());*/

		/*IGoal ca = createGoal("ams_create_agent");
		ca.getParameter("type").setValue("mycommunication.CommTv");
		dispatchSubgoalAndWait(ca);
		AgentIdentifier createdagent = (AgentIdentifier)
		ca.getParameter("agentidentifier").getValue();*/

		/*AgentIdentifier createdagent = (AgentIdentifier)
		ca.getParameter("agentidentifier").getValue();
		IGoal sa = createGoal("ams_start_agent");
		sa.getParameter("ams").setValue(ams); // Set ams in case of
remote platform
		sa.getParameter("agentidentifier").setValue(createdagent);
		dispatchSubgoalAndWait(sa);*/

		while(true){
```

```
                    //CommServer agent waits for CommA's first message.
                    System.out.println("CommServer is waiting for a msg
from Client A...");
                   IMessageEvent me = waitForMessageEvent("request_msg");
//On open, agent pauses here.
                    sth = (String)me.getContent();
                   System.out.println("CommServer received from Client A:
"+sth);


                    if (sth.equals("100")==true) {
                           //CommServer agent sends to CommTv a message.
                           ont1.msgS2 = "Client -A- is in.";
                           IMessageEvent re11 =
createMessageEvent("request_msg11");
                           str102=ont1.msgS2.toString() ; //msg from
ontology
                           //     String str10[] ={"nio-
mtp://192.168.100.33:9877","nio-mtp://pc01:9877","nio-
mtp://pc01:9876","nio-mtp://pc01:9875"}; // nio-mtp://

       re11.getParameterSet(SFipa.RECEIVERS).addValue(ag1);
                           re11.setContent(str102);
                           sendMessage(re11);
                           str12 = (String)re11.getContent();
                           System.out.println("CommServer is ready to
send to CommTv:  "+str12);

                           //CommServer agent waits for CommTv's reply
message.
                           System.out.println("CommServer is waiting for
a reply from CommTv...\n");
                           IMessageEvent me12 =
waitForMessageEvent("request_msg12"); //pauses here.
                           sth12 = (String)me12.getContent();
                           System.out.println("CommServer received from
CommTv:  "+sth12);

                           try {
                                  try {Thread.currentThread().sleep(4000);}
catch (InterruptedException ie) {}



                                   linein = new BufferedReader(new
InputStreamReader(System.in));
                                   do {
                                          System.out.print("\nPlease select
1 to change clients > ");
                                          mysel = linein.readLine();
                                   System.out.print("Bad Command. Please
try again."); }
                                   while(mysel.equals("1") == false);

                                   if (mysel.equals("1") == true) {
                                          ont1.selection=mysel;
                                          //System.out.println("SELECTION
IS>>>"+ont1.selection);


                                          //CommServer agent sends to CommA
a reply message.
```

```
                                        IMessageEvent re =
createMessageEvent("request_msg2");
                                        ont1.msgS1 = "Welcome Client -A-
.";

                                        str101=ont1.msgS1.toString() ;
//msg from ontology
                                        //AgentIdentifier tt[] =
{ag2,ag3};

      re.getParameterSet(SFipa.RECEIVERS).addValue(ag2);
                                        re.setContent(str101);
                                        sendMessage(re);
                                        str3 = (String)re.getContent();
                                        System.out.println("CommServer is
ready to reply to Client A:  "+str3+"\n");


                                        // Going to Client B------------
------------------------------




                                        try
{Thread.currentThread().sleep(3000);} catch (InterruptedException ie) {}
                                        IMessageEvent re20 =
createMessageEvent("request_msg20");
                                        //ont1.msgS1 = "";
                                        //str101=ont1.msgS1.toString() ;
//msg from ontology

      re20.getParameterSet(SFipa.RECEIVERS).addValue(ag3);


                                        Double klm = new Double
(AuxClass.ert2);
                                        re20.setContent(klm.toString());
                                        sendMessage(re20);
                                       str20 = (String)re20.getContent();
                                        System.out.println("CommServer
sends signal to Client B:  "+str20+"\n");


                                        System.out.println("\nCommServer
is waiting for a msg from Client B...");
                                        me =
waitForMessageEvent("request_msg"); //On open, agent pauses here.
                                        sth = (String)me.getContent();
                                        System.out.println("CommServer
received from Client B:  "+sth);

                                        ont1.msgS2 = "Client -B-
started.";
                                        re11 =
createMessageEvent("request_msg11");
                                        str102=ont1.msgS2.toString() ;
//msg from ontology
                                        //    String str10[] ={"nio-
mtp://192.168.100.33:9877","nio-mtp://pc01:9877","nio-
mtp://pc01:9876","nio-mtp://pc01:9875"}; // nio-mtp://

      re11.getParameterSet(SFipa.RECEIVERS).addValue(ag1);
                                        re11.setContent(str102);
                                        sendMessage(re11);
```

```
                                        str12 = (String)re11.getContent();
                                         System.out.println("CommServer is
ready to send to CommTv:  "+str12);

                                         //CommServer agent waits for
CommTv's reply message.
                                         System.out.println("CommServer is
waiting for a reply from CommTv...\n");
                                         me12 =
waitForMessageEvent("request_msg12"); //pauses here.
                                        sth12 = (String)me12.getContent();
                                         System.out.println("CommServer
received from CommTv:  "+sth12);

                                         //CommServer agent sends to CommA
a reply message.
                                         re =
createMessageEvent("request_msg2");
                                         ont1.msgS1 = "Welcome Client -B-
.";
                                         str101=ont1.msgS1.toString() ;
//msg from ontology

      re.getParameterSet(SFipa.RECEIVERS).addValue(ag3);
                                        re.setContent(str101);
                                        sendMessage(re);
                                        str3 = (String)re.getContent();
                                        System.out.println("CommServer is
ready to reply to Client B:  "+str3+"\n");


                                        System.out.println("-------
COMMSERVER FINALIZED SUCCESSFULLY-------\n");
                                        try
{Thread.currentThread().sleep(1000);} catch (InterruptedException ie) {}
                                    }//closes if input

                            }//closes try
                            catch (Exception e) { e.printStackTrace(); }
                        }//closes if 100

      //=============================================================
==========


                        else { System.out.println("-------COMMSERVER
FINALIZED W/E-------\n"); }
                        try {Thread.currentThread().sleep(1000);} catch
(InterruptedException ie) {}

                }//closes while true
      }//closes body
}//closes class
```

CommTv Plan (για Jadex)

Java πλάνο του CommTv xml πράκτορα

```
package jadex.examples.mycommunicationwgui;

import jadex.runtime.*;
import java.awt.Color;
import jadex.adapter.fipa.SFipa;
import javax.swing.JOptionPane;
import jadex.adapter.fipa.*;
import jadex.adapter.fipa.AgentIdentifier;

public class CommPlanTv extends Plan
{
	MyOntology ont1;
	String str1,str3,sth;
	String str201;

	public void body()
	{

		ont1=new MyOntology(); //ontology class

		str1="\n"+getBeliefbase().getBelief("msg").getFact();
		//Environment env2 =
(Environment)getBeliefbase().getBelief("env2").getFact();
		System.out.println(str1);
		//ccc.f.responseArea2.setText("Now Running");
		//JOptionPane.showMessageDialog(null, str1, "Welcome!!",
JOptionPane.INFORMATION_MESSAGE);
		AgentIdentifier ag1 = new AgentIdentifier("CommServer",
true);
		AMSCreateAgent ac1 = new AMSCreateAgent();
		ac1.setAgentIdentifier(ag1);
		ac1.setStart(true);



		while(true){
			//CommTv agent waits for CommServer's first message.
			System.out.println("CommTv is waiting for a msg from
CommServer...");
			IMessageEvent me =
waitForMessageEvent("request_msg11"); //On open, agent pauses here.
			sth = (String)me.getContent();
			System.out.println("CommTv received from CommServer:
"+sth);

			//CommServer agent sends to CommServer a reply
message.
			IMessageEvent re =
createMessageEvent("request_msg12");
			String ch1 = "Client -A- is in.";
			String ch2 = "Client -B- started.";
			String ch3 = ont1.msgS2.toString();
			if (sth.equals(ch1)==true) {ont1.msgT1 = "Got -A-.";}
			else if (sth.equals(ch2)==true) {ont1.msgT1 = "Got -
B-.";}
			else {ont1.msgT1 = "Got E--";}
			str201=ont1.msgT1.toString() ; //msg from ontology
```

```
                    //     String str11[] ={"nio-
mtp://192.168.100.33:9876","nio-mtp://pc01:9876","nio-
mtp://pc01:9877","nio-mtp://pc01:9875"};
                    re.getParameterSet(SFipa.RECEIVERS).addValue(ag1);
                    re.setContent(str201);
                    sendMessage(re);
                    str3 = (String)re.getContent();
                    System.out.println("CommTv is ready to reply to
CommServer:  "+str3+"\n");

                    System.out.println("----------------COMMTV FINALIZED
SUCCESSFULLY----------------\n");
                    try {Thread.currentThread().sleep(3000);} catch
(InterruptedException ie) {}
             }//closes while true
      }//closes body
}//closes class
```

CommA XML πράκτορας (για Jadex)

Ο CommA στέλνει μήνυμα στον CommServer και μετά ο CommA λαμβάνει απάντηση από τον CommServer

```
<agent xmlns="http://jadex.sourceforge.net/jadex"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                           http://jadex.sourceforge.net/jadex-0.96.xsd"
       name="CommA"
       package="jadex.examples.mycommunicationwgui">

       <imports>
              <import>jadex.planlib.*</import>
              <import>jadex.adapter.fipa.*</import>
       </imports>



       <beliefs>
              <belief name="msg" class="String" exported="true">
                     <fact>"Agent CommA is up and running."</fact>
              </belief>
              <!--
              <belief name="abc500" class="AgentIdentifier"
exported="true">
                     <fact>new AgentIdentifier("CommTv", true)</fact>
              </belief>
              <belief name="abc600" class="AgentIdentifier"
exported="true">
                     <fact>new AgentIdentifier("CommServer", true)</fact>
              </belief>
              -->
       </beliefs>



       <plans>
              <plan name="myplanA">
                     <body class="CommPlanA"/>
                     <!-- <body>new HelloWorldPlan()</body> -->
                     <trigger>
                            <messageevent ref="request_msg"/>
                            <messageevent ref="request_msg2"/>
                     </trigger>
              </plan>
       </plans>



       <events>
              <messageevent name="request_msg" direction="send"
type="fipa">
         <parameter name="performative" class="String" direction="fixed">
             <value>jadex.adapter.fipa.SFipa.REQUEST</value>
         </parameter>
         <parameter name="content" class="String">
                                          <value>"This is a req
message."</value>
                           </parameter>
```

```
    </messageevent>
    <messageevent name="request_msg2" direction="receive" type="fipa">
                    <parameter name="performative" class="String"
direction="fixed">

       <value>jadex.adapter.fipa.SFipa.REQUEST</value>
                    </parameter>
              </messageevent>
       </events>



       <configurations>
              <configuration name="default">
                    <plans>
                           <!-- Start myplan plan when agent is born. -->
                           <initialplan ref="myplanA"/>
                    </plans>
              </configuration>
       </configurations>

</agent>
```

CommB XML πράκτορας (για Jadex)

Ο CommB στέλνει μήνυμα στον CommServer και μετά ο CommB λαμβάνει απάντηση από τον CommServer

```
<agent xmlns="http://jadex.sourceforge.net/jadex"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                           http://jadex.sourceforge.net/jadex-0.96.xsd"
       name="CommB"
       package="jadex.examples.mycommunicationwgui">

       <imports>
              <import>jadex.planlib.*</import>
              <import>jadex.adapter.fipa.*</import>
       </imports>



       <beliefs>
              <belief name="msg" class="String" exported="true">
                     <fact>"Agent CommB is up and running."</fact>
              </belief>
              <!--
              <belief name="abc500" class="AgentIdentifier"
exported="true">
                     <fact>new AgentIdentifier("CommTv", true)</fact>
              </belief>
              <belief name="abc600" class="AgentIdentifier"
exported="true">
                     <fact>new AgentIdentifier("CommServer", true)</fact>
              </belief>
              -->
       </beliefs>



       <plans>
              <plan name="myplanB">
                     <body class="CommPlanB"/>
                     <!-- <body>new HelloWorldPlan()</body> -->
                     <trigger>
                            <messageevent ref="request_msg"/>
                            <messageevent ref="request_msg2"/>
                            <messageevent ref="request_msg20"/>
                     </trigger>
              </plan>
       </plans>



       <events>
              <messageevent name="request_msg" direction="send"
type="fipa">
         <parameter name="performative" class="String" direction="fixed">
             <value>jadex.adapter.fipa.SFipa.REQUEST</value>
         </parameter>
         <parameter name="content" class="String">
                                          <value>"This is a req
message."</value>
                           </parameter>
```

```
    </messageevent>

    <messageevent name="request_msg2" direction="receive" type="fipa">
                    <parameter name="performative" class="String"
direction="fixed">

       <value>jadex.adapter.fipa.SFipa.REQUEST</value>
                    </parameter>
              </messageevent>

    <messageevent name="request_msg20" direction="receive" type="fipa">
                    <parameter name="performative" class="String"
direction="fixed">
                           <value>jadex.adapter.fipa.SFipa.INFORM</value>
                    </parameter>
              </messageevent>

       </events>



       <configurations>
              <configuration name="default">
                    <plans>
                           <!-- Start myplan plan when agent is born. -->
                           <initialplan ref="myplanB"/>
                    </plans>
              </configuration>
       </configurations>

</agent>
```

CommServer XML πράκτορας (για Jadex)

Ο CommServer λαμβάνει μήνυμα από τον CommA, ενημερώνει το CommTv, λαμβάνει απάντηση από το CommTv και τέλος επιβεβαιώνει στον CommA. Μετά στέλνει ένα 'GO' μήνυμα προς τον CommB, ο οποίος βρίσκεται σε κατάσταση αναμονής, λαμβάνει απάντηση και τερματίζει

```
<agent xmlns="http://jadex.sourceforge.net/jadex"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                           http://jadex.sourceforge.net/jadex-0.96.xsd"
       name="CommServer"
       package="jadex.examples.mycommunicationwgui">

       <imports>
              <import>jadex.planlib.*</import>
              <import>jadex.adapter.fipa.*</import>
              <import>jadex.examples.garbagecollector.*</import>
       </imports>


       <capabilities>
              <capability name="amscap" file="jadex.planlib.AMS" />
       </capabilities>


       <beliefs>
              <belief name="msg" class="String" exported="true">
                     <fact>"CommServer is up and running."</fact>
              </belief>
              <!-- Environment object as singleton.-->
              <!--
              <belief name="env" class="Environment">
                     <fact>Environment.getInstance($agent.getType(),
$agent.getName())</fact>
              </belief>
              <belief name="abc100" class="AgentIdentifier"
exported="true">
                     <fact>new AgentIdentifier("CommTv", true)</fact>
              </belief>
              <belief name="abc200" class="AgentIdentifier"
exported="true">
                     <fact>new AgentIdentifier("CommA", true)</fact>
              </belief>
              -->
       </beliefs>


       <goals>
              <achievegoalref name="ams_create_agent">
                     <concrete ref="amscap.ams_create_agent" />
              </achievegoalref>
              <achievegoalref name="ams_start_agent">
                     <concrete ref="amscap.ams_start_agent" />
              </achievegoalref>
       </goals>

       <plans>
              <plan name="myplanServer">
                     <body class="CommPlanServer"/>
```

```
                        <!-- <body>new HelloWorldPlan()</body> -->
                        <trigger>
                               <messageevent ref="request_msg"/>
                               <messageevent ref="request_msg11"/>
                               <messageevent ref="request_msg12"/>
                               <messageevent ref="request_msg2"/>
                               <messageevent ref="request_msg20"/>
                        </trigger>
                 </plan>
          </plans>
          <events>
                 <!-- Specifies a translation request being all
                        messages with performative request. -->
                 <messageevent name="request_msg" direction="receive"
type="fipa">
                        <parameter name="performative" class="String"
direction="fixed">

          <value>jadex.adapter.fipa.SFipa.REQUEST</value>
                        </parameter>
                 </messageevent>

       <messageevent name="request_msg11" direction="send" type="fipa">
           <parameter name="performative" class="String" direction="fixed">
               <value>jadex.adapter.fipa.SFipa.REQUEST</value>
           </parameter>
           <parameter name="content" class="String">
                                          <value>"This is req
message11."</value>
                           </parameter>
       </messageevent>

       <messageevent name="request_msg12" direction="receive" type="fipa">
                           <parameter name="performative" class="String"
direction="fixed">

          <value>jadex.adapter.fipa.SFipa.INFORM</value>
                           </parameter>
                 </messageevent>
                 <messageevent name="request_msg2" direction="send"
type="fipa">
           <parameter name="performative" class="String" direction="fixed">
               <value>jadex.adapter.fipa.SFipa.REQUEST</value>
           </parameter>
           <parameter name="content" class="String">
                                          <value>"This is another req
message."</value>
          </parameter>
       </messageevent>
       <messageevent name="request_msg20" direction="send" type="fipa">
           <parameter name="performative" class="String" direction="fixed">
               <value>jadex.adapter.fipa.SFipa.INFORM</value>
           </parameter>
       </messageevent>
          </events>
          <configurations>
                 <configuration name="default">
                        <plans>
                               <!-- Start myplan plan when agent is born. -->
                               <initialplan ref="myplanServer"/>
                        </plans>
                 </configuration>
          </configurations>

</agent>
```

CommTv XML πράκτορας (για Jadex)

Ο CommTv λαμβάνει μήνυμα από τον CommServer και στέλνει απάντηση στον CommServer

```
<agent xmlns="http://jadex.sourceforge.net/jadex"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                           http://jadex.sourceforge.net/jadex-0.96.xsd"
       name="CommTv"
       package="jadex.examples.mycommunicationwgui">

       <imports>
              <import>jadex.planlib.*</import>
              <import>jadex.adapter.fipa.*</import>
              <import>jadex.examples.garbagecollector.*</import>
       </imports>



       <beliefs>
              <belief name="msg" class="String" exported="true">
                     <fact>"CommTv is up and running."</fact>
              </belief>
              <!--
              <belief name="env2" class="Environment">
                     <fact>Environment.getInstance($agent.getType(),
$agent.getName())</fact>
              </belief>
              <belief name="abc300" class="AgentIdentifier"
exported="true">
                     <fact>new AgentIdentifier("CommServer", true)</fact>
              </belief>
              <belief name="abc400" class="AgentIdentifier"
exported="true">
                     <fact>new AgentIdentifier("CommA", true)</fact>
              </belief>
              -->
       </beliefs>



       <plans>
              <plan name="myplanTv">
                     <body class="CommPlanTv"/>
                     <!-- <body>new HelloWorldPlan()</body> -->
                     <trigger>
                            <messageevent ref="request_msg11"/>
                            <messageevent ref="request_msg12"/>
                     </trigger>
              </plan>
       </plans>



       <events>
              <!-- Specifies a translation request being all
                     messages with performative request. -->
              <messageevent name="request_msg11" direction="receive"
type="fipa">
```

```
                    <parameter name="performative" class="String"
direction="fixed">

       <value>jadex.adapter.fipa.SFipa.REQUEST</value>
                    </parameter>
              </messageevent>

    <messageevent name="request_msg12" direction="send" type="fipa">
        <parameter name="performative" class="String" direction="fixed">
            <value>jadex.adapter.fipa.SFipa.INFORM</value>
        </parameter>
        <parameter name="content" class="String">
                                          <value>"This is req
message12."</value>
                         </parameter>
    </messageevent>

       </events>



       <configurations>
              <configuration name="default">
                    <plans>
                           <!-- Start myplan plan when agent is born. -->
                           <initialplan ref="myplanTv"/>
                    </plans>
              </configuration>
       </configurations>

</agent>
```

MyOntology.java

Βασίζεται στο εργαλείο δημιουργίας οντολογιών **Protege**

```
package jadex.examples.mycommunicationwgui;

/**
 *  Generated Java class for ontology MyOntology.
 */
public class MyOntology
{
	//-------- constants --------

	/** The name of the ontology. */
	public static final String ONTOLOGY_NAME= "MyOntology";

	/** The allowed java classes. */
	public static java.util.HashSet java_classes = new
java.util.HashSet();

	String msgA1 = "";		//e.g.100
	String msgS1 = "";		//e.g.Welcome A. TV is ready.
	String msgS2 = "";		//e.g.A is in.
	String msgT1 = "";		//e.g.Got A...
	String selection="0";
	double myd2,myd3;

	public void takeme(double jkl) { myd2=jkl; }

	public double thatis() { myd3=myd2; return myd3; }

	MyOntology(){}

	//-------- static part --------

	static
	{
	}
}//closes ontology class
```

AuxClass.java

Βοηθητική κλάση για αποθήκευση ορισμένων μεταβλητών και προσπέλαση αυτών από όλες τις εμπλεκόμενες κλάσεις

```
package jadex.examples.mycommunicationwgui;

abstract class AuxClass {

	static double ert2;
	static String ert3;
	static String ert1;

}//closes class
```