Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Αναγνώριση Σύνθετων Γεγονότων στη Ναυσιπλοΐα: Μία εφαρμογή λογισμικού.** **A software tool for detecting complex events in the maritime domain.**
Ονοματεπώνυμο Φοιτητή	**Μιχαήλ Βοσκάκης**
Πατρώνυμο	**Γεώργιος**
Αριθμός Μητρώου	**ΜΠΠΛ/13006**
Επιβλέπων	**Ιωάννης Θεοδωρίδης, Καθηγητής**

Ημερομηνία Παράδοσης **Οκτώβριος 2018**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ιωάννης Θεοδωρίδης Καθηγητής	Νικόλαος Πελέκης Επίκουρος Καθηγητής	Αλέξανδρος Αρτίκης Επίκουρος Καθηγητής

Περίληψη

Ο σκοπός της εργασίας είναι να παρουσιάσει μία εφαρμογή λογισμικού για εξομοίωση κίνησης πλοίων εφοδιασμένων με το σύστημα AIS, σε υδάτινες οδούς μεταξύ νήσων: Κατόπιν της διεξοδικής επαλήθευσης εισόδων χρήστη σχετικά με το περιβάλλον πλεύσης, τις τεχνικές προδιαγραφές των σκαφών και τις διαδρομές που ακολουθούν, αναλύονται δύο υλοποιήσεις εξομοίωσης, που διαφέρουν στο βαθμό αυτονομίας παραγωγής αποτελεσμάτων τους, ως εκπομπές θέσεως πλοίων, από τις δεδομένες εισόδους.

Επεκτείνοντας, η εφαρμογή σε πραγματικό χρόνο οπτικοποιεί τις θέσεις επί χάρτη ενώ, σημαντικότερα, αξιοποιεί τις παραγόμενες εκπομπές AIS ως γεγονότα χαμηλού επιπέδου (LLEs) μίας διαδικασίας αναγνώρισης γεγονότων υψηλού επιπέδου, με χρήση του δημοφιλούς συστήματος επεξεργασίας σύνθετων γεγονότων, Esper CEP.

Ερωτήματα του συστήματος αυτού, ενσωματωμένα στην εφαρμογή ως κανόνες αναγνώρισης, θέτουν το σύστημα σε «επαγρύπνηση» επί του ρεύματος των εκπομπών για την εύρεση συγκεκριμένων καταστάσεων ενδιαφέροντος όπως: απότομες αλλαγές κατεύθυνσης, εγγύτητα σε στεριά και άλλες συμπεριφορές, οι οποίες κρίνονται ασυνήθιστες, παραβατικές, επισφαλείς ή κακοήθεις.

Abstract

The essay at hand aims at introducing a software application simulating AIS - equipped marine traffic on routes among landmasses: Following extensive user input validation regarding the marine physical environment, ship technical details and performance, as well as their itineraries within said space, two simulation implementations are presented, differing on the degree of their respective input - output independence; the output being real-time AIS position broadcasts from the simulated fleet.

Furthermore, the broadcast positions are visualized in real-time on a map, while more significantly, as they comprise a stream of low-level events (LLEs), those are also channeled towards an instance of the popular complex event processing system, Esper CEP, thus triggering a high-level event recognition procedure.

Queries of that framework, embedded in the application as detection rules, render the system “vigilant”, observing the AIS broadcast stream for any matches on specific situations of interest, such as: abrupt direction changes, shore proximity and other behaviors deemed peculiar, illegal, unsafe or malicious.

Εισαγωγή

Το εύρος των αναγκών που καλύπτουν οι ναυτικές μετακινήσεις στις μέρες μας, και ως συνέπεια το μέγεθος του παγκόσμιου στόλου, καθιστούν επιτακτική την χρήση τεχνικών συνοπτικής, αφαιρετικής, και άλλοτε στοχευμένης παρουσίασης του υπέρογκου πρωτογενούς πληροφοριακού φορτίου των μετακινήσεων αυτών, με στόχο την ασφάλεια και αποτελεσματικότητα των πλωτών μεταφορών. Οι παραπάνω ανάγκες έφεραν στον κόσμο της ναυσιπλοΐας εφαρμογές του γνωστικού αντικειμένου της Αναγνώρισης Σύνθετων Γεγονότων, συνήθως με εφαλτήριο τα μηνύματα του συστήματος AIS που γνώρισε σχεδόν καθολική εγκατάσταση στον στόλο, κυρίως λόγω θεσμικής επιβολής.

Η εφαρμογή λογισμικού sailAway, που αναλύεται στις σελίδες αυτής της εργασίας, περιλαμβάνοντας μηχανισμούς παραγωγής μηνυμάτων AIS, παρέχει τη δυνατότητα στους χρήστες της να δημιουργήσουν τις συνθήκες ενός εικονικού ναυσιπλοϊκού σεναρίου. Ύστερα από την παραγωγή εξόδου από το σύστημα, τους δίδεται η δυνατότητα να διαπιστώσουν εάν ακολουθίες από τις εκπομπές του σεναρίου που εξομοιώθηκε απαρτίζουν συλλογικά κάποια από τις καταστάσεις ενδιαφέροντος, ή αλλιώς γεγονότα υψηλού επιπέδου, για τα οποία η εφαρμογή διατηρεί σαφείς περιγραφές ως κανόνες. Έτσι, αυτή η διαδικασία ανάδρασης έχει - μεταξύ άλλων - εκπαιδευτικό χαρακτήρα, όντας χρήσιμη σε διαφορετικές κατηγορίες χρηστών, ανάλογα με την ιδιότητα και τον απώτερο σκοπό τους. Ενδεικτικά η εφαρμογή μπορεί να χρησιμοποιηθεί από:

- Φοιτητές που μελετούν την Αναγνώριση Σύνθετων Γεγονότων και ιδιαίτερα εκείνους που ασχολούνται με τις εφαρμογές της στην θαλάσσια πλοήγηση.
- Μηχανικούς Λογισμικού που χρειάζονται μια πλατφόρμα εξοικείωσης με τους κανόνες περιγραφής γεγονότων στη γλώσσα που χρησιμοποιείται από την Esper.
- Επιστήμονες που αναζητούν μια γεννήτρια ναυτιλιακών δεδομένων για ερευνητική χρήση.
- Αξιωματικούς του Λιμενικού Σώματος, οι οποίοι πειραματιζόμενοι με διάφορα σενάρια παραβατικών ενεργειών και τους αντίστοιχους κανόνες αναγνώρισης αυτών, μπορούν να αποκτήσουν βαθύτερη κατανόηση για ενδεχόμενα «παραθυράκια» / αστοχίες αναγνώρισης, που ενδεχομένως παραβάτες έχουν ήδη ανακαλύψει και εκμεταλλεύονται. Η γνώση αυτή μπορεί να χρησιμοποιηθεί για εκσυγχρονισμό και προσαρμογή των συστημάτων που ήδη βρίσκονται σε υπηρεσία με σκοπό την βελτίωση της απόδοσής τους.

Από την περιγραφή των δύο μεθόδων παραγωγής εκπομπών σε επόμενα κεφάλαια, είναι σαφές πως μπορούν ακόμη να εισαχθούν πραγματικά ιστορικά δεδομένα κατόπιν λίγων απαραίτητων προσαρμογών, είτε ως απλοποιημένες διαδρομές, (π.χ. μόνο waypoints) ή ως πλήρες σειρές εκπομπών.

Για την επίτευξη των στόχων της, η εφαρμογή λαμβάνει από το χρήστη εκτενή πληροφόρηση σε μορφή διανυσμάτων αποθηκευμένων σε αρχεία. Τα δεδομένα αυτά περιλαμβάνουν τον χάρτη της εικονικής θάλασσας, όπου οι περιοχές που καλύπτονται από στεριά χαρακτηρίζονται στο επίπεδο, τη διαδρομή κάθε πλοίου στο χώρο αυτό, καθώς και κάποιες βασικές λεπτομέρειες και τεχνικές προδιαγραφές για κάθε σκάφος. Πρόσθετη πληροφόρηση περιλαμβάνει παραμέτρους λειτουργίας και γενικές ρυθμίσεις. Ανάλογα με το βαθμό αφαίρεσης (abstraction) κάθε εισηγμένης διαδρομής, παράγονται εκπομπές θέσεως είτε ως αναπαραγωγή των δεδομένων εισόδου, ή ως γνήσια εξομοίωση βάσει απλών μοντέλων φυσικής. Σε ζωντανό χρόνο οι θέσεις αυτές παρίστανται στο χάρτη, ενώ επί αυτού του ρεύματος αναγνωρίζονται καταστάσεις ενδιαφέροντος βάσει προδιαγεγραμμένων ερωτημάτων Esper EPL, ενσωματωμένων στην εφαρμογή.

Το υπόλοιπο της εργασίας είναι δομημένο ως εξής: Στο πρώτο κεφάλαιο, δίδονται οι απαραίτητες περιγραφές σε βασικές έννοιες που χρησιμοποιούνται μετέπειτα και επανειλημμένα. Στο δεύτερο κεφάλαιο, ερευνάται η επιστημονική βιβλιογραφία επί θεμάτων συναφών με την αναγνώριση σύνθετων ναυτιλιακών γεγονότων. Στο τρίτο κεφάλαιο περιγράφεται επισκοπικά η λειτουργία της εφαρμογής, τα εργαλεία που συντέλεσαν στην συγγραφή της, τα δομικά συστατικά της σε διάφορα επίπεδα, καθώς και το σύστημα κλήσης και περαιτέρω ανάπτυξής της. Το τέταρτο κεφάλαιο εμβαθύνει σε πληθώρα θεμάτων που αφορούν τη διαδικασία παραγωγής γεγονότων χαμηλού επιπέδου. Η ανάλυση του κεφαλαίου αυτού είναι επιλεκτική και περιλαμβάνει αποκλειστικά θέματα τα οποία δεν είναι προγραμματιστικώς τετριμμένα. Ενότητες του κεφαλαίου περιγράφουν διαδικασίες επαλήθευσης και ενσωμάτωσης εισόδου χρήστη, παραγωγής εκπομπών AIS από την εξομοίωση θαλάσσιας κίνησης, οπτικοποίησης και άλλες. Συμπληρωματικά, το πέμπτο κεφάλαιο αφορά αποκλειστικά την επεξήγηση κανόνων που ορίζουν τις σύνθετες καταστάσεις ενδιαφέροντος προς αναγνώριση στο ρεύμα των εκπομπών. Οι καταστάσεις υψηλού επιπέδου αναλύονται εξατομικευμένα και παρατίθενται για κάθε μία από αυτές περιορισμοί και λοιπές παρατηρήσεις. Το κεφάλαιο αυτό περιλαμβάνει επίσης τεχνικές λεπτομέρειες που είναι απαραίτητες για την ενσωμάτωση του συστήματος επεξεργασίας σύνθετων γεγονότων που αξιοποιείται, ενώ ακόμη παρουσιάζεται η διαδικασία παραγωγής γραπτών αναφορών για τα γεγονότα που εντοπίστηκαν, ανά είδος. Η εργασία ολοκληρώνεται στο έκτο κεφάλαιο όπου εκφράζονται ιδέες για μελλοντικές βελτιώσεις της εφαρμογής στην παρούσα μορφή της, αλλά και σκέψεις για την επέκτασή της.

Περιεχόμενα εργασίας

Σημείωση για τη λήψη της εφαρμογής / πρόσθετου υλικού:

Η δομή αρχείων της εφαρμογής, όπως περιγράφεται στο Κεφάλαιο 3, αλλά και τα συμπληρωματικά αρχεία της εργασίας, όπως παραρτήματα και ενδεικτικά datasets, περιέχονται στον συνοδευτικό οπτικό δίσκο.
Τα εν λόγω αρχεία είναι επίσης διαθέσιμα στο github.com/voskakism/sailAway ενώ οι ενδιαφερόμενοι μπορούν εναλλακτικά να επικοινωνήσουν στην διεύθυνση voskakism@gmail.com για την αποστολή ενός download link.

Κεφάλαιο 1: Επεξήγηση βασικών εννοιών

Περιεχόμενα κεφαλαίου

1.1: Συστήματα συντεταγμένων, αζιμούθιο και χαρακτηριστικές γωνίες πλοήγησης

Το αζιμούθιο είναι μία μέθοδος προσδιορισμού κατεύθυνσης και προσανατολισμού η οποία τυπικά χρησιμοποιείται στο σφαιρικό σύστημα συντεταγμένων. Στο σύστημα τριών διαστάσεων αυτό, ορίζεται ένα προκαθορισμένο επίπεδο αναφοράς στο οποίο ανήκει και η αρχή του συστήματος (σημείο παρατηρητή, μηδενικό διάνυσμα) και μία ημιευθεία αναφοράς επί του παραπάνω επιπέδου, με αρχή της την αρχή του συστήματος. Το σύστημα αυτό αποτελεί χώρο διανυσμάτων της μορφής v = (r, altitude, azimuth), όπου :

- v: το διάνυσμα με αρχή την αρχή του συστήματος και πέρας το εκάστοτε σημείο ενδιαφέροντος.
- r: η ευκλείδεια απόσταση μεταξύ της αρχής του συστήματος (σημείο παρατηρητή) και του σημείου ενδιαφέροντος, με άλλα λόγια το μέτρο του v.
- altitude: η γωνία μεταξύ του v και της ορθής προβολής αυτού (έστω v_{proj}) στο επίπεδο αναφοράς. Η γωνία αυτή έχει πρόσημο, αναλόγως του «ημισφαιρίου» που βρίσκεται το σημείο ενδιαφέροντος. Ως «ημισφαίρια» ορίζονται τα δύο διαμερίσματα του διανυσματικού χώρου όπως αυτός διαιρείται από το επίπεδο αναφοράς.
- azimuth (αζιμούθιο): η γωνία μεταξύ της ημιευθείας αναφοράς και του v_{proj}. Αυτή η γωνία δεν έχει πρόσημο, καταδεικνύοντας κάθε κατεύθυνση στο επίπεδο, με τιμές από 0 έως πλήρη κύκλο. Για τη μέτρησή της χρησιμοποιείται μία ημιευθεία (γνωστή και ως επιβατική ακτίνα) η οποία πάντα βρίσκεται επί του επιπέδου αναφοράς και για μηδενικές γωνίες συμπίπτει με την ημιευθεία αναφοράς. Με άλλα λόγια, η εκκίνηση της επιβατικής ακτίνας είναι η ημιευθεία αναφοράς. Η «ακτίνα» μετρά γωνίες περιστρεφόμενη με σημείο περιστροφής (το σταθερό της σημείο) την αρχή του συστήματος, και μόνο προς μία προκαθορισμένη φορά (είτε δεξιόστροφα ή αριστερόστροφα) έως ότου συμπέσει με το v_{proj}. Η γωνία που «διανύθηκε», είναι το αζιμούθιο ενδιαφέροντος.

Το σύστημα χρησιμοποιείται στην αστρονομία, ενώ πριν την έλευση τεχνολογιών όπως GNSS, χρησιμοποιήθηκε και στην θαλάσσια πλοήγηση. Σε εκείνη την εφαρμογή, το σημείο παρατηρητή είναι το εκάστοτε σκάφος, ενώ επίπεδο αναφοράς είναι το νοητό επίπεδο που εφάπτεται στην επιφάνεια της γης στο σημείο παρατηρητή. Η δε ημιευθεία αναφοράς εκτείνεται προς το βορρά και η επιβατική ακτίνα κινείται δεξιόστροφα, στη φορά των δεικτών του ρολογιού.

Μία άλλη σημαντική χρήση αυτού του συστήματος είναι ο προσδιορισμός επίγειων γεωγραφικών θέσεων. Σε εκείνη την εφαρμογή, το επίπεδο αναφοράς είναι αυτό που διέρχεται από τον ισημερινό κύκλο, χωρίζοντας τον πλανήτη στο βόρειο και νότιο ημισφαίριο. Αρχή του συστήματος είναι το κέντρο του πλανήτη, εκεί δηλαδή όπου ο άξονας περιστροφής της γης τέμνει το επίπεδο αναφοράς, ενώ ημιευθεία αναφοράς ορίζεται αυτή που τέμνει τον πρώτο μεσημβρινό (ορίζεται από την θέση του Greenwich, στο ανατολικό Λονδίνο). Οποιαδήποτε θέση στον φλοιό (επιφάνεια) της γης μπορεί να προσδιοριστεί με τη χρήση του σφαιρικού συστήματος όπως προσδιορίστηκε για τις ανάγκες της ειδικής εφαρμογής του. Έτσι, κάθε σημείο προσδιορίζεται πλήρως με τις γωνίες altitude και azimuth του διανύσματος που αντιπροσωπεύει. Στην συγκεκριμένη χρήση, το altitude είναι γνωστό ως γεωγραφικό πλάτος, ενώ το azimuth ως γεωγραφικό μήκος. Μία ιδιαιτερότητα προσδιορισμών θέσης

επί γης με χρήση του σφαιρικού συστήματος αποτελεί η προσημασμένη μέτρηση γεωγραφικού μήκους: Η επιβατική ακτίνα κινείται σε εύρος έως 180 μοιρών δεξιόστροφα ή αριστερόστροφα από τον πρώτο μεσημβρινό, διαιρώντας τη γη στο ανατολικό και δυτικό της ημισφαίριο.

Σε επέκταση τις περιγραφής του συστήματος, και χάριν πληρότητας, αξίζει μία σύντομη αναφορά σε στο κυλινδρικό σύστημα των τριών διαστάσεων, όπου η διάσταση του ύψους μετράται όχι ως η γωνία altitude του διανύσματος, αλλά «καρτεσιανά», ως το μέτρο της προβολής του v στον άξονα του ύψους. Φυσικά υπάρχει και το τρισδιάστατο καρτεσιανό, με ορθογώνιες προβολές σε όλους τους άξονες. Ακόμη, το σύστημα πολικών συντεταγμένων των δύο διαστάσεων είναι ένας χώρος διανυσμάτων της μορφής v = (r, azimuth). Τα v, r και azimuth ορίζονται όπως ακριβώς και στο στην περίπτωση του σφαιρικού συστήματος, η διάσταση του altitude δεν υφίσταται, ενώ επίσης ισχύει πως $v = v_{proj}$.

Η πλοήγηση σε μεγάλες αποστάσεις επί χερσαίων και θαλάσσιων οδών δημιούργησε ήδη από πολύ παλιά την ανάγκη χαρτογράφησης της γης. Καθώς η γη είναι κατά προσέγγιση σφαιρική, η αποτύπωση του φλοιού της στην πρακτική και εύχρηστη μορφή των δύο διαστάσεων είναι μία ενέργεια που μπορεί μόνο κατά προσέγγιση να πραγματοποιηθεί. Η πιο δημοφιλής προσέγγιση είναι η μερκατορική προβολή, μία απόπειρα αποτύπωσης του αναπτύγματος της σφαίρας ως ορθογώνιο παραλληλόγραμμο, υπονοώντας πως η περίμετρος όλων των παράλληλων κύκλων του πλανήτη είναι ίση με αυτή του ισημερινού. Ως παρενέργεια αυτής της αναληθούς υπόθεσης, οι γεωγραφικές αποστάσεις κατά μήκος κάποιου παράλληλου κύκλου παρίστανται μεγαλύτερες από τις πραγματικές. Αυτή η παραμόρφωση εντείνεται σε μεγάλα κατά απόλυτη τιμή γεωγραφικά πλάτη, δηλαδή κοντά στους πόλους. Στην προσπάθεια διατήρησης της μορφολογίας των γεωγραφικών σχηματισμών, οι αποστάσεις κατά μήκος των μεσημβρινών επίσης μεταβάλλονται κατά το ίδιο ποσοστό επιμήκυνσης με αυτό του εκάστοτε τοπικού παραλλήλου, καθιστώντας το ύψος του ορθογωνίου μεγαλύτερο από το μήκος του τμήματος του μεσημβρινού που απεικονίζει, φυσικά λαμβάνοντας υπόψη και το λόγο κλιμάκωσης της χαρτογράφησης. Ένα συναφές μειονέκτημα της προβολής αυτής είναι η αδυναμία χαρτογράφησης των πολικών περιοχών, καθώς εκεί η παραμόρφωση αγγίζει το άπειρο. Τυπικά, οι μερκατορικοί χάρτες δεν απεικονίζουν ακραία γεωγραφικά πλάτη. Φυσικά η προβολή αυτή έχει και πλεονεκτήματα που δικαιολογούν τη δημοτικότητά της: Οποιαδήποτε ευθεία γραμμή επί ενός μερκατορικού χάρτη έχει σταθερό course, και όλοι οι μεσημβρινοί εμφανίζονται παράλληλοι, ως αποτέλεσμα της εγγενούς παραμόρφωσης. Η μερκατορική προβολή, αν και προβάλλει τη γη σε δύο διαστάσεις, χρησιμοποιεί το σύστημα των σφαιρικών συντεταγμένων, διατηρώντας την αντιστοιχία των απεικονιζόμενων θέσεών της με αυτές της σφαιρικής προβολής του πλανήτη.

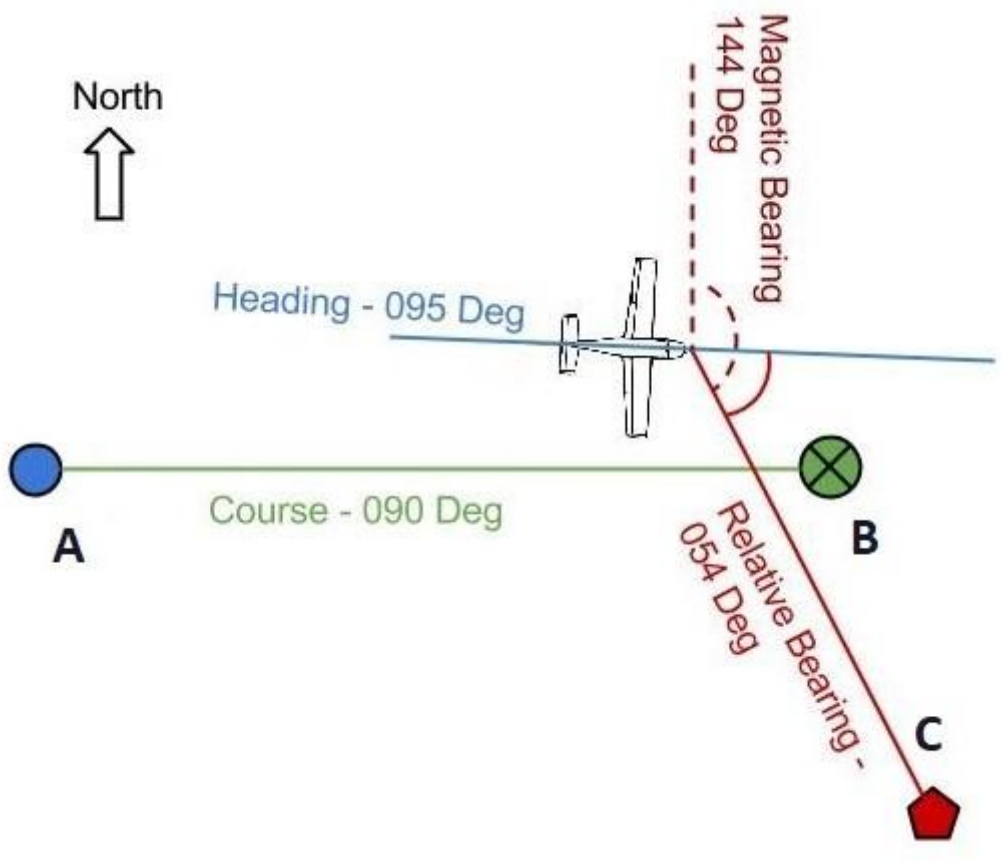
Για τις ανάγκες της εφαρμογής που παρουσιάζεται στα επόμενα κεφάλαια, έχει χρησιμοποιηθεί το καρτεσιανό (ορθοκανονικό) σύστημα συντεταγμένων των δύο διαστάσεων. Διατεταγμένα ζεύγη παριστούν θέσεις, με συντεταγμένες (1) γεωγραφικό μήκος (Longitude) και (2) γεωγραφικό πλάτος (Latitude), αντίστοιχα.

Οι έννοιες γωνιών και κατευθύνσεων πλοήγησης που αναλύονται στη συνέχεια, μεταφέρονται και στο γεωγραφικό σύστημα συντεταγμένων αυτής της εφαρμογής λογισμικού. Οι γωνίες της εφαρμογής που εκφράζονται ως αζιμούθια:

- Μετρώνται σε μοίρες.
- Δεν είναι προσημασμένες, έχοντας εύρος ολόκληρο τον κύκλο δηλαδή κάθε κατεύθυνση στο επίπεδο (0 έως 360 μοίρες).
- μετρώνται με την επιβατική τους ακτίνα να κινείται σύμφωνα με τη φορά των δεικτών του ρολογιού (clockwise).

Στην περίπτωση που ο χρήστης μεταφέρει ως είσοδο στην εφαρμογή γεωγραφικούς σχηματισμούς που προέρχονται από μερκατορικούς χάρτες, η όποια διόρθωση (αν κρίνεται αναγκαία) λόγω της παραμόρφωσης αποστάσεων και εμβαδών, αποτελεί ευθύνη του ίδιου και γίνεται εξωγενώς πριν την εισαγωγή (διορθωμένων ή μη) συντεταγμένων στην εφαρμογή.

Κατόπιν παρουσίασης της έννοιας του αζιμουθίου ως τρόπο προσδιορισμού κατεύθυνσης, παρατίθενται ορισμένες γωνίες / κατευθύνσεις ιδιαίτερου ενδιαφέροντος στην θαλάσσια ή εναέρια πλοήγηση:



Σχήμα 1: Παράδειγμα επεξήγησης heading, course και bearings.

Heading: Δείχνει τον προσανατολισμό του σκάφους σε κάθε δεδομένη στιγμή. Πρόκειται για τη γωνία το ένα σκέλος της οποίας εκτείνεται στην κατεύθυνση του βορρά, και το άλλο, με εκκίνηση την ίδια κατεύθυνση, κινείται αποκλειστικά δεξιόστροφα (clockwise) έως ότου καταστεί εκτεινόμενο στον ημιάξονα που έχει κατεύθυνση από την πρύμνη προς της πλώρη, παράλληλο με τον διαμήκη άξονα του σκάφους. Η γωνία που «διανύθηκε» κατά αυτόν τον τρόπο είναι το heading. Για παράδειγμα, ένα πλοίο στραμμένο ακριβώς στη δύση, έχει heading 270 μοίρες. Το heading αγνοεί την επιθυμητή διαδρομή και το διάνυσμα της ταχύτητας του σκάφους. Το σκάφος του ίδιου παραδείγματος θα μπορούσε να κινείται νοτιοδυτικά, υπό την επίδραση πολύ ισχυρών ανέμων / κυμάτων. Το heading αφορά αποκλειστικά «το που κοιτάει», όχι «το που πηγαίνει» κάποιο σκάφος.

Absolute Bearing: Χρησιμοποιείται για την κατάδειξη ενός σημείου ενδιαφέροντος (διαφορετικού του σκάφους, π.χ. φάρος). Πρόκειται για τη γωνία με κορυφή το «κέντρο» του σκάφους, ως τομή του διαμήκους και εγκάρσιου άξονά του, με το ένα σκέλος αυτής να εκτείνεται στην κατεύθυνση του βορρά, και το άλλο με εκκίνηση την ίδια κατεύθυνση κινείται αποκλειστικά δεξιόστροφα (clockwise) έως ότου τέμνει το εκάστοτε σημείο ενδιαφέροντος. Η γωνία που «διανύθηκε» κατά αυτόν τον τρόπο είναι το absolute Bearing. Εναλλακτικά, όταν το σκάφος είναι στραμμένο ακριβώς στο σημείο ενδιαφέροντος, το absolute bearing προς το σημείο αυτό ταυτίζεται με το heading του σκάφους.

Relative Bearing: Ορίζεται όπως ακριβώς και το absolute bearing, με τη μοναδική διαφορά πως το πρώτο (σταθερό) σκέλος της γωνίας μέτρησης δεν είναι στραμμένο προς το βορρά, αλλά βρίσκεται επί του διαμήκους άξονος του σκάφους, στην κατεύθυνση από την πρύμνη προς την πλώρη του. Προφανώς σχετίζεται με το heading και δείχνει το «σχετικό προσανατολισμό» μίας θέσης με την κατεύθυνση του σκάφους, όντας σε αρκετές περιπτώσεις πιο εύκολο στην κατανόηση και πιο χρήσιμο σε υπολογισμούς στροφών / ελιγμούς αποφυγής συγκρούσεων κ.α. Ισχύει πως το relative bearing μίας θέσης ισούται με το absolute bearing της ίδιας θέσης μείον το heading του σκάφους. Αρνητικά αποτελέσματα (δηλαδή για θέσεις που βρίσκονται στο ημιεπίπεδο της αριστερής, “portside”, πλευράς του σκάφους), αυτά αφαιρούνται από το 360.

Course (από γεωγραφική θέση Α σε Β): Ορίζεται όπως το absolute bearing με σημείο ενδιαφέροντος το Β, αλλά με τη μοναδική διαφορά πως η κορυφή της γωνίας δεν είναι το σκάφος, αλλά το σημείο Α. Δείχνει το heading που θα πρέπει να διατηρήσει ένα σκάφος ξεκινώντας από το Α ώστε να φτάσει στον προορισμό, Β. Σημεία τέτοια είναι συνήθως τα άκρα ευθυγράμμων τμημάτων που συνθέτουν την τεθλασμένη γραμμή μίας διαδρομής στο χάρτη.

Ύστερα από τις αναφορές στο βορρά, αξίζει να σημειωθεί πως οι αληθινές μαγνητικές πυξίδες δεν υποδεικνύουν την κατεύθυνση όπου οι μεσημβρινοί τέμνονται στο βόρειο ημισφαίριο, δηλαδή τον αληθινό (ή γεωγραφικό) δηλαδή βορρά, αλλά ένα άλλο σημείο επί της επιφάνειας της γης εντός του βορείου πολικού κύκλου που είναι γνωστό ως μαγνητικός βοράς. Ο μαγνητικός βορράς διαχρονικά μετακινείται εντός του πολικού κύκλου επηρεασμένος από την μεταβαλλόμενη κατανομή της μάζας της γης στα βαθύτερα ρευστά σιδηρούχα στρώματά της. Η απόκλιση αυτή ως γωνία, γνωστή και ως μαγνητική απόκλιση ή σφάλμα πυξίδας αυξάνεται σε μεγάλα γεωγραφικά πλάτη.

1.2: Η έννοια του περιγεγραμμένου ορθογωνίου στα πλαίσια της sailAway

Το όνομα του όρου δημιουργεί την υποψία πως η έννοια χρησιμοποιείται σε συνδυασμό με κάποια άλλη. Πράγματι, χρησιμεύει στην περιγραφή ενός συνόλου στοιχείων και αποτελεί ιδιότητα αυτού. Το σύνολο των στοιχείων μπορεί να αποτελείται από οποιοδήποτε πλήθος νησιών καθώς και από από οποιοδήποτε πλήθος πλοίων. Τα νησιά στο χάρτη είναι πολύγωνα των δύο διαστάσεων, με δεδομένο εμβαδό. Τα πλοία στην εφαρμογή έχουν γενικώς σημειακή υπόσταση και δεν καταλαμβάνουν έκταση. Όταν ο όρος χρησιμοποιείται στα πλαίσια της εφαρμογής sailAway, το περιγεγραμμένο ορθογώνιο ενός δεδομένου συνόλου πλοίων / νησιών είναι το παραλληλόγραμμο ορθογώνιο επί του χάρτη της εφαρμογής με επαρκές εμβαδό και θέση ώστε να υπερκαλύπτει εντελώς όλα τα στοιχεία του σχετικού συνόλου. Επιπλέον, όλες οι πλευρές του οφείλουν να εφάπτονται (και όχι να τέμνουν) με ένα ή περισσότερα στοιχεία του συνόλου. Ασφαλώς, μπορεί κάποιο στοιχείο να κατέχει πολλαπλά σημεία επαφής με το ορθογώνιο, ακόμη και εάν αυτά ανήκουν σε διαφορετικές πλευρές του ιδίου, και / ή εφάπτονται με διαφορετικές πλευρές του ορθογωνίου. Διαισθητικά, η έννοια μπορεί να γίνει αντιληπτή, ικανοποιώντας τους δύο παραπάνω περιορισμούς, ως ορθογώνια θαλάσσια περίφραξη / σύνορο που περικλείει όλα τα νησιά και πλοία στο ακέραιό τους, έχοντας επίσης την ελάχιστη δυνατή περίμετρο, ακουμπώντας τα στοιχεία ως αποτέλεσμα. Για την ολοκλήρωση του ορισμού της έννοιας, πάντα στα πλαίσια της εφαρμογής sailAway, απαιτείται ακόμη ένας περιορισμός που θα πρέπει το ορθογώνιο της ανάλυσης να καλύπτει: παραλληλία δύο (αντικριστών) πλευρών του με τον ισημερινό και τους παραλλήλους και κατά συνέπεια παραλληλία των υπολοίπων δύο πλευρών του με τους μεσημβρινούς του χάρτη. Στο σημείο αυτό υπενθυμίζεται πως ο νοητός χάρτης εδάφους της εφαρμογής είναι καρτεσιανός ορθοκανονικός και ως εκ τούτου δισδιάστατος με όλους τους μεσημβρινούς του παράλληλους και ισαπέχοντες.

Σε επέκταση των παραπάνω, αξίζει να σημειωθεί πως εάν ένα δεδομένο σύνολο αποτελείται από ακριβώς ένα πλοίο και επίσης κανένα νησί, η έννοια του περιγεγραμμένου ορθογωνίου δεν έχει νόημα στο σύνολο αυτό. Το προκύπτον σχήμα δεν θα είχε εμβαδό. Ακόμη, στον αρχεία κώδικα της εφαρμογής η έννοια αναφέρεται ως “containing rectangle”.

Ένας ακόμη τρόπος για να γίνει κατανοητή η έννοια, σε δεδομένο χάρτη δύο διαστάσεων, είναι η ανάπτυξη ενός παραδείγματος με χρήση της έννοιας των «ακραίων σημείων της πυξίδας», με την οποία σχετίζεται στενά. Η τελευταία είναι ορθότερα γνωστή και με την περιγραφή των «σημείων ορίζοντα» και αντανακλά τα ακραία σημεία μιας γεωγραφικής οντότητας στο χάρτη προς τις 4 βασικές κατευθύνσεις[1]. Παραδείγματος χάρη, εάν αναλογιστούμε πως η Ελλάδα έχει άκρα[2] της ως:

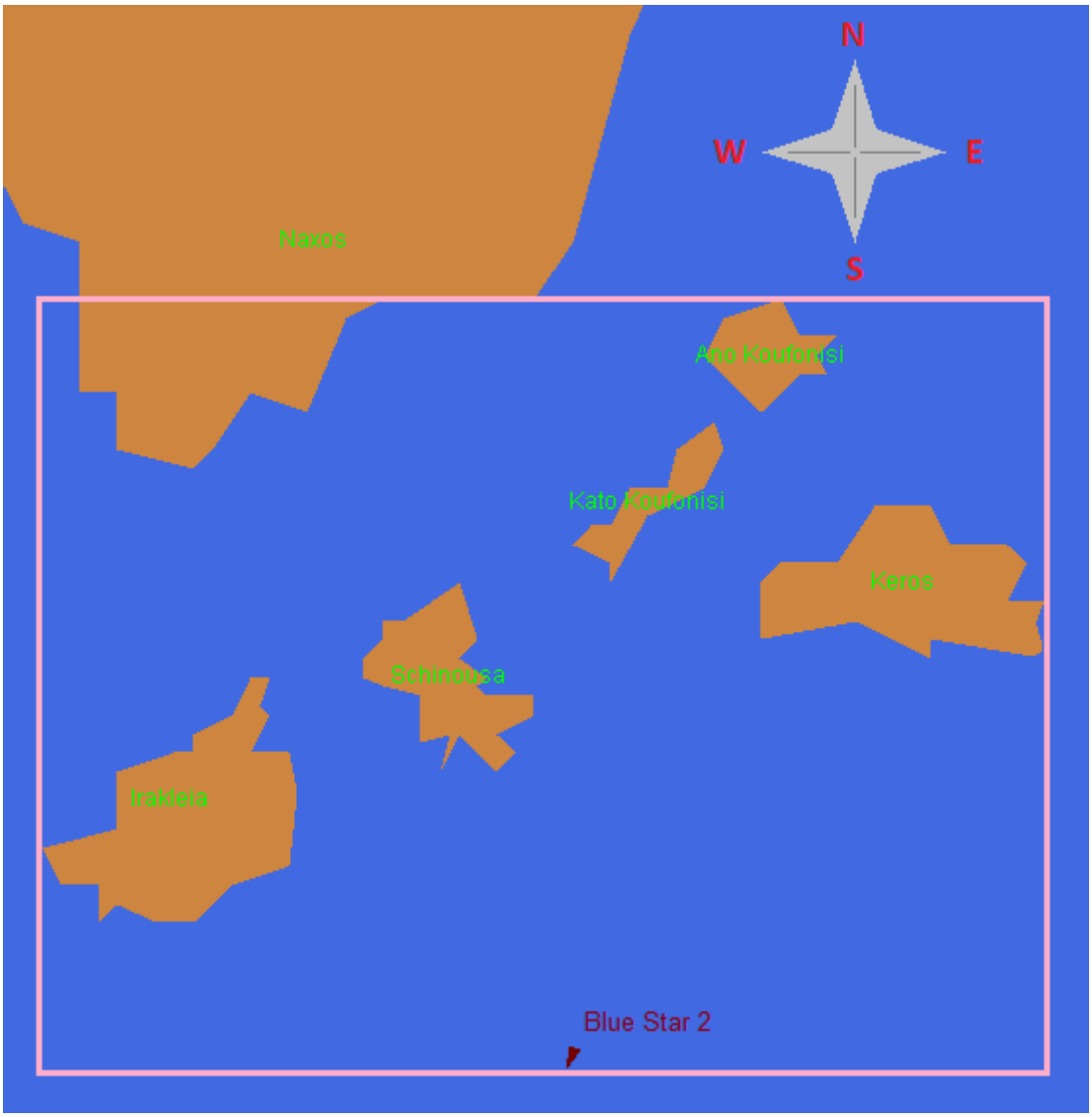
[1] Cardinal directions: Βοράς, Νότος, Δύση, Ανατολή.

[2] Τα ακραία σημεία στο παράδειγμα της Ελλάδος δεν είναι ιδιαίτερα ακριβή, αλλά αναφέρθηκαν ως έγινε καθώς είναι ευρύτερα αναγνωρίσιμα κατά αυτόν τον τρόπο. Έτσι π.χ. υπάρχει άλλο λίγο ακόμη έδαφος της Ελληνικής επικράτειας βορειότερα του Ορμενίου μέχρι τα σύνορα, το πραγματικό νοτιότερο άκρο της Επικράτειας είναι η νότια πλευρά της θαλάσσιας λωρίδας νοτίως της Γαύδου (θαλάσσιο σύνορο) κ.ο.κ.

- βορειότερο το χωριό Ορμένιο (σημείο της με το μεγαλύτερο γεωγραφικό πλάτος),
- νοτιότερο τη νήσο Γαύδο (σημείο της με το μικρότερο γεωγραφικό πλάτος),
- ανατολικότερο τη νήσο Στρογγυλή (σημείο της με το μεγαλύτερο γεωγραφικό μήκος),
- δυτικότερο τη νήσο Οθωνοί (σημείο της με το μικρότερο γεωγραφικό μήκος),

τότε εάν χαραχτούν οι δύο εφαπτόμενοι στο ανατολικότερο και δυτικότερο αντίστοιχα σημείο μεσημβρινοί, καθώς και οι δύο εφαπτόμενοι στο βοριότερο και νοτιότερο σημείο ισημερινοί, τότε το περιγεγραμμένο ορθογώνιο της Ελλάδος ορίζεται από τις 4 αυτές ευθείες.

Ένα ακόμη παράδειγμα στην Εικόνα 1: Εκεί, έχει χαρακτεί το περιγεγραμμένο ορθογώνιο του συνόλου με στοιχεία τη δεδομένη θέση του πλοίου Blue Star 2 και τις νήσους: Κέρος, Ηρακλειά, Άνω Κουφονήσι, Κάτω Κουφονήσι και Σχοινούσα (αλλά όχι Νάξο που μερικώς εμφανίζεται στην εικόνα).



Εικόνα 1

1.3: Το σύστημα μηνυμάτων AIS

Το AIS (Automatic Identification System) είναι ένα σύστημα συλλογής και μετάδοσης δεδομένων θαλάσσιας κυκλοφορίας σε πραγματικό χρόνο. Ο οργανισμός πίσω από την ανάπτυξη και θεσμοθέτησή του είναι ο Διεθνής Οργανισμός Ναυσιπλοΐας (IMO), όργανο του Οργανισμού Ηνωμένων Εθνών. Το σύστημα βασίζεται σε ασύρματες εκπομπές πακέτων δεδομένων από πλοία ανά τακτά χρονικά διαστήματα και περιλαμβάνει τόσο στατικές ή γενικώς αμετάβλητες όσο και άκρως δυναμικές, στιγμιαίας ισχύος, πληροφορίες. Οι σημαντικότερες από αυτές είναι:

- Όνομα πλοίου και MMSI (Maritime Mobile Service Identity, μοναδικός αριθμός 9 ψηφίων)
- Είδος πλοίου
- Διαστάσεις και βύθισμα
- Προορισμός και Αναμενόμενος Χρόνος Άφιξης
- Κατάσταση πλεύσης
- Ρυθμός και Φορά περιστροφής
- Τρέχουσα θέση: γεωγραφικό μήκος / πλάτος
- Προσανατολισμός: heading και course
- Ταχύτητα

Από πλευράς παραγωγής και εκπομπής μηνυμάτων, τα σκάφη διαθέτουν σχετικό εξοπλισμό, ο οποίος τροφοδοτείται με μετρήσεις από αισθητήρες πλοήγησης του σκάφους, όπως η γυροσκοπική πυξίδα και το σύστημα GPS. Ο μηχανισμός AIS τοποθετεί τις πληροφορίες αυτές στα κατάλληλα πεδία του υπό σύνθεση μηνύματος και η VHF κεραία του συστήματος το εκπέμπει ως ηλεκτρομαγνητικό κύμα σε προκαθορισμένη συχνότητα. Η λήψη του μηνύματος γίνεται από πλοία εντός εμβέλειας, από τους δικούς τους πομποδέκτες AIS που ακολουθούν την αντίστροφη διαδικασία: Αναγνωρίζουν το είδος του μηνύματος, το ψηφιοποιούν και αντλούν τις πληροφορίες των πεδίων του, παρέχοντας έτσι πολύτιμη πληροφόρηση για την κίνηση κοντινών πλοίων. Το πρόβλημα της πολυφωνίας και επικάλυψης εκπομπών λύνεται με χρήση ενός είδους χρονικής πολυπλεξίας. Πέρα από τα σκάφη που λαμβάνουν και αποστέλλουν μηνύματα, υπάρχει και δίκτυο παράκτιων ή επί νήσων σταθμών λήψης, που σχεδόν αποκλειστικά λαμβάνουν μηνύματα, εκτός αν πρόκειται για αποστολή κάποιου βοηθητικού ή ειδικού μηνύματος. Παραδείγματα μηνυμάτων από σταθμούς λήψης περιλαμβάνουν αιτήσεις πληροφοριών σε στοχευμένο σκάφος, μηνύματα συγχρονισμού κ.α. Εκπομπές σκαφών εκτός εμβέλειας των σταθμών αυτών, ενίοτε συλλέγονται και από κατάλληλα εξοπλισμένους δορυφόρους. Εν τέλει, όλα τα διαθέσιμα μηνύματα καταλήγουν στη γη όπου ενθυλακώνονται σε πακέτα όπως TCP και μέσω του διαδικτύου μεταφέρονται προς επεξεργασία ανά τον κόσμο. Το πρώτο βήμα αξιοποίησης αυτού του μεγάλου όγκου πληροφοριών διαρκούς εισροής, είναι η οπτικοποίηση για λήψη εικόνας (κυριολεκτικά και μεταφορικά) της θαλάσσιας κίνησης σε μία περιοχή. Για εξαγωγή πληροφοριών υψηλότερου επιπέδου με τρόπο αυτοματοποιημένο και εστιασμένο σε θέματα ενδιαφέροντος, το ρεύμα αυτών των μηνυμάτων αποτελεί το εφαλτήριο διαδικασιών αναγνώρισης σύνθετων γεγονότων, όπως άλλωστε καθίσταται προφανές από την πρόσφατη επιστημονική βιβλιογραφία.

Το σύστημα διευκολύνει και προάγει την ασφάλεια μετακινήσεων επί υδάτινων σωμάτων και χρησιμεύει σε πληθώρα εφαρμογών εμπορικού ενδιαφέροντος, όπως παρακολούθηση κίνησης εμπορευμάτων. Επίσης χρησιμεύει σε επιχειρήσεις έρευνας και διάσωσης, και παρέχει ένα μέσο εποπτείας για τις αρμόδιες Αρχές στην προσπάθειά τους να σταματήσουν διάφορες μορφές ναυτικών παραβάσεων. Ωστόσο, η χρήση τεχνολογιών και εξοπλισμού ανοικτών ή ιδιαίτερα διαδεδομένων προτύπων στα πλαίσια του AIS, αλλά και η έλλειψη μηχανισμών εξουσιοδότησης / ταυτοποίησης για τη χρήση του, συχνά επιτρέπουν και την ανεμπόδιστη παραποίηση του μεταδιδόμενου περιεχομένου πληροφοριών.

Στα πλαίσια της εφαρμογής λογισμικού αυτής της εργασίας, οι μεμονωμένες εκπομπές / πακέτα AIS εξομοιώνονται ως αντικείμενα της τάξης AISBroadcast.

1.4: Αναγνώριση Σύνθετων Γεγονότων, Γεγονότα Χαμηλού / Υψηλού Επιπέδου

Οι τρείς έννοιες είναι συνυφασμένες και οι περιγραφές κάθε μίας από αυτές τοποθετούνται σε σχέση με τις υπόλοιπες:

- Γεγονός χαμηλού επιπέδου, ή Low Level Event (LLE), είναι ένα άτομο (αδιάσπαστο) γεγονός μικρής αξίας και πληροφοριακού όγκου, με καθορισμένη οργάνωση του περιεχομένου του. Τυπικά, τα LLEs σε κάποια εφαρμογή ή σενάριο επεξεργασίας γεγονότων παρέχονται σε μεγάλο αριθμό αν πρόκειται για στατικά δεδομένα, ή καταφθάνουν με μεγάλο ρυθμό, ως ρεύμα.
- Γεγονός υψηλού επιπέδου, ή High Level Event (HLE), είναι το γεγονός το οποίο αποτελεί καθορισμένη δομή που ενσωματώνει πολλαπλά επιμέρους LLEs, ενός ή περισσότερων ειδών, ή ακόμη άλλα HLEs, και μπορεί να εκφραστεί σε όρους αυτών των υπογεγονότων κατά τρόπο που ενδέχεται να περιλαμβάνει περιορισμούς (χρονικούς ή άλλους) μεταξύ αυτών. Έτσι, ένα γεγονός υψηλού επιπέδου λέγεται και σύνθετο. Ένα HLE είναι κάτι περισσότερο από το σύνολο των LLEs που το δομούν: Αντιπροσωπεύει μία κατάσταση ιδιαίτερου ενδιαφέροντος η σημασία της οποίας ξεπερνά τη σημασία των δομικών του συστατικών, συνολικά.

Τα χαρακτηριστικά των παραπάνω εννοιών, όπως μέγεθος και αξία, ορίζονται συναρτήσει της διαδικασίας αναγνώρισης στην οποία συμμετέχουν, είναι σχετικά με αυτή, και όχι απόλυτα / αυτοτελή.

Η Αναγνώριση Σύνθετων Γεγονότων, εντασσόμενη στο γενικότερο πλαίσιο της Επεξεργασίας Γεγονότων, αφορά το σύνολο των ενεργειών εύρεσης HLEs από συλλογές ή ρεύματα σχετικών LLEs. Πέρα από τα γεγονότα εισόδου, το σύστημα που αναλαμβάνει την αναγνώριση αυτή, συνήθως λαμβάνει εξωγενώς και την αναλυτική περιγραφή των γεγονότων υψηλού επιπέδου ως αναγωγή ή ανάλυσή τους σε απλούστερες δομικές μονάδες γεγονότων χαμηλού επιπέδου, μαζί με όποιους περιορισμούς μπορεί να επιβάλλει η ανάλυση σε αυτές. Στη βιβλιογραφία οι παραπάνω έννοιες ορίζονται και συσχετίζονται με παρόμοιο τρόπο. Για παράδειγμα, σύμφωνα[3] με τους Artikis A., Sergot M. και Paliouras G. (2012), " *... τα συστήματα αναγνώρισης ταυτοποιούν σύνθετα γεγονότα ενδιαφέροντος, συλλογές γεγονότων που ικανοποιούν ένα μοτίβο. Ο 'ορισμός' ενός σύνθετου γεγονότος επιβάλλει περιορισμούς, χρονικούς και μη, στα υπογεγονότα του, δηλαδή απλά ή άλλα σύνθετα γεγονότα.*"

Η έρευνα στο πεδίο της Αναγνώρισης Σύνθετων Γεγονότων, ανταποκρινόμενη στις προκλήσεις του εύρους εφαρμογών της, έχει επεκταθεί σε διάφορες κατευθύνσεις που θεραπεύουν παθογένειες συναφείς με τη φύση των δεδομένων: όγκος, μορφολογική ασυνέπεια, πολλαπλές πηγές. Ακόμη, ζητήματα ενδιαφέροντος στο πεδίο αποτελούν η κλιμάκωση, η κατανεμημένη επεξεργασία, η αναγνώριση υπό συνθήκες αβεβαιότητας, η μάθηση σχέσεων μεταξύ γεγονότων με επαγωγικές τεχνικές, και άλλα.

Η κατηγοριοποίηση ενός είδους γεγονότος ως υψηλού ή χαμηλού επιπέδου είναι αδύνατη χωρίς περιγραφή του γενικότερου εννοιολογικού πλαισίου (context) της σχετικής με αυτό διαδικασίας. Γεγονότα που αναγνωρίζονται ως υψηλού επιπέδου από μία συγκεκριμένη διαδικασία αναγνώρισης, ενδέχεται να τελούν ως γεγονότα χαμηλού επιπέδου για μία περαιτέρω, διαφορετική διαδικασία. Το πρόβλημα αυτό της κατηγοριοποίησης θυμίζει την προσπάθεια κατάταξης ενός στοιχείου ως δεδομένο ή πληροφορία για ένα μηχανισμό επεξεργασίας: Δεδομένα εισρέουν σε ένα αλγόριθμο παράγοντας, από την οπτική του συγκεκριμένου αλγορίθμου, πληροφορίες. Αυτές οι εκροές ενδέχεται να αποτελούν εισροές (δεδομένα) κάποιου ανεξάρτητου συστήματος. Αυτός είναι ο λόγος για τον οποίο οι τρείς έννοιες επεξηγήθηκαν συλλογικά.

[3] Σε μετάφραση του πρωτότυπου.

Κεφάλαιο 2: Βιβλιογραφική ανασκόπηση

Περιεχόμενα κεφαλαίου

Στο παρόν κεφάλαιο εξετάζεται η υπάρχουσα επιστημονική βιβλιογραφία στο γνωστικό αντικείμενο της Αναγνώρισης Σύνθετων Γεγονότων στη Ναυσιπλοΐα. Επιλέχθηκε μία κατά το δυνατόν αντιπροσωπευτική ομάδα άρθρων τα οποία προσεγγίζουν πληθώρα επιμέρους ζητημάτων του αντικειμένου, μεθόδους και τεχνολογίες, αλλά και εφαρμογές του στις υδάτινες μετακινήσεις. Γενικά, η αναγνώριση σύνθετων συμβάντων μπορεί να γίνει από ανομοιογενείς πηγές δεδομένων, με χρήση ποικίλων θεωρητικών προσεγγίσεων, και προς εξυπηρέτηση ενός πολύ μεγάλου εύρους σκοπών.

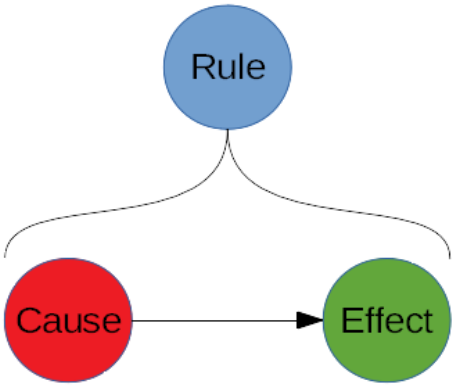
Της εξερεύνησης του χώρου της ναυσιπλοΐας, θα προηγηθεί η παρουσίαση θεμάτων προσανατολισμένων στην αναγνώριση ή επεξεργασία σύνθετων γεγονότων καθ' αυτή, ώστε εδραιωθεί σε πρώτο στάδιο μία βασική και υψηλού επιπέδου αντίληψη του οικοσυστήματος των τεχνολογιών που καθιστούν δυνατές τις εφαρμογές τις βιβλιογραφίας. Παράλληλα θα παρουσιαστούν μερικά σημαντικά άρθρα, σχετικά με αυτά τα θέματα.
Μετατοπίζοντας στη συνέχεια την εστίαση της ανάλυσης στον εντοπισμό ναυσιπλοϊκών γεγονότων, οι αντίστοιχες εργασίες παρουσιάζουν ενδιαφέρον τόσο από πλευράς χρησιμότητας των εφαρμογών σε τομείς όπως εξοικονόμηση πόρων, ασφάλεια παγίων υλικών, περιβάλλοντος και προσώπων όσο και από πλευράς καταπολέμησης εγκλήματος / νομικής συμμόρφωσης.
Σε επόμενο βήμα παρουσιάζονται λίγες επιστημονικές εργασίες που είναι συναφείς με την αναγνώριση σύνθετων γεγονότων ως εφαρμογές σε πεδία διαφορετικά από την ναυσιπλοΐα. Ο λόγος για τον οποίο η ανάλυση μεταβαίνει σε εκείνη την ενότητα, είναι η ανάδειξη του εύρους της αξιοποίησης που έχει η τεχνολογία επεξεργασίας γεγονότων στην καθημερινότητα, τις υποδομές, την οικονομία, την υγεία, και κατ' επέκταση αυτών, της αξίας της στην ποιότητα διαβίωσης.
Στο τέλος του κεφαλαίου, θα γίνει μια προσπάθεια να βρεθεί η σχετική θέση της παρούσας εφαρμογής στο υπάρχον σύνολο των εφαρμογών, η συνεισφορά της σε αυτό, οι ομοιότητες και οι διαφορές της με ένα επιλεγμένο μέρος των άρθρων που αναφέρονται.

2.1: Η τεχνολογία αναγνώρισης και επεξεργασίας γεγονότων

Καθώς οι εφαρμογές αναγνώρισης γεγονότων απαιτούν τη χρήση κάποιου είδους τεχνητής νοημοσύνης από το σύστημα, κρίνεται χρήσιμη η παρουσίαση των τριών μεθόδων λογικής συμπερασματολογίας (logical reasoning). Αυτές θα αποτελέσουν κριτήριο διάκρισης των εργασιών που θα παρουσιαστούν στο κεφάλαιο, επιτρέποντας παράλληλα την κατανόηση των θεμελιωδών τουλάχιστον αρχών λειτουργίας αυτών.
Συστατικά ενός λογικού προβλήματος, και κατ' επέκταση της διαδικασίας εξαγωγής ενός λογικού συμπεράσματος, αποτελούν τα γεγονότα ή αλήθειες (facts) και οι κανόνες που περιγράφουν την αιτιολογική σχέση μεταξύ τους. Έτσι, υπάρχουν οι αιτίες, τα αποτελέσματά τους και οι κανόνες που περιγράφουν την σχέση αυτή. Το σύνολο των αποτελεσμάτων δεν μπορεί να προηγείται χρονικά από την αιτία / τις αιτίες του.

Ένα λογικό πρόβλημα ανάγεται σε μία ή περισσότερες καταστάσεις όπου ένα από αυτά τα τρία χαρακτηριστικά μπορεί να μην είναι γνωστό. Ανάλογα με το ποιό είναι το απουσιάζον συστατικό, ακολουθείται και διαφορετική προσέγγιση επίλυσης για την εύρεσή του και την πλήρη εξήγηση του φαινομένου. Για να γίνει κατανοητή η ανάλυση, εισάγεται ένα παράδειγμα λογικού προβλήματος, με ένα κανόνα:

«Όταν ο ουρανός είναι συννεφιασμένος,
πάντα ακολουθεί βροχή».



Σχήμα 2: Λογική Συμπερασματολογία.

Οι τρεις μέθοδοι λογικής συμπερασματολογίας είναι οι:

- Deduction (πόρισμα): Είναι διαθέσιμος ο κανόνας και η γνώση πως έχει πραγματοποιηθεί η αιτία. Συμπεραίνεται πως θα συμβεί το αποτέλεσμα. «Γνωρίζω πως όταν έχει συννεφιά, βρέχει. Παρατηρώ πως έχει συννεφιά τώρα, άρα θα βρέξει». Με την υπόθεση πως ο κανόνας ισχύει απαρέγκλιτα, η πορισματική συμπερασματολογία είναι ντετερμινιστική όπως μία συνάρτηση, εν προκειμένω μπουλιανή.
- Abduction (απαγωγή): Είναι διαθέσιμος ο κανόνας και η γνώση πως έχει πραγματοποιηθεί το αποτέλεσμα. Συμπεραίνουμε πως έχει προηγουμένως συμβεί η αιτία. «Γνωρίζω πως όταν έχει συννεφιά, βρέχει. Παρατηρώ πως τώρα βρέχει, άρα μάλλον είχε συννεφιά νωρίτερα». Αξίζει να σημειώσουμε πως το αποτέλεσμά ενδέχεται να έχει πολλές ακόμη και ανεξάρτητες αιτίες, εκτός αν ο κανόνας δηλώνει πως το αποτέλεσμα προκαλείται εάν και μόνο εάν («ανν» / “iff”) συμβεί η συγκεκριμένη αιτία. Για το λόγο αυτό, η απαγωγική συμπερασματολογία δεν διέπεται απαραίτητα από ντετερμινισμό.
- Induction (επαγωγή): Είναι διαθέσιμα γεγονότα που αντιπροσωπεύουν αιτίες και αποτελέσματα, αλλά δεν υπάρχει γνώση περί αιτιολογικής σχέσης εκ των προτέρων. Όσο το φαινόμενο επαναλαμβάνεται, γίνεται ολοένα και πιο προφανής ο κανόνας, το ζητούμενο δηλαδή σε αυτήν την περίπτωση. «Πολλές φορές παρατήρησα τελευταία, πως αφού ο ουρανός γέμιζε σύννεφα, ακολουθούσε βροχή. Άρα μάλλον όποτε γενικώς έχουμε συννεφιά, ακολουθεί βροχή». Όπως συμβαίνει και με την απαγωγή, η επαγωγή αποτελεί μία επισφαλή προσέγγιση συμπερασματολογίας: Τα γεγονότα που απαρτίζουν τα διαθέσιμα στοιχεία του προβλήματος μπορεί να είναι απλώς συμπτώσεις.

Η επαγωγή είναι μια μετάβαση από τις ειδικές περιπτώσεις σε γενικές αλήθειες, ενώ το αντίθετο συμβαίνει με τις άλλες δύο μεθόδους λογικής συμπερασματολογίας, όπου ο κανόνας, η γενική αλήθεια δηλαδή, είναι γνωστός. Φυσικά στην περίπτωση όπου περισσότερα από ένα από τα συστατικά της συμπερασματολογίας απουσιάζουν, το πρόβλημα είναι αόριστο.

Σε μία προσπάθεια ταξινόμησης των διαφορετικών προσεγγίσεων της βιβλιογραφίας, με σκοπό τη σύσταση μίας συνολικής άποψης των συστημάτων αναγνώρισης σύνθετων γεγονότων, μπορούν να χρησιμοποιηθούν διακρίσεις βάσει διαφόρων κριτηρίων, όπως:

1. Η φύση της παραγόμενης εξόδου του εκάστοτε συστήματος, βάσει λογικής συμπερασματολογίας.
   Παραδείγματος χάρη, το σύστημα που παρουσιάζεται στην παρούσα εργασία είναι ξεκάθαρα και αποκλειστικά deductive (πορισματικό) καθώς ως είσοδο δέχεται LLEs (αιτίες), διαθέτει τους κανόνες αναγνώρισης ορισμένων HLEs, χωρίς δυνατότητα αναπροσαρμογής των κανόνων αυτών, και τελικώς παράγει αναφορές για τα αναγνωρισμένα σύνθετα γεγονότα (αποτελέσματα). Η διάκριση βάσει του κριτηρίου αυτού θα πρέπει να γίνεται ύστερα από συνολική εξέταση του συστήματος και όχι κρίνοντας από επιμέρους λειτουργίες του: Σε ορισμένες περιπτώσεις, πορισματικές υλοποιήσεις χρησιμοποιούν κάποιου είδους νοημοσύνη που εντάσσεται σε άλλου είδους συμπερασματολογία, αλλά μόνο για την εξυπηρέτηση συγκεκριμένων λειτουργιών αναπροσαρμογής.

2. Η χρήση, ή μη, τεχνικών Μηχανικής Μάθησης για αναμόρφωση ή ρύθμιση.
   Μηχανισμοί επεξεργασίας γεγονότων που βασίζονται σε μοντέλα γράφων, όπως νευρωνικά δίκτυα, DFA / NFA κ.α., χρησιμοποιούν σταθμίσεις στις συναρτήσεις μετάβασης μεταξύ των κόμβων τους, επιτρέποντας την σταδιακή αναγνώριση ενός γεγονότος καθώς αυτό ικανοποιεί συνθήκες εκφρασμένες από το state αυτών των δομών. Εάν αυτές οι τιμές δεν είναι κατάλληλα ορισμένες στην αρχική στάθμιση, υπάρχουν αλγόριθμοι όπως ο back-propagation και άλλοι στατιστικής, ως επί το πλείστον, προέλευσης και αναδρομικής, τυπικά, μορφής που μπορούν να τις διορθώσουν σταδιακά κατά το runtime, ανταποκρινόμενοι στα εισρέοντα δεδομένα / γεγονότα.
   Ακόμη, υπάρχουν και (υπό)συστήματα τα οποία δεν σκοπεύουν στην αναγνώριση HLEs, αλλά στην εύρεση κανόνων μεταξύ γεγονότων, με σκοπό τη χρήση τους εξωσυστημικώς, ή από άλλες μονάδες της υλοποίησης. Μία τέτοια προσέγγιση είναι ο επαγωγικός λογικός προγραμματισμός.
3. Οι σχέσεις που διέπουν τα επιμέρους γεγονότα χαμηλού επιπέδου, στα πλαίσια κανόνων αναγνώρισης.
   Ενδεικτικά, σε ορισμούς / κανόνες καταστάσεων ενδιαφέροντος (HLEs) που εκφράζονται ως σχέσεις μεταξύ LLEs αποκλειστικά χρονικές, τα αντίστοιχα συστήματα είναι γνωστά ως CRSs (Chronicle Recognition Systems), όπως είναι εκείνο που περιγράφουν στο "Chronicle recognition improvement using temporal focusing and hierarchization" οι Dousson C. και Le Maigat P. (2012). Άλλες προσεγγίσεις, όπως η παρούσα, βασίζονται βαρέως σε χώρο-χρονικές (spatiotemporal) σχέσεις LLEs, αφού άλλωστε αφορά κίνηση πλοίων.
   Λοιποί περιορισμοί μπορεί να είναι λογικοί, ποσοτικοί κ.α.

Επεκτείνοντας την απόπειρα κατηγοριοποίησης σε ότι αφορά τις προσεγγίσεις Μηχανικής Μάθησης συγκεκριμένα, ο Domingos παρουσίασε[4] τις βασικές «σχολές σκέψης» (paradigms / δόγματα) τις οποίες ονόμασε φυλές ("tribes") και εντόπισε τον επιστημονικό κλάδο προέλευσης αλλά και τη μεθοδολογία μάθησης για κάθε προσέγγιση:

Tribe	**Origins**	**Master Algorithm**
Symbolists	Logic, Philosophy	Inverse deduction[5]
Connectionists	Neuroscience	Back-propagation[6]
Evolutionaries	Evolutionary biology	Genetic programming
Bayesians	Statistics	Probabilistic inference
Analogizers	Psychology	Kernel machines

Η εμβάθυνση στις αρχές των παραπάνω τεχνικών είναι ίσως εκτός της θεματολογίας της παρούσας εργασίας, αν και στη βιβλιογραφία περί Αναγνώρισης Σύνθετων Γεγονότων, ορισμένες θεωρίες εφαρμόζονται πιο συχνά από άλλες. Για το λόγο αυτό θα αναλυθούν συνοπτικά εδώ, μερικές φορές υπό το πρίσμα δημοσιεύσεων που πραγματεύονται τις τεχνολογίες καθ' αυτές, ή τις χρησιμοποιούν στα πλαίσια κάποιας υλοποίησης.

Μία σημαντική κατασκευή αποτελούν τα Δίκτυα Bayes, Bayesian Networks, βασικό εργαλείο της «φυλής» των Bayesians. Πρόκειται για πιθανολογικό (probabilistic) στατιστικό μοντέλο που αναπαρίσταται σε άκυκλο κατευθυνόμενο γράφο[7] οριζόμενο από τυχαίες μεταβλητές ως κόμβους και συναρτήσεις δεσμευμένων πιθανοτήτων ως τόξα. Πέρας κάποιου τόξου είναι η posterior μεταβλητή, ενώ αρχή του η συνθήκη αυτής (prior). Στην πιο απλή περίπτωση, οι συναρτήσεις αυτές είναι μπουλιανές, οπότε αναπαρίστανται συνήθως με πίνακες αλήθειας, εάν το πλήθος των γονέων του κόμβου το επιτρέπει, καθώς κάθε κόμβος με n γονείς έχει 2^n πρότυπα στο πεδίο ορισμού της συνάρτησης πιθανότητάς του. Σε πιο σύνθετες περιπτώσεις, χρησιμοποιούνται συνεχείς συναρτήσεις

[4] "The Master Algorithm". Talks at Google, 20/11/2015.
[5] Εννοώντας «επαγωγή».
[6] Έννοια όχι πολύ διαφορετική από την «προς τα πίσω επαγωγή», στις «Βελτιώσεις», Κεφάλαιο 6 του παρόντος.
[7] Directed acyclic graph, DAG.

πυκνότητας πιθανότητας για κάθε κόμβο. Υπάρχουν αλγόριθμοι συμπερασματολογίας που μπορούν να τρέξουν επί BNs, «απαντώντας» ερωτήματα πιθανολογικού χαρακτήρα, ενώ επίσης υπάρχουν αλγόριθμοι μάθησης των παραμέτρων των συναρτήσεων κατανομής πιθανότητας ή ακόμη και της δομής / τοπολογίας του γράφου, δηλαδή των σχέσεων εξάρτησης μεταξύ των μεταβλητών.

Ερευνώντας το δόγμα της Λογικής, η ανάλυση δεν θα μπορούσε να ξεκινήσει παρά με μία εργασία η οποία δεν έχει ως σκοπό την προαγωγή της έρευνας στο πεδίο της αναγνώρισης σύνθετων γεγονότων, αλλά έχει ιδιαίτερη αξία για αυτή, αποτελώντας θεμελιακό εργαλείο για την υλοποίηση πολλών άλλων εργασιών της. Όπως μάλιστα παρατηρεί ο M. Shanahan (1999), το άρθρο των Kowalski R. και Sergot M., εισήγαγε για πρώτη φορά την έννοια του «Λογισμού Γεγονότων» (“Event Calculus”). Πρόκειται για ένα τυποποιημένο (“formal”) μηχανισμό λογικής πρώτου βαθμού που δύναται να συμπεράνει την αλήθεια γεγονότων ή καταστάσεων σε κάποια δεδομένη χρονική στιγμή, βασιζόμενος σε:

1. Ένα χρονικό ιστορικό γεγονότων, δηλαδή γεγονότα διατεταγμένα στον χρονικό άξονα. Ο Shanahan αναφέρει χαρακτηριστικά “narrative of events” ή πιο απλά “what happens when”. Τα γεγονότα αυτά μπορούν να έχουν χρονική διάρκεια, καθώς ορίζεται η έναρξη και η λήξη τους ως διαφορετικές χρονικές στιγμές.
2. Μία βάση γνώσης αποτελούμενη από κανόνες λογικής ισοδυναμίας μεταξύ των γεγονότων / δράσεων. Σύμφωνα με τον Shanahan: “effects of actions” ή “what actions do”.

Ο μηχανισμός, αναλόγως με τα δεδομένα που καλείται να συμπεράνει, μπορεί να κινηθεί επαγωγικά, πορισματικά ή απαγωγικά.

Ίσως με εξαίρεση την χρονική διάσταση του Λογισμού Γεγονότων, η ομοιότητά του με τις τεχνικές λογικού προγραμματισμού να είναι αρκετά προφανής. Με την «προσθήκη» του χρόνου, καθώς αυτός αποτελεί συστατικό της ρεαλιστικής περιγραφής ενός γεγονότος, δομείται ένα ικανό πλαίσιο συσχέτισης δράσεων και κατ’ επέκταση ένα πολύτιμο θεωρητικό εφόδιο στην έρευνα της αναγνώρισης σύνθετων γεγονότων. Πράγματι, η εργασία των Kowalski και Sergot, αποτέλεσε εφαλτήριο για μία ολόκληρη γενιά άλλων ερευνητικών έργων του κλάδου, έχοντας πολύ υψηλό αριθμό ετεροαναφορών από αυτές τις μεταγενέστερες δημοσιεύσεις ακόμη και δεκαετίες αργότερα. Χωρίς να προκαλεί έκπληξη, η εργασία αυτή έχει ιδιαίτερα αυξημένη αξία σε προσεγγίσεις που βασίζονται σε τεχνικές λογικού προγραμματισμού.
Αξίζει να σημειωθεί πως και άλλοι επιστήμονες διέκριναν τη σημασία της ζεύξης της λογικής με τη χρονική διάσταση καθώς η εργασία των Kowalski και Sergot, αν και πρωτοποριακή στην κατεύθυνση αυτή, δεν είναι το μοναδικό ερευνητικό άρθρο του είδους. Παράδειγμα αποτελεί η εργασία των C. Ghezzi, D. Mandrioli και A. Morzenti “TRIO: A logic language for executable specifications of real-time systems” που δημοσιεύτηκε το 1990 και αποτέλεσε και αυτή σημείο εκκίνησης άλλων ερευνητικών άρθρων συναφών με την αναγνώριση σύνθετων γεγονότων από νεότερους επιστήμονες, όπως οι Cugola και Margara.

Στο “Run-time Composite Event Recognition” των Artikis A., Sergot M. και Paliouras G. (2012) παρουσιάζεται η RTEC (Event Calculus for Run-time Reasoning), μία διάλεκτος του Λογισμού Γεγονότων και η υλοποίησή της στη γλώσσα λογικού προγραμματισμού Prolog. Η RTEC ενσωματώνει έναν αριθμό καινοτομιών που της επιτρέπουν αποδοτική αναγνώριση γεγονότων σε πραγματικό χρόνο και της δίνουν επεκτασιμότητα σε μεγάλες ροές γεγονότων.
Είναι κατάλληλη για χρήση σε εφαρμογές όπου τα δεδομένα εισόδου μπορεί να καταφθάνουν σε σειρά διαφορετική από εκείνη της πηγής τους, λόγω της μεταβλητής χρονικής υστέρησης που μπορεί να υπάρχει στη μετάδοσή τους. Με χρήση προσωρινής μνήμης αποφεύγονται περιττοί επαν-υπολογισμοί. Μηχανισμοί διαχείρισης χρονικών διαστημάτων απλοποιούν τον ορισμό των σύνθετων γεγονότων και αυξάνουν την αποδοτικότητα του συστήματος. Με χρήση δεικτών στα γεγονότα εισόδου (indexing), η RTEC μπορεί να λειτουργήσει με δεδομένα αφιλτράριστα, πρωτογενή. Τέλος, ένα «κινητό παράθυρο», εξασφαλίζει πως ο μηχανισμός μπορεί να «ξεχνά» τα παλαιότερα γεγονότα εισόδου, χάριν αποδοτικής λειτουργίας και εξοικονόμησης πόρων συστήματος, αλλά με ενδεχόμενο κίνδυνο να μην αναγνωριστούν σύνθετα γεγονότα τα οποία αποτελούνται από γεγονότα χαμηλού επιπέδου που εκτείνονται χρονικά σε εύρος μεγαλύτερο του «παραθύρου». Το σύστημα έχει αξιολογηθεί στα πλαίσια

του άρθρου εφαρμοζόμενο με σκοπό τη διαχείριση αστικής κυκλοφορίας. Χρησιμοποιήθηκαν τεχνητά και αληθινά datasets[8].

Επιστρέφοντας προς στιγμήν στα θεωρητικά πλαίσια που κατέστησαν δυνατή την πρόοδο στο αντικείμενο, οι Richardson M. και Domingos P. (2006) διέκριναν τις αδυναμίες αλλά και τις δυνατότητες των σημαντικότερων προσεγγίσεων στην Μηχανική Μάθηση και επιχείρησαν να θεμελιώσουν μία προσέγγιση, τα Markov Logic Networks, που φιλοδοξούσε να ενοποιήσει τα δημοφιλέστερα paradigms με τρόπο που θα διατηρούσε τα πλεονεκτήματά τους, ελαχιστοποιώντας τις αδυναμίες του αποτελέσματος σύντηξης αυτών. Το μεγαλεπήβολο σχέδιο ενοποίησης της Μηχανικής Μάθησης υπό ένα καθολικό δόγμα, μοιάζει να πέτυχε, κρίνοντας από την δημοτικότητα που τάχιστα απέκτησε μεταξύ ερευνητών ανά τα γνωστικά αντικείμενα, συμπεριλαμβανομένης της Αναγνώρισης Σύνθετων Γεγονότων.
Αξιοποιώντας την επεξήγηση που αναπτύχθηκε στις προηγούμενες παραγράφους επί σχετικών ζητημάτων, θα μπορούσε κάποιος να φανταστεί, καταχρηστικά ελαφρώς, τα MLNs ως επέκταση του Λογισμού Γεγονότων με στοιχεία κυρίως από δίκτυα Bayes, αλλά και διαφοροποιήσεις. Συγκεκριμένα, πρόκειται ασφαλώς για τρόπο αναπαράστασης ενός σεναρίου μάθησης με γράφο, ο οποίος όμως δεν είναι κατευθυνόμενος, ενώ οι κόμβοι του αποτελούνται όχι από λογικές μεταβλητές, αλλά όλες τις δυνατές τιμές μεταβλητών και κατηγορουμένων (ground variables and predicates), αναλόγως του εκάστοτε προβλήματος. Η σημαντικότερη «κληρονομιά» από τα BNs είναι η δυνατότητα στάθμισης μίας λογικής πρότασης, αντί της αυστηρής επαλήθευσης ή διάψευσής της, εισάγοντας έτσι στο μοντέλο την διάσταση της αβεβαιότητας, και αυξάνοντας κατακόρυφα την καταλληλότητά του για πληθώρα εφαρμογών. Σε ζητήματα αναπροσαρμογής και μάθησης επί μίας τέτοιας δομής, έχουν αναπτυχθεί αλγόριθμοι οι οποίοι έχουν επιρροές από άλλους συναφείς σχηματισμούς αποτύπωσης μερικώς εμφανούς state, όπως Hidden Markov Models.

Η TESLA (Trio-based Event Specification LAnguage) των Cugola G., Margara A. (2010) είναι μία γλώσσα περιγραφής σύνθετων γεγονότων. Κάθε κανόνας της αντιμετωπίζει τα εισερχόμενα δεδομένα ως ειδοποιήσεις γεγονότων και καθορίζει τον τρόπο με τον οποίο πρότυπα (“patterns”) μεταξύ γεγονότων οδηγούν στην πρόκληση άλλων, «σύνθετων» γεγονότων. Είναι μία υλοποίηση του λογισμού TRIO (Ghezzi et al, 1990) και ως εκ τούτου δύναται να αναπαραστήσει γεγονότα και σχέσεις αυτών όπου υπάρχουν χρονικοί περιορισμοί, έχοντας απλό συντακτικό και υψηλό βαθμό εκφραστικότητας. Προσφέρει φίλτρα, λογικές πράξεις όπως άρνηση, αθροιστικές λειτουργίες και άλλα. Στο τελευταίο μέρος του άρθρου, οι συγγραφείς περιγράφουν ένα σύστημα αναγνώρισης σύνθετων γεγονότων ως διερμηνέας (interpreter) κανόνων της TESLA. Το ημιτελές[9] σύστημα αυτό βασίζεται στη δημιουργία αυτομάτων.
Το “Complex Event Processing with T-REX” (2010) αποτελεί τη συνέχεια της έρευνας της ίδιας ομάδας επιστημόνων, παρουσιάζοντας το ολοκληρωμένο πια σύστημα αναγνώρισης σύνθετων γεγονότων ονόματι T-REX. Το σύστημα δέχεται ερωτήματα της TESLA και μετατρέποντάς τα σε αυτόματα, πραγματοποιεί αναγνώριση γεγονότων υψηλού επιπέδου ύστερα από εισροή γεγονότων χαμηλότερου επιπέδου. Στη συνέχεια οι συγγραφείς αναφέρονται σε θέματα επιδόσεων του συστήματος και προχωρούν και σε μία σύγκριση με την Esper, ένα πολύ δημοφιλές CEP σύστημα.

Η Cayuga των Brenna L., Demers A., Gehrke J. et al. (2007) είναι ένα σύστημα εποπτείας σύνθετων γεγονότων για ροές δεδομένων υψηλής ταχύτητας. Αποτελείται από μία απλή, βασισμένη στο συντακτικό της SQL γλώσσα και μία μηχανή επεξεργασίας των ερωτημάτων αυτής, που κάνει χρήση μη ντετερμινιστικών αυτομάτων πεπερασμένων καταστάσεων (NFAs) για την λειτουργία της. Στο άρθρο τους, οι ερευνητές παρουσιάζουν την υλοποίησή τους εφαρμόζοντάς τη σε ροή διαδικτυακών ροών γεγονότων (web feeds), επιδεικνύοντας την εκφραστικότητα, τις επιδόσεις και την επεκτασιμότητά της. Ιδιαίτερα σε ότι αφορά την τελευταία αυτή ιδιότητά της, η μηχανή μπορεί να χειριστεί όχι μόνο μεγάλο όγκο δεδομένων αλλά και πληθώρα ενεργών ερωτημάτων. Όπως πολλές υλοποιήσεις του χώρου, ανάμεσά τους και η Esper, χρησιμοποιεί κάποια διάλεκτο της SQL ως προγραμματιστική διεπαφή.

[8] Ελσίνκι, Νοέμβριος 2011.
[9] Κατά το χρόνο συγγραφής εκείνου του άρθρου.

Ενώ η αναγνώριση σύνθετων γεγονότων σε πραγματικού χρόνου ροές δεδομένων είναι ζήτημα πολύ υψηλού ενδιαφέροντος στην βιβλιογραφία που εξετάζουμε, το άρθρο που παρουσιάζει το μηχανισμό επεξεργασίας σύνθετων δεδομένων DejaVu, των Dindar N., Fischer P.M., Tatbul N. (2011) φέρνει στο επίκεντρο την αξία των ιστορικών δεδομένων, καθώς αυτά μπορούν να χρησιμοποιηθούν ως βάση γνώσης για την πρόβλεψη μελλοντικών τάσεων και ανακάλυψη σχέσεων αιτιώδους συνάφειας μεταξύ γεγονότων. Η DejaVu αναμιγνύει τα δεδομένα του παρόντος με τα αποθηκευμένα και παλαιότερα γεγονότα. Η ανάμειξη αυτή επιτυγχάνεται με τη χρήση ενός ιδιαίτερου τύπου ερωτημάτων τα οποία ονομάζονται Ερωτήματα Συσχετισμού Προτύπων (Pattern Correlation Queries, PCQs). Η DejaVu προτάσσει μια καινοτόμο και ολοκληρωμένη αρχιτεκτονική επεξεργασίας σύνθετων γεγονότων και εστιάζει σε τεχνικές επεκτασιμότητας.

2.2: Η προσφορά της Αναγνώρισης Σύνθετων Γεγονότων στη Ναυσιπλοΐα

Τα επιστημονικά άρθρα που ακολουθούν αποτελούν ένα υποσύνολο των εργασιών Αναγνώρισης Σύνθετων Γεγονότων, επιλεγμένων με κύριο κριτήριο την φύση των Γεγονότων Υψηλού επιπέδου που εντοπίζουν. Κάποια τέτοια HLEs σε αυτά προσέλκυσαν το ενδιαφέρον πολλαπλών συγγραφικών ομάδων της βιβλιογραφίας και έχουν υλοποιηθεί με διάφορες προσεγγίσεις. Πολλά από τα HLEs αυτά εντοπίζονται και από την εφαρμογή που παρουσιάζεται σε άλλες σελίδες του παρόντος. Φυσικά οι προσεγγίσεις κάθε υλοποίησης ανά σύνθετο γεγονός παρουσιάζουν μοναδική μεθοδολογία σε κάθε περίπτωση.

Όπως υποστηρίζεται στο Κεφάλαιο 5, ορισμένα HLEs αποτελούν μονάχα αρχή περαιτέρω εξωσυστημικής διερεύνησης, καθώς τα γεγονότα χαμηλού επιπέδου που τα συνθέτουν δεν δύνανται να συντελέσουν στην αναγνώριση της κατάστασης που έχει ουσιαστικό ενδιαφέρον σε αυτά, δηλαδή κάποιου γεγονότος «υψηλότερου» ακόμη επιπέδου από τα αναγνωρισμένα HLEs. Ένα τέτοιο παράδειγμα είναι το σύνθετο γεγονός της συνάντησης δύο ή περισσότερων σκαφών στην ανοικτή θάλασσα, γνωστό και ως “rendezvous”. Αυτήν την ύποπτη κατάσταση μελέτησαν αναλυτικά οι Shahir, Glasser, et al. (2015) οι οποίοι με το “Maritime Situation Analysis Framework” διέκριναν διαφορετικά είδη συναντήσεων, όπως παράκτιες, προσυμφωνημένες και άλλες, και ανέπτυξαν ένα πολυεπίπεδο μοντέλο εντοπισμού περιστατικών κάθε είδους. Βάσει αυτού, από τα εισερχόμενα LLEs (AIS, marine radar), εντοπίζονται ζεύγη πλοίων βάσει εγγύτητας και άλλων κριτηρίων, ενώ ένα HMM επιλέγει το πιο πιθανό σενάριο συνάντησης βάσει των παρατηρήσεων απόστασης, ταχύτητας και κατεύθυνσης. Με αυτό το μηχανισμό η υλοποίηση αποκτά δυνατότητες προσαρμογής, αλλά και δράσης υπό συνθήκες αβεβαιότητας. Τελικώς, λοιπές context-specific παράμετροι λαμβάνονται υπ' όψη για περεταίρω κατηγοριοποίηση, πριν την έκδοση οδηγιών.

Η επικείμενη σύγκρουση πλοίων είναι μεταξύ των «δημοφιλέστερων» γεγονότων που οι ερευνητές προσπαθούν να εντοπίσουν σε ρεύμα απλοϊκών ναυτικών γεγονότων. Το ενδιαφέρον για αυτό το HLE είναι δικαιολογημένο: μία τέτοια σύγκρουση μπορεί να έχει πολύ υψηλές υλικές ζημιές, σχεδόν πάντοτε καθυστερήσεις, ενώ σε σοβαρότερες εκδοχές, ναυάγια και μαζική απώλεια ζωής. Η αξία της έγκαιρης αναγνώρισης μίας τέτοιας χώρο χρονικής σύγκλισης σκαφών έχει πολύ μεγάλη αξία, δεδομένων κατάλληλων χειρισμών αποφυγής της. Η δε αναγνώριση μίας σύγκρουσης που μόλις έχει συμβεί, δεν είναι ασήμαντη, καθώς μπορεί να κινητοποιήσει σωστικά μέσα στο σημείο χωρίς χρονοτριβή.

Δύο από τα συναφή ερευνητικά έργα είναι εκείνο των Boubeta-Puig J., Medina-Bulo I., Ortiz G. και Fuentes-Landi G. (2012), “Complex event processing applied to early maritime threat detection”, και το “A complex event processing approach to detect abnormal behaviours in the marine environment” (2015) των Terroso-Saenz F., Valdes-Vela M., και Skarmeta-Gomez A.F.. Η τελευταία εργασία χρησιμοποιεί τεχνική πρόβλεψης μελλοντικών θέσεων για αναγνώριση επικείμενων συγκλισεων, ενώ μία άλλη εκδοχή της τεχνικής αυτής αξιοποιείται και από το παρόν, σε Esper EPL query υπό την ονομασία “Imminent Collision”. Η εργασία των Boubeta-Puig et al. ανιχνεύει επίσης περιστατικά πειρατείας, σφάλματα στο σύστημα πρόωσης ή στην εγκατάσταση AIS. Αντίθετα, εκείνη των Terroso-Saenz et al., πραγματεύεται και εντοπισμό γεγονότων ασυνήθηστα υψηλής ή χαμηλής ταχύτητας.

Ενώ οι υλοποιήσεις αναγνώρισης και επεξεργασίας γεγονότων χρησιμοποιούν μηχανισμούς χάρη στους οποίους είναι σε θέση να διαχειριστούν μεγάλο όγκο πληροφοριών σε πραγματικό χρόνο, η «συμπύκνωση» των LLEs, με την απόρριψη εκείνων που δεν μεταφέρουν πληροφοριακό περιεχόμενο χρήσιμο στην αναγνώριση HLEs παρουσιάζει πλεονεκτήματα με ελάχιστες συνέπειες στην λειτουργία αναγνώρισης. Η διατήρηση των χρήσιμων εκπομπών, που σε έργα των Patroumpas, Artikis, Theodoridis, Pelekis et al. ονομάζονται "critical points", πραγματοποιείται χάρη σε "compressor" αλγόριθμο της υλοποίησης που εντοπίζει μεταβολές σε heading ή ταχύτητα στη χρονική ακολουθία εκπομπών AIS ενός σκάφους.

Την παραπάνω τεχνική περιστολής του όγκου LLEs χρησιμοποιούν οι επόμενες δύο εργασίες, οι οποίες χρησιμοποιούν λογικό προγραμματισμό, διαμέσου του συστήματος RTEC, προς αναγνώριση σύνθετων ναυτιλιακών γεγονότων. Το "Event Recognition for Maritime Surveillance" των Patroumpas K., Artikis A., Katzouris N., Vodas M., Theodoridis Y. και Pelekis N. (2015), διακρίνει γεγονότα ενδιαφέροντος ως προς τη χρονική διάρκειά τους, με κάποιες καταστάσεις να είναι στιγμιαίες όπως μία πολύ απότομη αλλαγή διεύθυνσης, και άλλες αποτελούν φαινόμενο που αφορά πολλές διαδοχικές εκπομπές, όπως ομαλή στροφή σε καμπλύλη σχετικά μεγάλης ακτίνας. Σχεδόν η ίδια συγγραφική ομάδα, για την ακρίβεια οι Alevizos E., Artikis A., Patroumpas K., Vodas M., Theodoridis Y. και Pelekis N., παρουσίασαν το "How not to drown in a sea of information: An event recognition approach" (2015). Το άρθρο αυτό παρουσιάζει πληθώρα HLEs ναυτικού ενδιαφέροντος, όπως ύποπτη υστέρηση σκάφους, συνάντηση σκαφών, καταδίωξη σκάφους και πλεύση σε ελεγχόμενη περιοχή. Το τελευταίο είδος HLE έχει, όπως και η συνάντηση σκαφών, πολλαπλά κίνητρα εκ μέρους των παραβατών, αν και αυτά μπορούν σε πολλές περιπτώσεις να αποκαλυφθούν από το είδος της εκάστοτε περιορισμένης έκτασης. Έτσι, εάν πρόκειται για προστατευμένο βιότοπο οι παραβάτες μπορεί να είναι αλιείς, εάν πρόκειται για πεδίο βολής του Π.Ν., μπορεί να πρόκειται για απόπειρα κατασκοπείας κ.ο.κ.. Σε κάθε περίπτωση, η ικανότητα της υλοποίησης να εντοπίζει γεγονότα βάσει χωρικών περιορισμών, μπορεί να έχει πολλαπλά οφέλη. Ένα ακόμη HLE το οποίο εντοπίζεται στην υλοποίηση της εν λόγω εργασίας είναι η παραλαβή δέματος, η οποία θα μπορούσε ίσως να θεωρηθεί μία προσπάθεια παράνομης μεταβίβασης χωρίς ταυτόχρονη παρουσία στο χώρο (συνάντηση), προς αποφυγή έγερσης υποψίας. Ένας παρόμοιος τύπος HLE εντοπίζεται και από την παρούσα εφαρμογή, υπό την ονομασία "Package Delivery". Το HLE αυτό είναι επίσης πολύπλευρο και απαιτεί πρόσθετη διερεύνηση για την εξακρίβωση της παράβασης, εάν τελικά αυτή υφίσταται.

Σε μία προσπάθεια αποφυγής συγκρούσεων μεταξύ πλοίων ή τουλάχιστον την ταυτοποίηση όσων κινούνται επισφαλώς σε περιοχές αυξημένης ναυτιλιακής κυκλοφορίας, όπως στενά και πορθμοί, υπάρχει μία μερίδα προσεγγίσεων που επικεντρώνονται στην εύρεση διαδρόμων κυκλοφορίας σε τέτοιες περιοχές, με χρήση τεχνικών μη επιτηρούμενης (unsupervised) μηχανικής μάθησης από datasets εκπομπών σκαφών που διέρχονται στην περιοχή. Στη συνέχεια και έχοντας «μάθει» τους διαδρόμους αλλά και την κατεύθυνση σε αυτούς, εντοπίζουν σε άλλο σύνολο δεδομένων πλοία η κίνηση των οποίων δεν συμμορφώνεται με αυτή την θεσμοθετημένη ή μη τάξη, θέτοντας σε κίνδυνο τους διερχόμενους, αφού αυξάνεται έτσι η πιθανότητα κάποιας σύγκρουσης. Ως ανώμαλες τροχές σε τέτοιες περιπτώσεις, ορίζονται εκείνες που διασχίζουν έναν ή περισσότερους διαδρόμους, ή εκείνες που έχουν αντίθετη φορά κίνησης από κάποιο διάδρομο, ενώ χωρικά βρίσκονται εντός αυτού ή πολύ κοντά του. Τέτοιες δημοσιεύσεις αποδίδονται, ενδεικτικά, στους Laxhammar R. (2008) και Pallota G., Vespe M., Bryan K. (2014).

Το "On the Representation and Exploitation of Context Knowledge in a Harbor Surveillance Scenario" των Garcia J., Gomez-Romero J., Patricio M.A., Molina J.M. και Rogova G. (2011), αναπαριστά τους δρώντες του σεναρίου αναγνώρισης γεγονότων ως δομημένες οντότητες βάσει μοντέλου που επιτρέπει καταγραφή των αλληλεξαρτήσεών τους, όπως has-A, και κληρονομικών σχέσεων (is-A) επιτρέποντας με αυτό τον τρόπο διαφοροποίηση ορισμένων οντοτήτων, προσδίδοντάς τους ξεχωριστές ιδιότητες. Καθώς οι κύριες μορφές οντοτήτων αφορούν πλοία αλλά και σταθερές γεωγραφικές περιοχές, το μοντέλο ενδείκνυται για αναγνώριση γεγονότων ενδιαφέροντος σε λιμένες καθώς εκείνοι διέπονται από περισσότερους θεσμούς και κανόνες λειτουργίας, καθιστώντας γενικότερης χρήσης συστήματα ακατάλληλα για αυτές τις εξειδικευμένες ναυτιλιακές περιοχές. Παραδείγματος χάρη, οι προσπάθειες έγκαιρης αναγνώρισης επικίνδυνης σύγκλισης πλοίων επικεντρώνονται σε κριτήρια εγγύτητας. Οι λιμένες όμως είναι περιοχές που η χωρική πυκνότητα πλοίων είναι αναμενόμενη και φυσιολογική, προκαλώντας έτσι εσφαλμένη αναγνώριση συκρούσεων. Η εργασία δεν περιορίζεται σε αυτό το HLE,

με περιγραφές για τις ιδιαιτερότητες στα λιμάνια που απαιτεί η αναγνώριση όπως παράνομη μετανάστευση, τρομοκρατικές ενέργειες κ.α.

Η εργασία των Pan J., Jiang Q., Hu J. και Shao Z. (2012) αναδεικνύει την αξία της οπτικοποίησης της πρωτογενούς πληροφόρισης. Δεν αναγνωρίζει κανένα συγκεκριμένο HLE, αλλά υποστηρίζει τη λήψη αποφάσεων εκ μέρους ανθρώπινου παράγοντα, με επιλεκτική παρουσίαση πλοίων στο χάρτη βάσει μίας μετρικής που αποτελεί στάθμιση διαφορικών ταχύτητας, heading και άλλων κριτηρίων.

Μερίδα ερευνητών της σχετικής βιβλιογραφίας, φαίνεται από το άρθρο των Kazemi S., Abghari S., Lavesson N., Johnson H. και Ryman P. (2013), εστιάζουν στο σημαντικό ζήτημα της εγκυρότητας και αξιοπιστίας των πληροφοριών που μεταδίδονται με μηνύματα AIS: Το πληροφοριακό φορτίο των εκπομπών, μπορεί να αποκλίνει ελαφρώς από τις πραγματικές τιμές λόγο μικρών σφαλμάτων του εξοπλισμού, χρονικών υστερήσεων, θορύβου και λοιπών τεχνικών παραμέτρων. Χειρότερα, ενδέχεται να είναι απολύτως ψευδές ή ανύπαρκτο λόγω εσκεμένης αλλοίωσης ή καταστολής από το πλήρωμα κάποιου σκάφους, προς απόκρυψη επικείμενων παραβατικών ενεργειών. Τα εν λόγω άρθρα παρουσιάζονται στο τέλος του Κεφαλαίου 5, παράλληλα με τη σχετική ανάλυση για τις στρατηγικές προς αντιμετώπιση αυτού του θεμελιώδους προβλήματος στην αναγνώριση σύνθετων ναυτιλιακών γεγονότων.

2.3: Αναγνώριση Σύνθετων Γεγονότων σε λοιπούς κλάδους

Επεκτείνοντας την έρευνα σε επιστημονικές εργασίες Αναγνώρισης Σύνθετων Γεγονότων χωρίς τον περιορισμό των εφαρμογών τους στην ναυτιλία, οι υλοποιήσεις είναι πολυάριθμες και καλύπτουν πληθώρα πτυχών και δραστηριοτήτων. Ενδεικτικά και μη περιοριστικά, αναφέρονται λίγες ενδιαφέρουσες δημοσιεύσεις.

Από το χώρο της καρδιολογίας, η υλοποίηση που περιγράφουν οι Carrault G., Cordier M.-O., Quiniou R. και Wang F. (2003) αποτελεί ένα σύστημα αναγνώρισης καρδιακής αρρυθμίας, δεχόμενο ως είσοδο δεδομένα κάποιου ηλεκτροκαρδιογραφήματος (ECG). Πρόκειται για ένα CRS, δηλαδή για ένα σύστημα όπου η αναγνώριση βασίζεται σε σχέσεις χρονικές μεταξύ των LLEs, τα οποία είναι στην προκειμένη περίπτωση χρονικώς διατεταγμένα κύματα ενδιαφέροντος (συσπάσεις καρδιακών κόλπων / κοιλιών στο ECG). Ένα νευρωνικό δίκτυο επιλέγει ορισμένα από τα LLEs βάσει κάποιων κριτηρίων ιατρικού ενδιαφέροντος και αυτά εισρέουν σε μηχανισμό συμπερασματολογίας που τελικώς κρίνει εάν κάποια από αυτά συνιστούν άρρυθμους κτύπους. Μία ενδιαφέρουσα άποψη του συγκεκριμένου συστήματος είναι πως διαθέτει μηχανισμό αναπροσαρμογής της λειτουργίας του, αντλώντας νέους ορισμούς κανόνων HLEs, μαθαίνοντας με τεχνικές επαγωγικού λογικού προγραμματισμού, επιτηρούμενο, δεχόμενο ECGs με παθολογικά ευρήματα από μία σχετική ΒΔ.

Οι μέθοδοι αναγνώρισης κίνησης οχημάτων από λήψεις εικόνων κυκλοφορίας που παρουσιάζουν οι Hsu-Yung Cheng, Shih-Han Hsu (2011) στο “Intelligent Highway Traffic Surveillance With Self-Diagnosis Abilities”, αναδεικνύουν τη δυνατότητα των συστημάτων αναγνώρισης σύνθετων καταστάσεων να επαφίενται σε πολλαπλές εναλλακτικές προσεγγίσεις αναλόγως των ιδιαίτερων συνθηκών που επικρατούν σε κάποιο σενάριο. Έτσι, οι συγγραφείς χρησιμοποιούν τα χρώματα που εμφανίζονται στο viewport της κάμερας για αναγνώριση οδοστρώματος και αυτοκινήτων κατά τη διάρκεια της ημέρας, ενώ σε ώρες χαμηλού φωτισμού, η ίδια λειτουργικότητα επιτυγχάνεται με ζεύξη των δύο πρόσθιων φωτιστικών σωμάτων. Με ανάλογες μεθόδους υπολογίζονται οι εικονιζόμενες αποστάσεις κ.α.

Παραμένοντας στην χρήση εικόνων CCTV ως LLEs, οι Artikis A., Skarlatidis A. και Paliouras G. (2010) εστιάζουν περισσότερο στη φύση των γεγονότων υψηλού επιπέδου και στην αναγνώρισή τους, παρά στις τεχνικές θεμελίωσης και ορισμού LLEs, καθώς η υλοποίησή τους βασίζεται σε dataset χρονικώς διατεταγμένων εικόνων, ήδη συσχετισμένων με περιγραφές βραχυπρόθεσμων ανθρωπίνων συμπεριφορών, ως LLEs. Πρόκειται για υλοποίηση του Λογισμού Γεγονότων (EC) με χρήση λογικού προγραμματισμού για την αναγνώριση ανθρωπίνων συμπεριφορών μακράς διαρκείας, όπως χαρακτηρίζονται στη συγκεκριμένη εφαρμογή τα HLEs. Τέτοιες συμπεριφορές αποτελούν συναντήσεις και αντιπαραθέσεις μεταξύ προσώπων, αποθέσεις αντικειμένων στο χώρο, πτώσεις κ.α. Οι ορισμοί περιλαμβάνουν ως επί το πλείστον χρονικούς περιορισμούς μεταξύ βραχυχρόνιων συμπεριφορών.

Μέρος της υλοποίησης είναι μηχανισμός εύρεσης νέων κανόνων από το dataset με χρήση λογικού προγραμματισμού, απαγωγικά και επαγωγικά.

Ακόμη, το "Intelligent M2M: Complex event processing for machine-to-machine communication" των Bruns R., Dunkel J., Masbruch H. και Stipkovic S. (2015), δείχνει πως η Αναγνώριση Σύνθετων Γεγονότων έχει χρήσιμες εφαρμογές όπου αποδέκτης της υψηλού επιπέδου πληροφόρησης από μηχανές δεν είναι αποκλειστικά ο άνθρωπος αλλά περαιτέρω μηχανισμοί, στα πλαίσια συστημάτων και δικτύων αυτοματοποιημένου ελέγχου και εποπτείας. Η εργασία περιγράφει ένα αφαιρετικό μοντέλο επικοινωνίας μεταξύ συσκευών βασισμένο σε γεγονότα σχετικά με τη λειτουργία τους, με σκοπό τη υποστήριξη λήψης αποφάσεων από τα αρμόδια τμήματα ελέγχου τους. Εξηγείται η αξία του μοντέλου στη Διαχείριση Παγίων και παρουσιάζονται δύο υποθετικές υλοποιήσεις, περί παραγωγής ηλεκτρικής ενέργειας από φωτοβολταϊκά στοιχεία και περί υπηρεσιών εκτυπώσεων.

2.4: Συγκριτική τοποθέτηση της sailAway

Σε αντίθεση με τα άρθρα που παρουσιάστηκαν στην αρχή του παρόντος κεφαλαίου, η παρούσα εργασία δεν εισάγει νέες τεχνολογίες αναπαράστασης γεγονότων ή εύρεσης καταστάσεων ενδιαφέροντος από αυτά, παρά μόνο αποτελεί εφαρμογή υφιστάμενων και εδραιωμένων τεχνικών. Ακόμη, η χρήση Esper CEP από την sailAway την καθιστά σύστημα πορισματικής (deductive) συμπερασματολογίας, χωρίς δυνατότητες αναπροσαρμογής, τουλάχιστον όχι ενδοσυστημικώς. Αντίθετα, υπάρχει έντονη τάση, βάσει βιβλιογραφίας, για «έξυπνα» συστήματα που υλοποιούν τεχνικές Μηχανικής Μάθησης, είτε αποκλειστικά για εύρεση συσχετίσεων και αιτιολογικών συνδέσεων μεταξύ γεγονότων διαφορετικών επιπέδων αφαίρεσης, ή για την αναπροσαρμογή και βελτιστοποίηση υφιστάμενων σχέσεων, πορισματικής αξιοποίησης.

Σε ότι αφορά την προέλευση των γεγονότων χαμηλού επιπέδου ανά τα συστήματα της βιβλιογραφίας, συχνά παρατηρείται ανάλυση και εύρεση καταστάσεων ενδιαφέροντος επί ιστορικών datasets σε συγκεκριμένες θάλασσες και/ή για ορισμένες χρονικές περιόδους, όπως συνέβη παραδείγματος χάρη με την δημοσίευση των Miliou A., Demetriou M., Caprio C., Tsimpidis T. και Barnicoat S. (2010) για 3 συγκεκριμένα περάσματα στο Αιγαίο μεταξύ 2009 και 2010. Φυσικά υπάρχουν και υλοποιήσεις, όπως η sailAway, που εστιάζονται περισσότερο στο σύστημα αναγνώρισης σύνθετων γεγονότων παρά στις ιδιαιτερότητες κάποιας περιοχής / ιστορικού dataset, όντας ικανές να χρησιμοποιήσουν LLEs άγνωστης προέλευσης ως ρεύμα εισόδου σε πραγματικό χρόνο ή ακόμη και προδιαγεγραμμένο σύνολο στοιχείων.

Ένα συναφές κριτήριο διάκρισης των άρθρων είναι η πραγματική υπόσταση των γεγονότων εισόδου, ανεξάρτητα της ιστορικότητάς τους ή του μοντέλου τροφοδοσίας τους στο σύστημα. Στο σημείο αυτό, επιβάλλεται μία σύντομη συζήτηση για την ανάλυση επί γεγονότων αμφότερων των κατηγοριών. Τα συστήματα που επεξεργάζονται πραγματικά δεδομένα είναι - αξιωματικά σχεδόν - πολύτιμα καθώς εξυπηρετούν τον αυτοσκοπό για τον οποίο η Αναγνώριση Σύνθετων Γεγονότων τελικά προάγει το βιοτικό μας επίπεδο, ανά τις εφαρμογές της. Το παραπάνω όμως δεν υπονοεί πως η ανάλυση υποθετικών καταστάσεων από τεχνητά δεδομένα είναι ανούσια. Αντιθέτως, οι προσεγγίσεις αυτές, στην οποία εντάσσεται και η υλοποίηση των επομένων κεφαλαίων, έχουν αξία επειδή:

1. Επιτρέπουν εστίαση στη λειτουργία της εφαρμογής, για αξιολόγηση, έλεγχο αλλά και αποσφαλμάτωση κατά τη φάση ανάπτυξής της, αφού προδιαγεγραμμένα γεγονότα εισόδου αντανακλούν προβλέψιμα γεγονότα εξόδου, HLEs, που είναι και το ζητούμενο από το σύστημα. Η sailAway, διαθέτει η ίδια το μηχανισμό παραγωγής τεχνητών εκπομπών AIS για την αναγνώριση σύνθετων καταστάσεων από αυτά. Γενικότερα όμως, η παραγωγή τεχνητών γεγονότων χαμηλού επιπέδου για αξιολόγηση συστημάτων (benchmarking) έχει αναγνωριστεί ως ανάγκη από την επιστημονική βιβλιογραφία, όπως φαίνεται από τη δημοσίευση των Theodoridis Y., Silva J.R.O. και Nascimento M.A. (1999) περί αλγορίθμου παραγωγής κινούμενων αντικειμένων σημειακής υπόστασης ή καταλαμβάνοντα ορθογώνιο εμβαδό, διαχρονικώς κινούμενα προς την αξιολόγηση χωροχρονικών δομών δεδομένων. Έχοντας αποκτήσει εμπειρία από εφαρμογές τηλεματικής δικτύων, ο Brinkhoff T. (2002) στη δημοσίευση με τίτλο "A Framework for Generating Network-Based Moving Objects",

επεκτείνει αυτή την ιδέα σε GIS ΒΔ με αναπαράσταση οδικών δικτύων ως γραφήματα. Παρόλο που το domain (πεδίο εφαρμογής) για το οποίο προορίζονται οι γεννήτριες δεδομένων που περιγράφονται στο άρθρο είναι διαφορετικό της ναυτιλίας, η ανάγκη παραγωγής δεδομένων υφίσταται σε αυτή, επίσης.

2. Παρέχουν δυνατότητα εκπαίδευσης, μέσω διαφόρων σεναρίων εξομοίωσης, σε φοιτητές, ερευνητές του γνωστικού αντικειμένου αλλά και επαγγελματίες του κλάδου, Αξιωματικούς Λ.Σ., Π.Ν. και άλλους. Οι εκπαιδευτική προσφορά αυξάνεται ιδιαίτερα όταν τα σενάρια εξομοίωσης περιλαμβάνουν καταστάσεις που εμφανίζονται σπάνια ή είναι επικίνδυνες. Χαρακτηριστικό παράδειγμα υπέρ του επιχειρήματος είναι η διαδεδομένη χρήση εξομοιωτών στις αερομεταφορές και την αεράμυνα.

Στο τέλος του Κεφαλαίου 6 όπου παρατίθενται μεταξύ άλλων οι μελλοντικές κατευθύνσεις του παρόντος εγχειρήματος, καταγράφονται σκέψεις για τη μετατροπή προς παράλληλη χρήση πραγματικών δεδομένων από την εφαρμογή.

Κεφάλαιο 3: Επισκόπηση της εφαρμογής λογισμικού

Περιεχόμενα κεφαλαίου

3.1: Τα εργαλεία που χρησιμοποιήθηκαν

Η Java είναι μία σύγχρονη αντικειμενοστραφής γλώσσα προγραμματισμού, ο σχεδιασμός της οποίας επιτρέπει τον σαφή διαχωρισμό της διαδικασίας ανάπτυξης λογισμικού από την εκάστοτε πλατφόρμα-προορισμό (δηλαδή την αρχιτεκτονική υλικού και λειτουργικού συστήματος όπου πρόκειται να εκτελεστεί το λογισμικό της). Επιτυγχάνει δε αυτό με διαιρεμένη σε δύο στάδια επεξεργασία: Αρχικά γίνεται μεταγλώττιση του πηγαίου κώδικα σε μία ενδιάμεση γλώσσα, κοινή ανεξαρτήτως πλατφόρμας, γνωστή ως Java bytecode που αποθηκεύεται σε αρχεία .class. Στο επόμενο στάδιο, η απαραίτητη τελικώς διαφοροποίηση που απαιτείται για την εκτέλεση λογισμικού γραμμένου σε Java ανά τις διάφορες πλατφόρμες, επιτυγχάνεται από τον τρόπο διερμηνείας του bytecode. Συγκεκριμένα, διαφορετικοί και εξειδικευμένοι για κάθε πλατφόρμα διερμηνείς, διαβάζουν το κοινό σε κάθε περίπτωση bytecode για την παραγωγή τοπικών (native, platform-specific) εντολών μηχανής, κατά το χρόνο εκτέλεσης. Αυτοί οι διερμηνείς είναι γνωστοί ως Java Virtual Machines, JVMs, ενώ το ζεύγος του JVM κάθε πλατφόρμας με τις απαραίτητες συνοδευτικές βιβλιοθήκες είναι γνωστό ως Java Runtime Environment, JRE. Η επιλογή της Java ως μέσο ανάπτυξης εφαρμογών έχει το πλεονέκτημα της μετακύλησης του εργασιακού φόρτου που απαιτείται για την εύρυθμη λειτουργία οποιουδήποτε προγράμματος ανά τις πλατφόρμες, από τους προγραμματιστές αυτού προς την ομάδα ανάπτυξης της Java, μέσω της διατήρησης πολλαπλών εκδόσεων των Virtual Machines, μία ανά πλατφόρμα.

Η Java λοιπόν εστιάζεται στην καθολική συμβατότητα του κώδικά της στο οικοσύστημα των συσκευών εκτέλεσης και ως εκ τούτου στη φορητότητα. Ακόμη, η πολιτική που ακολουθεί διαχρονικώς δίνει έμφαση στη συμβατότητα των νεότερων βιβλιοθηκών της με παλαιότερα, υφιστάμενα προγράμματα που αναφέρονται σε αυτές[10]. Ενσωματώνει τεχνολογίες ανάπτυξης λογισμικού όπως πολυνηματικότητα, διαχείριση εξαιρέσεων, εκτενές μοντέλο εισόδου / εξόδου. Φυσικά περιλαμβάνει και όλα τα χαρακτηριστικά που θα περίμενε κανείς από μία αντικειμενοστραφή γλώσσα, όπως πολυμορφισμό και ενθυλάκωση. Αυτοί είναι μερικοί από τους παράγοντες που έκαναν τη γλώσσα δημοφιλή[11] και ίσως εξηγούν την επιλογή της ως μέσο ανάπτυξης του λογισμικού που θα παρουσιαστεί εν συντομία στη συνέχεια.

Η Esper είναι προϊόν λογισμικού που έχει υλοποιηθεί με Java και προορίζεται για χρήση σε προγράμματα γραμμένα στην ίδια γλώσσα[12], καθώς παρέχεται τυπικά ως συλλογή .jar[13] αρχείων. Τα αρχεία αυτά απαρτίζουν το κύριο μέρος του λογισμικού, μαζί με τμήματά του που μπορούν να χρησιμοποιηθούν προαιρετικά. Ακόμη περιλαμβάνονται κομμάτια λογισμικού γραμμένα από τρίτους, γενικώς γνωστά ως εξαρτήσεις (dependencies), προ-απαιτούμενα για τη λειτουργία του.

[10] Βλ. backwards compatibility.
[11] … παρόλο που διαθέτει και κάποια αδύναμα σημεία. Ένα από αυτά είναι ο τρόπος με τον οποίο υλοποιείται ο παραμετρικός προγραμματισμός σε αυτήν (generic programming). Βλ. type erasure.
[12] Η Esper δύναται να ενσωματωθεί και σε .NET λογισμικό με χρήση της σειράς λογισμικού NEsper από προγράμματα γραμμένα σε C#.
[13] Ενοποιημένος και μεταφέρσιμος τρόπος διάθεσης αρχείων .class.

Πρόκειται για μία μηχανή επεξεργασίας ροών σύνθετων γεγονότων, ανοικτού κώδικα[14]. Δύο συσχετισμένες με αυτή έννοιες οι οποίες περιγράφουν εύστοχα, τα αντικείμενα που θεραπεύει είναι οι CEP (Complex Event Processing, Επεξεργασία Σύνθετων Γεγονότων) και ESP (Event Stream Processing, Επεξεργασία Ροών Γεγονότων).

Η μηχανή της Esper μπορεί να δεχθεί πολλαπλές πηγές μηνυμάτων ως ροές ζωντανών (live) ή αποθηκευμένων (historical) γεγονότων από εύρος μορφών, όπως ενδεικτικά, αρχεία csv, xml, σχεσιακές βάσεις δεδομένων και αντικείμενα Java (ή Collections αυτών). Στην περίπτωση των αντικειμένων, θα πρέπει αυτά να πληρούν κάποιες προδιαγραφές προσπέλασης της κατάστασής τους (state) μέσω κατάλληλων μεθόδων τους[15]

Για χρήση της Esper, ο προγραμματιστής χρησιμοποιεί δύο εργαλεία ως διεπαφές με αυτή:

1. το API, για την αρχικοποίηση της μηχανής και ρύθμιση των παραμέτρων της, δήλωση τύπων δεδομένων και βοηθητικών συναρτήσεων, προσπέλαση πληροφοριών στο ρεύμα γεγονότων εξόδου / αναγνωρισμένων γεγονότων, κ.α.
2. την γλώσσα EPL (Event Processing Language) που είναι ειδικά ανεπτυγμένη για χρήση με αυτή τη μηχανή και αποτελεί τον τρόπο καθορισμού της επιθυμητής επεξεργασίας επί των γεγονότων εισόδου. Με αυτήν, ο προγραμματιστής μπορεί να συνδυάσει υπό συνθήκες ρεύματα εισόδου σε νέα, να τα φιλτράρει και να προχωρήσει σε αναγνώριση σύνθετων καταστάσεων από αυτά, εστιάζοντας σε καθορισμένα χρονικά ή βάσει πλήθους παράθυρα με πληθώρα κριτηρίων, λογικών, χώρο-χρονικών και άλλων. Ακόμη μπορεί να εφαρμόσει στρατηγικές πολλαπλών "βημάτων" για ενδεχόμενη αναγνώριση ιδιαίτερα σύνθετων γεγονότων. Η γλώσσα είναι ευέλικτη, καθώς πέρα από τελεστές και συναρτήσεις εγγενείς σε αυτή, στις οποίες περιλαμβάνονται και αθροιστικές, επιτρέπει στο χρήστη την κλήση συναρτήσεων ορισμένων από εκείνον, στα πλαίσια της συγγραφής του περιβάλλοντος προγράμματος (Java / C#). Η εκφραστική και επεκτάσιμη αυτή γλώσσα βασίζεται στην σύνταξη της SQL, όντας υπερσύνολό της.

Σε απόκριση των αναγνωρισμένων σύνθετων γεγονότων, ή άλλων συνθηκών, ο χρήστης μπορεί να προγραμματίσει την έξοδο της Esper με χρήση listeners / subscribers προσαρτημένων σε ερωτήματα. Ένα τέτοιο σενάριο χρήσης δίνει στην Esper τη μορφή ενός ECA[16] συστήματος, παρόλο που η EPL σαν γλώσσα ερωτημάτων βαθύτατα επηρεασμένη από την SQL, δεν έχει αυτόν τον προσανατολισμό, τουλάχιστον σε ότι αφορά την συνιστώσα της απόκρισης (Action). Η μηχανή χρησιμοποιεί, ενδεικτικά και αναλόγως των μηχανισμών που ο χρήστης επιστρατεύει, γραφικά μοντέλα επεξεργασίας γεγονότων όπως δυναμικά δένδρα και μηχανές πεπερασμένων καταστάσεων. Είναι επεκτάσιμη σε μεγάλη κλίμακα, αντιμετωπίζοντας όγκο γεγονότων και εύρος ερωτημάτων γραμμικά, παρέχοντας σε ζωντανό χρόνο επεξεργασία / αναγνώριση γεγονότων με μικρή χρονική υστέρηση και περιορισμένη δέσμευση πόρων συστήματος, ενώ δύναται να αναπτυχθεί και σε κατανεμημένα σχήματα πηγών γεγονότων και μονάδων επεξεργασίας, χωρίς να εξαρτάται από την εκάστοτε αρχιτεκτονική. Η Esper διατίθεται με εκτενέστατη τεκμηρίωση[17], που αφορά κυρίως τη γλώσσα EPL, αλλά επίσης το API της μηχανής επεξεργασίας γεγονότων, με τη μορφή Javadocs API Reference.

[14] Η EsperTech, συντελεστής ανάπτυξης της Esper, παρέχει επίσης εκδόσεις όπως τις Enterprise Edition και High Availability αλλά και περαιτέρω υπηρεσίες προσανατολισμένες σε επιχειρήσεις.

[15] … δηλαδή accessor methods, a.k.a. getters / setters. Σε γενικές γραμμές, η Esper είναι συμβατή με αντικείμενα Java που συμμορφώνονται, όχι απαραιτήτως με απόλυτη αυστηρότητα, στο πρότυπο JavaBeans. Όμοια είναι και η μορφή γεγονότων εισόδου (LLEs) που παρέχεται από την sailAway στην μηχανή αναγνώρισης σύνθετων γεγονότων.
Για περισσότερες πληροφορίες για το πρότυπο JavaBeans και τις μορφές εισόδου Esper:
http://www.oracle.com/technetwork/java/javase/documentation/spec-136004.html και
http://esper.espertech.com/release-6.1.0/esper-reference/html_single/index.html#eventrep_intro

[16] Event – Condition – Action.

[17] http://www.espertech.com/esper/esper-documentation/

3.2: Υψηλού επιπέδου παρουσίαση λειτουργίας

Η sailAway είναι μία εφαρμογή εξομοίωσης πλεύσης πλοίων και αναγνώρισης σύνθετων γεγονότων από τις κινήσεις τους. Είναι γραμμένη σε Java, με χρήση του Java Standard Edition Development Kit 8 (έκδοση 1.8.0_161), ενώ επίσης χρησιμοποιεί Esper, έκδοση 6.1.0. Αναπτύχθηκε με το εργαλείο σύνταξης κειμένου Notepad++ σε υπολογιστές λειτουργικού συστήματος Windows 7 / 10.

Ως είσοδο δέχεται από το χρήστη, μέσω αρχείων .csv, έναν υποθετικό χάρτη αρχιπελάγους, δρομολόγια πλοίων επί αυτού, αλλά και ρυθμίσεις λειτουργίας της εφαρμογής. Ειδικότερα, σε κατάλληλο φάκελο τοποθετούνται αρχεία που έκαστο αντιπροσωπεύουν το σχήμα και τη θέση ενός πολυγώνου το οποίο αναπαριστά νησί. Μπορεί κανείς να φανταστεί ακόμη και ολόκληρο τον πλανήτη ως ένα ενιαίο ωκεανό, όπου όλες οι στεριές (ακόμη και η Ευρασιατική ήπειρος) βρίσκονται επί αυτού ως νήσοι, αφού βρέχονται περιμετρικώς από θάλασσα. Έτσι η εφαρμογή θεωρεί πως ο χώρος μεταξύ των νήσων, αλλά και αυτός πέραν του περιγεγραμμένου ορθογωνίου του συνόλου τους, είναι νερό όπου τα πλοία μπορούν να κινηθούν. Σε άλλο φάκελο, τοποθετούνται πληροφορίες σχετικά με τα ταξίδια πλοίων (trips). Αυτά διαχωρίζονται σε δύο κατηγορίες, αναλόγως του τρόπου με τον οποίο εξομοιώνεται η κίνηση των πλοίων τους:

1. Γνησίως εξομοιούμενα ταξίδια κατά το χρόνο εκτέλεσης του προγράμματος:
   Τα ταξίδια αυτά χαρακτηρίζονται ως legitimate, χωρίς η ονομασία να υπονοεί κάτι για την παραβατικότητά τους. Απλώς, αντανακλά το γεγονός ότι πρόκειται για σειρές εκπομπών θέσης που παράγονται κατά την εκτέλεση του προγράμματος και, πράγματι, κατόπιν διαδικασίας εξομοίωσης των θέσεων του πλοίου σε πραγματικό χρόνο. Για να γίνει κατανοητή - έστω σε υψηλό επίπεδο - η διαδικασία, θα πρέπει αρχικά να εξεταστεί συνοπτικά η τριάδα των αρχείων που απαιτείται για την περιγραφή ενός τέτοιου ταξιδιού:
   a. Αρχείο περιγραφής πλοίου:
      Περιέχει τις στατικές και διαχρονικά αμετάβλητες πληροφορίες που σχετίζονται με το σκάφος όπως MMSI, μήκος, όνομα, σημαία και άλλα.
   b. Αρχείο διαδρομής:
      Με την υπόθεση ότι η ρότα κάθε πλοίου είναι, σε μία ελαφρώς απλοποιημένη εκδοχή της, μία τεθλασμένη γραμμή, το αρχείο αυτό περιέχει ένα διάνυσμα με συνιστώσες τις κορυφές αυτής της γραμμής (ως συντεταγμένες) στην σειρά που θα προσεγγισθούν από το πλοίο. Οι κορυφές αυτές αναφέρονται ως Waypoints, από τον αντίστοιχο τύπο δεδομένων. Η τεθλασμένη γραμμή της διαδρομής σχηματίζεται από ευθύγραμμα τμήματα (Legs) που έκαστο ενώνει δύο διαδοχικά Waypoints. Η σειρά των Waypoints καθορίζεται από το αρχείο διαδρομής, με την πρώτη καταχώρηση σε αυτό να αποτελεί το σημείο εκκίνησης και η τελευταία τον προορισμό. Οι ενδιάμεσες καταχωρήσεις είναι οι συντεταγμένες των Waypoints που αποτελούν κορυφές στροφών της διαδρομής, αφού κάθε μία αποτελεί το κοινό σημείο δύο διαδοχικών ευθυγράμμων τμημάτων της.
   c. Αρχείο παραμέτρων ταξιδιού:
      Το αρχείο αυτό δεν έχει σχέση με το αρχείο καθολικής παραμετροποίησης της εφαρμογής, αλλά έκαστο αφορά το γνησίως εξομοιούμενο ταξίδι στο φάκελο του οποίου βρίσκεται. Οι πληροφορίες κάθε τέτοιου αρχείου λειτουργούν συμπληρωματικά σε αυτές του αντίστοιχου αρχείου περιγραφής πλοίου, αλλά σε αντίθεση με εκείνες, αντιπροσωπεύουν ποσότητες μεταβλητές ανά ταξίδι, όπως φορτίο (εμπορευμάτων, οχημάτων, καυσίμων κ.λ.π.) και αρχική ταχύτητα.

   Με αφετηρία το πληροφοριακό περιεχόμενο των παραπάνω αρχείων, για κάθε εξομοιούμενο ταξίδι τίθενται σε λειτουργία αλγόριθμοι οι οποίοι μοντελοποιούν με βασικές αρχές φυσικής την επιτάχυνση των πλοίων, την επιβράδυνσή τους πριν από μία αλλαγή κατεύθυνσης, τα όρια ταχυτήτων που μπορούν να φθάσουν στρίβοντας με ασφάλεια σε καμπές δεδομένης ακτίνας και γωνίας, το «θόρυβο» που παρατηρείται στις θέσεις τους λόγω των ανέμων και τις διορθώσεις στις οποίες επιδίδονται με σκοπό να διατηρήσουν την πορεία τους επί των διαδοχικών ευθυγράμμων τμημάτων που ορίζονται από τα waypoints. Καθ' όλη τη διάρκεια της πλεύσης τους παράγονται μηνύματα (εκπομπές) AIS, ως αποτέλεσμα των αλγορίθμων αυτών, οι οποίοι ενσωματώνονται σε μεθόδους του τύπου δεδομένων Lnav.

2. Προδιαγεγραμμένα ταξίδια:
   Πρόκειται για το δίδυμο που αποτελείται αφ' ενός από αρχεία που περιγράφουν πλοία (όπως ακριβώς και στην περίπτωση των legitimates, αφού διαβάζονται από τον ίδιο parser), και αφ' ετέρου από αρχεία που περιγράφουν επακριβώς όλες τις εκπομπές θέσης κάθε πλοίου, σειριακά. Κάθε τέτοια εκπομπή περιλαμβάνει τις συντεταγμένες του πλοίου μαζί με την ταχύτητά του και τον προσανατολισμό του. Κατά κάποιο τρόπο, αυτή η κατηγορία ταξιδιών θα μπορούσε να είναι γνησίως εξομοιούμενα ταξίδια τα οποία όμως έχουν ήδη εκτελεσθεί από την εφαρμογή, με τα στίγματα που παρήχθησαν να έχουν αποθηκευτεί στο αρχείο εκπομπών θέσης. Ασφαλώς υπάρχουν πολλές μικρές τεχνικές διαφορές, αλλά και ουσιαστικότερες, με κυριότερη το γεγονός πως οι συνιστώσες του κάθε διανύσματος εκπομπής μπορούν να καθοριστούν τελείως αυθαίρετα από το χρήστη, όπως επίσης και η αλληλουχία τους μέσα σε ένα τέτοιο αρχείο. Σε αυτή την κατηγορία ταξιδιών δεν υπάρχει πρακτικά εξομοίωση. Η εφαρμογή απλώς διαβάζει τις πληροφορίες του πλοίου και των εκπομπών του, και προχωρά στην αναπαραγωγή των αντίστοιχων αντικειμένων που αντιπροσωπεύουν τις εκπομπές AIS. Κάθε προδιαγεγραμμένο ταξίδι ουσιαστικά παρακάμπτει ένα μέρος των προσπαθειών της εφαρμογής για αληθοφανή κατά το δυνατόν κίνηση πλοίων, αγνοώντας κάποιους ελέγχους ορθότητας των δεδομένων εισαγωγής, όπως έλεγχος για πλεύση σε στεριά. Ο λόγος όμως για τον οποίο έχει συμπεριληφθεί αυτός ο τρόπος παραγωγής LLEs, είναι η αμεσότητα που δίνει στη χρήση της εφαρμογής, αλλά και στην απόλυτη προβλεψιμότητα των εκπομπών, χαρακτηριστικό που επιτρέπει αυξημένη εστίαση επί της διαδικασίας αναγνώρισης HLEs. Σε αντιδιαστολή με τα ταξίδια της προηγούμενης κατηγορίας, τα δρομολόγια που παρουσιάστηκαν εδώ χαρακτηρίζονται από τον κώδικα της εφαρμογής ως illicit, ενώ ο τύπος δεδομένων που αναλαμβάνει την «εξομοίωσή» τους (ή ακριβέστερα: την αναπαραγωγή τους) είναι ο BroadcastPlayer.

Τα δεδομένα εισόδου διαβάζονται με ειδικά ανεπτυγμένους για αυτό το σκοπό CSV Parsers και στη συνέχεια ελέγχονται εκτενέστατα για την ορθότητά τους, σε πολλαπλά επίπεδα. Οι πληροφορίες των αρχείων μετατρέπονται ύστερα από τα πρώτα «κύματα» ελέγχων ορθότητας σε αντικείμενα των κατάλληλων τύπων, με γνώμονα το είδος της οντότητας που περιγράφουν. Θα μπορούσε κανείς να παρομοιάσει αυτή τη διαδικασία με object deserialization, ενώ ορθότερα πρόκειται για ενσωμάτωση των δεδομένων του χρήστη στην εφαρμογή ως αντικείμενα, ή αλλιώς user input integration. Τα αντικείμενα που παράγονται περνούν από πρόσθετους ελέγχους, αναλόγως του τύπου τους. Τελικώς τα άτομα που απέτυχαν κάποιου ελέγχου απορρίπτονται από τη μετέπειτα διαδικασία. Για ένα νησί λόγου χάρη, οντότητα που ελέγχεται εκτενέστερα από όλες, ελέγχεται το αρχείο .csv, ώστε να περιέχει αριθμούς και όχι αλφαριθμητικά ως συντεταγμένες. Στη συνέχεια γίνονται έλεγχοι επί του πολυγώνου που το διάνυσμα αναπαριστά (π.χ. απλότητα γραμμής, ύπαρξη τριών ή περισσοτέρων κορυφών). Επιτυγχάνοντας, το πολύγωνο δημιουργείται ως αντικείμενο του τύπου Island και αυτό στη συνέχεια ελέγχεται για το ενδεχόμενο συγκάλυψής του από άλλα υφιστάμενα νησιά στον ίδιο χώρο, κ.ο.κ.. Οι εν λόγω έλεγχοι της ορθότητας δεδομένων εισαγωγής από το χρήστη (user input validation) αφορούν ένα σημαντικό μέρος της εφαρμογής. Ως μηχανισμός ανάδρασης προς τον ανθρώπινο παράγοντα χρησιμοποιείται η τεχνική των εμφωλευμένων εξαιρέσεων[18] από τις οποίες μπορεί να διερευνηθεί σταδιακά η αιτία και προέλευση της όποιας αποτυχίας αξιοποίησης των δεδομένων εισαγωγής. Για αυτό το σκοπό έχουν γραφεί και ειδικοί τύποι εξαιρέσεων.

Στο ζήτημα της εισαγωγής δεδομένων από το χρήστη, εισάγονται ακόμη σε ειδικό αρχείο καθολικές παράμετροι λειτουργίας της εφαρμογής. Οι ρυθμίσεις που μπορεί ο χρήστης να ελέγξει μέσω του αρχείου αυτού καλύπτουν ένα εύρος θεμάτων όπως, ενδεικτικά, την ένταση των ανέμων (turbulence) που μπορούν να κάνουν ένα πλοίο να παρεκκλίνει της πορείας του και την εμφάνιση ή μη του παραθύρου γραφικών που παριστά το χάρτη με τις κινήσεις των πλοίων σε πραγματικό χρόνο.

[18] Από μία εναλλακτική προσέγγιση, σε χρόνο αποσφαλμάτωσης, η τεχνική αυτή είναι γνωστή ως exception chaining καθώς λαμβάνουμε μία σειρά εξαιρέσεων «ξεπακετάροντας» τις διαδοχικά.

3.2.1: Το σενάριο εξομοίωσης

Στο σημείο αυτό εισάγεται η έννοια του σεναρίου εξομοίωσης, οριζόμενη ως το σύνολο εκείνων των πληροφοριών εισόδου που με οποιοδήποτε τρόπο μπορούν να επηρεάσουν το παραγόμενο ρεύμα εκπομπών θέσεως. Πρόκειται δηλαδή για ένα υποσύνολο των πληροφοριών που παρέχονται από τα αρχεία εισόδου. Συγκεκριμένα, το σενάριο εξομοίωσης περιλαμβάνει όλες τις πληροφορίες των αρχείων νήσων, πλοίων, παραμέτρων και δρομολογίων, αλλά και μέρος από τις ρυθμίσεις του αρχείου παραμετροποίησης. Προτρέχοντας ελαφρώς, αναφέρουμε πως πρόκειται για όλες τις μεταβλητές σε αυτό, εκτός από τις:

- debugStatus (boolean flag που καθορίζει την έξοδο λεπτομερέστερων πληροφοριών για τη λειτουργία της εφαρμογής).
- globalSpeedLimit (δεκαδικός που αντανακλά το όριο ταχύτητας που επιβάλει στη θάλασσα η υποθετική Λιμενική Αρχή. Χρησιμεύει σε ερωτήματα για αναγνώριση παραβατικής συμπεριφοράς).
- guiMode (Java enum που καθορίζει την παραγωγή γραφικής εξόδου κατά τη διάρκεια της εξομοίωσης, αλλά και το είδος αυτής).

  Ακόμη, υπάρχουν οι εξής ρυθμίσεις στο αρχείο που επίσης δεν αποτελούν μέρος του σεναρίου εξομοίωσης και σκοπό έχουν να καθορίσουν με περισσότερη ακρίβεια τη μορφή της γραφικής απεικόνισης της εφαρμογής, εάν αυτή είναι ενεργοποιημένη:
- trailLength (το πλήθος των πιο πρόσφατων θέσεων κάθε πλοίου).
- immobileSpeedThreshold (το όριο ταχύτητας κάτω από το οποίο κάθε πλοίο απεικονίζεται με διαφορετικό σύμβολο στο χάρτη, υπονοώντας αδράνεια).

Η χρήση των μεταβλητών αυτών αναλύεται περεταίρω στις κατάλληλες ενότητες.

Βάσει του σεναρίου εξομοίωσης, ξεκινά ένα ανεξάρτητο νήμα επεξεργασίας (thread) για κάθε ταξίδι. Οι εκπομπές πραγματοποιούνται για κάθε νήμα ταξιδίου στην ίδια χρονική συχνότητα, δηλαδή ανά δεδομένη και καθολική για την εφαρμογή χρονική περίοδο. Η περίοδος αυτή, broadcastInterval, αποτελεί μία από τις διαθέσιμες ρυθμίσεις του αρχείου. Η παραγωγή των αντικειμένων ξεκινά συγχρονισμένα, αν και τα ταξίδια μπορούν να ολοκληρωθούν σε διαφορετικές χρονικές στιγμές, με κάποια πλοία να φθάνουν στον προορισμό τους νωρίτερα από άλλα. Η φύση των νημάτων παραγωγής (legitimate / illicit) των αντικειμένων δεν είναι «ορατή» από τη μηχανή της Esper, η οποία έχει «επίγνωση» μόνο του περιεχομένου των μηνυμάτων, τα οποία έχουν όλα την ίδια μορφή, αφού είναι αντικείμενα του ίδιου τύπου δεδομένων, ενώ επίσης δεν διαθέτουν πεδίο στο οποίο να αποσαφηνίζεται αυτή η διάκριση μεταξύ ταξιδιών.

Το αποτέλεσμα όλων αυτών των παράλληλων διαδικασιών είναι τελικώς οι εκπομπές θέσης των πλοίων ως AISBroadcast objects, μηνύματα πολύ όμοια με αυτά του προτύπου AIS. Όλα αυτά τα μηνύματα αποστέλλονται στην Esper, αποτελώντας για αυτήν το ρεύμα εισόδου (LLEs). Περισσότερα για την λειτουργία των ερωτημάτων EPL και των listeners της Esper ακολουθούν στο αντίστοιχο κεφάλαιο.

3.3: Υποσυστήματα, δομή αρχείων και αυτοματοποιημένη μεταγλώττιση

Καθώς η εφαρμογή έχει κάποια έκταση, τόσο με τη συμβατική μετρική των γραμμών κώδικα αλλά με βάση το πλήθος των τύπων δεδομένων που ορίζονται σε αυτή, κρίθηκε χρήσιμη, ήδη από πρώιμο στάδιο ανάπτυξης, η χρήση ενός τρόπου επιμερισμού των αλγορίθμων και τύπων της, με γνώμονα την συνάφεια των εργασιών και των ρόλων τους. Η Java ως γλώσσα που ενδείκνυται για εκτενή προγραμματιστικά έργα, αντιμετωπίζει το πρόβλημα αυτό παρέχοντας δομές και μηχανισμούς που εξυπηρετούν το σκοπό της τμηματικότητας διατηρώντας κάποιο βαθμό σχετικής ανεξαρτησίας των τμημάτων που προκύπτουν. Πρόκειται για μία έννοια που μπορεί καλύτερα να αποδοθεί - μονολεκτικά τουλάχιστον - στην αγγλική ως “modularity”. Οι εν λόγω δομές είναι τα Java packages[19] και ακόμη, αρχής γενομένης από την Java εκδ. 9, τα Java modules[20].

Η παρούσα εφαρμογή δεν κάνει χρήση των μηχανισμών αυτών, αφού δεν ορίζει ρητά packages ή modules. Παρόλα αυτά, διαθέτει τμηματική (modular) υπόσταση χάρη σε μία πιο απλή, ad hoc, προσέγγιση επιμερισμού / ομαδοποίησης η οποία στα πλαίσια της παρούσας εργασίας ονομάζεται υποσύστημα.

Υποσύστημα της sailAway είναι μία συλλογή τύπων δεδομένων που παρουσιάζουν συνάφεια ως προς τις οντότητες που αναπαριστούν και τις λειτουργίες που επιτελούν, ομοίως με Java package.

Η ιεραρχική δομή των κυριότερων φακέλων αρχείων της εφαρμογής φαίνεται στο παρακάτω δένδρο:

```
sailAway
├───application
│   ├───dataTypes
│   │   ├───ais
│   │   ├───configuration
│   │   ├───exceptions
│   │   ├───gui
│   │   ├───lowLevelTypes
│   │   ├───parsers
│   │   └───userInput
│   ├───listeners
│   └───main
├───esper
│   └───deps
├───helperApps
│   └───ScaleAndOffset
├───inputFiles
│   ├───islands
│   └───trips
│       ├───illicit
│       └───legitimate
├───media
│   └───icons
└───outputFiles
```

Καθώς αναλύεται σε υψηλό επίπεδο ο ρόλος των αρχείων, δεν γίνεται παρουσίαση όλων των περιεχομένων του \sailAway\, χάριν απλότητας. Έτσι, στο παραπάνω σχήμα δεν απεικονίζονται φάκελοι οι οποίοι έχουν δευτερεύοντα ρόλο, καθώς και κάποιοι που έχουν δυναμική δημιουργία κατά το runtime της εφαρμογής. Ακόμη, δεν εμφανίζονται αρχεία. Τα στοιχεία αυτά που απουσιάζουν,

[19] Συλλογές συναφών τάξεων με δυνατότητα ορισμού κοινής πολιτικής πρόσβασης (scope) από τον κώδικα της εφαρμογής.

[20] Συλλογές από Java packages που συνοδεύονται από λεπτομερή περιγραφή, χάρη σε ένα σχήμα μεταδεδομένων, ορίζοντας σχέσεις όπως εξαρτήσεις από άλλα modules κ.λ.π. Η ονομασία αυτής της δομής είναι μάλλον ατυχής, καθώς πολλές επιμέρους μονάδες κώδικα Java αποτελούν και αυτές (ασχέτως επιπέδου εξειδίκευσης) ανεξάρτητα τμήματα (modules). Έτσι παραδείγματος χάρη, μία μέθοδος (method) αποτελεί module με την ευρύτερη έννοια καθώς είναι προστατευμένη και ρητά ανεξάρτητη από το περιβάλλον της (private internals, parameter / return type contracts, encapsulation).

παρουσιάζονται επιλεκτικά κατά την επιμέρους εξέταση των αντίστοιχων top level[21] φακέλων, που ακολουθεί:

- application
  Στον φάκελο αυτό βρίσκεται η εφαρμογή διαμερισμένη σε υποφακέλους που έκαστος αντιπροσωπεύει κάποιο υποσύστημά της. Κάθε τέτοιος φάκελος περιέχει τον πηγαίο κώδικα του υποσυστήματος, το αποτέλεσμα μεταγλώττισης αυτού, αλλά και ένα βοηθητικό Windows batch file (.cmd). Σε επόμενη παράγραφο, παρουσιάζεται ενδεικτικά ένας τέτοιος φάκελος, αποκαλύπτοντας την εσωτερική οργάνωσή του και το ρόλο των αρχείων σε αυτό.
  Ο application περιέχει τρεις υποφακέλους:
  - main
  - dataTypes
  - listeners

  Οι main και listeners είναι φάκελοι που περιέχουν τα αντίστοιχα ομώνυμα υποσυστήματα. Ο φάκελος dataTypes απλώς περιέχει συγκεντρωμένους τους φακέλους των υπόλοιπων επτά υποσυστημάτων στα οποία βασίζεται η εφαρμογή, εξαρτώμενη από αυτά, αφού για τους τύπους δεδομένων εντός των main και listeners, οι τύποι υποσυστημάτων υπό τον dataTypes αποτελούν τύπους χαμηλού επιπέδου, ή εξαρτήσεις. Με άλλα λόγια οι κύριοι, high level αλγόριθμοι της εφαρμογής αναφέρονται στους low level τύπους εντός του dataTypes.
- esper
  Εδώ βρίσκεται το κύριο αρχείο του Esper CEP (esper-6.1.0.jar), το οποίο περιλαμβάνει συγκεντρωμένο το bytecode του εν λόγω συστήματος, απαραίτητο τόσο στο χρόνο μεταγλώττισης όσο και στο χρόνο εκτέλεσης της εφαρμογής. Στον υποφάκελο deps[22], βρίσκονται τέσσερα πρόσθετα .jar αρχεία, τα οποία καλούνται από το τους τύπους δεδομένων της Esper κατά το χρόνο εκτέλεσης. Τα πέντε συνολικώς αρχεία αυτά είναι τα ελάχιστα δυνατά που απαιτούνται για τη λειτουργία της Esper, αλλά καλύπτουν τις ανάγκες της εφαρμογής.
- helperApps
  Πρόκειται για την τοποθεσία εναπόθεσης βοηθητικών εφαρμογών που σκοπό έχουν να διευκολύνουν τη χρήση της εφαρμογής sailAway με διάφορους τρόπους. Κατά το χρόνο συγγραφής του παρόντος, περιέχεται μόνο η εφαρμογή ScaleAndOffset που σκοπό έχει να επιτρέψει στο χρήστη τη μαζική μεταβολή αρχείων εισόδου πραγματοποιώντας αλλαγές στην κλίμακα και τη θέση ακολουθιών από συντεταγμένες. Το utility αυτό παρουσιάζεται εκτενέστερα στο Παράρτημα Β.
- inputFiles
  Όπως το όνομα προδίδει, αυτός είναι ο φάκελος στον οποίο τοποθετούνται όλα τα αρχεία που περιέχουν δεδομένα εισαγωγής, δηλαδή το σενάριο εξομοίωσης και τις λοιπές ρυθμίσεις. Το αρχείο ρυθμίσεων είναι το inputFiles\configuration.csv. Εντός του υποφακέλου islands τοποθετείται σε top level ένα αρχείο .csv για κάθε νησί του σεναρίου εξομοίωσης. Το όνομα του νησιού λαμβάνεται από το όνομα του αρχείου του. Ο φάκελος trips αφορά τα ταξίδια και περιέχει τους υποφακέλους illicit και legitimate, αναλόγως του είδους του ταξιδιού, όπως αναλύθηκε προηγουμένως. Έτσι, στον κατάλληλο υποφάκελο του trips τοποθετείται σε top level φάκελος του ταξιδιού με κάποιο ενδεικτικό όνομα. Εντός αυτού, σε top level, τα αρχεία που τελικώς περιγράφουν το ταξίδι αυτό. Ένα illicit ταξίδι αποτελείται από τα αρχεία broadcasts.csv και vessel.csv, ενώ ένα γνησίως εξομοιούμενο από τα vessel.csv, parameters.csv και waypoints.csv. Υπενθυμίζεται πως τα αρχεία περιγραφής πλοίων έχουν την ίδια μορφή για κάθε είδος ταξιδιού και είναι ανταλλάξιμα μεταξύ ειδών, αφού διαβάζονται από τον ίδιο parser σε κάθε περίπτωση και παράγουν objects του ίδιου τύπου.
- media
  Πρόκειται για το φάκελο πολυμεσικών αρχείων της εφαρμογής, χάρη στον οποίο αυτά διαχωρίζονται από τον πηγαίο κώδικα και το bytecode. Ο μοναδικός φάκελος του media στο παρόν στάδιο ανάπτυξης είναι ο media\icons, περιέχοντας μόνο το anchor.png. Το αρχείο

[21] Top level περιεχόμενα ενός φακέλου Α, θεωρούνται τα περιεχόμενα του Α που δεν περιέχονται σε φακέλους οι οποίοι επίσης αποτελούν περιεχόμενα του Α.
[22] Συντομία του dependencies, «εξαρτήσεις».

αυτό αποτελεί το εικονίδιο του παραθύρου του χάρτη που εμφανίζεται στο runtime, αναλόγως ρυθμίσεων γραφικής εξόδου.

- outputFiles
  Στον κατάλογο αυτό, η εφαρμογή εγγράφει σε αρχεία κειμένου τα σύνθετα γεγονότα που έχει αναγνωρίσει από τη ροή εκπομπών θέσεως των πλοίων. Σε κάθε εκτέλεση δημιουργείται ένας υποφάκελος που αφορά το συγκεκριμένο αυτό run instance, εφ' εξής «φάκελος εκτέλεσης». Κάθε τέτοιος φάκελος είναι ονοματισμένος με την ημερομηνία και ακριβή ώρα[23] έναρξης εκτέλεσης, ώστε να αποφεύγεται η αντικατάσταση παλαιών αποτελεσμάτων από νεότερες εκτελέσεις (overwrites). Εντός του φακέλου εκτέλεσης βρίσκεται το αρχείο "AIS Broadcasts.txt" στο οποίο είναι καταγεγραμμένες όλες οι εκπομπές AIS των πλοίων (LLEs, από την οπτική της αναγνώρισης σύνθετων γεγονότων) σε αύξουσα χρονική σειρά, δηλαδή με το παλαιότερο στην αρχή. Ακόμη, εντός του υποφακέλου Event Reports τοποθετούνται τα αρχεία των αναγνωρισμένων γεγονότων, με ένα αρχείο για κάθε ερώτημα EPL που παρήγαγε αποτέλεσμα, δηλαδή αναγνώρισε ένα ή περισσότερα από τα γεγονότα που περιγράφει. Σε κάθε τέτοιο αρχείο, τα γεγονότα εγγράφονται σε αύξουσα χρονική σειρά.

- Ένα παράδειγμα των περιεχομένων του outputFiles:

```
outputFiles
├──Thursday the 12th of October 2017 03_03_45
│    │   AIS Broadcasts.txt
│    │
│    └──Event Reports
│           Package Delivery Events.txt
│           Reported Overspeed Events.txt
│
├──Thursday the 12th of October 2017 03_04_52
│    │   AIS Broadcasts.txt
│    │
│    └──Event Reports
│           Imminent Collision Events.txt
│           Reported Overspeed Events.txt
│           Sharp Turn Events.txt
```

Από τα περιεχόμενα στον φάκελο outputFiles του παραδείγματος, είναι εμφανές πως η εφαρμογή έτρεξε δύο φορές, την Πέμπτη 12 Οκτωβρίου 2017. Η πρώτη εκτέλεση ξεκίνησε στις 3:03 και 45'' π.μ.. Σε αυτή, πραγματοποιήθηκαν ένα ή περισσότερα Package Delivery HLEs και ένα ή περισσότερα Reported Overspeed HLEs. Στην επόμενη εκτέλεση, που ξεκίνησε στις 3:04 και 52'' π.μ., πραγματοποιήθηκαν ένα ή περισσότερα Imminent Collision HLEs, ένα ή περισσότερα Reported Overspeed HLEs και ένα ή περισσότερα Sharp Turn HLEs. Η διαφορά των αποτελεσμάτων μεταξύ των δύο εκτελέσεων, οφείλεται στο διαφορετικό σενάριο εξομοίωσης. Οι παραχθείσες εκπομπές των δύο εκτελέσεων βρίσκονται καταγεγραμμένες στα αντίστοιχα "AIS Broadcasts" αρχεία των δύο φακέλων. Είναι φυσικό διαφορετικά LLE streams να προκαλούν και την ανίχνευση διαφορετικών HLEs, με δεδομένα και αμετάβλητα τα ερωτήματα αναγνώρισης. Στο παραπάνω παράδειγμα, οι διαφορές είναι εμφανείς λόγω των αρχείων αναφοράς HLEs που εξέδωσε η εφαρμογή. Ασφαλώς θα μπορούσαν να υπάρχουν και δύο εκτελέσεις όπου παρήχθησαν αρχεία με ίδια ονόματα, αλλά ενδεχομένως με διαφορετικό πλήθος / χρόνο πραγματοποίησης και άλλα χαρακτηριστικά μεταξύ μεμονωμένων HLEs του ίδιου είδους. Με άλλα λόγια, διαφορετικό περιεχόμενο των αντίστοιχων αρχείων. Σε τέτοια περίπτωση, εξυπακούεται πως οι δύο εκτελέσεις δεν έχουν το ίδιο αποτέλεσμα και δεν θεωρούνται ταυτόσημες.

Έχοντας περιγράψει σε κάποιο βαθμό την χρήση των αρχείων της εφαρμογής, αξίζει να επισημανθεί πως όλες οι διαδρομές αρχείων που δηλώνονται στον κώδικα της εφαρμογής και στα scripts της είναι

[23] Η ώρα έχει μορφή 24 ωρών. Καθώς στην ονοματοδοσία των αρχείων ΛΣ της οικογένειας Windows, χαρακτήρες όπως η άνω-κάτω τελεία «:» δεν μπορούν να χρησιμοποιηθούν, επιλέχθηκε η κάτω παύλα «_» ως διαχωριστικό ωρών / λεπτών / δευτερολέπτων.

relative, με κοινό root το φάκελο sailAway. Αυτό επιτρέπει τη μετακίνηση της εφαρμογής με τη μετακίνηση ολόκληρου του ομώνυμου φακέλου σε άλλες τοποθεσίες ή υπολογιστές, χωρίς να διαταράσσεται η ικανότητά της να διαβάζει / γράφει αρχεία εισόδου / εξόδου. Ομοίως δεν επηρεάζεται η λειτουργία των scripts μεταγλώττισης και εκτέλεσης. Σε καμία περίπτωση όμως δεν μπορεί να υπάρξει αλλαγή στη δομή κάτω από το φάκελο sailAway ή στις σχετικές θέσεις των περιεχομένων του.

Εστιάζοντας στα υποσυστήματα, παρακάτω φαίνεται η οργάνωση των αρχείων τους. Χάριν απλότητας επιλέχθηκε ένα υποσύστημα με μικρό αριθμό αρχείων, το ais:

```
ais
│   build.cmd
├───classes
│       AISBroadcast.class
│       Broadcast.class
└───src
        AISBroadcast.java
        Broadcast.java
```

Σε κάθε φάκελο υποσυστήματος βρίσκονται σε top level οι υποφάκελοι src και classes καθώς και το αρχείο build.cmd. Ο src περιέχει σε top level αρχεία πηγαίου κώδικα που έκαστο ορίζουν και το ομώνυμο data type. Εξαίρεση αποτελούν αρχεία .java με τύπους δεδομένων που ορίζουν τοπικά και άλλους, πρόσθετους, βοηθητικούς τύπους (nested classes). Με χρήση του αρχείου αυτοματοποιημένης μεταγλώττισης build.cmd, το αποτέλεσμα αποθηκεύεται στον φάκελο classes κατά αντιστοιχία και σε top level. Στην περίπτωση εμφωλευμένων τύπων, ο compiler παράγει τα πρόσθετα αρχεία .class που αντιστοιχούν σε αυτούς.

Το αρχείο build.cmd κάθε υποσυστήματος είναι ένα εκτελέσιμο αρχείο δέσμης ενεργειών των Windows, περιέχει δηλαδή ένα script[24] της γραμμής εντολών του λειτουργικού συστήματος αυτού[25]. Ο σκοπός του αρχείου είναι η διευκόλυνση της μεταγλώττισης του κώδικα του υποσυστήματος ύστερα από προγραμματιστικές αλλαγές και επεκτάσεις. Η αυτοματοποίηση αυτή είναι χρήσιμη καθώς η διαδικασία μεταγλώττισης είναι ως επί το πλείστον αμετάβλητη ενώ είναι αναγκαίο να πραγματοποιείται συχνά (τουλάχιστον σε ενεργές φάσεις ανάπτυξης) και απαιτεί αρκετή πληκτρολόγηση ακριβείας, λόγω των σύνθετων αλληλεξαρτήσεων μεταξύ υποσυστημάτων που πρέπει να καθοριστούν με διαφορετικό τρόπο για κάθε ένα στο –classpath option[26] της εντολής μεταγλώττισης (πρόγραμμα javac). Ως απάντηση σε αυτές τις προκλήσεις, το script έχει αποθηκευμένες τις απαιτούμενες εντολές με τα κατάλληλα options, τα οποία μπορούν απλώς να τρέξουν συνολικά ως διαδικασία με ένα κλικ. Ένα πρόσθετο πλεονέκτημα είναι η συνέπεια που εξασφαλίζεται αφού σε κάθε εκτέλεση διεξάγονται οι ίδιες εργασίες, ενώ η πιθανότητα λανθασμένης ενέργειας από το χρήστη ελαχιστοποιείται. Ο αλγόριθμος σε κάθε τέτοιο script, συνοπτικά:

1. Απενεργοποιεί την παραγωγή εξόδου στην γραμμή εντολών.
   Αυτή η ενέργεια δεν είναι απαραίτητη, απλώς αποκρύπτει από το χρήστη τις περιεχόμενες εντολές του script, όχι όμως και την έξοδο που παράγεται από αυτές, διατηρώντας την γραμμή εντολών οπτικά «καθαρή», αλλά επιτρέποντας την εμφάνιση σημαντικών μηνυμάτων, όπως compilation errors / warnings.
2. Διαγράφει όλα τα περιεχόμενα του φακέλου classes του υποσυστήματος.
   Η ενέργεια δεν είναι απολύτως απαραίτητη, αλλά εξασφαλίζει πως μετά το πέρας της

[24] Αλγόριθμος αποτελούμενος από επιμέρους εντολές ενός λειτουργικού συστήματος και κλήσεις σε προγράμματά του μέσω γραμμής εντολών. Για την υλοποίηση τέτοιων αλγορίθμων, τα αντίστοιχα λειτουργικά συστήματα παρέχουν επίσης και εντολές ελέγχου ροής, επαναλήψεων κ.α.

[25] Κατά αντίστοιχο τρόπο μπορούν να γραφούν και εκτελέσιμα scripts για άλλα Λ.Σ. (όπως π.χ. εκείνα του προτύπου POSIX) που θα πραγματοποιούν τις ίδιες ακριβώς εργασίες.

[26] Επιλογή παραμετροποίησης της λειτουργίας ενός CLI προγράμματος. Στην περίπτωση του Java compiler, javac.exe, η επιλογή –classpath (ή –cp) χρησιμοποιείται για τον ορισμό των φακέλων στους οποίους βρίσκονται τα .class αρχεία των τύπων δεδομένων στους οποίους αναφέρεται ο προς μεταγλώττιση κώδικας. Το option χρησιμοποιείται κατά τον ίδιο τρόπο και κατά την κλήση του JVM (πρόγραμμα java.exe).

εκτέλεσης του script, το bytecode του φακέλου classes είναι, στο σύνολό του, προϊόν του τελευταίου build.

3. Μεταγλωττίζει όλα τα αρχεία πηγαίου κώδικα του υποσυστήματος.
   Πρόκειται για όλα τα αρχεία .java αυτού, δηλαδή τα περιεχόμενα του φακέλου src. Αυτό επιτυγχάνεται με κλήση του java compiler και χρήση του wildcard *.java. Στην κλήση αυτή ενσωματώνεται η κατάλληλη παραμετροποίηση για κάθε υποσύστημα. Συγκεκριμένα, μέσω του option –cp καταδεικνύονται στον compiler οι classes φάκελοι άλλων υποσυστημάτων στα οποία ο υπό μεταγλώττιση κώδικας αναφέρεται, ή άλλες τοποθεσίες που περιέχουν bytecode λοιπών εξαρτήσεων, όπως το λογισμικό Esper. Επιπλέον, με το option –d, το αποτέλεσμα μεταγλώττισης κατευθύνεται στον φάκελο classes.
4. «Παγώνει» τη ροή του script στο τέλος της διαδικασίας.
   Με τον τρόπο αυτό, διατηρείται το παράθυρο της γραμμής εντολών ανοικτό αντί να κλείσει με την ολοκλήρωση του script. Η τελευταία εντολή προκαλεί την εμφάνιση ενός prompt όπως «Press any key to continue…»[27], περιμένοντας είσοδο από το πληκτρολόγιο. Ως αποτέλεσμα, το script δεν μπορεί να ολοκληρωθεί πριν από την είσοδο αυτή και το παράθυρο διατηρείται, επιτρέποντας στο χρήστη να διαβάσει εκεί τις παρατηρήσεις του compiler. Εάν δεν επιστραφεί κάποιο error στην γραμμή εντολών, ως αποτέλεσμα της εκτέλεσης του javac.exe, η διαδικασία είναι επιτυχής[28].

Στην περίπτωση σφαλμάτων μεταγλώττισης, αξίζει να παρατηρήσουμε πως ο φάκελος classes του υποσυστήματος θα είναι άδειος ή ελλιπής και δεν θα περιέχει την προηγούμενη έκδοση των αρχείων .classes, εάν κάποια υπήρχε ήδη εκεί. Ο λόγος είναι η σειρά των εντολών του script, όπου πρώτα διαγράφονται όλα τα περιεχόμενα του φακέλου και στη συνέχεια καλείται ο μεταγλωττιστής, ο οποίος δεν παράγει τα αναμενόμενα αρχεία .class εάν αντιμετωπίσει οποιοδήποτε σφάλμα σε έστω και ένα από τα αρχεία πηγαίου κώδικα.

Ύστερα από αλλαγές στον κώδικα πολλαπλών υποσυστημάτων, ο χρήστης θα πρέπει να εκτελέσει με την κατάλληλη σειρά τα build scripts αυτών, αφού τα υποσυστήματα εξαρτώνται το ένα από το άλλο. Για μια πλήρη μεταγλώττιση της εφαρμογής, θα πρέπει να εξεταστούν οι σχέσεις εξάρτησης των υποσυστημάτων και σε κάθε βήμα να μεταγλωττίζονται υποσυστήματα που δεν εξαρτώνται από άλλα που δεν έχουν ακόμη μεταγλωττιστεί στα πλαίσια αυτής της διαδικασίας. Έτσι η διαδικασία ξεκινά από εκείνα που εξαρτώνται μόνο από τύπους δεδομένων που παρέχονται από το JRE και από τύπους εντός του ίδιου υποσυστήματος, ενώ με την επιτυχή μεταγλώττιση αυτών καθίσταται δυνατή η συνέχιση με άλλα υποσυστήματα, αναλόγως των σχέσεων εξάρτησης, μέχρις ότου η εφαρμογή έχει πλήρως μεταγλωττισθεί.

Η σειρά μεταγλώττισης έχει ως εξής:

1. Όλα τα υποσυστήματα του φακέλου dataTypes
2. Υποσύστημα listeners
3. Υποσύστημα main

Για την αποφυγή αυτής της ιδιαίτερα επίπονης και χρονοβόρας διαδικασίας, υπάρχει ακόμη ένα build script, το sailAway\application\dataTypes\build.cmd που αναλαμβάνει την αυτοματοποιημένη μεταγλώττιση των υποσυστημάτων του dataTypes, εκτελώντας απλώς με την κατάλληλη σειρά εντολές από τα επιμέρους scripts των επτά υπό το φάκελο αυτό υποσυστημάτων, απλοποιώντας κατά πολύ τις ενέργειες του χρήστη. Ακόμη, στα σχόλια του κώδικά του, περιγράφονται αναλυτικά οι σχέσεις εξάρτησης των υποσυστημάτων αυτών. Το κατευθυνόμενο γράφημα που θα μπορούσε να περιγράψει τις σχέσεις είναι άκυκλο με την εξαίρεση του κύκλου των userInput και ais. Αναλυτικότερα, ο τύπος δεδομένων BroadcastSequence του υποσυστήματος userInput αναφέρεται στον Broadcast του υποσυστήματος ais και ο τύπος AISBroadcast του ais αναφέρεται στον Vessel του υποσυστήματος userInput. Η ιδιαίτερη σχέση εξάρτησης αυτών των δύο στοιχείων προκαλεί πρόβλημα μόνο σε περιπτώσεις που ως αποτέλεσμα αφήνουν το φάκελο classes του υποσυστήματος ais κενό, χωρίς να περιέχει κάποιες παλαιότερες εκδόσεις αρχείων ώστε να μπορέσει να γίνει η αναφορά από το

[27] Η ακριβής μορφή του μηνύματος εξαρτάται από την πλατφόρμα-προορισμό και τις ρυθμίσεις γλώσσας σε αυτήν.
[28] Πέρα από errors, ο compiler παράγει επίσης και απλά warnings (προειδοποιήσεις) σε κάποιες περιπτώσεις. Τα warnings δεν σταματούν τη διαδικασία μεταγλώττισης και δεν αποτελούν ένδειξη αποτυχίας.

userInput. Από την άλλη πλευρά του ίδιου προβλήματος, δεν είναι δυνατή η μεμονωμένη μεταγλώττιση του ais, καθώς δεν μπορεί να γίνει η απαιτούμενη αναφορά στο userInput, καθώς η διαδικασία μεταγλώττισης του τελευταίου επίσης αποτυγχάνει λόγω της δικής του αδυναμίας αναφοράς. Μια διαδικασία επίλυσης αυτής της κατάστασης, η οποία θα μπορούσε να χαρακτηριστεί ως "co-dependency deadlock", περιγράφεται στα σχόλια του κώδικα του γενικού αυτού build script του dataTypes. Καθώς η αλληλεξάρτηση υφίσταται μόνο σε επίπεδο υποσυστημάτων και όχι συγκεκριμένων τύπων, η λύση αυτή περιλαμβάνει ως ενδιάμεσο βήμα τη μεταγλώττιση του userInput χωρίς τον τύπο που προκαλεί την εξάρτηση του υποσυστήματος από το ais.

Οι ίδιοι λόγοι που οδήγησαν στην ανάπτυξη build scripts, και ιδιαίτερα η ανάγκη για πληκτρολόγηση θέσεων bytecode των τύπων από τους οποίους εξαρτάται η τάξη του entry point[29] της εφαρμογής, οδήγησαν στη συγγραφή ενός ακόμη script για την εκτέλεσή της: το sailAway\run.cmd. Το script αυτό είναι πιο απλό από τα build scripts, καθώς δεν διαγράφει αρχεία, απλώς καλεί το JVM (java.exe) με όρισμα τον τύπο Application στο φάκελο classes του υποσυστήματος main. Ο τύπος αυτός περιέχει το entry point της εφαρμογής. Προς αποφυγή πολυπλοκότητας στο script αυτό, η κλήση του Application μέσω java.exe δεν δέχεται command line arguments. Η εφαρμογή απλώς διαβάζει τις επιθυμητές ρυθμίσεις που ο χρήστης ορίζει στο sailAway\inputFiles\configuration.csv. Στο option –cp καταδεικνύονται τα υπόλοιπα οκτώ υποσυστήματα, αλλά και η τοποθεσία της Esper και των εξαρτήσεών της.

Η χρήση των παραπάνω scripts θα μπορούσε να αποφευχθεί με τη χρήση κάποιου σύγχρονου IDE κατάλληλου για Java projects, παραδείγματα αποτελούν τα NetBeans και Eclipse. Παρόλα αυτά, έχει και η παρούσα προσέγγιση τα δικά της πλεονεκτήματα όπως αμεσότητα, διαφάνεια και υψηλότερη φορητότητα.

3.4: Οι συνιστώσες τάξεις ανά υποσύστημα: Οι ρόλοι και οι σχέσεις τους

Με τις παραγράφους που προηγήθηκαν, σχεδόν ολοκληρώνεται η υψηλού επιπέδου περιγραφή της εφαρμογής. Η έως εδώ επεξήγηση αποτελεί προϋπόθεση για την παράθεση μίας λεπτομερέστερης αλλά επιλεκτικής ανάλυσης ορισμένων πιο πολύπλοκων ή κρίσιμων σημείων που χρήζουν περαιτέρω εμβάθυνσης. Ωστόσο, υπάρχουν δύο ακόμη προαπαιτούμενα: Αφ' ενός, η αντιστοίχιση των αρχείων πηγαίου κώδικα με τους τύπους που ορίζονται σε αυτά, μαζί με σύντομη περιγραφή τους, για την εξήγηση των ρόλων τους και / ή των οντοτήτων που αναπαριστούν. Αφ' ετέρου, κρίνεται χρήσιμη η αποσαφήνιση, μέσω απλοποιημένων[30] UML Class Diagrams, των σχέσεων κληρονομικότητας μεταξύ τους.

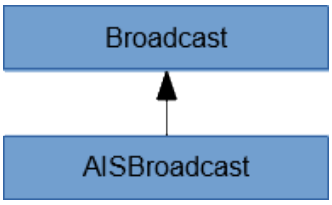
Στα διαγράμματα, με κόκκινο συμβολίζονται οι τύποι δεδομένων που ορίζονται στα παρεχόμενα Java Packages του JRE Standard Edition, αποτελούν δηλαδή πρακτικά κομμάτι της γλώσσας. Με πράσινο αντιστοίχως συμβολίζονται οι τύποι δεδομένων που ορίζονται από την διανομή Esper CEP που χρησιμοποιήθηκε, ενώ τα υπόλοιπα data types, που συμβολίζονται με μπλε, είναι οι τύποι δεδομένων της παρούσας εφαρμογής. Τα διαγράμματα πέραν της ιδιαιτερότητας της μη εμφάνισης των μελών των τάξεων, διέπονται από τις υπόλοιπες συμβάσεις των UML του είδους. Σχετικά με αυτές, υπενθυμίζεται πως οι αφηρημένες τάξεις συμβολίζονται με πλάγια γραμματοσειρά, ενώ τα interfaces με την ένδειξη "interface" πριν το όνομα της τάξης σε πλάγια γραμματοσειρά.

[29] Το entry point είναι το μπλοκ κώδικα από το οποίο ξεκινά η εκτέλεση του προγράμματος. Κατά σύμβαση στα προγράμματα Java, είναι η μέθοδος public static void main(String[]).

[30] Η απλοποίηση έγκειται στην παράλειψη των μελών των τάξεων στα διαγράμματα αυτά. Για μία πληρέστερη εικόνα της σύνθεσης των τύπων δεδομένων της εφαρμογής, ο αναγνώστης παρακινείται να εξετάσει το Παράρτημα Α' (Javadocs).

Υποσύστημα ais

Αρχείο πηγαίου κώδικα	Οριζόμενος τύπος δεδομένων
AISBroadcast.java	AISBroadcast
Broadcast.java	Broadcast



AISBroadcast: Τύπος δεδομένων που αναπαριστά μήνυμα AIS, δηλαδή εκπομπή θέσης[31] πλοίου. Αποτελείται από τα πεδία timeStamp (long, χρονική στιγμή εκπομπής), vessel (Vessel, αναφορά στο αντικείμενο του εκπέμποντος πλοίου), position (Coordinates, συντεταγμένες), heading (Azimuth, προσανατολισμός) και speed (double, η ταχύτητα σε μέτρα ανά δευτερόλεπτο, m/s). Το ακατέργαστο ρεύμα εισόδου γεγονότων στην Esper (LLEs) αποτελείται αποκλειστικά από μηνύματα τέτοιου τύπου.

Broadcast: Γονικός τύπος του AISBroadcast. Σε σύγκριση με τον απόγονό του, περιέχει λιγότερα πεδία (position, heading, speed), τα οποία ο απόγονος κληρονομεί. Δεν είναι αφηρημένος τύπος, καθώς επιτρέπει τη δημιουργία αντικειμένων του. Κάθε τέτοιο αντικείμενο περιέχει τις πληροφορίες μίας γραμμής (καταχώρησης) του broadcasts.csv. Τα αντικείμενα του τύπου αυτού έχουν περιορισμένη χρήση στην εφαρμογή καθώς αφορούν μόνο προκαθορισμένα ταξίδια. Έχουν επίσης περιορισμένη διάρκεια ζωής, αφού αποτελούν απλώς ενδιάμεσο βήμα παραγωγής AISBroadcast, "εμπλουτιζόμενα" με τα πεδία που λείπουν κατά την αναπαραγωγή τους από τον BroadcastPlayer. Για τη διαδικασία αυτή άλλωστε υπάρχει πρόσθετος εξειδικευμένος constructor στην τάξη AISBroadcast.

Υποσύστημα configuration

Αρχείο πηγαίου κώδικα	Οριζόμενος τύπος δεδομένων
AppConfiguration.java	AppConfiguration
GuiMode.java	GuiMode

AppConfiguration: Τάξη που να αποθηκεύει τις ρυθμίσεις της εφαρμογής που προέρχονται από το αρχείο configuration.csv. Έχει 9 διαφορετικά πεδία, όσα δηλαδή και αυτά που ο χρήστης θέτει στο αρχείο. Υπάρχουν επιπλέον 9 πεδία, τα “defaults”, με τις προκαθορισμένες τιμές για κάθε ρύθμιση και χρησιμεύουν ως μηχανισμός ασφαλείας σε περίπτωση λανθασμένης εισαγωγής του χρήστη. Η αρχιτεκτονική της εφαρμογής είναι όμοια singleton pattern, με τη βασική διαφορά πως δεν υπάρχει ούτε ένα αντικείμενο του τύπου. Αντιθέτως, όλα τα μέλη είναι μέλη της τάξης, αφού είναι δηλωμένα ως static. Ο τύπος έχει constructor ιδιωτικής πρόσβασης και αυτός δεν καλείται ποτέ. Τα “defaults” είναι final[32], και οι μόνες μέθοδοι που διαθέτει είναι getters / setters[33] για τα πεδία ρυθμίσεων και getters για τα αντίστοιχα “defaults”. Οι μέθοδοι είναι και τα μοναδικά μέλη δημοσίου πρόσβασης (public). Τα πεδία είναι «ζωντανά» από την έναρξη της εκτέλεσης της εφαρμογής και αρχικοποιούνται χάρη σε ένα static initialization block, εντός του οποίου σε κάθε πεδίο ρύθμισης ανατίθεται η τιμή του αντίστοιχου “default”. Μόνο μετά την ανάγνωση του αρχείου ρυθμίσεων από τον τύπο AppConfigurationCSVParser οι τιμές των ρυθμίσεων αλλάζουν[34],εάν δεν υπάρχουν σφάλματα στις τιμές του χρήστη. Χάρη σε αυτήν τη τάξη, οι ρυθμίσεις που επηρεάζουν ποικιλοτρόπως τη λειτουργία της

[31] Ο όρος «εκπομπή θέσης» είναι, στην περίπτωση του AISBroadcast ή ενός πραγματικού μηνύματος AIS, υπεραπλούστευση καθώς ένα τέτοιο μήνυμα περιέχει πολύ περισσότερες πληροφορίες από τη γεωγραφική θέση του σκάφους τη δεδομένη στιγμή.
[32] Ο modifier αυτός καθιστά το μέλος αμετάβλητο, ύστερα από την εισαγωγή αρχικής τιμής.
[33] Μέθοδοι, τυπικά δημόσιας πρόσβασης, που διαβάζουν (“get”) ή θέτουν (“set”), τα πεδία ιδιωτικής πρόσβασης της τάξης τους, συνεισφέροντας έτσι στις καλές πρακτικές του αντικειμενοστραφούς προγραμματισμού, που επιτάσσουν ενθυλάκωση / απόκρυψη της εσωτερικής δομής τύπων δεδομένων.
[34] Ο AppConfigurationCSVParser είναι ο μοναδικός τύπος που γράφει στα πεδία του AppConfiguration.

εφαρμογής, βρίσκονται συγκεντρωμένες στο state της, ενώ είναι ορατές σε όσα υποσυστήματα αναφέρονται σε αυτή.

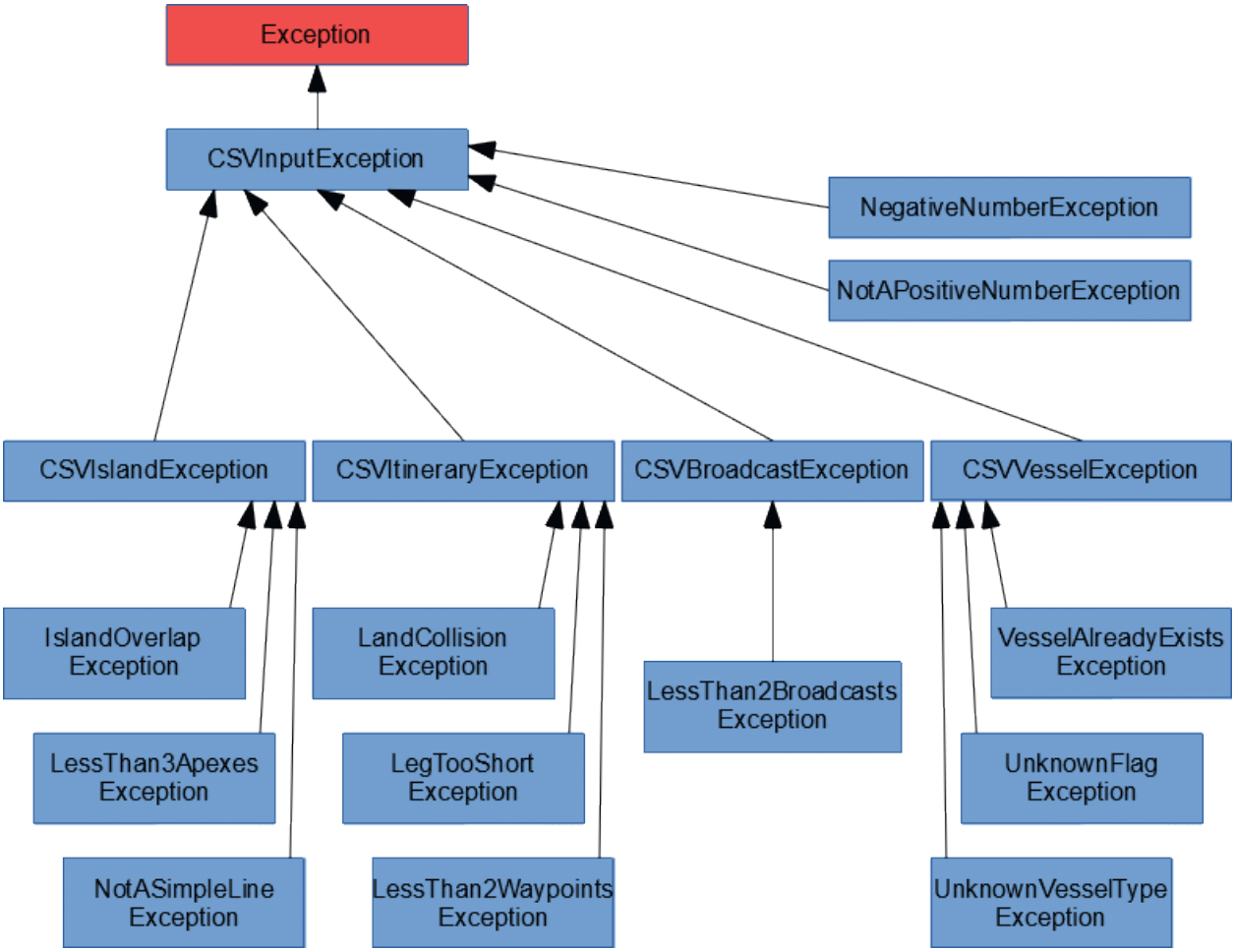
GuiMode: Java enum που περιέχει τις προκαθορισμένες τιμές που δέχεται το πεδίο GuiMode του AppConfiguration. Αυτές είναι οι:

- NOGUI
- FIXED
- RESIZABLE_TO_TRAILS
- RESIZABLE_TO_ICONS

Η πρώτη τιμή ως αποτέλεσμα έχει τη μη εμφάνιση γραφικών, ενώ με τις υπόλοιπες τρείς εμφανίζεται JFrame που απεικονίζει το χάρτη των νησιών και τις κινήσεις των πλοίων σε ζωντανό χρόνο. Στην κατάλληλη ενότητα, αναλύεται η λειτουργικότητα των γραφικών.

Υποσύστημα exceptions

Αρχείο πηγαίου κώδικα	**Οριζόμενος τύπος δεδομένων**
CSVBroadcastException.java	CSVBroadcastException
CSVInputException.java	CSVInputException
CSVIslandException.java	CSVIslandException
CSVItineraryException.java	CSVItineraryException
CSVVesselException.java	CSVVesselException
IslandOverlapException.java	IslandOverlapException
LandCollisionException.java	LandCollisionException
LegTooShortException.java	LegTooShortException
LessThan2BroadcastsException.java	LessThan2BroadcastsException
LessThan2WaypointsException.java	LessThan2WaypointsException
LessThan3ApexesException.java	LessThan3ApexesException
NegativeNumberException.java	NegativeNumberException
NotAPositiveNumberException.java	NotAPositiveNumberException
NotASimpleLineException.java	NotASimpleLineException
UnknownFlagException.java	UnknownFlagException
UnknownVesselTypeException.java	UnknownVesselTypeException
VesselAlreadyExistsException.java	VesselAlreadyExistsException



IslandOverlapException: Το υπό επεξεργασία / προς ενσωμάτωση νησί επικαλύπτει χωρικά (μερικώς ή ολικώς) κάποιο άλλο νησί που έχει ήδη τοποθετηθεί στο χάρτη.

LandCollisionException: Υπάρχει χωρική επικάλυψη, μερική ή ολική, ενός ή περισσοτέρων Legs του υπό επεξεργασία δρομολογίου Lnav, με στεριά. Υπενθυμίζεται πως τα νησιά παράγονται και τοποθετούνται πριν από τα trips και πως δεν υπάρχει αντίστοιχος έλεγχος για προδιαγεγραμμένα scripted / illicit ταξίδια.

LegTooShortException: Το μήκος ενός Leg (τύπος που χρησιμοποιείται μόνο σε Lnav ταξίδι), έχει μήκος μικρότερο του lane width[35]. Για την ομαλή λειτουργία των αλγορίθμων πλεύσης πρέπει κάθε ευθύγραμμο τμήμα της τεθλασμένης γραμμής της διαδρομής να έχει μήκος τουλάχιστον όσο το πλάτος του διαδρόμου.

LessThan2BroadcastsException: Το προδιαγεγραμμένο (scripted) ταξίδι αποτελείται από λιγότερες των δύο εκπομπών θέσης, Broadcasts. Ένα ταξίδι οφείλει να αντιπροσωπεύει κίνηση πλοίου και ως εκ τούτου να αποτελείται από δύο τουλάχιστον εκπομπές θέσης, με την πρώτη από αυτές να αφορά το σημείο απόπλου ή γενικότερα εκκίνησης, και το τελευταίο τον προορισμό, είτε πρόκειται για σημείο κοντά στην περίμετρο νησιού (λιμένα) ή σημείο στην ανοικτή θάλασσα.

LessThan2WaypointsException: Το γνησίως εξομοιούμενο ταξίδι (Lnav) αποτελείται από λιγότερα των δύο Waypoints.

[35] Πλάτος θαλάσσιου διαδρόμου πλεύσης.

LessThan3ApexesException: Το υπό δημιουργία νησί, ως πολύγωνο στο χάρτη, αποτελείται από 2 ή λιγότερες κορυφές. Κάθε πολύγωνο αποτελείται τουλάχιστον από 3.

NegativeNumberException: Εισαγωγή αρνητικού αριθμού σε πεδία όπου απαιτείται μηδέν ή θετικός.

NotAPositiveNumberException: Εισαγωγή μηδέν ή αρνητικού αριθμού σε πεδία όπου απαιτείται θετικός.

NotASimpleLineException: Εξαίρεση που μπορεί να εμφανισθεί κατά τη δημιουργία νησιού: Η τεθλασμένη γραμμή που περιγράφει την περίμετρό του, δεν είναι απλή, δηλαδή τέμνει τον εαυτό της. Ως αποτέλεσμα, προκύπτει ένα αυτο-τεμνόμενο πολύγωνο, γνωστό και ως μη απλό, το οποίο ασφαλώς δεν μπορεί να αναπαριστά ένα νησί.

UnknownFlagException: Η σημαία του πλοίου που καθορίζεται στο αρχείο vessel.csv δεν υπάρχει στο σύνολο που αντιπροσωπεύει το enum Flag.

UnknownVesselTypeException: Το είδος του πλοίου που καθορίζεται στο αρχείο vessel.csv δεν υπάρχει στο σύνολο που αντιπροσωπεύει το enum VesselType.

VesselAlreadyExistsException: Το υπό ενσωμάτωση αντικείμενο πλοίου (Vessel), περιγράφει σκάφος το οποίο έχει το ίδιο MMSI με κάποιο άλλο του στόλου (αντικείμενο Fleet) στο υπό διαμόρφωση σενάριο εξομοίωσης. Δεν επιτρέπονται τα διπλότυπα / κλώνοι πλοίων, καθώς αυτά δεν μπορούν να πλέουν ταυτόχρονα σε δύο ή περισσότερες διαδρομές. Ο έλεγχος αφορά και τους δύο τύπους ταξιδιών (Lnav / BroadcastPlayer).

CSVBroadcastException: Base class εξαιρέσεων σχετικών με τη δημιουργία στιγμάτων θέσης προδιαγεγραμμένου ταξιδιού.

CSVIslandException: Base class εξαιρέσεων σχετικών με τη δημιουργία νησιού και τοποθέτησής του στο αρχιπέλαγος.

CSVItineraryException: Base class εξαιρέσεων σχετικών με τη δημιουργία διαδρομής γνησίως εξομοιούμενου ταξιδιού.

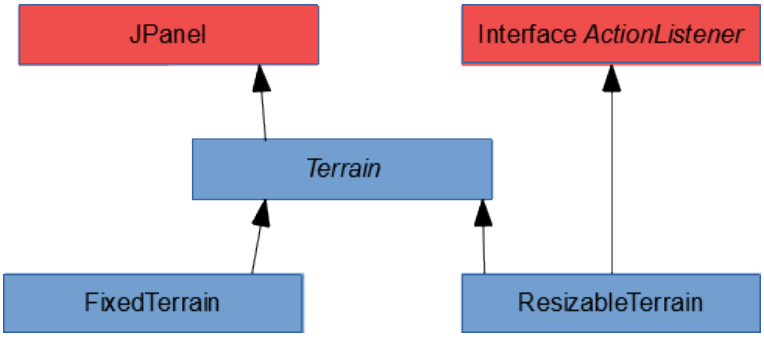
CSVVesselException: Base class εξαιρέσεων σχετικών με τη δημιουργία πλοίου (Vessel).

Οι παραπάνω τέσσερις είναι γονικοί τύποι συγκεκριμένων επιμέρους ομάδων εξαιρέσεων, αναλόγως της οντότητας στης οποίας την ενσωμάτωση χρησιμοποιούνται. Παρόλο που δεν είναι αφηρημένοι τύποι, ο ρόλος τους είναι ως επί το πλείστον οργανωτικός, καθώς μέσω του πολυμορφισμού που παρέχουν στους απογόνους τους, τους ομαδοποιούν επιτρέποντας σχετική ευελιξία στη συγγραφή των τύπων ανάγνωσης αρχείων εισαγωγής (parsers).

CSVInputException: Ο γονικός τύπος όλων των εξαιρέσεων που ορίζονται από την εφαρμογή. Στα πλαίσια της τεχνικής της εμφώλευσης εξαιρέσεων που χρησιμοποιείται στην εφαρμογή, οι απόγονοί του εμφωλεύονται σε αυτόν. Ο τύπος αυτός δεν είναι αφηρημένος καθώς εξαιρέσεις του παρόντος τύπου εγείρονται ως αποτέλεσμα και άλλων, java native εξαιρέσεων.

Υποσύστημα gui

Αρχείο πηγαίου κώδικα	Οριζόμενος τύπος δεδομένων
FixedTerrain.java	FixedTerrain
	FixedTerrain.Isle
Terrain.java	Terrain
	Terrain.Ship
	Terrain.Beep
	Terrain.IntPair
ResizableTerrain.java	ResizableTerrain



Terrain: Αφηρημένος τύπος που επεκτείνει το javax.swing.JPanel αποτελώντας το μέσο εμφάνισης γραφικών από την εφαρμογή. Περιέχει τοπικούς (εμφωλευμένους) βοηθητικούς τύπους δεδομένων τους οποίους κληρονομούν οι δύο μη αφηρημένες τάξεις που τον επεκτείνουν, FixedTerrain και ResizableTerrain.

Terrain.Ship: Βοηθητικός τύπος που συγκεντρώνει τις πληροφορίες που απαιτούνται για την ορθή αναπαράσταση εικονιδίου πλοίου: χρώμα, τρέχουσα θέση, προσανατολισμός, όνομα, MMSI (για λόγους αναφοράς) και ιστορικό προσφάτων θέσεων για χάραξη του «ίχνους» του στην επιφάνεια του νερού.

Terrain.Beep: Βοηθητικός τύπος που αποτελεί απλοποιημένη μορφή του τύπου AISBroadcast, περιέχοντας μόνο τις απαραίτητες πληροφορίες για τη γραφική απεικόνιση των κινήσεων των σκαφών ως αλληλουχία στιγμάτων: θέση, προσανατολισμός, όνομα, MMSI, ταχύτητα.

Terrain.IntPair: Βοηθητικός τοπικός τύπος που αναπαριστά, όπως ο Coordinates, συντεταγμένες. Η διαφορά είναι πως ο IntPair περικλείει διατεταγμένο ζεύγος ακεραίων και όχι δεκαδικών. Αποτελεί τρόπο απεικόνισης των συντεταγμένων του χάρτη, ύστερα από τη μετατροπή σε συντεταγμένες εικονοστοιχείων οθόνης. Η μετατροπή αυτή λαμβάνει υπόψη την ανάλυση οθόνης του υπολογιστή όπου εκτελείται η εφαρμογή.

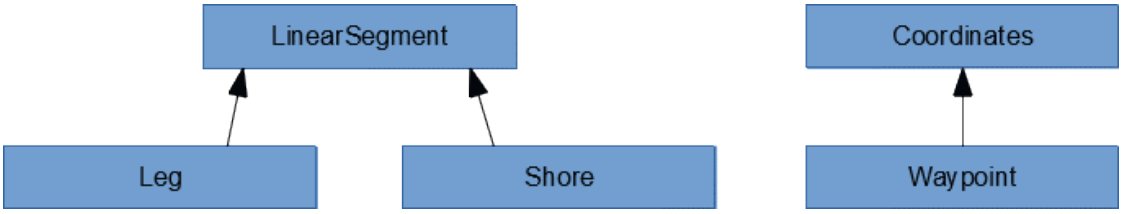
FixedTerrain: Πρόκειται για το χάρτη που ύστερα από την αρχική εκτίμηση της κλίμακας και θέσης του αρχιπελάγους, δεν προσαρμόζει το ορατό του πεδίο αλλάζοντας εστίαση (zoom) και θέση (pan). Δεν ακολουθεί τις κινήσεις των πλοίων ώστε αυτά να είναι στο σύνολό τους ορατά εντός του παραθύρου.

FixedTerrain.Isle: Βοηθητικός τύπος που χρησιμοποιείται μόνο από FixedTerrain χάρτη και αναπαριστά νησί. Καθώς η εκτίμηση της κλίμακας και θέσης γίνεται μόνο μία φορά, στην αρχή του κύκλου ζωής ενός αντικειμένου FixedTerrain, οι συντεταγμένες των ακρωτηρίων και κόλπων κάθε νησιού μπορούν χάριν αποδοτικότητας να «μεταφραστούν» σε πολύγωνα του παρόντος τύπου με συντεταγμένες κορυφών οθόνης, τύπου IntPair.

ResizableTerrain: Τύπος που χρησιμοποιείται ως εναλλακτικός του FixedTerrain. Σε αντίθεση με αυτόν, το viewport του παρόντος ακολουθεί τις κινήσεις των πλοίων, διατηρώντας τα εικονίδια των πλοίων ορατά, ακόμη και εάν πλεύσουν πέραν του αρχιπελάγους. Προσαρμόζει σε πραγματικό χρόνο εστίαση και θέση, υπολογίζοντάς τις εκ νέου ύστερα από κάθε εκπομπή θέσης που φθάνει από τους γεννήτορες εκπομπών (Lnav / BroadcastPlayer).

Υποσύστημα lowLevelTypes

Αρχείο πηγαίου κώδικα	Οριζόμενος τύπος δεδομένων
Azimuth.java	Azimuth
Coordinates.java	Coordinates
Flag.java	Flag
Leg.java	Leg
LinearSegment.java	LinearSegment
	LinearSegment. IntersectionStatus
Shore.java	Shore
Turn.java	Turn
TurnDirection.java	TurnDirection
VesselType.java	VesselType
Waypoint.java	Waypoint



Azimuth: Αζιμούθιο σε μοίρες. Χρησιμοποιείται για την σήμανση κατεύθυνσης, με σημαντικότερη εφαρμογή το heading των πλοίων. Η τάξη ενθυλακώνει έναν ιδιωτικής προσπέλασης αριθμό διπλής ακριβείας, ενώ επίσης περιέχει μεθόδους που επιτρέπουν αλλαγές κατεύθυνσης, υπολογισμούς σχετικού / απόλυτου αζιμουθίου και άλλα, διατηρώντας πάντα το αζιμούθιο "κανονικοποιημένο", ως υπόλοιπο ακέραιης διαίρεσης με το 360.

Coordinates: Συντεταγμένες στο νοητό χάρτη. Αποτελεί βασικότατο για την εφαρμογή τύπο, αφού χάρη σε αυτόν ορίζονται στεριές, θέσεις πλοίων, πορείες κ.λ.π. Στην καρδιά αυτής της τάξης βρίσκεται ένα διατεταγμένο ζεύγος δεκαδικών διπλής ακριβείας, κρατώντας τιμές γεωγραφικού μήκους και γεωγραφικού πλάτους. Το σύστημα συντεταγμένων της εφαρμογής είναι το καρτεσιανό ορθοκανονικό επίπεδο. Όλες οι μετρήσεις είναι σε μέτρα του S.I. (m).

Flag: Java enum για την αποτύπωση του συνόλου που ορίζει το πρότυπο ISO 3166-1 alpha-2. Πρόκειται για κωδικοποίηση των χωρών του κόσμου σε αλφαριθμητικά 2 χαρακτήρων. Ίσως η πιο γνωστή χρήση του προτύπου αυτού είναι τα Internet top level domains (TLDs).

LinearSegment: Τύπος που αναπαριστά ευθύγραμμο τμήμα, οριζόμενος από δύο αντικείμενα Coordinates, δηλαδή δύο σημεία. Διαθέτει μεθόδους εύρεσης σημείων τομής μεταξύ ευθυγράμμων τμημάτω, αποστάσεων τους από σημεία κ.α..

LinearSegment.IntersectionStatus: Βοηθητικός, ανεξάρτητος εμφωλευμένος τύπος ο οποίος χρησιμοποιείται για να εμπλουτίσει την έννοια της τομής μεταξύ δύο ευθυγράμμων τμημάτων. Ενθυλακώνει μία boolean τιμή και ένα σημείο (αντικείμενο Coordinates). Το σημείο του τύπου αποτελεί το σημείο τομής των ευθειών-φορέων των δύο ευθυγράμμων τμημάτων, εφ' όσον ασφαλώς αυτές τέμνονται, δηλαδή δεν είναι παράλληλες. Η μπουλιανή τιμή είναι αληθής εάν το σημείο αυτό ανήκει από κοινού στα δύο ευθύγραμμα τμήματα. Με αυτόν τον τρόπο, αποτυπώνονται πληροφορίες τομής τόσο για τα τμήματα, όσο και για τις ευθείες που τα φέρουν.

Leg: Κληρονομικός απόγονος του LinearSegment. Αποτελείται από δύο Waypoints (απόγονοι του Coordinates) και αντιπροσωπεύει ευθύγραμμο τμήμα μεταξύ τους, δηλαδή το ευθύγραμμο μέρος της διαδρομής ενός Lnav, μεταξύ δύο διαδοχικών στροφών (ή σημείων εκκίνησης / προορισμού).

Shore: Κληρονομικός απόγονος του LinearSegment. Αντιπροσωπεύει πλευρά πολυγώνου νησιού.

TurnDirection: Java enum με τιμές: {LEFT, RIGHT} που λειτουργεί ως ιδιότητα του τύπου Turn. Δείχνει την κατεύθυνση κάποιας αλλαγής πορείας ενός πλοίου από την δική του οπτική (αριστερή, “portside”, ή δεξιά, “starboard”, στροφή).

Turn: Τύπος που αποτυπώνει πληροφορίες μίας καμπύλης πορείας στην επιφάνεια της θάλασσας. Στην προσπάθεια για κατά το δυνατόν αληθοφανείς κινήσεις κατά την εξομοίωση των Lnav ταξιδιών, οι πορείες που στην πραγματικότητα ακολουθούν τα γνησίως εξομοιούμενα ταξίδια δεν ακολουθούν πιστά την τεθλασμένη γραμμή κοντά στις κορυφές της. Αντίθετα, αλλάζουν πορεία τμηματικά και ομαλά, διαγράφοντας ένα τόξο κύκλου σε κάθε στροφή, σε μία διαδικασία που περιγράφεται αναλυτικότερα σε κατάλληλη ενότητα. Για την εκτέλεση αυτής της στροφής απαιτείται ο υπολογισμός παραμέτρων και η διατήρηση αυτών των πρόσθετων στοιχείων σε ένα κατάλληλο τύπο δεδομένων, ο οποίος είναι ο Turn. Τα πεδία του είναι:

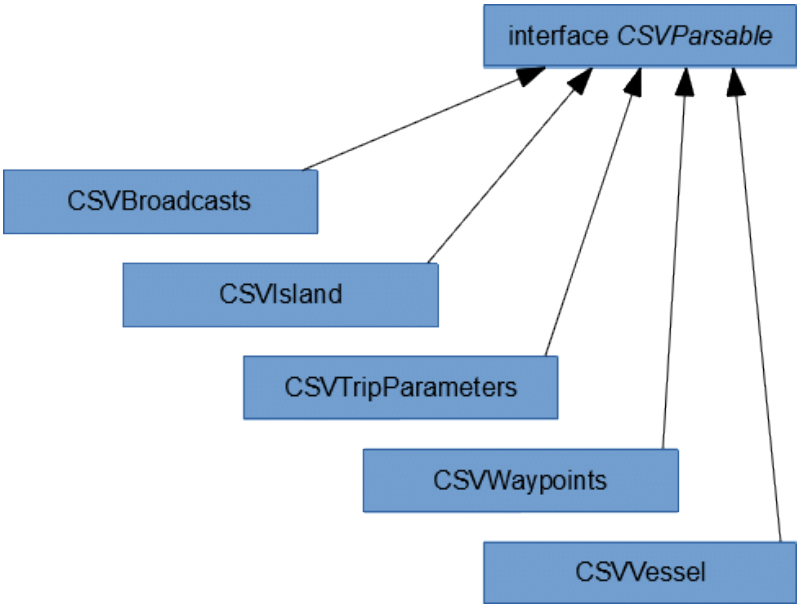
- turnDirection (TurnDirection, δεξιά / αριστερή στροφή)
- turnPoint (Coordinates, το σημείο περιστροφής, δηλαδή το κέντρο του κύκλου στον οποίο ανήκει το τόξο περιστροφής. Δεν πρόκειται για το Waypoint)
- turnRadius(double, ακτίνα περιστροφής σε μέτρα)
- turnAngle (double, επίκεντρη γωνία τόξου σε μοίρες. Ισούται με την γωνία αλλαγής κατεύθυνσης, ή το relative bearing της πλεύσης μεταξύ των δύο Legs της στροφής)
- turnEntryApex (Coordinates, το σημείο εισόδου στη στροφή, δηλαδή το μοναδικό κοινό σημείο του τόξου με το Leg που αφήνει πίσω του το πλοίο, για να πλεύσει στο επόμενο)
- speedCap (double, αριθμός που δείχνει το όριο ταχύτητας στη δεδομένη στροφή από το συγκεκριμένο πλοίο του Lnav. Χρησιμοποιείται από τον αλγόριθμο πλεύσης)

VesselType: Java enum με διάφορα είδη πλοίων όπως αυτά καθορίζονται με περιγραφή και κωδικό αριθμό αναφοράς, βάσει του IMO SOLAS, 46 U.S.C. 2101 or CFR 140.10. Το σύνολο δεν είναι πλήρες, αλλά περιέχει τα κυριότερα. Πέρα από τα enum constants τα οποία κατονομάζουν τα είδη σκαφών, η τάξη ορίζει και το πεδίο code (αντιπροσωπεύοντας τον κωδικό αριθμό αναφοράς του προτύπου), και αντιστοιχίζει μία μοναδική τιμή του σε κάθε constant. Ο CSVVessel parser μπορεί να διαβάσει από κάποιο αρχείο vessel.csv το είδος του πλοίου, είτε ως constant / περιγραφή ή ως κωδικό χάρη στη μέθοδό του, getVesselTypeForCode(int).

Waypoint: Τύπος που επεκτείνει το Coordinates και αντιπροσωπεύει καταχώρηση μίας μόνο γραμμής εντός αρχείου waypoints.csv. Πρόκειται δηλαδή για μία κορυφή της τεθλασμένης γραμμής που καταδεικνύει την πορεία ενός πλοίου Lnav.

Υποσύστημα parsers

Αρχείο πηγαίου κώδικα	**Οριζόμενος τύπος δεδομένων**
AppConfigurationCSVParser.java	AppConfigurationCSVParser
CSVBroadcasts.java	CSVBroadcasts
CSVIsland.java	CSVIsland
CSVParsable.java	CSVParsable
CSVTripParameters.java	CSVTripParameters
CSVVessel.java	CSVVessel
CSVWaypoints.java	CSVWaypoints



CSVParsable: Αφηρημένος τύπος, interface για την ακρίβεια, που λειτουργεί ως άμεσος και κοινός κληρονομικά πρόγονος για τους υπόλοιπους τύπους του υποσυστήματος parsers, με εξαίρεση τον AppConfigurationCSVParser. Οι parsers που υλοποιούν το interface, είναι υποχρεωμένοι να διαθέτουν σαφή ορισμό (concrete implementation) της μεθόδου getData(), η οποία, ως αποτέλεσμα ανάγνωσης των αρχείων εισαγωγής, παράγει κάποιο userInput τύπο ή προκαλεί κάποια CSVInputException εξαίρεση σε περίπτωση σφάλματος.

CSVIsland: Τύπος υπεύθυνος για την ανάγνωση .csv αρχείων περιγραφής νήσων (inputFiles\islands\). Αποτέλεσμα της ανάγνωσης ενός ορθώς ορισμένου νησιού είναι ένα αντικείμενο Island, με πρόσθετους ελέγχους από την τάξη εκείνη.

CSVWaypoints: Τύπος υπεύθυνος για την ανάγνωση αρχείων διαδρομής, waypoints.csv. Αποτέλεσμα της ανάγνωσης μίας ορθώς ορισμένης διαδρομής είναι ένα αντικείμενο Itinerary, με πρόσθετους ελέγχους από την τάξη εκείνη. Χρησιμοποιείται σε γνησίως εξομοιούμενα ταξίδια, legitimate trips, Lnavs.

CSVBroadcasts: Τύπος υπεύθυνος για την ανάγνωση αρχείων προκαθορισμένων εκπομπών, broadcasts.csv. Αποτέλεσμα της ανάγνωσης μίας ορθώς ορισμένης ακολουθίας εκπομπών του είδους είναι ένα αντικείμενο BroadcastSequence, με πρόσθετους ελέγχους από την τάξη εκείνη. Χρησιμοποιείται σε προκαθορισμένα ταξίδια, illicit trips.

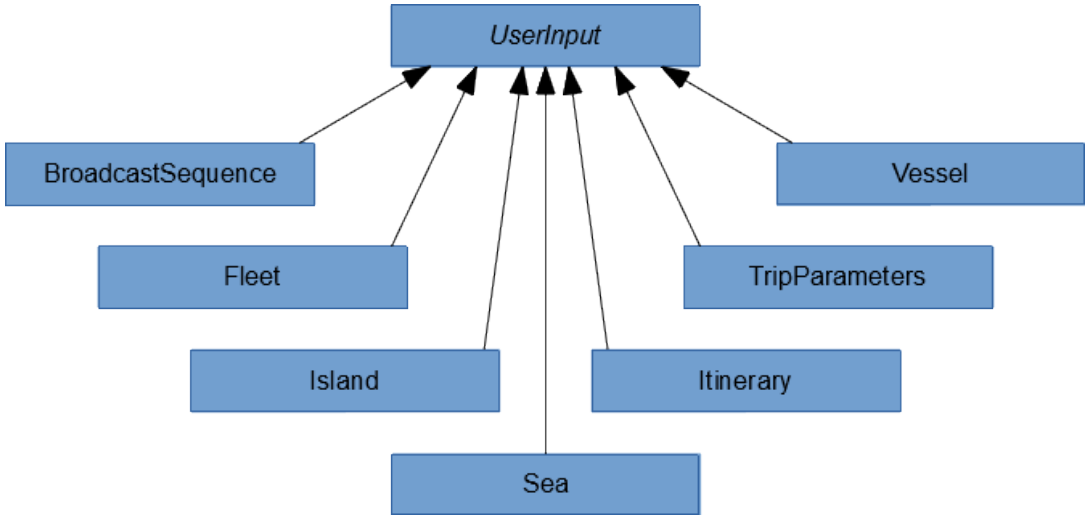
CSVVessel: Τύπος υπεύθυνος για την ανάγνωση αρχείων περιγραφής πλοίου, vessel.csv. Αποτέλεσμα της ανάγνωσης μιας έγκυρης περιγραφής πλοίου είναι ένα αντικείμενο Vessel.

CSVTripParameters: Τύπος υπεύθυνος για την ανάγνωση αρχείων παραμέτρων πλεύσης πλοίου, parameters.csv. Αποτέλεσμα της ανάγνωσης έγκυρων τέτοιων παραμέτρων είναι ένα αντικείμενο TripParameters.

AppConfigurationCSVParser: Τύπος υπεύθυνος για την ανάγνωση του αρχείου ρυθμίσεων της εφαρμογής. Δεν υλοποιεί το interface CSVParsable και ως εκ τούτου έχει απλώς λογική συσχέτιση με τους άλλους τύπους του υποσυστήματός του. Διαθέτει μέθοδο ανάγνωσης (που επίσης ονομάζεται getData()), μα αυτή δεν έχει τύπο επιστροφής (void return type) αφού δεν είναι δυνατό να δημιουργηθεί αντικείμενο ρυθμίσεων AppConfiguration, ενώ τα μέλη της τάξης αυτής είναι ήδη «ζωντανά» ως static. Αλλά και ανεξάρτητα ακόμη από αυτό, η τάξη AppConfiguration δεν είναι απόγονος του userInput. Επιπλέον, η μέθοδος δεν εγείρει exceptions, αλλά χρησιμοποιεί μηνύματα προς το χρήστη στη γραμμή εντολών, ως μέσο ανάδρασης της εισόδου του.

Υποσύστημα userInput

Αρχείο πηγαίου κώδικα	Οριζόμενος τύπος δεδομένων
BroadcastSequence.java	BroadcastSequence
Fleet.java	Fleet
Island.java	Island
Itinerary.java	Itinerary
Sea.java	Sea
TripParameters.java	TripParameters
UserInput.java	UserInput
Vessel.java	Vessel



UserInput: Αφηρημένη τάξη που λειτουργεί ως άμεσος και κοινός κληρονομικά πρόγονος για όλους τους υπόλοιπους τύπους του υποσυστήματος userInput, δίδοντάς τους ένα κοινό τύπο χάριν πολυμορφισμού.

BroadcastSequence: Τύπος που ενθυλακώνει λίστα από τα αντικείμενα Broadcast, αντληθέντα από αρχείο broadcasts.csv με χρήση του parser CSVBroadcasts. Ο constructor αυτής της τάξης είναι ιδιωτικού scope, δηλαδή μπορεί να κληθεί μόνο από τον κώδικα εντός της ίδιας τάξης. Τα αντικείμενα Broadcast δίδονται μαζί με κλήση σε μία μέθοδο τάξης δημόσιας πρόσβασης η οποία κατόπιν ελέγχων καλεί τον constructor ή εγείρει την σχετική εξαίρεση.

Island: Αντιπροσωπεύει νησί ως πολύγωνο. Περιέχει μεθόδους που μπορούν να διερευνήσουν την απλότητα της τεθλασμένης γραμμής που ορίζεται από την ακολουθία των κορυφών στο αρχείο εισόδου του, την επικάλυψή του με άλλα νησιά κ.λ.π. Ο constructor καλείται ιδιωτικά, από δημόσια μέθοδο της τάξης (public static) και όχι αντικειμένου, εξασφαλίζοντας τη δημιουργία του αντίστοιχου Island μόνο όταν υπάρχουν οι σχετικές προϋποθέσεις επί των ορισμάτων / κορυφών. Έκαστο αντιπροσωπεύει τις πληροφορίες που δίδονται σε αρχείο .csv νήσου του inputFiles\islands\. Τα αρχεία αυτά διαβάζει parser του τύπου CSVIsland.

Itinerary: Τύπος που ενθυλακώνει διασυνδεδεμένη λίστα από Waypoints, προερχόμενα από αρχείο waypoints.csv, ύστερα από ανάγνωσή του με χρήση του parser CSVWaypoints. Αντιπροσωπεύει τη διαδρομή κάποιου ταξιδιού ως ακολουθία από συντεταγμένες προς τις οποίες το πλοίο κατευθύνεται ευθύγραμμα και με τη σειρά που αυτές διατάσσονται. Το πρώτο σημείο είναι το σημείο εκκίνησης και το τελευταίο ο προορισμός. Τα ενδιάμεσα σημεία, εάν υπάρχουν, δείχνουν θέσεις όπου το πλοίο πραγματοποιεί αλλαγή κατεύθυνσης και πλέει στο επόμενο ευθύγραμμο τμήμα, Leg. Όμοια με άλλους τύπους του υποσυστήματος, ο constructor της τάξης είναι ιδιωτικής πρόσβασης, και για τη δημιουργία αντικειμένων Itinerary, δίδονται Waypoints ως ορίσματα με κλήση σε δημόσια μέθοδο τάξης. Η

μέθοδος πραγματοποιώντας τους κατάλληλους ελέγχους δημιουργεί το αντικείμενο ή εγείρει εξαιρέσεις.

TripParameters: Αντιπροσωπεύει το αντικείμενο που ως πεδία του έχει παραμέτρους που αφορούν ένα πλοίο. Αυτές δεν είναι σταθερές διαχρονικά, αλλά μπορεί να μεταβάλλονται σε διαφορετικά ταξίδια. Πρόκειται για φορτία (όπως καύσιμα, έρματα και ωφέλιμο) και αρχική ταχύτητα. Οι παράμετροι αντλούνται από αρχείο parameters.csv με χρήση του parser CSVTripParameters.

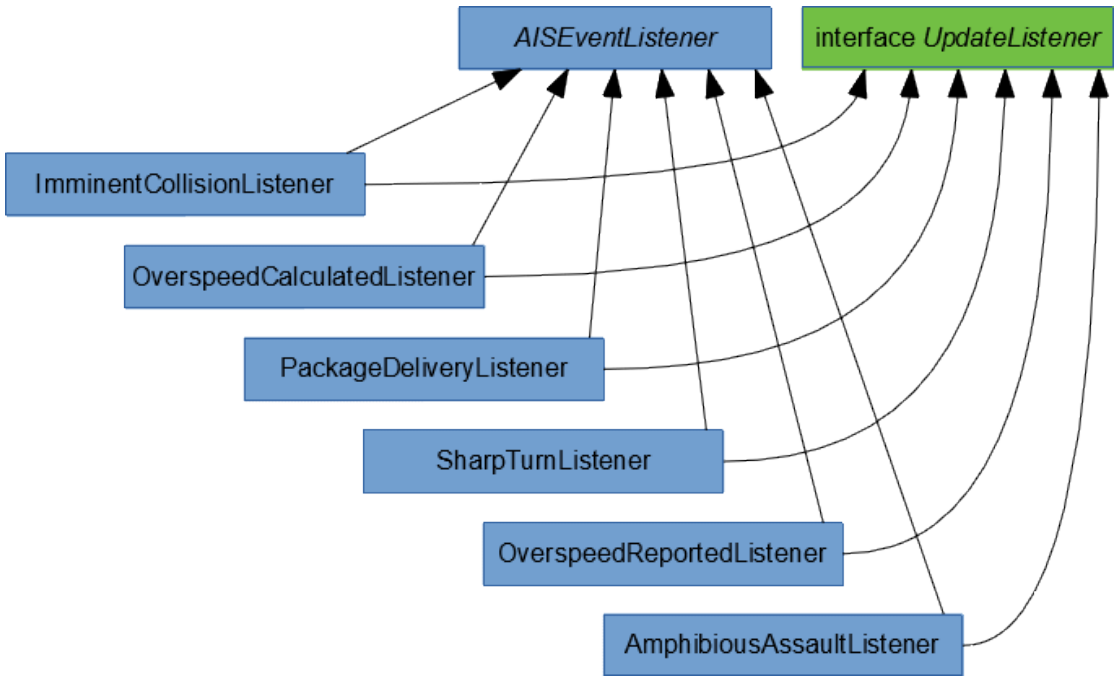
Vessel: Τάξη που περιγράφει πλοίο. Αντιπροσωπεύει τις πληροφορίες κάποιου αρχείου vessel.csv, κατόπιν ανάγνωσης με τον parser CSVVessel και επιτυχίας των σχετικών ελέγχων ενσωμάτωσης.

Fleet: Συλλογή όλων των Vessel αντικειμένων του σεναρίου εξομοίωσης. Χρησιμεύει σε έλεγχο για διπλότυπα πλοίων.

Sea: Συλλογή όλων των Island αντικειμένων του σεναρίου εξομοίωσης. Χρησιμεύει στον έλεγχο χωρικής επικάλυψης των υποψήφιων προς ενσωμάτωση νήσων, με τις ήδη ενσωματωμένες.

Υποσύστημα listeners

Αρχείο πηγαίου κώδικα	**Οριζόμενος τύπος δεδομένων**
AISEventListener.java	AISEventListener
ImminentCollisionListener.java	ImminentCollisionListener
OverspeedCalculatedListener.java	OverspeedCalculatedListener
OverspeedReportedListener.java	OverspeedReportedListener
PackageDeliveryListener.java	PackageDeliveryListener
SharpTurnListener.java	SharpTurnListener
AmphibiousAssaultListener.java	AmphibiousAssaultListener



Το υποσύστημα των listeners αποτελείται από τύπους δεδομένων με κατάλληλες μεθόδους που απλώς αναλαμβάνουν δράση σε απόκριση κάποιου HLE που έχει ανιχνευθεί. Είναι σημαντικό να ξεκαθαριστεί πως οι δομές αυτές δεν αποτελούν μέρος της μηχανής αναγνώρισης σύνθετων γεγονότων, ούτε περιγράφουν τα σύνθετα γεγονότα αυτά. Η εν λόγω περιγραφή HLEs σε όρους LLEs γίνεται από τα αντίστοιχά τους ερωτήματα EPL, που ορίζονται στη μέθοδο Application.main(). Υπάρχει αμφιμονοσήμαντη[36] αντιστοιχία μεταξύ ερωτημάτων και listeners. Οι listeners αυτοί καταγράφουν σε ανθρωπίνως αναγνώσιμη μορφή στα αντίστοιχα αρχεία κάθε ερωτήματος, μία καταχώρηση για κάθε εκδήλωση του γεγονότος αυτού. Οι καταχωρήσεις βρίσκονται σε αύξουσα χρονική σειρά και η μορφή της αναφοράς - καταχώρησης εξαρτάται από τη φύση του κάθε σύνθετου γεγονότος.

AISEventListener: Πρόκειται για τον γονικό και αφηρημένο τύπο που περιέχει πεδία τα οποία κληρονομούν όλοι οι Listeners (αρχείο καταγραφής εξόδου, χαρακτήρας αλλαγής γραμμής) αλλά κυριότερα, τη μέθοδο εγγραφής στο αρχείο.

ImminentCollisionListener: Αφορά το ερώτημα imminentCollisionEPL και γράφει τα αναγνωρισμένα HLEs στο αρχείο Imminent Collision Events.txt.

OverspeedCalculatedListener: Αφορά το ερώτημα overspeedCalculatedEPL και γράφει τα αναγνωρισμένα HLEs στο αρχείο Calculated Overspeed Events.txt.

OverspeedReportedListener: Αφορά το ερώτημα overspeedReportedEPL και γράφει τα αναγνωρισμένα HLEs στο αρχείο Reported Overspeed Events.txt.

PackageDeliveryListener: Αφορά το ερώτημα packageDeliveryEPL και γράφει τα αναγνωρισμένα HLEs στο αρχείο Package Delivery Events.txt.

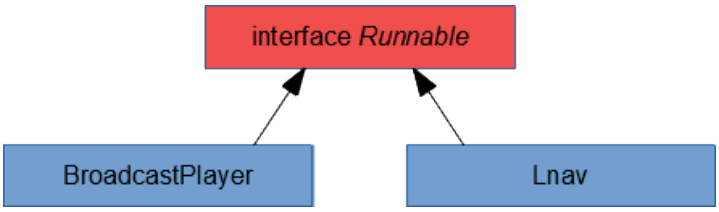
SharpTurnListener: Αφορά το ερώτημα sharpTurnEPL και γράφει τα αναγνωρισμένα HLEs στο αρχείο Sharp Turn Events.txt.

AmphibiousAssaultListener: Αφορά το ερώτημα amphibiousAssaultEPL και γράφει τα αναγνωρισμένα HLEs στο αρχείο Amphibious Assault Events.txt.

[36] «Ένα προς ένα» (1-1).

Υποσύστημα main

Αρχείο πηγαίου κώδικα	Οριζόμενος τύπος δεδομένων
Application.java	Application
BroadcastPlayer.java	BroadcastPlayer
Lnav.java	Lnav



Application: Ο τύπος που περιέχει το entry point της εφαρμογής, τη μέθοδο main(), η οποία αποτελεί τη «ραχοκοκαλιά» του προγράμματος. Ο κώδικάς της δίνει μία υψηλή επισκόπηση του συνόλου των εργασιών, αφού με κλήσεις αναθέτει τις επιμέρους ειδικότερες διαδικασίες σε μεθόδους του Application ή άλλων τύπων, οι οποίες σε κάποιες περιπτώσεις καλούν με τη σειρά τους άλλες. Η μέθοδος αρχικοποιεί την μηχανή Esper, και περιέχει τον κώδικα των ερωτημάτων EPL. Καλεί τις διαδικασίες ανάγνωσης των ρυθμίσεων και του σεναρίου εξομοίωσης από την είσοδο του χρήστη, δημιουργώντας το terrain και τα ταξίδια επί αυτού. Δημιουργεί το φάκελο εκτέλεσης και ξεκινά σε αυτόνομα threads τα ταξίδια, από όπου παράγονται οι εκπομπές AISBroadcast. Σε αυτό το datatype περιέχονται ακόμη βοηθητικές συναρτήσεις για ενδιάμεσους υπολογισμούς που απαιτούνται από κάποια ερωτήματα EPL, και η μέθοδος investigateException() που αναλαμβάνει να εμφανίσει στην γραμμή εντολών τα ακριβή αίτια μίας εξαίρεσης, αποδομώντας τη σταδιακά / αναδρομικά και αποκαλύπτοντας τις εσωτερικές εμφωλευμένες εξαιρέσεις, παρουσιάζοντας έτσι στον χρήση «από τα γενικά προς τα ειδικά» το σφάλμα του στα δεδομένα εισόδου.

Οι επόμενοι δύο τύποι διαθέτουν τις μεθόδους που παράγουν σε ζωντανό χρόνο εκπομπές AIS οι οποίες τρέχουν παράλληλα σε ξεχωριστό thread για κάθε διαφορετικό πλοίο / ταξίδι, ανεξαρτήτως του τρόπου παραγωγής των εκπομπών. Κάθε τύπος δέχεται ως ορίσματα στον constructor του τα ανάλογα αντικείμενα εισόδου (πλοίο, διαδρομή, …), αναφορές της αρχικοποιημένης μηχανής της Esper για αποστολή των γεγονότων προς επεξεργασία / αναγνώριση HLEs, και αναφορά του JPanel για γραφική απόδοση των στιγμάτων θέσης. Άλλα ορίσματα περιλαμβάνουν αναφορές προς αντικείμενα βοηθητικά, για μία υποτυπώδη επικοινωνία μεταξύ των νημάτων επεξεργασίας με στόχο το συγχρονισμό τους, όπως έχει περιγραφεί. Τελικώς, υπάρχει αναφορά προς το αρχείο AIS Broadcasts.txt εντός του τρέχοντος φακέλου εκτέλεσης για καταγραφή των παραγόμενων LLEs.

BroadcastPlayer: Πρόκειται για αναπαραγωγό των προκαθορισμένων θέσεων, όπως προκύπτουν από τα BroadcastSequence και Vessel που δέχεται ως ορίσματα. Αντιπροσωπεύει τα illicit, scripted ταξίδια.

Lnav: Τύπος δεδομένων που περιέχει αλγορίθμους εξομοίωσης πλεύσης πλοίου και παραγωγής εκπομπών AIS βάσει των υπολογισμών τους. Αντιπροσωπεύει τα γνησίως εξομοιούμενα, legitimate, ταξίδια όπως αυτά περιγράφονται από τις αναφορές των αντίστοιχων Vessel, Itinerary και TripParameters αντικειμένων που μεταβιβάζονται σε αυτόν.

Σε επέκταση της περιγραφής των τάξεων της εφαρμογής, παρουσιάζονται στο Παράρτημα Α τα μέλη αυτών, μέσω ιστοσελίδων Javadocs. Σε αυτές συμπεριλαμβάνονται μέθοδοι / πεδία ιδιωτικής πρόσβασης.

Κεφάλαιο 4: Παραγωγή γεγονότων AIS

Περιεχόμενα κεφαλαίου

4.1: Τα πεδία αρχείων εισαγωγής και η ενσωμάτωσή τους

Σε αυτή την ενότητα γίνεται η τυπική παρουσίαση της μορφής των αρχείων εισαγωγής, αλλά και μία υψηλού επιπέδου ανάλυση των αλγορίθμων ανάγνωσής τους, οι οποίοι υλοποιούνται από τους αντίστοιχους parsers. Ακόμη, περιγράφονται ορισμένα πεδία των οποίων η χρήση δεν κατέστη προφανής από την μέχρι στιγμής ανάλυση. Απώτερος στόχος των ενεργειών αυτών είναι η ελαχιστοποίηση των σφαλμάτων από πλευράς του χρήστη.

Στα αρχεία .csv που συνοδεύουν την εφαρμογή, η πρώτη γραμμή δεν περιέχει δεδομένα εισαγωγής, αλλά ανθρωπίνως αναγνώσιμες κεφαλίδες στηλών των πεδίων. Χάρη σε αυτές, γίνεται σαφέστερη η φύση των απαιτούμενων δεδομένων, η μορφή τους, ο τύπος στον οποίο θα μετατραπούν, καθώς και το επιτρεπτό εύρος τιμών τους. Αποσαφηνίζεται η σειρά με την οποία πρέπει να εισάγονται τα πεδία σε κάθε γραμμή των αρχείων, η οποία είναι ίδια με αυτή των αντίστοιχων κεφαλίδων τους. Η σειρά αυτή αντανακλά και τη σειρά με την οποία ο parser κάποιου αρχείου προσδοκά τις πληροφορίες για να τις μετατρέψει στο αντίστοιχο datatype και να τις αποθηκεύσει στην κατάλληλη μεταβλητή, ξεκινώντας τη διαδικασία ενσωμάτωσης των δεδομένων. Η πρώτη γραμμή κάθε τέτοιου αρχείου δεν προκαλεί σφάλμα ανάγνωσης, διότι οι parsers την αγνοούν[37]. Για αυτό το λόγο, τυχόν νέα αρχεία εισαγωγής, ακόμη και αν δεν περιέχουν κεφαλίδες, θα πρέπει να ξεκινούν από τη δεύτερη γραμμή.

Όπως ισχύει και με τις υπόλοιπες γραμμές τέτοιων αρχείων, οι κεφαλίδες είναι χωρισμένες μεταξύ τους με κόμμα («,»). Κάθε επικεφαλίδα έχει τη μορφή: key(dataType:description), όπου:

- key: Το όνομα της μεταβλητής. Τα ονόματα είναι κατατοπιστικά ώστε να μην προκαλούν σύγχυση σε ότι αφορά τη φύση της ζητούμενης πληροφορίας.
- dataType: Ο τύπος δεδομένων στον οποίο θα μετατραπεί το πεδίο κειμένου. Η ενσωμάτωση γίνεται από μεθόδους τάξης των αντίστοιχων τύπων. Σε περίπτωση σφάλματος εγείρονται οι αντίστοιχες εξαιρέσεις. Οι τύποι είναι Java primitives ή τύποι ορισμένοι από την sailAway.
- description: Εμφανίζεται σε κάποιες μόνο κεφαλίδες, για να περιγράψει περεταίρω τη ζητούμενη εισαγωγή. Ανάλογα με τη μεταβλητή, δείχνει είτε το εύρος αποδεκτών τιμών ή τη μονάδα μέτρησης[38]. Ο χαρακτήρας «:» εμφανίζεται μόνο όταν υπάρχει description, ως διαχωριστικό μεταξύ αυτού και του dataType.

Τα αρχεία εισαγωγής, με τη γραμμή κεφαλίδων τους (πρώτη):

configuration.csv:
turbulence(double:[0-1]), centripetalForce(double:Newtons), guiMode(GuiMode), trailLength(int:numberOfLastPositions), laneWidth(int:m), broadcastInterval(int:milliseconds), debugStatus(boolean), immobileSpeedThreshold(double:m/s), globalSpeedLimit(double:m/s)

vessel.csv:
mmsi(long), flag(Flag), name(String), type(VesselType), length(double:m), width(double:m), lightshipWeight(double:kg), maximumSpeed(double:m/s), minimumSpeed(double:m/s), acceleration(double:m/s^2), deceleration(double:m/s^2)

parameters.csv:
initialSpeed(double:m/s), ballast(double:kg), fuel(double:kg), payload(double:kg)

broadcasts.csv:
longitude(double:m), latitude(double:m), heading(Azimuth), speed(double:m/s)

[37] Στους parsers με εξαίρεση τον AppConfigurationCSVParser, η μέθοδος ανάγνωσης και ενσωμάτωσης είναι αυτή με signature UserInput getData(Boolean omitFirstLine) throws CSVInputException. Η getData() καλείται με παράμετρο “true”, για παράλειψη της πρώτης γραμμής. Σε αντίθετη περίπτωση, τα αρχεία εισαγωγής μπορούν να περιέχουν αναγνώσιμο περιεχόμενο από την πρώτη γραμμή. Ανάλογη μετατροπή μπορεί να γίνει και στην κλήση του parser του αρχείου καθολικών ρυθμίσεων.

[38] Η εφαρμογή κάνει χρήση αποκλειστικά των μονάδων του S.I., θεμελιωδών αλλά και παραγώγων. Δεν χρησιμοποιούνται πολλαπλάσια ή υποδιαιρέσεις των μονάδων, με εξαίρεση την περίοδο εκπομπής (broadcastInterval) που μετράται σε milliseconds (χιλιοστά του δευτερολέπτου, sec × 10^{-3}), για λόγους ακριβείας.

waypoints.csv / island files:
longitude(double:m), latitude(double:m)

Το αρχείο configuration.csv, τα αρχεία parameters.csv και vessel.csv είναι αρχεία εισαγωγής που έχουν μόνο μία γραμμή πληροφοριών, ή ακριβέστερα αρχεία των οποίων ο parser σταματά αφού διαβάσει μία μόνο γραμμή (τη δεύτερη, αφού η πρώτη αγνοείται). Με άλλα λόγια, τα αρχεία αυτά αποτυπώνουν έκαστο ένα διάνυσμα μίας γραμμής και κάποιων στηλών. Αυτά τα αρχεία στα πλαίσια της εφαρμογής λέγονται και single line, ή αρχεία εισαγωγής μίας γραμμής. Αντιθέτως, τα υπόλοιπα τρία είδη αρχείων εισαγωγής, δηλαδή τα εκάστοτε αρχεία νησιών, τα αρχεία waypoints.csv και broadcasts.csv, είναι αρχεία πολλαπλών γραμμών, που μεταφέρουν πίνακες, ή ακριβέστερα: διανύσματα διανυσμάτων. Σε αυτά, οι αντίστοιχοι parsers διαθέτουν επαναληπτικές δομές οι οποίες προχωρούν στην ανάγνωση της επόμενης γραμμής, εφ' όσον αυτή υπάρχει, και όσο δεν συντρέχουν κάποιοι λόγοι που μπορούν να τερματίσουν πρόωρα την ανάγνωσή του. Τα αρχεία αυτά είναι γνωστά και ως multi line, ή αρχεία εισαγωγής πολλαπλών γραμμών.

Με σκοπό την κατανόηση του χειρισμού ενός αρχείου εισαγωγής, εξηγείται συνοπτικά ο αλγόριθμος ανάγνωσης σε αριθμημένα βήματα. Έτσι, κάποιος parser:

1. Αγνοεί την πρώτη γραμμή.
2. Διαβάζει την επόμενη γραμμή και δημιουργεί ένα αντίγραφό της στη μνήμη[39].
3. Εξαλείφει από το αντίγραφο της γραμμής τους χαρακτήρες κενού (whitespace).
4. Αν πρόκειται για parser multi line αρχείου και επίσης εάν το αντίγραφο της γραμμής είναι κενό, εγκαταλείπει την ανάγνωση και πηγαίνει στο βήμα 10.
5. Αν πρόκειται για parser multi line αρχείου και επίσης εάν ο πρώτος χαρακτήρας του αντιγράφου της γραμμής είναι «#», παραλείπει την τρέχουσα γραμμή. Αν υπάρχει άλλη γραμμή στο αρχείο, πηγαίνει στο βήμα 2, αλλιώς στο βήμα 10.
6. Χωρίζει το αλφαριθμητικό του αντιγράφου στα επιμέρους αλφαριθμητικά που οριοθετούνται από τους χαρακτήρες «,», την αρχή και το τέλος της γραμμής. Στη συνέχεια, αποθηκεύει τα αλφαριθμητικά αυτά σε array, με αύξον index σειράς εμφάνισης στη γραμμή, ξεκινώντας από το 0.
7. Αν πρόκειται για parser multi line αρχείου και επίσης εάν τα επιμέρους αλφαριθμητικά είναι λιγότερα σε πλήθος από τα ζητούμενα, εμφανίζει μήνυμα σφάλματος και παραλείπει την τρέχουσα γραμμή. Αν υπάρχει άλλη γραμμή στο αρχείο, πηγαίνει στο βήμα 2, αλλιώς στο βήμα 10.
8. Διαβάζει τα k πρώτα επιμέρους αλφαριθμητικά του array (όπου k, ο αριθμός των ζητούμενων πεδίων), τα μετατρέπει στους κατάλληλους τύπους και τα αποθηκεύει σε μεταβλητές.
9. Αν πρόκειται για parser multi line αρχείου, δημιουργείται το αντικείμενο που προκύπτει από τις μεταβλητές του βήματος 8 με κλήση στον constructor του αντικειμένου που αντιπροσωπεύει κάθε γραμμή (Waypoint/Broadcast/Coordinates). Το αντικείμενο αυτό προστίθεται σε διασυνδεδεμένη λίστα. Αν υπάρχει άλλη γραμμή στο αρχείο, πηγαίνει στο βήμα 2, αλλιώς στο βήμα 10.
10. Επιστρέφει αντικείμενο του τύπου απογόνου του UserInput, αναλόγως του αρχείου / parser:
    a. Αν πρόκειται για parser multi line αρχείου, με κλήση στην μέθοδο τάξης τύπου επιστροφής (Itinerary/BroadcastSequence/Island), περνώντας σε αυτή ως παράμετρο τη διασυνδεδεμένη λίστα του βήματος 9.
    b. Αν πρόκειται για parser single line αρχείου, με άμεση κλήση στον constructor του τύπου (TripParameters/Vessel), περνώντας σε αυτόν ως παραμέτρους τις μεταβλητές του βήματος 8.

Από τον παραπάνω αλγόριθμο, καθίσταται σαφές πως:

- Στα multi line αρχεία, μία λάθος γραμμή δεν σταματά την ανάγνωση, αλλά αναγκάζει τον parser να παραλείψει τις πληροφορίες της συγκεκριμένης γραμμής από το dataset που αντιπροσωπεύει το αρχείο.

[39] Οι parsers δεν πραγματοποιούν αλλαγές (εγγραφές) στα αρχεία προέλευσης. Η όποια μετέπειτα επεξεργασία των γραμμών, πραγματοποιείται στο αντίγραφό που δημιουργούν.

- Για σχολιασμό μίας γραμμής σε multi line αρχεία μπορεί να χρησιμοποιηθεί ο χαρακτήρας «#», αρκεί μεταξύ αυτού και της αρχής της γραμμής να βρίσκονται μόνο χαρακτήρες κενού, κανένας ή περισσότεροι.
- Σχόλια πολλαπλών γραμμών μπορούν να υπάρξουν στο τέλος των multi line αρχείων, ύστερα από μία κενή γραμμή.
- Σχόλια πολλαπλών γραμμών μπορούν επίσης να υπάρξουν στο τέλος των single line αρχείων, απλώς με την τοποθέτησή τους στην $3^{η}$ γραμμή ή μετά αυτής.
- Εάν υπάρχουν περισσότερα πεδία από τα ζητούμενα κάθε γραμμής, αγνοούνται. Αυτός είναι ένας τρόπος προσθήκης σχολιασμού σε μία γραμμής που περιέχει δεδομένα, χωρίς να επηρεάζει την αναγνωσιμότητά τους από τον parser.

4.1.1: Επιλεκτική επεξήγηση πεδίων εισαγωγής

Ακολούθως, εξηγείται η φύση του πληροφοριακού περιεχομένου των αρχείων εισαγωγής, επιλεκτικά.

Οι ρυθμίσεις του αρχείου configuration.csv καλύπτουν όλο το πλάτος της εφαρμογής (app-wide) και δεν αφορούν π.χ., μόνο μία διαδρομή πλοίου:

- turbulence: Τυρβώδης ροή. Πρόκειται για μέρος ως ποσοστό (λαμβάνει τιμές από 0 έως 1) μίας ποσότητας που καθορίζει διακύμανση. Αυτή χρησιμοποιείται από αλγόριθμο του Lnav για την προσομοίωση των συνεπειών των αναταράξεων / ανέμων / κυματισμού στην πορεία των πλοίων. Μεγαλύτερες τιμές, προκαλούν σημαντικότερες αποκλίσεις πορείας.
- centripetalForce: Κεντρομόλος δύναμη. Χρησιμοποιείται από αλγόριθμο του Lnav, για τον υπολογισμό της μέγιστης ταχύτητας με την οποία ένα πλοίο δεδομένης συνολικής μάζας δύναται να στρίψει σε καμπή δεδομένης ακτίνας.
- trailLength: Ο αριθμός των πιο πρόσφατων θέσεων ενός πλοίου, συμπεριλαμβανομένης της τρέχουσας. Κάθε πλοίο στο χάρτη εμφανίζεται με ένα εικονίδιο, το οποίο κινούμενο αφήνει πίσω του ένα ίχνος που δεν είναι παρά μία τεθλασμένη γραμμή. Αυτή αποτελείται από τα ευθύγραμμα τμήματα που έκαστο ενώνουν δύο πρόσφατα διαδοχικά σημεία θέσης του ίδιου πλοίου. Προφανώς, όσο μεγαλύτερη η τιμή της μεταβλητής, τόσο μακρύτερο και το ίχνος στο χάρτη.
- laneWidth: Το πλάτος του διαδρόμου πλεύσης. Πρόκειται για το νοητό διάδρομο στον διαμήκη άξονα του οποίου βρίσκεται κάθε Leg διαδρομής, ισαπέχοντας από τα όριά του διαδρόμου του. Το ήμισυ του lane width δείχνει τη μέγιστη απόσταση κατά την οποία ένα πλοίο επιτρέπεται να παρεκκλίνει από το τρέχον Leg της πορείας του.
- broadcastInterval: Ο χρόνος σε milliseconds που μεσολαβεί μεταξύ δύο διαδοχικών εκπομπών κάθε νήματος επεξεργασίας ταξιδιού, προδιαγεγραμμένου ή εξομοιούμενου.
- debugStatus: Πρόσθετη πληροφόρηση αποσφαλμάτωσης. Όταν η τιμή της ορίζεται ως αληθής, η εφαρμογή παράγει αναλυτικότερη έξοδο στην γραμμή εντολών για διάφορα ενδογενή θέματα σχετικά με τη λειτουργία της, όπως την πρόοδο της ενσωμάτωσης του σεναρίου εξομοίωσης και ιδιαίτερα τη δημιουργία και τοποθέτηση νησιών. Ακόμη, στην περίπτωση που είναι ενεργοποιημένη η γραφική έξοδος, εμφανίζονται τα όρια των διαφόρων περιοχών σχεδίασης όπως το περιγεγραμμένο ορθογώνιο των νησιών, το περιθώριο έξω από αυτό κ.α..
- immobileSpeedThreshold: Το όριο ταχύτητας κάτω από το οποίο κάθε πλοίο απεικονίζεται με κύκλο στο χάρτη, αντί του τριγώνου που χρησιμοποιείται τυπικά. Έτσι υπονοείται αδράνεια ή αμελητέα ταχύτητα, όπως συμβαίνει για παράδειγμα κατά τη διάρκεια ελιγμών πρόσδεσης σε λιμένα.
- globalSpeedLimit: Καθολικό όριο ταχύτητας στην θάλασσα του σεναρίου εξομοίωσης. Δεν πρόκειται για όριο το οποίο αντανακλά τις φυσικές / τεχνικές δυνατότητες των σκαφών. Πρόκειται για κάποιο υποθετικό θεσμικό περιορισμό εκ μέρους της αρμόδιας Αρχής και χρησιμεύει σε ερωτήματα EPL για αναγνώριση παραβατικής συμπεριφοράς ναυτιλλομένων.

Οι πληροφορίες σε κάθε αρχείο vessel.csv αφορούν αποκλειστικά το πλοίο στο οποίο αναφέρονται:

- mmsi: Maritime Mobile Service Identity. Αριθμός 9 ψηφίων μοναδικός για κάθε πλοίο παγκοσμίως. Ο αριθμός αποτελεί κωδικοποίηση πληροφοριών αφού είναι δομημένος ώστε συγκεκριμένα ψηφία να αφορούν τη φύση του φέροντος σκάφους, της χώρας στην οποία υπάγεται κ.α.. Πέραν των πλοίων, αριθμοί MMSI αποδίδονται και σε άλλες ναυτιλιακές οντότητες, όπως παράκτιοι ραδιοσταθμοί.
- type: Ο τύπος (το είδος) του πλοίου. Το αρχείο πηγαίου κώδικα του Java enum που δέχεται το πεδίο, sailAway\application\dataTypes\lowLevelTypes\src\VesselType.java, παρέχει σχολιασμό εξηγώντας εκτενώς το ρόλο του πεδίου και το εύρος τιμών που γίνεται δεκτό ως εισαγωγή.
- lightshipWeight: Η μάζα του πλοίου χωρίς φορτία. Συμπεριλαμβάνονται οι μάζες του λειτουργικού εξοπλισμού, μονίμου έρματος, κινητήρων, συστημάτων ελέγχου / επικοινωνιών, ηλεκτρικών / υδραυλικών συστημάτων. Από τον υπολογισμό του μεγέθους αυτού εξαιρούνται μάζες καυσίμων, πληρώματος, επιβατών, αυτοκινήτων, εμπορευμάτων, μεταβλητού έρματος, και αναλωσίμων.
- maximumSpeed: Η μέγιστη ταχύτητα που μπορεί να αναπτύξει το πλοίο υπό ιδανικές συνθήκες, η τελική ταχύτητά του.
- minimumSpeed: Η ελάχιστη ταχύτητα του πλοίου με την οποία διατηρείται ένας ικανοποιητικός βαθμός κατευθυντικότητας και ελεγξιμότητας.
- acceleration: Ο μέγιστος ρυθμός επιτάχυνσης του πλοίου.
- deceleration: Ο μέγιστος ρυθμός επιβράδυνσης του πλοίου.

Οι τιμές των αρχείων parameters.csv που συνοδεύουν γνησίως εξομοιούμενα ταξίδια:

- initialSpeed: Η αρχική ταχύτητα. Με αρχική ταχύτητα μεγαλύτερη του μηδενός, δίδεται πρακτικά η δυνατότητα εξομοίωσης ενός ταξιδιού από στιγμή μεταγενέστερη του απόπλου του.
- ballast: Η μάζα του μεταβλητού έρματος[40]. Συνήθως χρησιμοποιείται θαλασσινό νερό ως έρμα, ώστε να υπάρχει η δυνατότητα άμεσης μεταβολής της ποσότητάς του στις ειδικές δεξαμενές ερματισμού, με χρήση αντλιών.
- fuel: Η συνολική μάζα των καυσίμων, που βρίσκονται στις δεξαμενές καυσίμων του πλοίου προορίζονται για κίνηση του ιδίου, όχι καυσίμων που τυχόν μεταφέρονται ως εμπορεύματα / αγαθά για χρήση τρίτων.
- payload: Το ωφέλιμο φορτίο. Στην περίπτωση ενός Ro-Ro Passenger πλοίου, είναι η συνολική μάζα αυτοκινήτων, επιβατών, αποσκευών και εμπορευμάτων.

Η συνολική μάζα του πλοίου που είναι γνωστή και ως εκτόπισμα[41], ορίζεται ως το άθροισμα των lightship weight και deadweight. Το τελευταίο είναι το άθροισμα καυσίμων, μεταβλητού έρματος και ωφέλιμου φορτίου. Για τη συνολική μάζα θα πρέπει να προστεθεί και η μάζα κάποιων αναλωσίμων, όπως λιπαντικά σε χρήση, προμήθεια πόσιμου νερού, τροφίμων και άλλων. Η μάζα αυτή θεωρήθηκε από την παρούσα ανάλυση αμελητέα, ιδιαίτερα ως ποσοστό της συνολικής, και παραλήφθηκε.

[40] Το έρμα χρησιμοποιείται για αύξηση της ευσταθείας του πλοίου. Τοποθετούμενο καταλλήλως, μετατοπίζει χαμηλά το κέντρο μάζας του σκάφους και αποτρέπει ροπές στρέψης (π.χ. λόγω κακής φόρτωσης). Πολλά πλοία έχουν μόνιμο έρμα (μόλυβδος / χυτοσίδηρος) και επίσης μεταβλητό (νερό στα διπύθμενα).

[41] … δηλαδή εκτόπισμα ύδατος. Γενικά ισχύει πως η μάζα του υγρού που εκτοπίζει ένα εμβαπτιζόμενο σε αυτό σώμα, ισούται με τη μάζα του σώματος.

4.2: Οι διαδικασίες επαλήθευσης και ενσωμάτωσης νησιών

Στο σημείο αυτό παρουσιάζεται λεπτομερέστερα η διαδικασία ενσωμάτωσης του αρχιπελάγους στο σενάριο εξομοίωσης. Από όλες τις διαδικασίες επαλήθευσης εισαγωγών του χρήστη και ενσωμάτωσης των αντικειμένων που προκύπτουν, επιλέχθηκε το υποσύνολο εκείνο των διαδικασιών που αφορούν τα νησιά. Ο λόγος είναι διττός: Οι σχετικές με τη δημιουργία εδάφους διαδικασίες αφ' ενός αποτελούν ένα πολύ μεγάλο μέρος της συνόλου τις επεξεργασίας δεδομένων εισόδου, και αφ' ετέρου οι υπόλοιπες ενσωματώσεις δεν χρησιμοποιούν επί της ουσίας τεχνικές που δεν αξιοποιούνται με κάποιο τρόπο από την εν λόγω διαδικασία. Η δημιουργία του εδάφους απαιτεί τη συνεργασία μεθόδων τύπων από διάφορα υποσυστήματα και είναι η πιο πολύπλοκη της ενσωμάτωσης εισόδου χρήστη, ίσως και της εφαρμογής συνολικά.

4.2.1: Parsing και έλεγχοι αριθμητικών πεδίων

Αρχικά, η Application.main() δημιουργεί ένα αντικείμενο τύπου Sea για την αποθήκευση όλων των νησιών που θα επιτύχουν τόσο στη δημιουργία τους αλλά και στην τοποθέτησή τους στο αρχιπέλαγος. Στη συνέχεια, διαβάζει όλα τα top level περιεχόμενα του φακέλου sailAway\inputFiles\islands, αγνοώντας τους φακέλους. Σειριακά, για κάθε αρχείο στο φάκελο νησιών, δημιουργεί ένα αντικείμενο CSVIsland περνώντας στον constructor του ως όρισμα μία αναφορά του εκάστοτε αρχείου, η οποία αποθηκεύεται σε πεδίο της τάξης εκείνης, διατηρώντας με τον τρόπο αυτό σύνδεση / αντιστοίχηση μεταξύ parser και αρχείου. Στη συνέχεια καλεί τη μέθοδο getData() του parser η οποία προχωρά στην ανάγνωση της περιγραφής του νησιού. Η λειτουργία του αλγορίθμου ανάγνωσης που διαθέτουν οι parsers έχει ήδη καλυφθεί, εδώ αναλύεται συνοπτικότερα και κυρίως με εστίαση στην ανάγνωση αρχείου νήσου από τον CSVIsland, που είναι ένας single line parser.

Η μέθοδος CSVIsland.getData(), αγνοεί την πρώτη γραμμή του αρχείου νήσου και όσο υπάρχουν άλλες γραμμές σε αυτό, τότε για κάθε μία, σειριακά:

1. Διαβάζει την επόμενη και δημιουργεί ένα αντίγραφό της στη μνήμη.
2. Εξαλείφει τους χαρακτήρες κενού από το αντίγραφο.
3. Εάν ο πρώτος χαρακτήρας του αντιγράφου της γραμμής είναι «#», παραλείπει την συγκεκριμένη γραμμή.
4. Χωρίζει το αλφαριθμητικό του αντιγράφου στα επιμέρους αλφαριθμητικά που οριοθετούνται από τους χαρακτήρες «,» αποθηκεύοντάς τα σε πίνακα, με index 0 για το πρώτο, 1 για το δεύτερο, κ.ο.κ.
5. Εάν τα επιμέρους αλφαριθμητικά είναι λιγότερα από δύο, εμφανίζει μήνυμα σφάλματος και παραλείπει την τρέχουσα γραμμή.
6. Διαβάζει το αλφαριθμητικό του array με index 0, το μετατρέπει σε δεκαδικό διπλής ακριβείας και αποθηκεύει αυτόν στη μεταβλητή longitude.
7. Διαβάζει το αλφαριθμητικό του array με index 1, το μετατρέπει σε δεκαδικό διπλής ακριβείας και αποθηκεύει αυτόν στη μεταβλητή latitude.
8. Από τις μεταβλητές γεωγραφικού μήκους και πλάτους δημιουργεί ένα αντικείμενο Coordinates με κλήση στον constructor: Coordinates(double longitude, double latitude).
9. Το αντικείμενο Coordinates του προηγούμενου βήματος προστίθεται σε διασυνδεδεμένη λίστα ονομαζόμενη apexes (διασυνδεδεμένες λίστες, όπως java.util.LinkedList<T> instances, αποτελούν συνήθη τρόπο υλοποίησης δομών όπως ουρές και στοίβες).

Με το πέρας της ανάγνωσης των γραμμών έχει ολοκληρωθεί η λίστα apexes, στον πρώτο κόμβο της οποίας βρίσκεται το Coordinates αντικείμενο που δημιουργήθηκε πρώτο (δηλαδή η πρώτη έγκυρη καταχώρηση του αρχείου), στον δεύτερο το επόμενο Coordinates κ.ο.κ.

Η μέθοδος .getData() διαβάζει ύστερα το όνομα του αρχείου και με χειρισμούς στο αλφαριθμητικό εκείνο, όπως απόρριψη της επέκτασης ".csv", εξάγει το όνομα του νησιού και επιστρέφει αντικείμενο τύπου Island, μέσω κλήσης σε άλλη μέθοδο, περνώντας σε αυτή ως παράμετρο τις κορυφές του πολυγώνου διατεταγμένες (λίστα apexes) αλλά και το αλφαριθμητικό που αντιπροσωπεύει το όνομα του νησιού.

4.2.2: Έλεγχοι πολυγώνου και δημιουργία αντικειμένου

Υπενθυμίζεται πως τα αρχεία .csv περιγραφής νησιών ονοματίζονται με το όνομα του νησιού που περιγράφουν, ακολουθούμενο από την επέκταση του αρχείου τιμών χωρισμένων με κόμμα. Επιπλέον, και όπως έχει ήδη υποστηριχθεί, δεν είναι δυνατή η κλήση του constructor της τάξης Island έξω από το μπλοκ κώδικα ορισμού της, καθώς εκείνος έχει περιορισμένη, ιδιωτική πρόσβαση. Η δημιουργία αντικειμένων γίνεται από τη δημοσίου πρόσβασης μέθοδο της τάξης (και όχι αντικειμένου) Island, με signature public static Island createIsland(LinkedList<Coordinates> apexes, String name) throws CSVIslandException. Αυτή η μέθοδος δύναται να επιστρέψει το επιθυμητό αντικείμενο νησιού υπό συνθήκες, έχοντας δηλαδή πραγματοποιήσει πρώτα πρόσθετους ελέγχους που εστιάζονται στη φύση του νησιού ως πολύγωνο δύο διαστάσεων. Ανάλογα με την έκβαση των ελέγχων, η μέθοδος Island.createIsland() επιστρέφει αντικείμενο της τάξης της καλώντας η ίδια τον constructor, ο οποίος έχει signature: private Island(LinkedList<Shore> shores, String name), περνώντας σε αυτόν τα ίδια ορίσματα που έλαβε και έλεγξε από την καλούσα .getData(), ή εγείρει τις εξαιρέσεις που σχετίζονται με τις αποτυχίες που εντοπίζονται από τους εξειδικευμένους ελέγχους δημιουργίας νήσου.

Ίσως κάποιος αναρωτηθεί για το λόγο που ο parser δεν αναλαμβάνει όλους τους σχετικούς ελέγχους χάρη απλότητος, για κάποιο νησί εν προκειμένω. Από τεχνικής πλευράς κάτι τέτοιο θα ήταν σίγουρα εφικτό, με την CSVIsland.getData() να διεξάγει τους ελέγχους, τόσο εκείνους που περιλαμβάνει ήδη, αλλά και εκείνους που πραγματοποιούνται στη συνέχεια από την Island.createIsland(). Με τον τρόπο αυτό θα μπορούσε η ίδια να καλέσει τον constructor της Island, ο οποίος θα ήταν δημοσίου πρόσβασης, καταργώντας την ανάγκη υλοποίησης «ενδιάμεσων» μεθόδων όπως της Island.createIsland(). Ωστόσο, ο εργασιακός φόρτος είναι κατανεμημένος με τον υφιστάμενο τρόπο μεταξύ των τύπων δεδομένων, από συνειδητή επιλογή και χωρίς η τεχνική αυτή να αποτελεί εφεύρεση του συγγραφέα. Αντιθέτως, επιβάλλεται από τις ενδεδειγμένες πρακτικές του αντικειμενοστραφούς μοντέλου ανάπτυξης εφαρμογών. Στα πλαίσιά τους, οι ιδιότητες και οι συμπεριφορές κάθε οντότητας ορίζονται εντός των «συνόρων» της, ως πεδία και μέθοδοι αντιστοίχως, αποτελώντας μέλη της. Είναι προς όφελος του προγραμματιστή η ανάθεση των διαφόρων δραστηριοτήτων στις οντότητες με τις οποίες σχετίζονται. Έτσι η εφαρμογή αποκτά τμηματικότητα και επεκτασιμότητα. Στην περίπτωση της παρούσας εφαρμογής, οντότητες αποτελούν τα νησιά αλλά και οι parsers που διαβάζουν της πληροφορίες τους από αρχεία. Η ανάγνωση του αρχείου είναι αρμοδιότητα του parser, όπως και οι έλεγχοι για την ορθότητα της μορφής του διανύσματος σε αυτά. Έτσι, ο εν λόγω τύπος δεδομένων ελέγχει για το πλήθος των πεδίων, την μη ύπαρξη αλφαριθμητικών εκεί όπου αναμένονται αριθμητικές τιμές κ.ο.κ, με σκοπό την παραγωγή μίας ακολουθίας συντεταγμένων. Οι περαιτέρω έλεγχοι, που αφορούν το ενδεχόμενο η ακολουθία συντεταγμένων να είναι αρμόζουσα ενός πολυγώνου που θα μπορούσε να αναπαριστά νησί, αποτελούν αρμοδιότητα της τάξης Island. Επεκτείνοντας, το τελικό κύμα ελέγχων που αφορά όχι την δημιουργία αλλά την ενδεχόμενη επικάλυψη μεταξύ νήσων, αποδίδεται στον κώδικα ενός άλλου τύπου δεδομένων, του Sea και όχι του Island.

Έτσι λοιπόν, καλείται η Island.createIsland(). Η μέθοδος αρχικά εξετάζει το πλήθος των κορυφών της λίστας που δέχθηκε ως όρισμα, apexes. Εάν αυτό είναι μικρότερο του 3, η διαδικασία σταματά με LessThan3ApexesException, καθώς κάθε πολύγωνο πρέπει να έχει τρείς ή περισσότερες κορυφές, επιστρέφοντας στην καλούσα. Σε διαφορετική περίπτωση, μία αναφορά της πρώτης κορυφής αποθηκεύεται για μετέπειτα προσπέλαση και η μέθοδος προχωρά σε μία επαναληπτική δομή εντός της οποίας επεξεργάζονται οι κορυφές, δύο σε κάθε κύκλο επανάληψης, διαδοχικές στη λίστα apexes, με την προπορευόμενη κάθε κύκλου να αποθηκεύεται σε δείκτη με όνομα previous και η επόμενη στον δείκτη next. Κάθε κύκλος επανάληψης δημιουργεί ένα νέο Shore από τα Coordinates previous και next με κλήση constructor: new Shore(previous, next). Υπενθυμίζεται πως η Shore είναι απόγονος του τύπου δεδομένων LinearSegment και αντιπροσωπεύει πλευρά πολυγώνου / ακτή νησιού. Ύστερα από τη δημιουργία της ακτής κάθε κύκλου, αυτή τοποθετείται στην διασυνδεδεμένη λίστα potentialShores που χρησιμοποιεί η createIsland() και στην οποία οι ακτές βρίσκονται δεικτοδοτημένες με τη σειρά δημιουργίας τους. Πρέπει να επισημανθεί πως αμέσως πριν την προσθήκη της ακτής στη λίστα προηγείται ο έλεγχος απλότητας γραμμής. Ο έλεγχος πραγματοποιείται με κλήση στην μέθοδο lineRemainsSimple(), στην οποία δίδονται ως παράμετροι το νεοσυσταθέν Shore, η λίστα potentialShores και ένα boolean flag, αληθές μόνο στον τελευταίο κύκλο. Η lineRemainsSimple()

επιστρέφει boolean τιμή, true εάν η γραμμή παραμένει απλή από το ενδεχόμενο προσθήκης του νέου Shore σε αυτή. Σε διαφορετική περίπτωση, επιστρέφει false και η καλούσα μέθοδος εγείρει NotASimpleLineException, επιστρέφοντας τη ροή εκτέλεσης στη δική της καλούσα, CSVIsland.getData(), η οποία θα κληθεί να χειριστεί την εξαίρεση. Στο τέλος κάθε επανάληψης, η αναφορά της κορυφής του δείκτη next αποθηκεύεται στον δείκτη previous, και στην αρχή του επόμενου κύκλου διαβάζεται η επόμενη κορυφή της λίστας apexes και αναφορά της αποθηκεύεται στο δείκτη next. Με αυτόν τον τρόπο κάθε κύκλος έχει αναφορά δύο διαδοχικών κορυφών, αλλά επίσης κάθε κορυφή επεξεργάζεται από δύο διαδοχικούς κύκλους επανάληψης, ως κορυφή δύο διαδοχικών ευθυγράμμων τμημάτων. Έτσι, η λίστα potentialShores έχει αποθηκευμένα με τη σειρά τμήματα μίας συνεχούς τεθλασμένης γραμμής. Στην αρχή του τελευταίου κύκλου, όταν δηλαδή δεν υπάρχει άλλη κορυφή της λίστας apexes που δεν έχει ήδη διαβαστεί, ως next ορίζεται η πρώτη κορυφή της λίστας, μία αναφορά της οποίας είχε κρατηθεί ξεχωριστά πριν την επαναληπτική δομή. Με αυτόν τον τρόπο εξασφαλίζεται το «κλείσιμο» του πολυγώνου του νησιού. Ύστερα από την δομή επανάληψης, όπου η ροή της εκτέλεσης μπορεί να φτάσει μόνο εάν η τεθλασμένη γραμμή που παρήχθηκε είναι απλή, η μέθοδος επιστρέφει τελικά το αντικείμενο Island με κλήση στον constructor της τάξης, ο οποίος έχει signature: private Island(LinkedList<Shore> shores, String name).

Η μέθοδος με signature private static boolean lineRemainsSimple(Shore s, LinkedList<Shore> potentialShores, boolean lastApex) της τάξης Island καλείται από την createIsland(), κάθε φορά που εκείνη σκοπεύει να επεκτείνει την τεθλασμένη potentialShores (ακτογραμμή νήσου), προσαρτώντας σε αυτή το επόμενο ευθύγραμμο τμήμα (ακτή). Σε κάθε κλήση της, η lineRemainsSimple() λαμβάνει ως παραμέτρους την υπάρχουσα ακτογραμμή ως ουρά ακτών, την υποψήφια ακτή και ένα bit το οποίο είναι αληθές μόνο όταν η καλούσα βρίσκεται στην επεξεργασία του τελευταίου ευθυγράμμου τμήματος που κλείνει το πολύγωνο του νησιού, ενώνοντας την τελευταία κορυφή με την πρώτη. Η lineRemainsSimple() χρησιμοποιεί μία επαναληπτική δομή η οποία διατρέχει τις ακτές της potentialShores με τη σειρά της λίστας, ξεκινώντας από την πρώτη. Η δομή αυτή επεξεργάζεται μία ακτή κάθε φορά και επαναλαμβάνει τις διαδικασίες της μία φορά λιγότερη από το πλήθος της potentialShores, αφού σκοπίμως παραλείπει την τελευταία κάθε φορά ακτή στη λίστα. Σε κάθε κύκλο επανάληψης ελέγχει για το ενδεχόμενο κοινού σημείου εντός του ευθύγραμμου τμήματος της υποψήφιας ακτής με την εκάστοτε ακτή της επανάληψης, μέσω κλήσης στην LinearSegment.getIntersectionWith(). Εάν διαπιστωθεί η τομή, η επανάληψη σταματά και η μέθοδος επιστρέφει την τιμή false στην καλούσα της. Η τελευταία της λίστας ακτή ελέγχεται αμέσως μετά την επαναληπτική δομή ξεχωριστά και έξω από αυτή. Ο λόγος για τον οποίο επιλέχθηκε αυτή η προσέγγιση είναι πως η τελευταία ακτή, ως το τελευταίο ευθύγραμμο τμήμα της γραμμής, εξ' ορισμού τέμνεται με το υποψήφιο προς προσάρτηση στο τέλος της τεθλασμένης ευθύγραμμο τμήμα. Ως εκ τούτου, ελέγχεται με τη χρήση της μεθόδου LinearSegment.intersectsAdjacent() και όχι με την LinearSegment.getIntersectionWith().
Ενδιαφέρον παρουσιάζει και η κλήση της lineRemainsSimple() όταν η createIsland() πρόκειται να «κλείσει» το πολύγωνο του νησιού, εξετάζοντας την προσάρτηση της ακτής που ορίζεται από το τελευταίο με το πρώτο apex. Στην περίπτωση εκείνη, το boolean flag lastApex που η lineRemainsSimple() λαμβάνει ως παράμετρο, είναι αληθές. Τότε, η λειτουργία της αλλάζει ελαφρώς αφού εφαρμόζει την ειδική διαδικασία ελέγχου, όχι μόνο για την τελευταία ακτή της λίστας, αλλά και για την πρώτη επίσης. Ο λόγος είναι ο ίδιος που επιβάλλει την ειδική μεταχείριση της εκάστοτε τελευταίας: το προς προσάρτηση ευθύγραμμο τμήμα έχει κοινά σημεία τόσο με την τελευταία ακτή της λίστας, όσο και με την πρώτη. Αυτά είναι τα δύο άκρα του.

4.2.3: Εργαλεία χαμηλού επιπέδου της τάξης LinearSegment

Σε προσπάθεια παροχής μίας ολοκληρωμένης εικόνας της διαδικασίας δημιουργίας και τοποθέτησης νησιών, κρίνεται χρήσιμη η σύντομη περιγραφή των μεθόδων που εξειδικεύονται στην τομή ευθυγράμμων τμημάτων και στην αλληλοεπικάλυψη συνεχόμενων ευθυγράμμων τμημάτων, getIntersectionWith() και intersectsAdjacent(), αντίστοιχα. Είναι και οι δύο μέλη της τάξης LinearSegment με signatures:

- public IntersectionStatus getIntersectionWith(LinearSegment s)

- public boolean intersectsAdjacent(LinearSegment s)

Είναι εμφανές πως έχουν περεταίρω ομοιότητες, αφού είναι δημοσίου πρόσβασης μέθοδοι αντικειμένου (δηλαδή δεν είναι static) και επίσης αμφότερες δέχονται ακριβώς ένα αντικείμενο της τάξης τους ως παράμετρο. Οι διαφορές τους είναι επίσης αρκετές και σε τεχνικό επίπεδο αφορούν τον επιστρεφόμενο τύπο τους, μα και τον τρόπο λειτουργίας τους. Διαφέρουν ακόμη στη σκοπιμότητά τους, και τις συνθήκες υπό τις οποίες πρέπει να καλούνται.

Η getIntersectionWith() χρησιμοποιείται για την εύρεση πληροφοριών επί της σχέσης μεταξύ δύο ευθυγράμμων τμημάτων και των ευθειών που τα φέρουν, δηλαδή των ευθειών που σχηματίζονται από την επέκτασή τους. Συγκεκριμένα, η πληροφόρηση περιλαμβάνει το σημείο τομής των φέροντων ευθειών αλλά το εάν αυτό ανήκει εντός του εύρους των ευθειών που ορίζουν τα ευθύγραμμα τμήματα. Καθώς η πληροφορία εκτείνεται σε δύο πεδία, η μέθοδος επιστρέφει ένα αντικείμενο LinearSegment.IntersectionStatus, που τα ενθυλακώνει. Πρόκειται για εμφωλευμένο[42] τύπο δεδομένων της LinearSegment, ο οποίος περιέχει τα πεδία:

- intersection, τύπου Coordinates. Αποτελεί το σημείο τομής των ευθειών-φορέων των δύο ευθυγράμμων τμημάτων, εφ' όσον ασφαλώς αυτές τέμνονται, δηλαδή δεν είναι παράλληλες. Η παραλληλία (κανένα σημείο τομής), αλλά και η σύμπτωσή τους (άπειρα σημεία τομής) έχουν ως αποτέλεσμα null τιμή στο πεδίο.
- intersecting, boolean. Είναι αληθές μόνο εάν τα ευθύγραμμα τμήματα έχουν έστω ένα κοινό σημείο. Αυτό ισχύει σε συγγραμμικά μα επικαλυπτόμενα τμήματα, επίσης τμήματα διαδοχικά σε τεθλασμένη γραμμή (συγγραμμικά ή μη), ή τεμνόμενα σε σημεία εσωτερικά του εύρους τους (για αμφότερα ή ένα από τα δύο).

Με τη συνδυασμένη ερμηνεία των τιμών των παραπάνω πεδίων[43], αποτυπώνονται πληροφορίες τομής τόσο για τα τμήματα, όσο και για τις ευθείες που τα φέρουν. Για την επιστροφή του κατάλληλου IntersectionStatus αντικειμένου, η μέθοδος χρησιμοποιεί ένα δένδρο απόφασης, με την πρώτη γενιά των κόμβων του να αποτελεί τις ρίζες τριών υπό-δένδρων, αναλόγως με το ενδεχόμενο τα ευθύγραμμα τμήματα να είναι κατακόρυφα. Η κλίση δεν ορίζεται σε κατακόρυφες με αποτέλεσμα να απαιτούνται άλλοι χειρισμοί στον υπολογισμό τομών, παραλληλίας κ.λ.π. Το πρώτο υποδένδρο αφορά την περίπτωση όπου οι φέρουσες ευθείες έχουν πεπερασμένη κλίση, στο δεύτερο ένα από τα δύο ευθύγραμμα τμήματα είναι κατακόρυφο, ενώ στο τρίτο υποδένδρο και τα δύο είναι κατακόρυφα. Η επόμενη διάκριση αφορά την ισότητα της κλίσης των δύο τμημάτων και υφίσταται μόνο[44] στο πρώτο υποδένδρο. Σε περιπτώσεις διαφορετικής κλίσης των φέροντων ευθειών, η διερεύνηση προχωρά με την εύρεση του σημείου τομής τους και στον περεταίρω έλεγχο εγκλεισμού του σημείου αυτού στο εύρος αμφότερων των ευθυγράμμων τμημάτων. Σε περιπτώσεις ομοίου κλίσης (παραλληλία) γίνεται έλεγχος για την ειδική περίπτωση της συγγραμμικότητας και εάν αυτή ισχύει, ελέγχεται η επικάλυψη (άπειρα σημεία τομής) των τμημάτων βάσει του μήκους των προβολών τους στον κατάλληλο άξονα (αυτή η τεχνική χρησιμοποιείται επίσης από την intersectsAdjacent() οπότε εξηγείται αργότερα). Οι συνδυασμοί των τιμών που ενθυλακώνονται στον τύπο επιστροφής περιγράφονται αναλυτικά στο σχολιασμό[45] και αντανακλούν τις διάφορες πιθανές σχέσεις που μπορεί να υπάρχουν μεταξύ δύο ευθυγράμμων τμημάτων. Η getIntersectionWith() είναι σχεδιασμένη για ευθύγραμμα τμήματα η σχέση των οποίων δεν είναι γνωστή, και για αυτό το λόγο χρησιμοποιεί το πολύπλοκο δένδρο καλύπτει κάθε ενδεχόμενο. Είναι, με άλλα λόγια, γενικής χρήσης.

Ενδιαφέρον παρουσιάζει ο έλεγχος απλότητας μίας τεθλασμένης γραμμής: αφ' ενός ζητείται η εύρεση τμημάτων που τέμνονται με οποιοδήποτε[46] τρόπο με τα μη διαδοχικά τους (οπότε το πεδίο intersecting στο επιστρεφόμενο αντικείμενο της getIntersectionWith() είναι true), και αφ' ετέρου η

[42] Instance inner class (όχι static) τάξη, η οποία ορίζεται ως μέλος της LinearSegment. Δεν είναι ανώνυμη, ούτε τοπική μεταβλητή της getIntersectionWith().
[43] Από τεχνικής πλευράς, τα πεδία είναι ιδιωτικής πρόσβασης και η εγγραφή τους γίνεται από τον constructor τους, ενώ η ανάγνωσή τους μέσω κατάλληλων getter μεθόδων public boolean intersecting() και public Coordinates getIntersection().
[44] Στο δεύτερο εξυπακούεται πως οι κλίσεις είναι άνισες, ενώ στο τρίτο ίσες.
[45] Αρχή της μεθόδου, αρχείο sailAway\application\dataTypes\lowLevelTypes\src\LinearSegment.java.
[46] Συμπεριλαμβανομένης επικάλυψης.

εύρεση τμημάτων που παρουσιάζουν επικάλυψη με τα γειτονικά τους (προηγούμενο / επόμενο). Η ειδική αυτή περίπτωση αντιπροσωπεύει μία ακόμη πρόκληση, καθώς αυτά εξ' ορισμού τέμνονται σε κάθε περίπτωση, έχοντας κοινά σημεία τα άκρα τους, κατ' ελάχιστο. Στα πλαίσια της δημιουργίας νήσου, όπου και ελέγχεται η απλότητα της ακτογραμμής, υπάρχει το ενδεχόμενο στο αρχείο .csv δύο διαδοχικά ευθύγραμμα τμήματα (ακτές) να είναι συγγραμμικά, με το δεύτερο τμήμα να εκτείνεται στην ημιευθεία που ορίζεται από το κοινό άκρο των δύο τμημάτων, την κοινή τους ευθεία, και τη φορά εκείνη που υποδεικνύει το έτερο άκρο του πρώτου τμήματος, προκαλώντας έτσι επικάλυψη με αυτό και έχοντας άπειρα κοινά τμήματα μαζί του. Κάτι τέτοιο θα μπορούσε να ελεγχθεί από null πεδίο intersection και ταυτοχρόνως true στο πεδίο intersecting του αντικειμένου επιστροφής της μεθόδου getIntersectionWith().

Η καλούσα lineRemainsSimple() έχει «επίγνωση» της σειράς διαδοχής των συγκρινόμενων τμημάτων, δεξιόστροφα και αριστερόστροφα, οπότε εφαρμόζει ανά περίπτωση τους κατάλληλους ελέγχους για τομή ξένων, μη διαδοχικών, τμημάτων ή επικάλυψη διαδοχικών, ερμηνεύοντας καταλλήλως την επιστροφή της getIntersectionWith(). Καθώς όμως ο κώδικας ελέγχου στην καλούσα για το ενδεχόμενο επικάλυψης διαδοχικών τμημάτων απαιτεί την ερμηνεία δύο πεδίων, αλλά και αναγκάζει σε εκτέλεση αυτήν τη σχετικά πολύπλοκη μέθοδο, προτιμήθηκε μία προσέγγιση όπου ο παραπάνω έλεγχος επικάλυψης για εξ΄ ορισμού διαδοχικά τμήματα ανετέθη σε μία λιγότερο σύνθετη μέθοδο, την intersectsAdjacent(), με μονοδιάστατη επιστροφή που συμβάλει σε πιο αναγνώσιμο consumer code (κώδικας της καλούσας).

Η μέθοδος intersectsAdjacent() επίσης αντιπροσωπεύει πράξη μεταξύ δύο ευθυγράμμων τμημάτων, ή φυσικά κληρονομικών απογόνων τους όπως Shore. Παρά το όνομά της δεν χρησιμοποιείται για να ερευνήσει το ενδεχόμενο τομής δύο ευθυγράμμων τμημάτων ή των ευθειών που τα φέρουν, ούτε για να υπολογίσει το σημείο αυτό. Η μέθοδος παράγει χρήσιμα αποτελέσματα μόνο όταν καλείται μεταξύ δύο τμημάτων που έχουν σίγουρα ένα κοινό σημείο και μόνο εάν αυτό αποτελεί άκρο και για τα δύο. Χαρακτηριστικό παράδειγμα αυτής της περίπτωσης αποτελούν δύο διαδοχικά ευθύγραμμα τμήματα μίας τεθλασμένης γραμμής. Επιστρέφει true εάν βρεθεί πως τα τμήματα έχουν περισσότερα κοινά σημεία από το γνωστό, ή false εάν μοιράζονται μόνο κάποια άκρη τους. Ο κώδικας της μεθόδου αποφεύγει τη χρήση ενός ιδιαίτερα σύνθετου δένδρου αποφάσεων καθώς εκμεταλλεύεται το γεγονός πως τα τμήματα ήδη τέμνονται. Συνοπτικά, εάν τα τμήματα έχουν διαφορετική κλίση, επιστρέψει false, ενώ προχωρά σε περαιτέρω διερεύνηση εάν τα τμήματα έχουν την ίδια κλίση (γίνεται ξεχωριστός έλεγχος για την περίπτωση των κατακόρυφων τμημάτων). Η διερεύνηση αυτή εξετάζει το μήκος της προβολής της τεθλασμένης γραμμής που σχηματίζεται από τα δύο τμήματα σε έναν από τους δύο άξονες και το συγκρίνει με το άθροισμα των επιμέρους προβολών των τμημάτων στον ίδιο άξονα. Εάν τα τμήματα επικαλύπτονται, το μήκος της προβολής της τεθλασμένης θα είναι μικρότερο από το άθροισμα των δύο επιμέρους προβολών.

4.2.4: Έλεγχοι επικάλυψης και ενσωμάτωση στο αρχιπέλαγος

Πίσω στη main(), ξεκινά η διαδικασία συμπερίληψης του νεοσυσταθέντος νησιού στο αρχιπέλαγος, για την ακρίβεια στη λίστα islands του αντικειμένου του τύπου Sea. Η λίστα είναι ιδιωτικής πρόσβασης καθώς η προσθήκη νήσων δεν πρέπει να γίνεται αυθαίρετα, αλλά μόνο κατόπιν έλεγχου επικάλυψης με τα υφιστάμενα νησιά της λίστας. Έτσι, η υπό συνθήκες προσθήκη πραγματοποιείται μέσω κλήσης στη μέθοδο Sea.addIsland() του αντικειμένου. Το προς ενσωμάτωση νησί δίδεται ως παράμετρος στην addIsland(), η οποία έχει signature public void addIsland(Island isle) throws CSVIslandException. Η μέθοδος αρχικοποιεί μία νέα λίστα στην οποία μπορούν να αποθηκευτούν αναφορές νησιών, την overlappingIsles, η οποία είναι αρχικά κενή. Με χρήση επαναληπτικής δομής εκτελούμενη για κάθε ένα από τα υφιστάμενα και ήδη επιτυχώς τοποθετημένα νησιά, ερευνάται η επικάλυψη του νησιού που έχει δοθεί ως παράμετρος στην μέθοδο, με το εκάστοτε νησί του αρχιπελάγους. Ο έλεγχος αυτός πραγματοποιείται με τη μέθοδο αντικειμένου (instance method) Island.overlapsWith(), η οποία καλείται από το αντικείμενο του υποψηφίου νησιού με όρισμα το εκάστοτε νησί της θάλασσας. Κάθε νησί της θάλασσας με το οποίο το προς ενσωμάτωση παρουσιάζει επικάλυψη, προστίθεται στη λίστα overlappingIsles. Έτσι με το πέρας της επανάληψης, η λίστα overlappingIsles περιέχει το υποσύνολο των νησιών του αρχιπελάγους με τα οποία το εκάστοτε προς τοποθέτηση πολύγωνο παρουσιάζει

επικάλυψη. Εάν η λίστα παραμείνει κενή, σημαίνει πως το υπό προσάρτηση νησί μπορεί να τοποθετηθεί στη λίστα islands, και πράγματι τοποθετείται. Διαφορετικά, η addIsland() συνθέτει ένα αλφαριθμητικό που κατονομάζει όλα τα νησιά που επικαλύπτονται με το υποψήφιο και το περνά ως μήνυμα (πεδίο message) στη νέα IslandOverlapException που εγείρεται, τερματίζοντας τη λειτουργία της και επιστρέφοντας στην καλούσα main(), η οποία επιφορτίζεται με το χειρισμό της εξαίρεσης.

Ίσως άξιο προσοχής είναι το γεγονός πως στη main(), τόσο η κλήση για τη δημιουργία κάποιου νησιού, όσο και η κλήση για την προσθήκη του στο αρχιπέλαγος γίνεται στο ίδιο try block, με catch statement που ανταποκρίνεται στις ενδεχόμενες εξαιρέσεις που πηγάζουν από αμφότερες τις δύο υποστοίβες κλήσης.

Η Sea.addIsland() που παρουσιάστηκε προηγουμένως, βασίζεται εκτενέστατα στη μέθοδο με signature public boolean overlapsWith(Island i), η οποία κρίνεται σκόπιμο να επεξηγηθεί, κατά το δυνατόν συνοπτικά, καθώς η λειτουργία της δεν είναι ιδιαίτερα απλή.

Η μέθοδος overlapsWith() της τάξης Island ελέγχει την επικάλυψη του νησιού που αντιπροσωπεύει το αντικείμενο από το οποίο καλείται, με αυτό που δέχεται ως παράμετρο. Επιστρέφει true εάν πράγματι τα νησιά επικαλύπτονται χωρικά, false σε διαφορετική περίπτωση. Εξάγει συμπέρασμα εξετάζοντας τις σχετικές θέσεις των περιγεγραμμένων ορθογωνίων των δύο νησιών, ενώ όπου δεν είναι δυνατό από τη σχέση αυτή να διαπιστωθεί οριστικά η επικάλυψη των νησιών, διερευνάται περαιτέρω το ενδεχόμενο τομής των ακτών τους, ή ακόμη και του εγκλεισμού σημείου (κορυφής) του ενός στο πολύγωνο του άλλου νησιού. Η μέθοδος περιέχει σε επιλεγμένη σειρά ελέγχους κριτηρίων επί των περιγεγραμμένων ορθογωνίων που εάν ισχύουν και αποτελούν ικανές συνθήκες εξαγωγής συμπεράσματος επιστρέφουν με return statements στην καλούσα, διακόπτοντας έτσι τη ροή προς τους επόμενους ελέγχους. Κατά αυτόν τον τρόπο, εάν ο κώδικας ενός ελέγχου εκτελεστεί στα πλαίσια της μεθόδου, σημαίνει πως τα κριτήρια των ελέγχων που προηγούνται αυτού, έχουν ήδη ελεγχθεί και δεν ικανοποιήθηκαν. Φυσικά κάθε έλεγχος είναι γραμμένος «έχοντας επίγνωση» αυτού του μηχανισμού, αποφεύγοντας τον περιττό επάν-έλεγχο προηγούμενων κριτηρίων. Οι εν λόγω έλεγχοι έχουν ως εξής, παρουσιαζόμενοι εδώ με τη σειρά εκτέλεσής τους:

1. Μη επικάλυψη των περιγεγραμμένων ορθογωνίων. Ελέγχεται από την ανεξαρτησία του εύρους που αυτά καταλαμβάνουν στους άξονες συντεταγμένων. Πραγματοποιούνται δύο έλεγχοι, ένας για κάθε άξονα (γεωγραφικό πλάτος / μήκος). Εάν σε οποιοδήποτε από τους δύο διαπιστωθεί η εν λόγω ανεξαρτησία, η μέθοδος επιστρέφει στην καλούσα με τιμή false, καθώς αφού τα περιγεγραμμένα ορθογώνια των νήσων είναι ξένα, δεν είναι δυνατόν να υπάρχει επικάλυψη των νήσων που περικλείουν.
2. Πλήρης επικάλυψη περιγεγραμμένου ορθογωνίου ενός νησιού από το ορθογώνιο του έτερου (το ένα ορθογώνιο βρίσκεται ολόκληρο «μέσα» στο άλλο). Κατ' αναλογία με τη μεθοδολογία του προηγούμενου ελέγχου, εξετάζεται ο πλήρης εγκλεισμός του εύρους ενός νησιού σε κάποιο άξονα από το εύρος του άλλου νησιού στον ίδιο άξονα. Για την ικανοποίηση των συνθηκών του ελέγχου, θα πρέπει η ίδια διάταξη σχέσης μεταξύ των ορθογωνίων να ισχύει και στον άλλο άξονα, υποχρεωτικά (το ορθογώνιο του ίδιου και πάλι νησιού να περικλείεται από το άλλο). Εάν ικανοποιηθούν αυτές οι συνθήκες,
    a. ελέγχεται το ενδεχόμενο τομής των ακτογραμμών των δύο νήσων με κλήση στη μέθοδο με signature private boolean shoreIntersectionWith(Island i), η οποία μέσω εμφωλευμένων επαναλήψεων απλώς καλεί τη μέθοδο LinearSegment.getIntersectionWith().intersecting() από το αντικείμενο της εκάστοτε ακτής του ενός νησιού με ορίσματα τις διάφορες ακτές του άλλου, ώστε να ελεγχθούν όλοι οι δυνατοί συνδυασμοί[47] των ακτών από κάθε νησί. Εάν εκείνη η μέθοδος διαπιστώσει έστω και μία τομή, επιστρέφει με true στην overlapsWith(), η οποία επίσης επιστρέφει με την ίδια τιμή.
    b. εάν η shoreIntersectionWith() επιστρέψει false, χωρίς να έχει διαπιστωθεί τομή δηλαδή, δεν μπορεί στη φάση εκείνη να αποφασιστεί οριστικά και άνευ αμφιβολίας η επικάλυψη των νησιών, οπότε απαιτείται περαιτέρω διερεύνηση. Για να γίνει

[47] Δηλαδή ελέγχεται η τομή μεταξύ των διατεταγμένων ζευγών w = (x, y) : x ε (x.getShores()), y ε (y.getShores()), με x, y νησιά, ή με άλλα λόγια το καρτεσιανό γινόμενο συνόλου των ακτών του ενός νησιού με το σύνολο των ακτών του άλλου νησιού.

κατανοητή η κατάσταση στην οποία έχει περιέλθει ο αλγόριθμος, υπάρχουν δύο νησιά με πλήρη αλληλοεπικάλυψη των περιγεγραμμένων ορθογωνίων τους, αλλά χωρίς κάποιο κοινό σημείο μεταξύ των ευθυγράμμων τμημάτων που αναπαριστούν τις ακτές τους. Σε αυτή την περίπτωση, αρκεί να ελεγχθεί το ενδεχόμενο όπου οποιοδήποτε σημείο του «εσωτερικού» νησιού (δηλαδή εκείνου του οποίου το περιγεγραμμένο ορθογώνιο επικαλύπτεται πλήρως από το εκείνο του άλλου νησιού) βρίσκεται εντός του πολυγώνου του «εξωτερικού» νησιού. Εάν κάτι τέτοιο ισχύει, τότε ολόκληρο το «εσωτερικό» πολύγωνο επικαλύπτεται από το εξωτερικό, όπως ένα περιφραγμένο οικόπεδο ενός νησιού. Αντίθετα, εάν το σημείο εκείνο δεν περιέχεται στο πολύγωνο με βεβαιότητα συμπεραίνεται πως ολόκληρο το «εσωτερικό» νησί δεν παρουσιάζει χωρική επικάλυψη με το «εξωτερικό», οπότε τα δύο νησιά είναι ανεξάρτητα χωρικά και μπορούν αμφότερα να συμπεριληφθούν στη λίστα Sea.islands και στο σενάριο εξομοίωσης. Διαισθητικά κάτι τέτοιο μοιάζει με σπάνια περίπτωση, αλλά σίγουρα όχι απίθανη, υπάρχει δε και ένα χαρακτηριστικό παράδειγμα στις Κυκλάδες: Σαντορίνη και Νέα Καμένη. Η διερεύνηση του εγκλεισμού σημείου σε πολύγωνο ανατίθεται από την overlapsWith() στην μέθοδο με signature public boolean contains(Coordinates apex). Αυτή καλείται από το αντικείμενο του «εξωτερικού» νησιού και της δίδεται ως όρισμα - αυθαιρέτως - το βόρειο ακρωτήρι του «εσωτερικού» νησιού. Η point-in-polygon μέθοδος αυτή υλοποιεί τον αλγόριθμο ακτίνας / ημιευθείας (ray-casting p-i-p implementation, υπάρχουν και άλλοι αλγόριθμοι όπως ο winding number) μετρώντας των αριθμό τομών της ημιευθείας με αφετηρία το βόρειο ακρωτήρι του «εσωτερικού» νησιού, με τις ακτές του «εξωτερικού» νησιού καθώς αυτή εκτείνεται πέραν του περιγεγραμμένου ορθογωνίου του «περιβάλλοντος» νησιού. Μονός αριθμός δείχνει εγκλεισμό, ενώ ζυγός αριθμός τομών δείχνει ανεξαρτησία / μη επικάλυψη. Ασφαλώς υπάρχουν ειδικές περιπτώσεις (... όπου κάποιο ευρεθέν σημείο μπορεί να αποτελεί σημείο επαφής της γραμμής του πολυγώνου και όχι τομής της, οπότε δεν πρέπει να καταμετρηθεί. Τέτοια σημεία είναι εκείνα που επίσης αποτελούν κορυφές του πολυγώνου και διερευνώνται περισσότερο για το ενδεχόμενο απλής επαφής, εξετάζοντας εάν τα φέροντα ευθύγραμμα τμήματα ανήκουν ανήκουν σε διαφορετικά ημιεπίπεδα, εκατέρωθεν της ακτίνας του αλγορίθμου) τις οποίες ο αλγόριθμος της μεθόδου χειρίζεται ειδικά. Τα σχόλια στην αρχή του κώδικά της, στο αρχείο Island.java, περιγράφουν λεπτομερώς την υλοποίηση.

3. Μερική επικάλυψη των περιγεγραμμένων ορθογωνίων. Σε αυτήν την περίπτωση η κλήση στην shoreIntersectionWith() παράγει το συμπέρασμα με βεβαιότητα, με την τιμή που επιστρέφει η μέθοδος εκείνη να επιστρέφεται και από την παρούσα.

Από τη λειτουργία της, η overlapsWith() αναδεικνύει μία άποψη της αξίας των περιγεγραμμένων ορθογωνίων, αξιοποιώντας εν προκειμένω τρόπους επιτάχυνσης διαδικασιών ελέγχου: Διακρίνονται περιπτώσεις οι οποίες μπορούν να αντιμετωπιστούν έκαστη ειδικά, με μειωμένη πολυπλοκότητα, συμβάλλοντας στην εξοικονόμηση πόρων συστήματος. Στον αντίποδα, η «εξατομικευμένη» αυτή προσέγγιση απαιτεί τη συγγραφή περισσότερου κώδικα. Στο παράδειγμα της overlapsWith(), η μέθοδος διακρίνει τρεις περιπτώσεις διαφορετικών βαθμών αλληλοεπικάλυψης των ορθογωνίων των νησιών. Η πρώτη περίπτωση (ανεξαρτησία) θα μπορούσε να παραλειφθεί, αναθέτοντας των έλεγχο επικάλυψης των σχετικών ζευγών νησιών στην τελευταία περίπτωση (που θα αφορούσε τότε όχι μόνο μερική επικάλυψη, αλλά απλώς όχι πλήρη επικάλυψη περιγεγραμμένων ορθογωνίων). Έτσι, ο κώδικας της μεθόδου θα ήταν σημαντικά λιγότερος, αλλά η πολυπλοκότητα του αλγορίθμου αρκετά υψηλότερη, πραγματοποιώντας ελέγχους τομών μεταξύ όλων των συνδυασμών ακτών για νησιά που αποδεδειγμένα δεν γίνεται να επικαλύπτονται, ή να έχουν τεμνόμενες ακτογραμμές.

Αρκετοί τύποι δεδομένων της εφαρμογής, που αναπαριστούν γεωγραφικές οντότητες με έκταση και χωρική υπόσταση, κάνουν χρήση της έννοιας για ανάλογες σχετικές διευκολύνσεις σε αλγορίθμους οι οποίοι πραγματοποιούν κάποιου είδους επεξεργασία επί αυτών. Έτσι, ενδεικτικά, ο LinearSegment και οι απόγονοί του υπολογίζουν το περιγεγραμμένο ορθογώνιό τους στον κώδικα του constructor τους, κατά τη δημιουργία αντικειμένων τους. Το ίδιο συμβαίνει με τα νησιά (τύπος Island), ενώ το αντικείμενο αρχιπελάγους, Sea, υπολογίζει εκ νέου τα όρια έκτασής του στο χάρτη, ύστερα από κάθε

επιτυχή κλήση της addIsland(), δηλαδή ύστερα από την προσθήκη κάποιου νησιού σε αυτό. Το περιγεγραμμένο ορθογώνιο του αρχιπελάγους χρησιμεύει στους αλγορίθμους γραφικής αναπαράστασης του υποσυστήματος gui.

4.3: Ανάδραση εισόδου χρήστη: Η τεχνική των εμφωλευμένων εξαιρέσεων

Ο μηχανισμός ανάδρασης που χρησιμοποιεί η εφαρμογή σε απόκριση των δεδομένων εισαγωγής χρήστη είναι η τεχνική των αλυσωτών / εμφωλευμένων εξαιρέσεων. Χάρη σε αυτή, καθίσταται δυνατή η «από τα γενικά προς τα ειδικά» διερεύνηση της προέλευσης και αιτίας της όποιας αποτυχίας αξιοποίησης των αρχείων εισόδου, εφαρμόζοντας διαδοχική εκφώλευση («άνοιγμα») των εξαιρέσεων που έχουν εγερθεί κατά την ανάγνωση ή τη μετέπειτα ενσωμάτωση των οντοτήτων. Στη συνέχεια, και πριν παρουσιαστεί ένα παράδειγμα της τεχνικής στα πλαίσια λειτουργίας της εφαρμογής, εξηγείται το τεχνικό υπόβαθρο που καθιστά δυνατό αυτό το μηχανισμό.

Όλα τα Exception types είναι απόγονοι της τάξης Throwable, δύο από τους constructors της οποίας είναι οι: Throwable(String message) και Throwable(String message, Throwable cause). Από αυτούς, γίνεται εμφανές πως η τάξη διαθέτει τα αντίστοιχα πεδία ιδιωτικής πρόσβασης message και cause. Το πρώτο χρησιμοποιείται για την αποθήκευση ενός ανθρωπίνως αναγνώσιμου μηνύματος για την φύση του προβλήματος που προκάλεσε την εξαίρεση, ενώ το cause αποθηκεύει μία αναφορά προς ένα άλλο Throwable αντικείμενο που ειδικότερα ευθύνεται για το σφάλμα. Η ύπαρξη αυτού του πεδίου είναι η βάση της λειτουργίας της τεχνικής των αλυσωτών εξαιρέσεων. Είναι γνωστό πως τα μέλη ιδιωτικής πρόσβασης δεν μεταφέρονται κληρονομικά στους απογόνους, αλλά παραμένουν εμμέσως διαθέσιμα σε αυτούς χάρη σε μεθόδους που έχουν πρόσβαση διαφορετική από ιδιωτική και αναφέρονται σε αυτά. Ενώ η Throwable διαθέτει “getters” (μέθοδοι ανάγνωσης) για τα message και cause, δεν διαθέτει “setters” ή άλλες μεθόδους που μπορούν να γράψουν τις τιμές αυτών των πεδίων. Ο μόνος τρόπος ορισμού της τιμής τους είναι το πέρασμα τους ως ορίσματα στο κατάλληλο constructor, δηλαδή κατά το χρόνο δημιουργίας των αντικειμένων στα οποία ανήκουν. Οι constructors δεν κληρονομούνται, αλλά στην Java υπάρχει τρόπος αποστολής ορισμάτων προς αυτούς, από constructors ίδιου διανύσματος παραμέτρων των άμεσων απογόνων τους (με χρήση της δεσμευμένης λέξης super). Πράγματι, οι τύποι εξαιρέσεων της εφαρμογής ορίζουν όλοι τους δύο constructors με λίστα παραμέτρων (String message) και (String message, Throwable cause), ενώ μέσω αυτών αποστέλλουν τα ορίσματά τους κατά τον τρόπο που περιγράφηκε. Σε μία κληρονομική αλυσίδα, καλούνται ούτως ή άλλως όλοι οι constructors της με φθίνουσα σειρά αρχαιότητας, αλλά με τη χρήση της λέξης super, προηγείται της σειράς αυτής το διαδοχικό πέρασμα παραμέτρων με την αντίθετη φορά. Αυτός είναι ο τρόπος ορισμού τιμών στα πεδία message και cause από τις εξαιρέσεις της εφαρμογής. Για την ανάκτηση των τιμών τους, χρησιμοποιούνται οι “getters” που κληρονομούνται από την Throwable. Σε ότι αφορά αυτές τις μεθόδους, η Throwable ορίζει τις public Throwable getCause(), public String getMessage() και επίσης και την public String toString(). Η τελευταία έχει πρακτικά το ίδιο αποτέλεσμα με την getMessage(), με τη διαφορά ότι στην αρχή του String που επιστρέφει, θέτει ως πρόθεμα το όνομα του τύπου δεδομένων του αντικειμένου από το οποίο κλήθηκε, δηλαδή τον τύπο της εξαίρεσης. Από τα παραπάνω το σημαντικότερο είναι πως υφίσταται τρόπος εμφώλευσης κατά το construction time κάποιας εξαίρεσης με περιεχόμενο την αιτία που την προκάλεσε, ενώ ακόμη για κάθε εξαίρεση υπάρχει η δυνατότητα εκ-φώλευσης / «ξεπακεταρίσματος» με τη μέθοδο getCause().

Προχωρώντας στο αναλυτικό παράδειγμα, γίνεται αναφορά στο σφάλμα εισαγωγής που υπάρχει σε κάποιο σενάριο εξομοίωσης. Έτσι, ως μέρος του τελευταίου δόθηκε ένα προδιαγεγραμμένο ταξίδι αποκαλούμενο “overSpeeder”. Τα σχετιζόμενα με αυτό αρχεία συνεπώς βρίσκονται στη διαδρομή inputFiles\trips\illicit\overSpeeder\. Στο αρχείο περιγραφής του αντίστοιχου πλοίου, vessel.csv, και συγκεκριμένα στο πεδίο “type”, το οποίο καθορίζει το είδος του πλοίου, δόθηκε η τιμή “TRIREME” (Τριήρης). Υπενθυμίζεται πως το πεδίο αυτό είναι του τύπου VesselType, ένα Java enum με τιμές που φαίνονται στο αντίστοιχο .java αρχείο, και δεν περιλαμβάνουν τη δοθείσα τιμή. Με την εκτέλεση της εφαρμογής εμφανίζεται, μεταξύ άλλων, στην γραμμή εντολών:

--> CSVInputException: inputFiles\trips\illicit\overSpeeder\vessel.csv: Unknown Vessel description
----> Caused by:

----> UnknownVesselTypeException: Vessel type not among listed.
------> Caused by:
------> java.lang.IllegalArgumentException: No enum constant VesselType.TRIREME

Το παραπάνω μήνυμα είναι τυπικό παράδειγμα εξόδου της συνάρτησης private static void investigateException(Throwable t) και ίσως προδίδει τη χρήση της τεχνικής των εμφωλευμένων / αλυσωτών εξαιρέσεων εκ μέρους της εφαρμογής. Για την κατανόηση της λειτουργίας της συνάρτησης, αλλά σημαντικότερα, του συνολικού μοντέλου ανάδρασης, εξηγείται η ακολουθία των συμβάντων ενδιαφέροντος που αφορούν το παράδειγμα αυτό, ξεκινώντας από την εκκίνηση της εκτέλεσης και με την υπόθεση πως το ταξίδι “overSpeeder”, με το σφάλμα που περιγράφηκε βρίσκεται ήδη στον αντίστοιχο υποφάκελο του inputFiles\:

Η μέθοδος Application.main() καλεί τη CSVVessel.getData() και αυτή ξεκινά ανάγνωση του vessel.csv, φθάνοντας στο πεδίο “type”. Καθώς το “TRIEREME” δεν υπάρχει ως όνομα αντικείμενου του enum VesselType, η valueOf() (implicit μέθοδος όλων των Java enums, που διαβάζει αλφαριθμητικά και τα αντιστοιχίζει με ομώνυμες τιμές του enum από το οποίο καλείται) εγείρει IllegalArgumentException με μήνυμα “No enum constant VesselType.TRIREME”. Η εξαίρεση αυτή είναι η αρχική, η «εσωτερικότερη» στη δομή εμφώλευσης και η ειδικότερη όλων των υπολοίπων που θα ακολουθήσουν, διαδοχικά περικλείοντάς τη. Εν προκειμένω, είναι μία εξαίρεση που ορίζεται από τις βιβλιοθήκες Standard Edition της γλώσσας Java και όχι από την ίδια τη sailAway. Ασφαλώς αυτό δεν συμβαίνει σε κάθε περίπτωση αλυσίδας εξαιρέσεων της εφαρμογής. Η εξαίρεση προκαλεί ανταπόκριση από πλευράς της μεθόδου, καθώς «πιάνεται» από το catch statement της γραμμής 58 και χειρίζεται από το προσκείμενο μπλοκ κώδικα. Σε αυτό, το statement throw εγείρει νέα UnknownVesselTypeException με message “Vessel type not among listed.” και ως cause εμφωλεύει μέσα της το Throwable από το οποία ξεκίνησαν όλα, το παραπάνω instance IllegalArgumentException.
Η νέα, γενικότερη εξαίρεση «πιάνεται» και αυτή στη γραμμή 81 της ίδιας μεθόδου από catch statement που περικλείει τις δομές χειρισμού εξαιρέσεων στις οποίες έγινε μέχρι τώρα αναφορά, δηλαδή βρίσκεται «ένα βήμα πιο έξω» από πλευράς εμφώλευσης μπλοκ κώδικα. Το catch αυτό δεν επαγρυπνά μόνο για UnknownVesselTypeException τύπους, αλλά γενικότερα για εξαιρέσεις του γονικού τύπου αυτής CSVVesselException, ο οποίος είναι ένας «ενδιάμεσος» κληρονομικά και εννοιολογικά τύπος. Η επιλογή αυτή δίνει στον κώδικα μία σχετική αφαιρετικότητα και επεκτασιμότητα, επιτρέποντας τη μελλοντική έγερση και άλλων σχετικών εξαιρέσεων από το ίδιο μπλοκ. Εν τέλει, η εξαίρεση χειρίζεται, εμφωλευόμενη και αυτή εντός μίας νέας CSVInputException που εγείρεται με ένα ταιριαστό για το είδος της μήνυμα όσον αφορά τη γενικότητα της περιγραφής,
“inputFiles\trips\illicit\overSpeeder\vessel.csv: Unknown Vessel description”. Η τελευταία αυτή εξαίρεση περιέχει μέσα της την UnknownVesselTypeException, η οποία περιέχει[48] την IllegalArgumentException. Η εμφώλευση γίνεται «από μέσα προς τα έξω» και «από τα ειδικά προς τα γενικά». Η νέα συνολική εξαίρεση δεν «πιάνεται» πάλι εντός της μεθόδου, και για το λόγο αυτό το signature της CSVVessel.getData() δηλώνει πως εγείρει εξαιρέσεις του τύπου CSVInputException. Οι εξαιρέσεις που «πετιούνται» σε επίπεδο μεθόδου, υποχρεώνουν σε χειρισμό τους όχι τις ίδιες αλλά τις καλούσες αυτών. Έτσι, επιστέφοντας στην καλούσα .main(), βλέπουμε πως πράγματι αυτή κάλεσε την CSVVessel.getData() εντός ενός try block και με προσκείμενο catch statement για τον κατάλληλο τύπο. Η main() χειρίζεται λοιπόν την εγερθείσα CSVInputException περνώντας την ως όρισμα στην investigateException().
Στη μέθοδο αυτή, που όπως η main(), είναι μέλος της τάξης Application και είναι δηλωμένη ως static, ακολουθείται η αντίστροφη διαδικασία όπου η συνολική εξαίρεση δεν εμφωλεύεται σε κάποια μεγαλύτερη κατασκευή, αλλά αποδομείται με την αντίθετη σειρά («από έξω προς τα μέσα» και «από γενικά προς ειδικά»), τμηματικά και αναδρομικά, όσο κάθε εκ-φωλευόμενη εξαίρεση περιέχει με τη σειρά της κάποια άλλη. Σε κάθε βήμα της διαδικασίας παρουσιάζεται ο τύπος της εξαίρεσης με το σχετικό μήνυμα. Με κατάλληλη μορφοποίηση της εξόδου της, όπως βέλη μεταβλητού μήκους που λειτουργούν ως εισθέσεις και με συνδετικές λέξεις μεταξύ των βημάτων - εξαιρέσεων, καθίσταται προφανής η δομή εμφώλευσης και κατ' επέκταση η αλυσίδα υπαιτιότητας μεταξύ των εξαιρέσεων.

[48] Ασφαλώς δεν πρόκειται κυριολεκτικά για σχέσεις εγκλεισμού, απλώς γίνεται χρήση διασυνδεδεμένης λίστας μίας κατεύθυνσης, υλοποιημένη με δείκτες, ή ειδικότερα στην Java, αναφορές προς αντικείμενα. Κάθε κόμβος αυτής της λίστας «δείχνει» προς το αντικείμενο που κατέχει εμφωλευμένο «μέσα» του, χάρη στο πεδίο cause.

Ακόμη, σε κάποιες περιπτώσεις η έξοδος αυτή αποτελεί μία υποτυπώδη ματιά σε τμήμα της στοίβας κλήσεων της εφαρμογής, χωρίς όμως να έχει τέτοιο σκοπό.

4.4: Οπτικοποίηση σεναρίου / Παραγωγή γραφικής εξόδου

Σε αυτό το στάδιο ανάλυσης είναι πια απολύτως προφανές πως οι δύο κεντρικοί ρόλοι της εφαρμογής είναι η παραγωγή ενός ρεύματος ναυτιλιακών εκπομπών θέσεως και η αναγνώριση σύνθετων γεγονότων επί του ρεύματος αυτού, σε πραγματικό χρόνο. Θέματα που αφορούν τον πρώτο αναπτύσσονται στο παρόν κεφάλαιο, ενώ τα ζητήματα εκείνα που αφορούν την αναγνώριση γεγονότων, παρουσιάζονται λεπτομερέστερα στο επόμενο. Τα παραπάνω περιγράφουν περισσότερα από όλα όσα η εφαρμογή απαιτεί, από τεχνικής πλευράς, για την επίτευξη του στόχου της. Καθώς όμως πρόκειται για ένα end user πρόγραμμα, αφού ανταλλάσσει πληροφόρηση με άνθρωπο, και δεν αποτελεί κάποιου είδους middleware εφαρμογή που επικοινωνεί με άλλες εφαρμογές / συστήματα, κρίθηκε χρήσιμος ο εμπλουτισμός της παρούσας υλοποίησης με περισσότερες λειτουργίες που επιτρέπουν την ανάδραση προς το χρήστη. Υπενθυμίζεται πως τέτοιες δυνατότητες υπάρχουν ήδη, και βασίζονται σε τεχνικές εξαιρέσεων που τελικώς παράγουν σε μορφή κειμένου αναλυτικά μηνύματα στο παράθυρο της γραμμής εντολών. Αυτές αφορούν δυσλειτουργίες που εμποδίζουν την εκτέλεση της εξομοίωσης ή μέρους αυτής, αφού καλύπτουν τους ελέγχους ορθότητας εισαγωγών και ενεργοποιούνται ύστερα από κάποιο σφάλμα σε αρχείο εισαγωγής. Τα μηνύματα αυτά μπορεί ο χρήστης να μελετήσει σε χρόνο της δικής του επιλογής και με όποια ταχύτητα επιθυμεί, με σκοπό να προσαρμόσει τα δεδομένα σε μεταγενέστερες εκτελέσεις του προγράμματος.

Η εξομοίωση, ως διαδικασία που παράγει τις εκπομπές θέσεως αλλά και ως το σύνολο των ορθών δεδομένων εισόδου, μπορεί να επωφεληθεί και αυτή από κάποια μορφή ανάδρασης, καθώς ο χρήστης χρησιμοποιεί συντεταγμένες και άλλες αριθμητικές ποσότητες για την περιγραφή της. Όσο ο όγκος των πληροφοριών αυτών αυξάνεται, εύκολα μπορούν να δημιουργηθούν παρανοήσεις στη νοητή εικόνα του αρχιπελάγους που ο χειριστής ίσως αναγκάζεται να διατηρεί, στη σκέψη του ή εξωτερικά.

Η υπάρχουσα τεχνική εξαιρέσεων δεν καλύπτει τις πληροφορίες εισόδου που είναι ορθές και καταλήγουν να λαμβάνουν επιτυχώς μέρος στην εξομοίωση (π.χ. νησιά που δεν επικαλύπτονται, δρομολόγια που δεν «πλέουν» στη στεριά κ.ο.κ.), αλλά επίσης θα ήταν ακατάλληλη για αυτά ούτως ή άλλως και δεν επεκτείνεται προς εκείνη την κατεύθυνση. Ο περιορισμός με αυτή είναι πως παρέχει κείμενο στην γραμμή εντολών, και ενώ μπορεί αυτό να είναι αναλυτικό ως προς τη φύση του εκάστοτε προβλήματος, δεν ταιριάζει στην πραγματικού χρόνου φύση της παρακολούθησης της εξομοίωσης. Ιδανικά θα πρέπει να παρέχεται πληροφόρηση ουσιαστική αλλά συνοπτική / επισκοπική η οποία θα αφομοιώνεται από το χρήστη αρκετά γρήγορα, δεδομένου του ρυθμού με τον οποίο καταφθάνουν οι νέες εκπομπές θέσεως, αλλάζοντας τα δεδομένα στο αρχιπέλαγος και φέρνοντας το επόμενο «κύμα» πληροφοριών ανάδρασης.

4.4.1: Επισκόπηση λειτουργίας

Αυτό το κενό ανάδρασης καλύπτεται από την οπτικοποίηση των νησιών και της διαδικασίας παραγωγής εκπομπών θέσεως η οποία είναι συνυφασμένη με την πλεύση των πλοίων, βάσει σεναρίου εξομοίωσης. Έτσι, η εφαρμογή παράγει προαιρετικά, αναλόγως ρυθμίσεων, γραφική έξοδο στην οποία αναπαρίσταται το αρχιπέλαγος και τα δρομολόγια πλοίων επί αυτού, με τις κινήσεις τους σε ζωντανό χρόνο χάρη σε μηχανισμούς ανασχεδίασης. Η ενεργοποίηση αυτής της δυνατότητας απεικόνισης δεν επηρεάζει με κανένα τρόπο την παραγωγή εκπομπών ούτε την επακόλουθη αναγνώριση σύνθετων καταστάσεων από αυτή.

Σε ότι αφορά τις παραγόμενες εικόνες, τα νησιά προβάλλονται ως πολύγωνα γεμισμένα με ωχρό καφέ, σε μίμηση του εδάφους. Φέρουν το όνομά τους στο κέντρο του περιγεγραμμένου ορθογωνίου τους με πράσινη γραμματοσειρά. Ο χώρος ανάμεσά τους γεμίζεται με μία ανοικτή μπλε απόχρωση συμβολίζοντας κάλυψη ύδατος. Τα πλοία του σεναρίου, γνησίως εξομοιούμενα και μη, συμβολίζονται

με μικρά ισοσκελή τρίγωνα[49], με το εικονοστοιχείο που καλύπτει το κεντροειδές[50] κάθε τριγώνου να παριστά την τελευταία θέση του πλοίου που αντιπροσωπεύεται από αυτό. Από το εν λόγω σημείο διέρχεται επίσης ο άξονας περιστροφής του τριγώνου, το οποίο είναι περιστραμμένο καταδεικνύοντας[51], εν είδει βέλους, το τρέχον heading του πλοίου. Όσο η τρέχουσα ταχύτητα του πλοίου βρίσκεται κάτω από κάποιο - πολύ χαμηλό συνήθως - όριο, τότε αντί τριγώνου χρησιμοποιείται κύκλος, με το κέντρο του να παριστά τη θέση του σκάφους. Αυτό το όριο ταχύτητας (κάτω από το οποίο το πλοίο θεωρείται πρακτικά στάσιμο) ορίζεται από το χρήστη και πρόκειται για το πεδίο immobileSpeedThreshold στο configuration.csv. Με αυτόν τον τρόπο, οι συμβολισμοί πέραν θέσης και προσανατολισμού, αποκτούν υποτυπωδώς έστω και τη διάσταση της αδράνειας. Στην περίπτωση του κύκλου η κατεύθυνση δεν μπορεί να παρασταθεί γραφικά, όμως αυτό δεν έχει μεγάλη σημασία λόγω ακινησίας. Καθώς τα πλοία έχουν σημειακή υπόσταση στα πλαίσια της εφαρμογής, τα σύμβολά τους (τρίγωνα / κύκλοι) έχουν σταθερό μέγεθος[52] στην οθόνη, ασχέτως όποιας κλίμακας σμίκρυνσης. Ανάλογα με την τιμή του πεδίου trailLength του αρχείου ρυθμίσεων, ενδέχεται να σχεδιάζεται και το ίχνος του πλοίου, το οποίο είναι το ιστορικό των πιο πρόσφατων θέσεών του. Η τιμή του πεδίου καθορίζει τον μέγιστο αριθμό των εμφανιζόμενων θέσεων στην οθόνη για κάθε ένα από τα πλοία του σεναρίου. Ο ακέραιος αριθμός αυτός λογίζει και την τρέχουσα (τελευταία / πιο πρόσφατη) θέση. Για το λόγο αυτό, δεν επιτρέπεται αρνητική ή μηδενική τιμή εισόδου. Το ίχνος αναπαρίσταται ως μία τεθλασμένη γραμμή που ενώνει διαδοχικές χρονικά θέσεις, με την τρέχουσα να συμβολίζεται με ένα από τα σχήματα που περιγράφηκαν. Το σύμβολο του πλοίου και το ίχνος του παρουσιάζονται με το ίδιο χρώμα, το οποίο με κάθε εκτέλεση της εφαρμογής επιλέγεται τυχαία[53]. Καθώς οι θέσεις κάθε πλοίου ανανεώνονται με νεότερες εκπομπές, το ιστορικό των θέσεων, το οποίο υλοποιείται εσωτερικά με ουρά (fifo) ονόματι trail, επεκτείνεται ελέγχοντας παράλληλα το μέγεθός της. Εάν φτάσει το καθορισμένο όριο, αφαιρεί την παλαιότερη χρονικά καταχώρηση από την κεφαλή της πριν από κάθε μεταγενέστερη προσθήκη θέσης στο τέλος της, διατηρώντας έτσι το μέγεθος της ρύθμισης. Οπτικά, όσο το ιστορικό δεν είναι κορεσμένο, η τεθλασμένη γραμμή επεκτείνεται με ένα νέο ευθύγραμμο τμήμα, ενώ ύστερα από τον κορεσμό του, κάθε επέκταση της γραμμής προς τη νεότερη θέση συνοδεύεται από την απώλεια του παλαιότερου ευθύγραμμου τμήματος στο άλλο άκρο της γραμμής.

Το σύνολο των παραγόμενων γραφικών περιλαμβάνεται σε ένα παράθυρο όπου προβάλλεται η κάτοψη του συνόλου των νησιών και, αναλόγως ρυθμίσεων, οι τρέχουσες και πρόσφατες θέσεις των πλοίων. Η κάτοψη είναι όμοια με χάρτη, με το πάνω μέρος της οθόνης να αντιστοιχεί στο Βορρά[54] και

[49] Ίσομεγέθη για κάθε πλοίο, οξυγώνια, με τα δύο ίσα τους σκέλη μακρύτερα από τη βάση.

[50] Πρόκειται για το σημείο τομής των τριών διαμέσων οποιουδήποτε τριγώνου. Καταχρηστικά γνωστό και ως "κέντρο μάζας, αφού σε σχήματα των δύο διαστάσεων η μάζα δεν υφίσταται. Αποδεικνύεται, πως για κάθε διάμεσο οποιουδήποτε τριγώνου το κεντροειδές τη χωρίζει σε αναλογία 2:1, με το μήκος του ευθυγράμμου τμήματος που ορίζεται από το κεντροειδές και την κορυφή που βρίσκεται απέναντι από την πλευρά που ισομοιράζεται από την εν λόγω διάμεσο, να αποτελεί τα 2/3 του συνολικού μήκους της. Στην περίπτωση του ισοσκελούς τριγώνου που χρησιμοποιείται για το συμβολισμό πλοίων στο χάρτη της εφαρμογής, η διάμεσος που ισομοιράζει τη μικρή βάση αποτελεί επίσης ύψος και διχοτόμο. Έτσι, το κεντροειδές βρίσκεται στο 1/3 του (μέγιστου) ύψους από τη (μικρότερη) βάση. Σχεδιάζοντας το τρίγωνο στην οθόνη έτσι ώστε να εφάπτεται στην άνω και στην αριστερή πλευρά της και με την μικρή βάση του παράλληλη στον οριζόντιο άξονα της οθόνης και επίσης έχουσα τη μεγαλύτερη τεταγμένη οθόνης από τα υπόλοιπα εικονοστοιχεία του τριγώνου (δείχνοντας heading 360), το κεντροειδές του θα βρισκόταν κάτω από το εικονοστοιχείο με συντεταγμένες (3,9). Πράγματι, η εφαρμογή διαθέτει σχεδίαση του τριγώνου σε αυτήν ακριβώς τη θέση και προσανατολισμό, αποθηκευμένη στο array shipTriangle. Χρησιμοποιώντας τη ως αφετηρία, την μετατοπίζει και την περιστρέφει παράγοντας τελικώς τα εικονίδια που προβάλλονται στην οθόνη.

[51] Το heading που αντιπροσωπεύει το τρίγωνο με τον τρόπο που είναι περιστραμένο, είναι αυτό που δείχνει ο φορέας του μεγαλύτερου ύψους του, και η φορά είναι από την βάση του ύψους αυτού, προς την κορυφή, η οποία έχει και τη μικρότερη γωνία στο τρίγωνο.

[52] Στην περίπτωση των τριγώνων, το κάθετο στη μικρή βάση ύψος έχει μήκος 14 εικονοστοιχείων, ενώ η εν λόγω βάση, 7. Οι κύκλοι έχουν διάμετρο 9 εικονοστοιχείων.

[53] Από ένα περιορισμένο εύρος της παλέτας RGB, ώστε να είναι σχετικά σκοτεινό έχοντας αντίθεση με το ανοικτό μπλε της θάλασσας.

[54] Ακριβέστερα, ο κατακόρυφος άξονας ψ της οθόνης είναι παράλληλος με τους νοητούς μεσημβρινούς του δισδιάστατου χάρτη που παράγει η εφαρμογή, με φθίνουσα σχέση μεταξύ τεταγμένης εικονοστοιχείων και εικονιζόμενου γεωγραφικού πλάτους, αφού στο σύστημα απόδοσης θέσης εικονοστοιχείων η αρχή των αξόνων - pixel (0,0) - βρίσκεται στην άνω αριστερή γωνία της οθόνης.

με όλα τα προβαλλόμενα στοιχεία να βρίσκονται υπό κλίμακα. Σε κάθε δεδομένη στιγμή / frame, η κλίμακα αυτή παραμένει ακριβώς ίδια για τους δύο άξονες και ασφαλώς ίδια για κάθε εικονιζόμενο στοιχείο, διατηρώντας έτσι την αναλογία των σχετικών τους θέσεων, το λόγο πλευρών τους και αποφεύγοντας, πέραν της απαραίτητης σμίκρυνσης[55], κάθε παραμόρφωση / στρέβλωση και αποτελώντας με τον τρόπο αυτό αξιόπιστη εικόνα της εξομοίωσης. Η κλίμακα εξαρτάται από την έκταση της περιοχής που προβάλλεται και ακριβέστερα από τις πλευρές του περιγεγραμμένου ορθογωνίου της, αλλά και από την ανάλυση της οθόνης. Ενώ στα πλαίσια της ίδιας εκτέλεσης η οθόνη παραμένει αμετάβλητη, η προβαλλόμενη περιοχή ενδέχεται, αναλόγως ρυθμίσεων και σεναρίου εξομοίωσης, να μεταβάλλεται. Σαν αποτέλεσμα μεταβάλλεται και η κλίμακα, δυναμικά. Η σημαντικότερη από τις ρυθμίσεις που αφορούν τη γραφική έξοδο είναι το enum GuiMode, οι διάφορες τιμές του οποίου υπαγορεύουν τον ορισμό της εν λόγω προβαλλόμενης περιοχής:

- FIXED: Η προβαλλόμενη περιοχή ορίζεται ως το περιγεγραμμένο ορθογώνιο του συνόλου των νησιών. Καθώς αυτά είναι αμετακίνητα, το ίδιο αμετάβλητη παραμένει και η προβαλλόμενη περιοχή. Οι υπολογισμοί κλίμακας και λοιπές σχετικές διεργασίες πραγματοποιούνται μόνο κατά την αρχικοποίηση του υπεύθυνου τύπου δεδομένων και όχι σε κάθε επανασχεδίαση της οθόνης, εξοικονομώντας πόρους συστήματος.
- RESIZABLE_TO_TRAILS: Η προβαλλόμενη περιοχή ορίζεται ως το περιγεγραμμένο ορθογώνιο του συνόλου των νησιών και του συνόλου των θέσεων των πλοίων. Συμπεριλαμβάνονται, πέραν από τις τελευταίες θέσεις κάθε πλοίου, οι παλαιές θέσεις τους. Ο αριθμός αυτών δεν περιορίζεται από τη ρύθμιση trailLength, αλλά αφορά όλες τις εκπομπές από την αρχή της εκτέλεσης.
- RESIZABLE_TO_ICONS: Η προβαλλόμενη περιοχή ορίζεται ως το περιγεγραμμένο ορθογώνιο του συνόλου των νησιών και του συνόλου της τρέχουσας - μόνο - θέσης κάθε πλοίου.
- NOGUI: Η εφαρμογή δεν παράγει γραφική έξοδο. Όλοι οι σχετικοί αλγόριθμοι γραφικών και οι προαπαιτούμενοι υπολογισμοί τους δεν εκτελούνται.

Για την παραγωγή γραφικών υπεύθυνο είναι το υποσύστημα gui[56]. Όπως έχει ήδη γραφεί, αυτό απαρτίζεται από τρεις τύπους δεδομένων, οι δύο εκ των οποίων είναι concrete[57], με τον τύπο Terrain να αποτελεί τον αφηρημένο πρόγονό τους, ενσωματώνοντας κοινές δυνατότητες και χαρακτηριστικά των απογόνων του ως μέλη τάξης: πεδία, μέθοδοι και εμφωλευμένοι τύποι.

Θέτοντας τις τιμές RESIZABLE_TO_TRAILS ή RESIZABLE_TO_ICONS στη GuiMode, στις οποίες η προβαλλόμενη περιοχή ορίζεται, πέραν των νήσων, και από τις θέσεις των πλοίων τα οποία ενδέχεται να πλέουν και εκτός του περιγεγραμμένου ορθογωνίου των νησιών, είναι απαραίτητος ο εκ νέου υπολογισμός κλίμακας και λοιπών παραμέτρων πριν από κάθε ανασχεδίαση της οθόνης. Λόγω των ομοιοτήτων στη διαδικασία παραγωγής γραφικών που επιφέρουν οι δύο παραπάνω τιμές, αλλά και λόγω της εκτενούς διαφοροποίησης αυτής με την αντίστοιχη διαδικασία που προκαλεί η τιμή FIXED, οι δύο “RESIZABLE_TO_...” επιλογές προκαλούν τη δημιουργία ενός αντικειμένου ResizableTerrain. Αυτό, όντας τελικώς ένα JPanel, αναλαμβάνει την οπτικοποίηση με το δυναμικό τρόπο που επιβάλλουν αυτές οι τιμές. Η λεπτή διαφορά μεταξύ των δύο τους λαμβάνεται υπ' όψη και αποκτά σημασία ύστερα από τη δημιουργία του εν λόγω JPanel, αρκετά αργότερα από τη δημιουργία του αντικειμένου και κατά την ανανέωση των ορίων της προβαλλόμενης περιοχής, που συμβαίνει ύστερα από κάθε λήψη εκπομπής θέσης. Αντίθετα, η επιλογή FIXED με τις δικές της ιδιαιτερότητες πραγματοποιείται από ένα FixedTerrain JPanel.

Πριν εξεταστεί αναλυτικότερα ο τρόπος με τον οποίο λειτουργούν οι FixedTerrain και ResizableTerrain, θα περιγραφούν οι λίγες αλλά ουσιαστικές διεργασίες που είναι κοινές μεταξύ τους, τόσο αυτές που είναι κληρονομικά κεκτημένες, αλλά και εκείνες που ορίζονται από τα ίδια τα concrete JPanels. Επεκτείνοντας προς αυτήν την κατεύθυνση, παρουσιάζονται επίσης τα μέλη του κοινού τους

[55] Θεωρητικά είναι δυνατή και η μεγέθυνση, σε περιπτώσεις όπου η προβαλόμενη περιοχή είναι μικρότερη από το ωφέλιμο τμήμα του παραθύρου προβολής.

[56] Graphical user interface, γραφική διεπαφή χρήστη. Παρόλο που ο όρος χρησιμοποιείται και για αμφίδρομη επικοινωνία, ο χρήστης μονάχα λαμβάνει πληροφορίες από το παράθυρο γραφικών, χωρίς δυνατότητα επέμβασης στη ροή της εξομοίωσης. Ομοίως δεν υπάρχει τρόπος προσαρμογής των παραγόμενων γραφικών, π.χ. αλλαγή μεγέθυνσης, ή κύλιση εικόνας στους άξονες / εστίαση.

[57] Μη - αφηρημένοι: παρέχουν τη δυνατότητα δημιουργίας αντικειμένου τους, instantiation.

προγόνου, Terrain, στα οποία περιλαμβάνονται και τοπικοί τύποι δεδομένων. Κατόπιν εξήγησης των μελών αυτών, η ανάλυση προχωρά σε μερικές ακόμη κοινές διεργασίες, ακολουθούμενες από τη συνολική παρουσίαση της διαδικασίας παραγωγής γραφικής εξόδου.

4.4.2: Κοινές διεργασίες μεταξύ JPanels

Διεργασία υπολογισμού κλίμακας:
Η πρώτη διεργασία είναι ο υπολογισμός της κλίμακας, που επιτρέπει την περίληψη ολόκληρης της εκάστοτε προβαλλόμενης περιοχής στο παράθυρο, αλλά και ο υπολογισμός των απαραίτητων εισθέσεων (offsets) σε εικονοστοιχεία από τα άκρα της οθόνης, που τοποθετούν την προβαλλόμενη περιοχή του χάρτη στο κέντρο της, κατά ύψος και πλάτος. Η ανάλυση της οθόνης μεταφέρεται στον αλγόριθμο ως δύο μεταβλητές, τις width και height ως πλάτος και ύψος σε εικονοστοιχεία, αντίστοιχα. Από αυτές αφαιρείται το συνολικό πάχος των κατακόρυφων και οριζόντιων - αντιστοίχως - πλαισίων παραθύρου JFrame και τα υπόλοιπά τους αποτελούν την περιοχή που στον κώδικα αναφέρεται ως viewport, δηλαδή η «εσωτερική» επιφάνεια του παραθύρου, που καλύπτεται ολόκληρη από το JPanel. Από το πλάτος αλλά και το ύψος του viewport αφαιρείται το διπλάσιο του margin (αριθμός εικονοστοιχείων που αποτελούν περιθώριο, «μπορντούρα»), με αποτέλεσμα μια άλλη περιοχή, που στον κώδικα των JPanels αναφέρεται ως canvas. Εντός αυτής της περιοχής σχεδιάζονται οι πρωταγωνιστές του σεναρίου εξομοίωσης. Η μη σχεδίαση νησιών και πλοίων σε εικονοστοιχεία που απέχουν όσο margin ή λιγότερο από τις εσωτερικές άκρες του παραθύρου, περιμετρικώς, γίνεται για αισθητικούς και πρακτικούς λόγους, παρά το γεγονός ότι η έκταση της τελικά ωφέλιμης επιφάνειας στην οθόνη, περιορίζεται ελαφρώς. Αυτό το πλαίσιο εμφανίζεται να καλύπτεται με νερό στον προβαλλόμενο χάρτη.

Καθώς, όπως έχει υποστηριχθεί, η προβαλλόμενη περιοχή περιλαμβάνεται στο ακέραιο εντός του canvas και χωρίς καμία παραμόρφωση στο λόγο πλευρών του περιγεγραμμένου ορθογωνίου της, αξίζει να σημειωθεί πως μόνο από σύμπτωση θα μπορούσε ο λόγος αυτός να συμπίπτει με το λόγο πλευρών του canvas. Εάν ο λόγος πλευρών του περιγεγραμμένου ορθογωνίου (πλάτος/ύψος) της εκάστοτε προβαλλόμενης περιοχής είναι σε σχέση με τον αντίστοιχο λόγο του canvas:

- μικρότερος: η προβολή της περιοχής εφάπτεται με τις δύο οριζόντιες πλευρές του canvas, αφήνοντας αχρησιμοποίητο χώρο δεξιά και αριστερά της, ο οποίος ισομοιράζεται φέρνοντας την προβολή στο κέντρο της οθόνης.
- μεγαλύτερος: η προβολή της περιοχής εφάπτεται με τις δύο κατακόρυφες πλευρές του canvas, αφήνοντας αχρησιμοποίητο χώρο πάνω και κάτω της, ο οποίος ισομοιράζεται φέρνοντας την προβολή στο κέντρο της οθόνης.

Το αριστερό μισό του αχρησιμοποίητου χώρου στον οριζόντιο άξονα (leftPadding) αλλά και το άνω μισό του αχρησιμοποίητου χώρου στον κατακόρυφο άξονα (topPadding), αποτελούν offsets ενδιαφέροντος για την εφαρμογή, αφού η αρχή των αξόνων του συστήματος συντεταγμένων της οθόνης είναι η άνω αριστερή γωνία της. Ασφαλώς, σε κάθε περίπτωση και ανάλογα με τη σχέση των πλευρών των δύο ορθογωνίων, ένα από τα δύο offsets θα είναι μηδενικό. Στην περίπτωση που οι δύο λόγοι είναι ίσοι, τότε και τα δύο paddings θα είναι μηδενικά.
Είναι εμφανές πως εξ' αιτίας της ανισότητας των δύο λόγων, είτε το ύψος ή το πλάτος αποτελούν περιοριστικές διαστάσεις στο μέγεθος της προβολής νησιών και πλοίων. Στην πρώτη περίπτωση, η διάσταση που περιορίζει το μέγεθος της προβολής είναι το ύψος, με την εικόνα να εκτείνεται σε όλο το διαθέσιμο ύψος του canvas και με περίσσεια χώρου στον οριζόντιο άξονα (πλάτος) που υπολογίζεται ως padding (γέμισμα) από την εφαρμογή και ισομοιράζεται δεξιά και αριστερά της προβολής, ενώ επίσης τα paddings στον κατακόρυφο άξονα είναι μηδενικά.
Το αντίθετο συμβαίνει στην περίπτωση όπου ο λόγος που υπολογίσθηκε για την προβαλλόμενη περιοχή είναι μεγαλύτερος του λόγου πλευρών του canvas. Εκεί, η περιοριστική διάσταση είναι το πλάτος, με περίσσεια χώρου στον κατακόρυφο άξονα και θετικό topPadding, μηδενικό leftPadding.

Η κλίμακα υπολογίζεται ως ο συντελεστής, ο οποίος πολλαπλασιαζόμενος με την πλευρά εκείνη του περιγεγραμμένου ορθογωνίου της προβαλλόμενης περιοχής που εκτείνεται στην εκάστοτε περιοριστική διάσταση, δίνει γινόμενο την πλευρά του canvas της ίδιας διάστασης. Στη συνέχεια, το

μέγεθος της άλλης πλευράς του περιγεγραμμένου ορθογωνίου της προβολής υπολογίζεται από το scale αυτό, ενώ ο αχρησιμοποίητος χώρος υπολογίζεται ως η διαφορά της άλλης πλευράς αυτής από τη μη περιοριστική πλευρά του canvas. Αναλόγως της σχέσης των λόγων πλευρών των δύο ορθογωνίων ο χώρος αυτός ισομοιράζεται και αποθηκεύεται στο κατάλληλο padding.

Η περιοχή εντός του canvas που δεν περιλαμβάνει τα εκάστοτε paddings οριζόντιων ή κατακόρυφων, ονομάζεται clip. Αποτελεί το χώρο απεικόνισης της προβαλλόμενης περιοχής. Τρέχοντας την εφαρμογή σε debugStatus = true, σχεδιάζονται τα όρια των περιοχών clip και canvas με αποτέλεσμα να φαίνονται και οι αχρησιμοποίητοι χώροι. Κατ' επέκταση είναι αμέσως κατανοητή όχι μόνο η σχέση των λόγων των δύο ορθογωνίων, αλλά και η διαδικασία της διαμέρισης της οθόνης στις επιμέρους περιοχές.

Διεργασία μετατροπής σε συντεταγμένες οθόνης:
Οι υπολογισμοί που περιγράφηκαν προηγουμένως αποτελούν μόνο τη μια όψη του νομίσματος. Η άλλη, είναι η αξιοποίηση των αριθμών που εξήχθησαν για την αποτελεσματική μετατροπή συντεταγμένων του χάρτη σεναρίου εξομοίωσης στις αντίστοιχές τους, σε συντεταγμένες εικονοστοιχείων οθόνης. Η παρακάτω διεργασία επαναλαμβάνεται για μία στρατηγικώς επιλεγμένη μερίδα των σημείων. Παραδείγματος χάρη, για τη μετατροπή ενός ευθυγράμμου τμήματος αρκεί να μετατραπούν μόνο τα δύο άκρα του σε συντεταγμένες οθόνης, τα οποία μετά δίδονται ως ορίσματα σε μέθοδο που σχεδιάζει γραμμές. Κατά ανάλογο τρόπο σχεδιάζονται και σύνθετα σχήματα όπως πολύγωνα, μετατρέποντας σε συντεταγμένες οθόνης μόνο τα σημεία που οι αντίστοιχες μέθοδοι σχεδίασης απαιτούν ως ορίσματα.

Η μέθοδος που πραγματοποιεί την μετατροπή είναι η protected IntPair
cartesianToMonitor(Coordinates c) του γονικού τύπου Terrain. Ο τύπος επιστροφής IntPair αποτελεί απλώς ένα ζεύγος ακεραίων x, y που αντιπροσωπεύουν την οριζόντια και κατακόρυφη διάσταση εικονοστοιχείων, αντίστοιχα. Η μέθοδος εφαρμόζει σειριακά τα παρακάτω επτά βήματα:

1. Από την παράμετρο της μεθόδου εξάγονται το γεωγραφικό μήκος και πλάτος.
2. Από το διάνυσμα του προηγούμενου βήματος αφαιρείται το διάνυσμα που αποτελεί το νοτιοδυτικό άκρο του περιγεγραμμένου ορθογωνίου της προβαλλόμενης περιοχής. Με άλλα λόγια, όλη η προβαλλόμενη περιοχή μετατοπίζεται στην αρχή των αξόνων, για την ακρίβεια τοποθετείται στο θετικό τεταρτημόριο του συστήματος με τις πλευρές του περιγεγραμμένου ορθογωνίου της να εφάπτονται στους άξονες. Ο λόγος για τον οποίο πρέπει[58] να συμβεί αυτό είναι μία ανεπιθύμητη παρ-ενέργεια του πολλαπλασιασμού με την κλίμακα που λαμβάνει χώρα στο επόμενο βήμα. Αυτός ως αποτέλεσμα έχει, πέρα από την επιθυμητή μετατροπή των σχετικών αποστάσεων των σημείων, και τη μετατόπισή τους ανάλογα με την απόστασή τους από την αρχή των αξόνων. Ένα όμοιο πρόβλημα περιγράφεται αναλυτικότερα στο Παράρτημα Β'. Σημειώνεται πως το άκρο αυτό της περιοχής, που στον κώδικα αναφέρεται ως southWestCorner, υπολογίζεται στα JPanels, από το constructor argument Sea.
3. Οι συντεταγμένες πολλαπλασιάζονται με την κλίμακα.
4. Στρογγυλοποιούνται στον πλησιέστερο ακέραιο.
5. Εντός του χώρου της οθόνης όπου τελικά πραγματοποιείται η σχεδίαση νησιών και πλοίων, clip, το σημείο αντικαθίσταται από το συμμετρικό του, με άξονα συμμετρίας τη νοητή οριζόντια γραμμή που διατρέχει την περιοχή clip στο ήμισυ του ύψους της. Για την ακρίβεια, η τεταγμένη οθόνης (y) λαμβάνει ως τιμή το ύψος του clip, μειωμένο κατά την προηγούμενη τιμή της y. Αυτό είναι απαραίτητο καθώς οι συντεταγμένες στο χάρτη λογίζονται βάσει ορθοκανονικού καρτεσιανού επιπέδου, ενώ στο σύστημα συντεταγμένων των εικονοστοιχείων οθόνης, η όποια αύξηση στις τεταγμένες μετακινεί τα σημεία προς την κατεύθυνση που στο χάρτη της η εφαρμογή ορίζει ως Νότο.

[58] Μία εναλλακτική προσέγγιση θα ήταν η τοποθέτηση του βήματος αυτού μετά το βήμα του πολλαπλασιασμό των σημείων με την κλίμακα. Εκεί, θα πρέπει να αφαιρεθεί από την τετμημένη και τεταγμένη του προκύπτοντος σημείου η ποσότητα (scale * southWestCorner.getLongitude()) και (scale * southWestCorner.getLatitude()), αντίστοιχα.

6. Προστίθενται τα offsets στις συντεταγμένες. Το leftPadding στην οριζόντια, το topPadding στην κατακόρυφη, το margin και στις δύο. Καθώς αυτή η διαδικασία πραγματοποιείται για όλα τα σημεία, η προκύπτουσα εικόνα είναι «κεντραρισμένη» στην οθόνη.
7. Η μέθοδος παράγει και επιστρέφει το αντικείμενο IntPair από τις συντεταγμένες που προκύπτουν από την διαδικασία.

4.4.3: Κληρονομικώς κοινά μέλη μεταξύ JPanels

Ενώ τα δύο JPanels του υποσυστήματος έχουν θεμελιώδεις διαφορές, εξακολουθούν να μοιράζονται μεγάλο βαθμό λειτουργικότητας και αυτό αντανακλάται στις ομοιότητες που τα ίδια έχουν ως τάξεις αλλά και στα στοιχεία που κληρονομούν από τον αφηρημένο τύπο, Terrain. Αυτός περιλαμβάνει ως μέλη εμφωλευμένους τύπους, πεδία και μεθόδους. Για την κατανόηση της λειτουργίας του gui, κρίνεται χρήσιμη η εξέταση των σημαντικότερων εκ των μελών αυτών:

1. Εμφωλευμένοι τύποι:
   Πλέον του IntPair, στους υπο-τύπους του Terrain εντάσσονται οι τάξεις Ship<T> και Beep<T> που αποτελούν απλουστευμένες μορφές των Vessel και AISBroadcast, αντίστοιχα. Η απλούστευση αυτή δεν είναι απολύτως αναγκαία από τεχνικής πλευράς, αλλά περιστέλλει την δέσμευση μνήμης, αφού οι παράγωγοι αυτοί τύποι περιέχουν μόνο το υποσύνολο εκείνο των πληροφοριών των αρχικών τους που είναι χρήσιμο κατά την παραγωγή γραφικών. Ακόμη, πεδία σε αυτούς έχουν απλοποιηθεί περαιτέρω και/ή έχουν προστεθεί άλλα, απαραίτητα στην οπτικοποίηση της πλεύσης. Παρόλο που αμφότεροι αυτοί οι τύποι χρησιμοποιούνται από τα JPanels, λόγω της διαφορετικής προσέγγισης που τα τελευταία ακολουθούν σε ότι αφορά υπολογισμούς κλίμακας / offsets – που με τη σειρά της είναι απόρροια των διαφορετικών ορισμών της προβαλλόμενης περιοχής - προκύπτει κάποιου είδους διένεξη στην προσπάθεια επαναχρησιμοποίησης του κώδικα των Ship<T> και Beep<T>, από τους δύο απογόνους του Terrain.
   Για την ακρίβεια, το FixedTerrain το οποίο ως προβαλλόμενη περιοχή θεωρεί το αρχιπέλαγος, περιοχή αμετάβλητη, πραγματοποιεί τους υπολογισμούς κλίμακας στον constructor για μία και μοναδική φορά. Κατ' επέκταση, πληροφορίες όπως δεδομένες ιστορικές θέσεις πλοίων θα αντιστοιχούν στο ίδιο εικονοστοιχείο οθόνης καθ' όλη τη διάρκεια της εκτέλεσης. Για το λόγο αυτό, το FixedTerrain μπορεί να μετατρέψει θέσεις σε εικονοστοιχεία και να αναφέρεται σε αυτές αορίστως. Αντίθετα, στο ResizableTerrain, λόγω της ανάγκης επαν-υπολογισμού των μεταβλητών αυτών, η αντιστοιχία θέσεων χάρτη με συντεταγμένες οθόνης, θα μεταβάλλεται διαρκώς. Έτσι, το ResizableTerrain δεν μπορεί να αποθηκεύσει ιστορικές θέσεις πλοίων και κορυφές πολυγώνων αφού τις μετατρέψει. Φυσικά η μετατροπή συντεταγμένων είναι απαραίτητη και σε αυτή την περίπτωση, αλλά συμβαίνει μόνο εντός της paintComponent(), εκ νέου για κάθε frame. Προφανώς το σημείο διένεξης για τα JPanels είναι ο τρόπος με τον οποίο οι εμφωλευμένες τάξεις ενθυλακώνουν χωρικές θέσεις, με τον FixedTerrain να απαιτεί IntPair συντεταγμένες οθόνης, και τον ResizableTerrain να είναι αναγκασμένος να λειτουργεί με συντεταγμένες χάρτη, Coordinates. Η επίλυση που προτιμήθηκε είναι η συγγραφή των εμφωλευμένων τύπων με χρήση γενικού προγραμματισμού[59], μία έκφανση του οποίου είναι η απόδοση μεταβλητών / wildcards (το δημοφιλέστερο σύμβολο: T) αντί για σαφείς δηλώσεις τύπων.
   a. Ο Terrain.Beep<T> ως απλοποιημένη μορφή του AISBroadcast, ενθυλακώνει τα πεδία:
      - Long mmsi

[59] Ο γενικός (generic) προγραμματισμός στην Java υλοποιείται με την προσέγγιση που είναι γνωστή ως type erasure, κατά την οποία κάθε εμπλεκόμενο πεδίο μεταγλωττίζεται λαμβάνοντας ως τύπο δεδομένων το νεότερο πρόγονο της κληρονομικής ιεραρχίας ο οποίος είναι κοινός μεταξύ των τύπων των αντικειμένων προς τα οποία καταδεικνύει. Για το λόγο αυτό δεν μπορούν σε generic πεδία να δωθούν primitive types ως δηλώσεις τύπων. Το προκύπτον bytecode και κατ' επέκταση το runtime δεν έχουν «επίγνωση» του πραγματικού τύπου της αναφοράς που κρατά ένα generic πεδίο και ο προγραμματιστής συνήθως τον γνωστοποιεί περνώντας παραμέτρους που τον αποκαλύπτουν σε κώδικα που πραγματοποιεί downcasting τέτοιων πεδίων.

- String name
- T position
- double heading
- double speed

Πέραν από τη χρήση γενικού τύπου για την απεικόνιση της θέσης, η τάξη έχει απλουστευθεί καθώς δεν περιέχει αναφορά σε Vessel, αλλά το mmsi του αντίστοιχου πλοίου, και σε διαφορετικό πεδίο, το όνομά του. Το mmsi αποθηκεύεται εδώ όχι ως primitive long όπως στο Vessel, αλλά ενθυλακώνεται στο αντίστοιχο reference type, Long, για λόγους συμβατότητας με άλλα στοιχεία του υποσυστήματος. Αντίθετα, για την αποθήκευση του προσανατολισμού, απλά μεταφέρεται το σχετικό αζιμούθιο ως αριθμητική τιμή, αντί αντικειμένου Azimuth. Έχουν παραλειφθεί τα timeStamp και message.

b. Ο Terrain.Ship<T> ως απλοποιημένη μορφή του Vessel, ενθυλακώνει τα πεδία:
   - String name
   - LinkedList<T> trail
   - double heading
   - double speed
   - Color color
   - Random r

   Τα πεδία των αντικειμένων Ship<T> μεταφέρουν σε μεγάλο βαθμό πληροφορίες που προέρχονται από τα Beeps (προερχόμενα από AISBroadcasts) που εξέπεμψαν τα ταξίδια των πλοίων που αντιπροσωπεύουν. Έτσι, η ομοιότητα του Ship<T> με τον αρχικό τύπο Vessel είναι μικρότερη από αυτή μεταξύ των Beep<T> και AISBroadcast, αλλά αυτό είναι συνέπεια των πληροφοριών που απαιτούν οι διαδικασίες παραγωγής γραφικών. Συγκρίνοντας με τον αρχικό τύπο, έχει διατηρηθεί το πεδίο name, ενώ έχουν παραλειφθεί τα πεδία φυσικών διαστάσεων (length, width, height), τα βοηθητικά πεδία καθορισμού ταχύτητας (maximumSpeed, minimumSpeed, acceleration, deceleration) και τα πεδία flag, type. Το αναγνωριστικού ρόλου πεδίο mmsi έχει - φαινομενικά - επίσης παραληφθεί, αλλά στην πραγματικότητα το υποσύστημα διατηρεί αμφιμονοσήμαντη συσχέτιση μεταξύ mmsi και αντικειμένων Ship<T> χάρη σε άλλο μέλος του Terrain που παρουσιάζεται αργότερα. Νέα πεδία του τύπου είναι τα speed και heading, ο τύπος και οι τιμές των οποίων έρχονται ευθέως από κάποιο Beep<T> αντικείμενο. Άλλες προσθήκες είναι τα πεδία color (java.awt.Color) και r (java.util.Random). Κατά την αρχικοποίηση κάποιου Ship<T>, το Random αντικείμενο χρησιμοποιείται για την παραγωγή ενός τυχαίου χρώματος με το οποίο θα απεικονίζεται το πλοίο και το ίχνος του στην οθόνη, με το προκύπτον χρώμα να αποθηκεύεται στο color. Το τελευταίο και πιο ενδιαφέρον νέο πεδίο του τύπου είναι το trail, η ουρά μήκους trailLength που αντιπροσωπεύει το ιστορικό θέσεων του πλοίου. Το τέλος της είναι η τελευταία (τρέχουσα θέση) του πλοίου ενώ ο τύπος δεδομένων των θέσεών της εξαρτάται από το είδος του JPanel που θα αρχικοποιηθεί από την εφαρμογή, βάσει ρυθμίσεων χρήστη.
   Ένα άλλο μέλος του Terrain.Ship<T> είναι η μέθοδος void update(Beep b, Class<T> type) που χρησιμοποιεί την παράμετρο τύπου Beep<T> για να ανανεώσει τις τιμές των πεδίων του αντικειμένου Ship<T> αντιγράφοντας αυτές του αντικειμένου Beep<T>. Εν πολλοίς, αυτή η μέθοδος αποτελεί «συνδετικό κρίκο» από λειτουργικής πλευράς των δύο τύπων. Οι τιμές που αντιγράφονται είναι οι heading και speed. Ακόμη, η μέθοδος αυτή ελέγχει το μέγεθος της λίστας trail και εάν εκείνη έχει φτάσει το όριο θέσεων, αφαιρεί από την αρχή της μία καταχώρηση. Στη συνέχεια, και ανεξαρτήτως του προηγούμενου ελέγχου, προσθέτει στο τέλος της τη θέση που ενθυλακώνεται στο αντικείμενο Beep<T>, αφού γίνει η κατάλληλη μετατροπή στον ακριβή τύπο θέσεων που κρατά η λίστα trail. Σε αυτήν την ενέργεια χρησιμοποιείται η άλλη παράμετρος που αποκαλύπτει τη φύση του εν λόγω είδους θέσης.

Σε ότι αφορά θέματα πρόσβασης των δύο τύπων, όλα τα πεδία τους είναι δηλωμένα ως private, επιτρέποντας προσπέλαση μέσω μεθόδων δημοσίου πρόσβασης ("getters") που επίσης αποτελούν μέλη των τάξεων. Εξαίρεση αποτελεί το πεδίο Ship<T>.r (τύπου Random), το οποίο έχει εσωτερική χρήση. Η απόδοση τιμών στα πεδία του Beep<T> γίνεται μέσω του constructor του από τις τιμές που δέχονται ως παραμέτρους. Αντίθετα, στον τύπο Ship<T> το μόνο πεδίο που τίθεται κατά το construction είναι το name, με τα color και r να λαμβάνουν ενδογενώς τιμή και υπόλοιπα να τίθενται από τη μέθοδο Ship<T>.update(), η πρόσβαση της οποίας επιτρέπει κλήση από εξωτερικό κώδικα.

2. Πεδία:
   Από τον Terrain, τα δύο JPanels κληρονομούν πληθώρα πεδίων τα οποία παρουσιάζονται ομαδοποιημένα, αναλόγως της μεταβλητότητάς τους. Τα παρακάτω πεδία χρησιμεύουν στην διαδικασία εύρεσης της κλίμακας και λοιπών offsets όπως περιγράφηκε νωρίτερα.
   a. Σταθερές, δηλωμένες ως protected static final:
      - int leftBezel, rightBezel, topBezel, bottomBezel: Περιθώρια παραθύρου
      - int margin
      - double[] shipTriangle: Πρόκειται για το εικονίδιο (τρίγωνο) πλοίων. Αποθηκεύεται ως διάνυσμα πλήθους 2×N των συντεταγμένων των N κορυφών του με διάταξη {x1, y1, x2, y2, ..., xN, yN}, όπου 1,2,...,N το αναγνωριστικό σημείου / κορυφής, x η τετμημένη και y τεταγμένη οθόνης.
      - Color seaColor, landColor
   b. Μεταβλητές, δηλωμένες ως protected:
      - double scale
      - int canvasWidth, canvasHeight
      - int leftPadding, topPadding
      - int clipWidth, clipHeight
      - Coordinates southWestCorner
      - HashMap<Long, Ship> ships: Πρόκειται για ένα σύνολο καταχωρήσεων, κάθε μία εκ των οποίων αποτελεί μονοσήμαντη αντιστοίχηση μεταξύ αριθμού mmsi και του αντικείμενου Ship<T> που αντιπροσωπεύει το πλοίο που φέρει το mmsi της ίδιας καταχώρησης. Κάθε JPanel διατηρεί όλα τα πλοία που σχεδιάζει επί οθόνης σε αυτή τη δομή, εξηγώντας έτσι τη φαινομενική παράλειψη του πεδίου mmsi κατά την απλοποίηση του Vessel σε Ship<T>. Ο τύπος java.util.HashMap<K, V>[60] που χρησιμοποιείται για την υλοποίηση των

[60] Η αντιστοίχηση πραγματοποιείται μεταξύ ενός κλειδιού (key) και μίας τιμής (value) συσχετισμένης με αυτό. Το κλειδί οφείλει να είναι μοναδικό στο σύνολο των κλειδιών: η προσθήκη εγγραφής με κλειδί που υπάρχει ήδη στο σύνολο κλειδιών της δομής, προκαλεί απλώς την αντικατάσταση της τιμής του υπάρχοντος κλειδιού με την τιμή της νέας εγγραφής. Αυτή η δομή δεδομένων σε σημεία υπερτερεί έναντι άλλων δομών όπως πίνακες ή διασυνδεδεμένες λίστες, αφού δεν αποθηκεύει τις καταχωρήσεις με κάποια διάταξη και έχει σταθερή πολυπλοκότητα αναζήτησης, ανεξαρτήτως του αριθμού των καταχωρήσεων σε αυτή. Αυτό επιτυγχάνεται από την τεχνική της αντιστοίχησης: Το κλειδί κάθε καταχώρησης παρέχεται ως εισοδος σε μία συνάρτηση κατακερματισμού (hash function, συνάρτηση που επιστρέφει αριθμούς σταθερού μήκους με ελάχιστη πιθανότητα επιστροφής του ίδιου αριθμού από διαφορετικές εισόδους) η έξοδος της οποίας διαθέτει μονοσήμαντη αντιστοίχηση προς διεύθυνση μνήμης του συστήματος. Στην αντιστοιχούσα θέση μνήμης κάθε κλειδιού αποθηκεύεται το συσχετισμένο με αυτό value. Ακόμη, το HashMap κάνει χρήση μίας λίστας για καταλογράφηση των όλων καταχωρήσεών του, που επιστρέφει με κλήση στη μέθοδο entrySet(). Σε περίπτωση που για δύο διαφορετικά keys η συνάρτηση κατακερματισμού επιστρέψει την ίδια τιμή, έχουμε το λεγόμενο collision όπου οι αντίστοιχες τιμές θα αποθηκευτούν σε λίστα η κεφαλή τη οποίας θα βρίσκεται στην θέση μνήμης που αντιστοιχεί στην, κοινή για τα δύο keys, έξοδο της hash function, hashCode. Έτσι, οι επικαλύψεις αυτές δεν καταστρέφουν τη δομή, αλλά αυξάνουν την πολυπλοκότητα αναζήτησης καθώς αναγκάζονται να εμφωλεύσουν άλλες δομές. Τα collisions αποφεύγονται με επάρκεια μνήμης, και αποτελεσματικές συναρτήσεις κατακερματισμού.

Σε υλοποιήσεις με java.util.HashMap<K, V>, η αρχικοποίηση απαιτεί τη δήλωση των τύπων δεδομένων των προς αντιστοίχηση αντικειμένων, Συγκεκριμένα:

- K: ο τύπος του κλειδιού.
- V: ο τύπος της τιμής.

αντιστοιχήσεων, δεν εγγυάται ασφάλεια των δεδομένων που αποθηκεύει από ταυτόχρονη προσπέλαση νημάτων επεξεργασίας, κοινώς δεν είναι thread-safe. Ωστόσο, και παρά το γεγονός ότι διάφορα threads ενδέχεται πράγματι να εγγράφουν στη δομή ταυτοχρόνως, στο παρόν σενάριο χρήσης αυτή η εξασφάλιση δεν είναι απαραίτητη αφού κάθε νήμα επεξεργασίας (εν προκειμένω, ταξίδι) πάντοτε γράφει μόνο στην καταχώρηση[61] που αφορά το πλοίο του οποίου το ταξίδι προσομοιώνει, και καμία άλλη.

3. Μέθοδοι:
   a. Terrain.cartesianToMonitor(), με signature protected IntPair cartesianToMonitor(Coordinates c):
      Είναι η μέθοδος εκείνη που πραγματοποιεί τον αλγόριθμο μετατροπής θέσεων του χάρτη σε συντεταγμένες οθόνης. Πρόκειται για τον αλγόριθμο των επτά βημάτων που παρουσιάστηκε νωρίτερα και έπεται του υπολογισμού κλίμακας.
   b. Terrain.beep(), με signature protected abstract void beep(AISBroadcast ais) (*Δεν θα πρέπει να συγχέεται με τον εμφωλευμένο τύπο Beep<T>, επίσης μέλος της ίδιας τάξης*):
      Είναι αφηρημένη μέθοδος η οποία απλώς δηλώνεται από τον κώδικα του Terrain, χωρίς να ορίζεται. Τα JPanels FixedTerrain και ResizableTerrain που την κληρονομούν, ορίζουν ελάχιστα διαφοροποιημένες υλοποιήσεις της, και λόγω της μεγάλης ομοιότητάς τους, το μεγαλύτερο, κοινό, μέρος της λειτουργικότητας τους αναλύεται εδώ, με την εστίαση στις λιγοστές ιδιαιτερότητες των δύο εκδοχών να λαμβάνει χώρα στα πλαίσια των επιμέρους αναλύσεων των JPanels.
      Η μέθοδος αποτελεί τη - μονόδρομη - διεπαφή του υποσυστήματος gui με το περιβάλλον της. Καλείται από τα νήματα επεξεργασίας ταξιδιών (τύποι Lnav και BroadcastPlayer) τα οποία κατά τη δημιουργία τους λαμβάνουν μία αναφορά του αντικειμένου Terrain που δημιουργήθηκε (ασφαλώς πρόκειται για απόγονο της αφηρημένης τάξης Terrain, ήτοι FixedTerrain ή ResizableTerrain) και αμέσως μετά από την παραγωγή κάθε εκπομπής AISBroadcast, καλούν την beep() περνώντας ως παράμετρο σε αυτή το αντικείμενο της εν λόγω εκπομπής. Η μέθοδος αποτελεί ακόμα ένα σημείο σύνδεσης μεταξύ των τύπων Ship<T> και Beep<T> αλλά σημαντικότερα, την αφετηρία της διαδικασίας ανανέωσης της προβολής στην οθόνη που καθιστά την οπτική έξοδο σύγχρονη με τη διαδικασία εξομοίωσης / παραγωγής εκπομπών θέσεως από τα νήματα επεξεργασίας. Στους ορισμούς της beep(), ο κώδικας είναι χωρισμένος σε τρία μέρη που εκτελούνται σειριακά. Το τελευταίο αποτελεί το κύριο σημείο διαφοροποίησης μεταξύ των δύο εκδοχών, ενώ τα δύο πρώτα, που παρουσιάζονται εδώ, είναι ίδια με εξαίρεση τον διαφορετικό τύπο που χρησιμοποιείται για την αναπαράσταση θέσεων από τα δύο JPanels.
      Στο πρώτο μέρος, το αντικείμενο AISBroadcast «αποδομείται», με την μέθοδο να αποθηκεύει σε εσωτερικές μεταβλητές τα δομικά πεδία του απλοποιημένου τύπου, τα οποία «επανασυσκευάζονται» στον constructor του Beep<T>. Σε ότι αφορά τη γεωγραφική θέση που αναφέρει το AISBroadcast, αυτή μετατρέπεται σε IntPair με κλήση στην cartesianToMonitor() στην εκδοχή του FixedTerrain, ενώ σε διαφορετική περίπτωση χρησιμοποιείται αντικείμενο τύπου Coordinates. Ακόμη, ανεξαρτήτως τύπου JPanel, το πεδίο mmsi που εξάγεται από το AISBroadcast ενθυλακώνεται σε reference type που απαιτείται για τη δημιουργία Beep<T>.
      Στο δεύτερο μέρος, ταυτοποιείται το εκπέμπον πλοίο από το mmsi του, που αποτελεί κλειδί (key) των καταχωρήσεων του HashMap ships. Εάν η καταχώρηση υπάρχει ήδη,

---

τα παραπάνω θα πρέπει να είναι reference types και όχι primitives. Αυτός είναι και ο λόγος που το mmsi ενθυλακώνεται από τον Beep<T> σε Long.

Μία όμοια, παλαιότερη, δομή στην Java είναι και το java.util.Hashtable<K, V> το οποίο ενώ είναι thread-safe, είναι πιο αργό και δεν επιτρέπει null key / null values.

[61] Εντός του HashMap<K, V>, οι καταχωρήσεις είναι ενθυλακωμένες ως αναφορές σε ξεχωριστά αντικείμενα, του τύπου Map.Entry<K, V>.

ανακτάται από το ships η τιμή της, δηλαδή το αντικείμενο Ship<T>, και καλείται η μέθοδος update() του τελευταίου, με όρισμα το νεοσυσταθέν Beep<T>. Εάν η καταχώρηση δε βρεθεί, όπως συμβαίνει πριν την πρώτη εκπομπή κάθε πλοίου, τότε απλώς το Ship<T> δημιουργείται με όρισμα στον constructor το όνομά του. Στη συνέχεια καλείται η μέθοδος update() όπως ακριβώς θα συνέβαινε εναλλακτικά, και τελικώς η αντιστοίχηση mmsi και του νέου Ship<T> εγγράφεται στο νηολόγιο ships.

4.4.4: Ο τύπος δεδομένων FixedTerrain.Isle

Ένας ακόμη τύπος ενδιαφέροντος στο υποσύστημα gui είναι ο FixedTerrain.Isle, ο οποίος δεν είναι κληρονομικά κοινός, αλλά τοπικός (nested) στο JPanel FixedTerrain, το οποίο εκμεταλλευόμενο την αμετάβλητη περιοχή που προβάλλει, υπολογίζει τις σχετικές μεταβλητές όπως κλίμακα για μία μοναδική φορά, στον constructor του. Στη συνέχεια, μετατρέπει όλα τα νησιά του αρχιπελάγους σε πολύγωνα του παρόντος τύπου, συμβατά με τις μεθόδους σχεδίασης που καλεί από την paintComponent(), και σημαντικότερα: σε συντεταγμένες οθόνης. Οι τελευταίες φυσικά παραμένουν αμετάβλητες για νησιά, όταν το GuiMode λαμβάνει τιμή FIXED και ως εκ τούτου αρχικοποιείται ένα FixedTerrain. Κατ' αναλογία με την περιγραφή που δόθηκε προηγουμένως για τους τύπους Beep<T> και Ship<T>, ο Isle είναι απλοποιημένος τύπος του Island. Περιέχει τα ιδιωτικής πρόσβασης πεδία:

- String title
- int[] lons
- int[] lats
- IntPair titlePosition

Άλλα μέλη του τύπου είναι τέσσερις προσβάσιμοι “getters” που επιτρέπουν ανάγνωση των παραπάνω πεδίων. Η εγγραφή στα title, lons και lats γίνεται μέσω του constructor του από τις τιμές που δέχονται ως παραμέτρους. Το πεδίο titlePosition υπολογίζεται εσωτερικά, καθώς η τελευταία εντολή του constructor είναι μία κλήση στην private void calculateTitlePosition(), που αποτελεί το τελευταίο μέλος της τάξης Isle. Η μέθοδος υπολογίζει το σημείο αυτό, που βρίσκεται στο μέσον του πολυγώνου καθ ύψος και πλάτος, αφού το σημείο αυτό είναι απαραίτητο για την εύρεση της κατάλληλης θέσης του caption[62] ώστε το όνομα του νησιού να εμφανίζεται τοποθετημένο στο κέντρο του του περιγεγραμμένου ορθογωνίου του πολυγώνου.

Τα αντικείμενα του τύπου Isle, τα οποία αρχικοποιούνται μόνο εφ' όσον ο χρήστης επιλέξει FIXED GuiMode και μόνο εάν υπάρχουν νησιά στην εφαρμογή, διατηρούνται στο σύνολό τους σε μία διασυνδεδεμένη λίστα η οποία αποτελεί αποκλειστικά μέλος του FixedTerrain και δεν έχει περιέλθει σε αυτόν κληρονομικά. Η λίστα αυτή ονομάζεται landmass και εκεί αποθηκεύονται τα αντικείμενα αμέσως μετά από τη δημιουργία τους, ενώ επίσης από αυτή τη λίστα πραγματοποιείται η ανάκτησή τους, τυπικά με σκοπό τη σχεδίαση τους στην οθόνη.

4.4.5: Κοινών διεργασιών, συνέχεια

Με την ανάλυση των παραπάνω τύπων και των μελών τους, κληρονομούμενων ή μη, η ανάλυση μπορεί να προχωρήσει σε μερικές ακόμη κοινές - ως επί το πλείστον - διεργασίες ή τμήματα αυτών, με σκοπό η παρουσίαση της λειτουργίας των δύο JPanels να είναι κατά το δυνατόν σύντομη και να δύναται να εστιαστεί στις διαφορές τους, παρά στην επανάληψη κοινών χαρακτηριστικών. Οι επόμενες δύο υπό-διεργασίες εμπλέκονται σε κάποιο βαθμό στην ευρύτερη διαδικασία σχεδίασης των νησιών στην οθόνη, ως προαπαιτούμενες:

Διεργασία μετατροπής από Island σε διανύσματα συντεταγμένων:
Καθώς η παρεχόμενη από το σύστημα γραφικών της Java συνάρτηση που καλεί η εφαρμογή για σχεδίαση πολυγώνων, δεν δέχεται ως παραμέτρους αντικείμενα Island ή Isle, θα πρέπει να υπάρξει

[62] Ετικέττα κειμένου επί οθόνης.

ένας μηχανισμός μετατροπής στην απαιτούμενη μορφή παραμέτρων της μεθόδου αυτής, η οποία δέχεται τα εξής ορίσματα:

- int[] x: Διάνυσμα των τετμημένων.
- int[] y: Διάνυσμα των τεταγμένων.
- int count: Πλήθος σημείων.

Τα παραπάνω δύο διανύσματα αφορούν συντεταγμένες οθόνης των κορυφών του προς σχεδίαση πολυγώνου. Συντεταγμένες με ίσο δείκτη μεταξύ των δύο διανυσμάτων αφορούν το ίδιο σημείο, ενώ η συνάρτηση ξεκινά τη σχεδίαση από τα σημεία με το μικρότερο δείκτη (index). Ο φυσικός αριθμός count δηλώνει το πλήθος των πρώτων (με το μικρότερο index) σημείων που τελικά θα σχεδιαστούν.

Η διεργασία μετατροπής έχει ως εξής, και πραγματοποιείται για κάθε νησί του αρχιπελάγους:

1. Ανακτούνται και αποθηκεύονται τοπικά, το όνομα[63] του νησιού και το πλήθος των ακτών (πλευρών) του. Στα κλειστά πολύγωνα, ο αριθμός αυτός ισούται με τον αριθμό των κορυφών τους.
2. Αρχικοποιούνται μονοδιάστατοι πίνακες ακεραίων: x[] για τις τετμημένες, y[] για τις τεταγμένες. Το μέγεθος που δηλώνουν προέρχεται από το προηγούμενο βήμα.
3. Για κάθε κορυφή του νησιού (συγκεκριμένα για το σημείο Α κάθε πλευράς του):
    a. Με κλήση στην cartesianToMonitor() το σημείο μετατρέπεται σε IntPair οθόνης.
    b. Με κλήση στους “getters” του παραπάνω IntPair, εξάγονται πλάτος και ύψος τα οποία αποθηκεύονται στα x[] και y[], αντίστοιχα, με index τον αριθμό της τρέχουσας επανάληψης, ξεκινώντας από το 0.

Διεργασία εύρεσης θέσης τίτλου:
Η διεργασία αυτή γίνεται με σκοπό το «κεντράρισμα» σε πλάτος και ύψος του caption που αναγράφει το όνομα του εκάστοτε νησιού. Η διαδικασία ξεκινά με την εύρεση του σημείου που βρίσκεται στο μέσον του ύψους και πλάτους του περιγεγραμμένου ορθογωνίου του νησιού σε συντεταγμένες οθόνης. Στην περίπτωση που η εφαρμογή χρησιμοποιεί FixedTerrain, και κατ' επέκταση αντικείμενα Isle, αυτά υπολογίζουν κατά το χρόνο δημιουργίας τους το μέσον αυτό, χάρη κλήσης στην μέθοδό τους calculateTitlePosition(). Σε διαφορετική περίπτωση, αυτό συμβαίνει εντός της μεθόδου σχεδίασης paintComponent(), που αναλύεται στη συνέχεια. Απαιτείται και ένα ακόμη στάδιο, καθώς η μέθοδος σχεδίασης αλφαριθμητικών (java.awt.Graphics.drawString()) γράφει προς τα δεξιά και κάτω από τη θέση που της δίνεται, θα πρέπει αυτή να μεταφερθεί αριστερά κατά το μισό πλάτος, και πάνω κατά το μισό ύψος του caption. Οι διαστάσεις του caption αντλούνται με τη χρήση ενός αντικειμένου java.awt.FontMetrics και της μεθόδου αυτού getStringBounds(), στην οποία το αλφαριθμητικό δίνεται ως παράμετρος. Το δεύτερο στάδιο της διεργασίας αυτό λαμβάνει χώρα εντός της συνάρτησης σχεδίασης, για αμφότερα τα JPanels.

Διεργασία σχεδίασης: η paintComponent():
Η διαδικασία γραφικής σχεδίασης πραγματοποιείται από τον κώδικα αμφότερων των JPanels και για την ακρίβεια από τη μέθοδο public void paintComponent(Graphics g) την οποία κληρονομούν από την τάξη javax.swing.JComponent αλλά ορίζουν εκ νέου (override). Ο κώδικας εντός της μεθόδου αυτής περιγράφει τη σχεδίαση που πραγματοποιείται στην οθόνη κατόπιν κλήσης σε μία δεδομένη στιγμή, παράγοντας δηλαδή ένα frame το οποίο ενώ παραμένει[64] στην οθόνη, αποτελεί σε κάθε περίπτωση στιγμιότυπο των προβαλλόμενων στοιχείων βάσει της κατάστασής τους κατά την κλήση της paintComponent() από τον κώδικα της εφαρμογής. Για προβολή αλλαγών όπως κινήσεις, απαιτείται η εκ νέου κλήση της είτε σε τακτά χρονικά διαστήματα, ή σε απόκριση μίας αλλαγής / γεγονότος. Αυτός ο μηχανισμός ανανέωσης υπάρχει στα JPanels της εφαρμογής αν και υλοποιείται με χρήση διαφορετικών προσεγγίσεων για το καθένα, οι οποίες θα αναλυθούν σε άλλη παράγραφο.

Η λειτουργία, ο τρόπος χρήσης / κλήσης αυτής της μεθόδου, του ορίσματός της και των εξειδικευμένων συναρτήσεων σχεδίασης που καλούνται από αυτό, εντάσσονται στα πλαίσια του δημοφιλέστερου συστήματος γραφικής εξόδου της Java, Swing, του παλαιότερου συστήματος AWT,

[63] Χρησιμεύει στη δημιουργία αντικειμένου Isle, στην περίπτωση FixedTerrain, και για την εύρεση θέσεως τίτλου που ακολουθεί σε κάθε περίπτωση ανεξαρτήτως JPanel.
[64] Δηλαδή ανασχεδιάζεται αυτομάτως όταν το παράθυρο μετακινείται, αλλάζει μέγεθος ή επανέρχεται στο προσκήνιο με ή χωρίς μερική επικάλυψη από άλλα παράθυρα, κ.ο.κ.

αλλά και των σχετικών με αυτά APIs. Εκεί επίσης ανήκουν και οι τύποι που το υπό εξέταση υποσύστημα επεκτείνει και χρησιμοποιεί, όπως JPanel και JFrame. Η ανάλυση της παραπάνω εργαλειοθήκης γραφικής απεικόνισης είναι πέραν[65] της θεματολογίας του παρόντος κειμένου, ωστόσο υπάρχουν μερικά σημεία που αξίζει να παρατεθούν χάριν κατανόησης της διαδικασίας:

- Η κλήση στην paintComponent() συνήθως πάντοτε πρέπει να γίνεται έμμεσα, με κλήση της repaint()[66]. Ο λόγος για τον οποίο η paintComponent() δεν καλείται ευθέως από "consumer" code, είναι αρκετά πολύπλοκος, αλλά συνοπτικά, έτσι εξασφαλίζεται η κλήση και προς άλλες μεθόδους γραφικών που αναλαμβάνουν χρήσιμους ρόλους όπως σχεδίαση αναδρομικώς εμφωλευμένων στοιχείων του αντικειμένου, περιθωρίων κ.α.
- Το αντικείμενο Graphics μπορεί να θεωρηθεί ως το μεταφορικό «μολύβι» που σχεδιάζει τα γραφικά στην οθόνη σε κάθε κλήση της μεθόδου. Σε αυτό ρυθμίζεται το εκάστοτε χρώμα σχεδίασης και από αυτό καλούνται διαφορετικές μέθοδοι που εξειδικεύονται στη σχεδίαση ευθυγράμμων τμημάτων αλλά και σχημάτων όπως κύκλοι/οβάλ, ή πολύγωνα.
- Για κάθε είδος σχήματος υπάρχει μέθοδος του Graphics που το σχεδιάζει με τα εσωτερικά του σημεία στο χρώμα που έχει επιλεχθεί (γέμισμα) και διαφορετική που σχεδιάζει μόνο τα περιμετρικά.
- Οι μέθοδοι σχεδίασης πολυγώνων καλούνται με ορίσματα τα οποία αποτελούν συντεταγμένες των κορυφών, εκφρασμένες σε εικονοστοιχεία. Ως ορίσματα δίνονται δύο ισομεγέθεις και μονοδιάστατοι πίνακες ακεραίων με τον πρώτο να περιέχει τετμημένες και τον δεύτερο τεταγμένες οθόνης. Οι δείκτες των στοιχείων μεταξύ των πινάκων αυτών είναι ίσοι για τις ίδιες κορυφές, αντιστοιχίζοντας έτσι τις συντεταγμένες. Τα σημεία επεξεργάζονται από τις μεθόδους σε αύξουσα σειρά αυτού του δείκτη τους.
- Όπως ήδη αναφέρθηκε, κάθε κλήση της παράγει στιγμιότυπο των διαθέσιμων δεδομένων, οι αλλαγές των οποίων θα πρέπει να προβληθούν με νέες κλήσεις από μηχανισμό ανανέωσης.

Όσο για τη διαδικασία σχεδίασης που εφαρμόζει η paintComponent() στα πλαίσια της sailAway, αυτή χωρίζεται σε τρία σειριακώς εκτελούμενα βήματα:

1. Σχεδίαση των νησιών: Το πρώτο βήμα διαφοροποιείται ελαφρώς αναλόγως του JPanel που αρχικοποιείται. Για το λόγο αυτό εξετάζεται ειδικά για κάθε ένα από αυτά σε άλλη παράγραφο.
2. Σχεδίαση των ορίων των περιοχών σχεδίασης. Το μεσαίο αυτό βήμα είναι προαιρετικό, αναλόγως ρυθμίσεων. Όπως αναφέρθηκε νωρίτερα, εάν το debugStatus τεθεί ως αληθές στο αρχείο configuration.csv, σχεδιάζονται τα όρια των clip και canvas με μαύρες γραμμές πάχους ενός εικονοστοιχείου. Η σημαντικότερη πληροφόρηση που λαμβάνει ο χρήστης από τις γραμμές είναι η σχέση των λόγων πλευρών όπως αναλύθηκε, καθώς και ο ορισμός και η διαρκής μεταβολή της προβαλλόμενης περιοχής.
3. Σχεδίαση των πλοίων και του ίχνους τους. Το παρόν βήμα, όπως και το προηγούμενο, είναι κοινό για τα δύο JPanels και έτσι περιγράφεται εδώ:
   Ουσιαστικά πρόκειται για μία επαναληπτική δομή που εκτελείται για κάθε μία καταχώρηση του HashMap ships και για την ακρίβεια της τιμής σε αυτή, δηλαδή για κάθε αντικείμενο Ship<T>. Από αυτό ανακτάται το χρώμα και τίθεται στο Graphics. Ακόμη, ανακτάται το trail του ανεστραμμένο, ύστερα από κλήση στη μέθοδο descendingIterator() της λίστας trail. Έτσι, το πρώτο στοιχείο είναι η τρέχουσα θέση του πλοίου. Στη συνέχεια και όσο υπάρχουν θέσεις στο trail, ο αλγόριθμος περνά σε επανάληψη όπου σε κάθε κύκλο αυτής επεξεργάζεται το επόμενο στοιχείο της - ανεστραμμένης - λίστας, διατρέχοντας δηλαδή το ιστορικό από τις πιο πρόσφατες προς τις παλαιότερες θέσεις. Σε κάθε επανάληψη από τη δεύτερη και έπειτα, διατηρείται και μία αναφορά προς την ιστορική θέση του αμέσως προηγούμενου κύκλου επανάληψης. Οι συντεταγμένες των δύο αυτών σημείων παρέχονται ως ορίσματα στην μέθοδο Graphics.drawline() και με τον τρόπο αυτό, παράγεται τμηματικά η τεθλασμένη γραμμή του ίχνους. Εξαίρεση αποτελεί η πρώτη επανάληψη, όπου το μοναδικό σημείο που έχει ανακτηθεί αποτελεί την τρέχουσα / τελευταία θέση του σκάφους η οποία απεικονίζεται

[65] Για μία διεξοδική περιγραφή του εν λόγω συστήματος γραφικών:
http://www.oracle.com/technetwork/java/painting-140037.html
[66] public void repaint() της τάξης java.awt.Component.

με χρήση των συμβόλων όπως εξηγήθηκε στην αρχή της ενότητας: Ελέγχεται η ταχύτητα του πλοίου, αφού αντληθεί από το αντικείμενο Ship<T> της καταχώρησης, και εάν είναι μικρότερη ή ίση του ορίου immobileSpeedThreshold (εκ ρυθμίσεων), τότε σχεδιάζεται κύκλος κατά τα γνωστά. Εναλλακτικά, από το Ship<T> αναγιγνώσκεται επιπλέον το heading και μετατρέπεται σε ακτίνια για λόγους συμβατότητας του, ως παράμετρο. Η σχεδίαση του τριγώνου του σκάφους στην κατάλληλη θέση και προσανατολισμό στην οθόνη γίνεται με χρήση ενός αντικειμένου AffineTransform, χάρη στις μεθόδους του οποίου πραγματοποιούνται γραμμικοί μετασχηματισμοί. Για την ακρίβεια, αντίγραφο του shipTriangle, ονόματι shipIcon, μεταφέρεται έτσι ώστε το κεντροειδές του να συμπίπτει με την αρχή των αξόνων οθόνης και περιστρέφεται γύρω από το σημείο αυτό σύμφωνα με το heading του πλοίου. Τελικώς, μεταφέρεται ξανά έτσι ώστε το κεντροειδές του να συμπίπτει με την τρέχουσα θέση του σκάφους στην οθόνη. Το shipIcon, όπως έχει παρουσιαστεί στην παράγραφο των κληρονομούμενων εκ Terrain πεδίων, αποθηκεύει τις κορυφές του στη μορφή {x1, y1, x2, y2,...} χάριν συμβατότητας με τις μεθόδους μεταφοράς / περιστροφής του AffineTransform. Έτσι, πριν δοθεί ως όρισμα της Graphics.fillPolygon() για την σχεδίασή του επί οθόνης, υπόκειται σε μία ακόμη μετατροπή χάρη συμμόρφωσης με τη μορφή ορισμάτων της μεθόδου αυτής, η οποία είναι εκείνη που περιγράφηκε στα πλαίσια της διεργασίας μετατροπής νησιών σε μονοδιάστατα διανύσματα συντεταγμένων οθόνης, προηγουμένως. Καθώς πρόκειται για τρία μόνο σημεία, οι ζυγές κατά array index[67] συντεταγμένες αποθηκεύονται στο διάνυσμα τετμημένων και οι μονές σε αυτό των τεταγμένων.

4.4.6: Η συνολική διαδικασία παραγωγής γραφικών

Κατόπιν περιγραφής των παραπάνω μελών τάξης και διεργασιών, ειδικών ή κληρονομούμενων, η ανάλυση μπορεί πλέον να προχωρήσει στην συνολική διαδικασία παραγωγής γραφικών σε υψηλό επίπεδο, αναφερόμενη στις διεργασίες και τύπους που έχουν πια ήδη επεξηγηθεί. Ωστόσο, δίνεται έμφαση σε θέματα που δεν έχουν παρουσιαστεί καθώς είναι ειδικά για κάποιο από τα JPanels. Το σημείο εκκίνησης του παρόντος σταδίου ανάλυσης είναι στη συνάρτηση main() της εφαρμογής: Ύστερα από τη ρύθμιση βασικών παραμέτρων και τοποθέτηση νησιών, αλλά πριν από τον ορισμό ερωτημάτων EPL και την εκκίνηση των νημάτων δρομολογίων, γίνεται η προετοιμασία που απαιτείται για την προβολή γραφικών.

Συγκεκριμένα, ελέγχεται η τιμή του πεδίου GuiMode από τις ρυθμίσεις και εάν η τιμή που έχει αποδοθεί σε αυτό είναι η NOGUI, τότε καμία άλλη διαδικασία σχετική με γραφικά δεν πραγματοποιείται και η εκτέλεση συνεχίζει στη main() με τις απαραίτητες διεργασίες για την ανάγνωση των πληροφοριών των ταξιδιών από αρχεία εισόδου και στη μετέπειτα εκκίνηση των νημάτων επεξεργασίας τους. Η παράμετρος Terrain που αυτά απαιτούν ως είσοδο στον constructor τους παραμένει null. Σε διαφορετική περίπτωση[68]:

1. Αρχικοποιείται ένα JFrame με τίτλο παραθύρου το όνομα της εφαρμογής, και favicon το /sailAway/media/icons/anchor.png.
2. Με χρήση παρεχόμενων από τη γλώσσα εργαλείων, διαπιστώνεται η ανάλυση οθόνης του συστήματος, και αποθηκεύεται σε δύο ακεραίους για πλάτος και ύψος, σε πλήθος εικονοστοιχείων.
3. Το μέγεθος του JFrame ταυτίζεται με τις διαστάσεις του προηγούμενου βήματος, ενώ η σχετική του θέση με άλλα containers παραμένει αόριστη. Αυτό επιτρέπει την τοποθέτησή του στο μέσον της οθόνης, κίνηση όχι απαραίτητη καθώς εκτείνεται σε όλη την ανάλυσή της.
4. Το JFrame ρυθμίζεται ώστε το κλείσιμο του παραθύρου να μην τερματίζει τη λειτουργία της εφαρμογής. Κλείνοντας το παράθυρο, τόσο η παραγωγή εκπομπών θέσεως από τα νήματα

[67] Με εκκίνηση από το 0.

[68] Χάρη στην επαλήθευση εισόδου του χρήστη από την AppConfigurationCSVParser.getData(), η τιμή της GuiMode δεν μπορεί να είναι κάτι άλλο πέρα από τις τέσσερις προβλεπόμενες τιμές της. Σε περίπτωση λανθασμένης εισαγωγής, χρησιμοποιείται η προκαθορισμένη τιμή, FIXED.

εξομοίωσης, όσο και η αναγνώριση γεγονότων από αυτά, συνεχίζεται κανονικά και τα αποτελέσματα εξακολουθούν να εγγράφονται στα αρχεία εξόδου, σε πραγματικό χρόνο.

5. Το JFrame ρυθμίζεται ώστε να μην επιτρέπει αλλαγή μεγέθους από το χρήστη. Διαφορετικά, δεν θα είναι η προβαλλόμενη περιοχή ορατή στο σύνολό της, καθώς η παρούσα έκδοση δεν υποστηρίζει εκ νέου υπολογισμό κλίμακας και offsets σε απόκριση αλλαγών μεγέθους παραθύρου.
6. Εξετάζεται ο αριθμός των νησιών του αρχιπελάγους (αντικείμενο Sea) και εάν αυτός είναι θετικός, εξετάζεται εκ νέου η τιμή του GuiMode για τη δημιουργία του κατάλληλου Terrain / JPanel αντικειμένου: Εάν αυτή είναι FIXED, αρχικοποιείται ένα FixedTerrain. Αλλιώς (RESIZABLE_TO_ICONS ή RESIZABLE_TO_TRAILS), δημιουργείται ένα ResizableTerrain. Στους constructors αυτών δίνονται ως παράμετροι, πέρα από τη συλλογή νησιών Sea, οι διαστάσεις οθόνης του βήματος 2, απαραίτητες για τη διεργασία εύρεσης κλίμακας και offsets. Παρόλο που δύο διαφορετικές τιμές του GuiMode προκαλούν την χρήση του ίδιου τύπου, αυτή η διαφοροποίηση δεν αποτυπώνεται με κάποιο όρισμα στον constructor του ResizableTerrain. Αντίθετα οι διαφορές στη λειτουργία του συμβαίνουν κατόπιν ανάγνωσης του GuiMode, όπου απαιτείται.
   Στην περίπτωση δεν υπάρχουν καθόλου νησιά στο σενάριο εξομοίωσης και εάν επίσης η τιμή του GuiMode είναι διαφορετική της RESIZABLE_TO_TRAILS, τότε το εν λόγω πεδίο λαμβάνει την τιμή αυτή παρά την όποια άλλη επιλογή του χρήστη, ο οποίος ειδοποιείται για την αλλαγή από ένα μήνυμα στην κονσόλα. Στη συνέχεια αρχικοποιείται ένα ResizableTerrain, λαμβάνοντας τα ορίσματα στον constructor του που θα δεχόταν σε κάθε άλλη περίπτωση.
7. Το αντικείμενο Terrain (JPanel) που δημιουργήθηκε ορίζεται ως εμφωλευμένο[69] (περιεχόμενο) στο JFrame.
8. Το JFrame ορίζεται ως ορατό.
9. Το JFrame ορίζεται ως μεγιστοποιημένο.

Ο λόγος για τον οποίο αψηφάται, υπό συνθήκες, η είσοδος χρήστη στο βήμα 6 είναι η ακαταλληλότητα των υπολοίπων GuiModes κατά την εξέχουσα σχετική συγκυρία: Με την παραδοχή πως είναι επιθυμητή η οπτικοποίηση κάποιου είδους, καθώς ο χρήστης δεν επέλεξε NOGUI, η τιμή αυτή απορρίπτεται. Η FIXED δε, ορίζει ως προβαλλόμενη περιοχή το περιγεγραμμένο ορθογώνιο των νησιών, τα οποία όμως δεν υπάρχουν. Οι υπόλοιπες δύο επιλογές (Resizable to ...) είναι τεχνικώς κατάλληλες, περιλαμβάνοντας στον ορισμό τους για την περιοχή, θέσεις πλοίων. Τελικώς, προτιμήθηκε η RESIZABLE_TO_TRAILS καθώς εκεί προκαλούνται μεταβολές στην έκταση της προβαλλόμενης περιοχής τυπικά σπανιότερα, παρέχοντας - υποκειμενικά - μια πιο ευχάριστη εμπειρία στο χρήστη και δεσμεύοντας λιγότερους πόρους συστήματος.

Τα JPanels εξετάζονται παρακάτω ξεχωριστά ως διαδικασίες, οι οποίες χωρίζονται στις ενέργειες που πραγματοποιούνται κατά τη δημιουργία των αντίστοιχων αντικειμένων και σε εκείνες που αφορούν την ανανέωση οθόνης.

FixedTerrain:
Στον constructor του FixedTerrain πραγματοποιείται αρχικά (και για μοναδική φορά) η διεργασία υπολογισμού κλίμακας και offsets. Τα ευρήματα αποθηκεύονται στα κατάλληλα πεδία - μέλη της τάξης. Στη συνέχεια, λαμβάνει χώρα η διεργασία μετατροπής από τις Island παραμέτρους σε διανύσματα συντεταγμένων οθόνης, συμβατά με τις μεθόδους σχεδίασης της paintComponent(). Κάθε νησί που μετατρέπεται, αποθηκεύεται ως αντικείμενο του εμφωλευμένου τύπου Isle και προστίθεται στη λίστα landmass, που είναι αποκλειστικό μέλος του FixedTerrain. Από τον constructor του τύπου Isle καλείται η Isle.calculateTitlePosition() που πραγματοποιεί ένα μέρος της διεργασίας εύρεσης θέσης τίτλου, συγκεκριμένα την εύρεση του μέσου σε πλάτος και ύψος του Isle. Το υπόλοιπο αυτής της διεργασίας ολοκληρώνεται στην paintComponent().

Ύστερα από κάθε κλήση της μεθόδου FixedTerrain.beep() από κάποιο νήμα παραγωγής εκπομπών θέσης, η μέθοδος πραγματοποιεί τα πρώτα δύο μέρη της που είναι κοινά για τα Jpanels, όπως έχει ήδη περιγραφεί: Μετατρέπει τη ληφθείσα ως παράμετρο εκπομπή AISBroadcast στον τοπικό Beep<T> τύπο, και στη συνέχεια ανανεώνει τις πληροφορίες που αυτό μεταφέρει για το πλοίο, στην αντίστοιχη καταχώρησή του στο HashMap ships. Το τρίτο, τελευταίο, και ειδικό για το FixedTerrain βήμα της

[69] Από το σχεδιασμό του API γραφικών, τα βήματα 7 και 8 καθίστανται απαραίτητα για την προβολή του JPanel.

beep() είναι η κλήση στη repaint(), δηλαδή έμμεσα στην paintComponent(). Η μέθοδος σχεδίασης αυτή, αρχικά σχεδιάζει τα νησιά διαβάζοντας όλα τα στοιχεία του landmass, σχεδιάζοντάς τα ως πολύγωνα με κλήση στην Graphics.fillPolygon() και κατόπιν ολοκλήρωσης της διεργασίας εύρεσης θέσης τίτλου, (λαμβάνοντας δηλαδή υπ' όψη τις διαστάσεις του caption) εγγράφει στην οθόνη τα ονόματά τους με κλήση στην Graphics.drawString(). Η paintComponent() συνεχίζει με τα υπόλοιπα δύο τμήματα που είναι κοινά για τα JPanels και αφορούν την προαιρετική χάραξη γραμμών μεταξύ των νοητών χωρίων στην οθόνη (debug) και τελικώς τη σχεδίαση των πλοίων του ships με το ίχνος τους, αναλόγως ρυθμίσεων. Εξυπακούεται πως εάν οι θέσεις αυτών - ιστορικές ή μη - βρίσκονται εκτός του περιγεγραμμένου ορθογωνίου του αρχιπελάγους, ενδέχεται να μην είναι ορατές.

ResizableTerrain:
Παρόλο που το JPanel αυτό δεν ορίζει κάποιο εμφωλευμένο τύπο, είναι αρκετά πιο πολύπλοκο από το FixedTerrain λόγω της διαρκούς προσαρμογής των προβαλλόμενων στοιχείων του στην οθόνη. Για το λόγο αυτό, παρουσιάζονται εδώ τα αποκλειστικά στο ResizableTerrain μέλη:

1. Πεδία ιδιωτικής πρόσβασης:
    - Sea sea
    - Timer t
    - boolean noIslandsMode_firstBeep

Λόγω της συχνής αναφοράς στα νησιά του αρχιπελάγους από τις μεθόδους του τύπου, το αντικείμενο Sea αποθηκεύεται και τοπικά. Το αντικείμενο javax.swing.Timer χρησιμεύσει στη διαδικασία ανανέωσης οθόνης. Παράγει γεγονότα (ticks) σε καθορισμένη περίοδο: Στην περίπτωση της εφαρμογής, αυτή ορίζεται ίση με την περίοδο εκπομπών από τα νήματα επεξεργασίας, broadcastInterval, με συνήθη τιμή τα 2 δευτερόλεπτα. Το boolean flag noIslandsMode_firstBeep χρησιμεύει στους ειδικούς χειρισμούς που απαιτούνται στην περίπτωση που το σενάριο εξομοίωσης δεν περιλαμβάνει νησιά.

    - double seaMinX, seaMinY, seaMaxX, seaMaxY
    - double minX, minY, maxX, maxY

Οι παραπάνω συντεταγμένες χάρτη αποτελούν τιμές γεωγραφικού μήκους και πλάτους (με κατάληξη X και Y, αντίστοιχα). Το min εντός ονόματος δηλώνει ελάχιστο ενώ το max μέγιστο. Η πρώτη τετράδα ορίζει το περιγεγραμμένο ορθογώνιο του αρχιπελάγους και έτσι δε μεταβάλλεται, ενώ η δεύτερη χωρίς το πρόθεμα “sea”, το περιγεγραμμένο ορθογώνιο της μεταβλητής περιοχής που ορίζεται από το σύνολο των νησιών και τις θέσεις των πλοίων, αναλόγως επιλεγμένου GuiMode. Έτσι, πρόκειται για τις πιο πρόσφατες μόνο θέσεις εκπομπών πλοίων, για την επιλογή RESIZABLE_TO_ICONS, ή όλες τις θέσεις για την RESIZABLE_TO_TRAILS. Η σταθερή περιοχή αναφέρεται ως “sea” στον κώδικα του ResizableTerrain, ενώ ή μεταβλητή - και προβαλλόμενη - περιοχή ως “theatre”.

2. Μέθοδοι:
    - private void resetTheatreExtremes():
      Ταυτίζει το περιγεγραμμένο ορθογώνιο του theatre με αυτό του αρχιπελάγους.
    - private void calculateScaleOriginOffset():
      Εφαρμόζει το μεγαλύτερο μέρος της διεργασίας υπολογισμού κλίμακας και offsets εικονοστοιχείων, συγκεκριμένα αυτό που έπεται του προσδιορισμού του χωρίου canvas.
    - private void updateTheatreExtremes(AISBroadcast ais):
      Η μέθοδος αυτή καλείται από την beep(), ως το τρίτο και τελευταίο βήμα εκείνης της μεθόδου που είναι ειδικό για κάθε JPanel. Κατ' επέκταση τρέχει ύστερα από κάθε λήψη εκπομπής θέσης, ενώ επίσης λαμβάνει ως παράμετρο από την beep() την ίδια αναφορά αντικειμένου AISBroadcast που έλαβε και εκείνη. Η updateTheatreExtremes() αποτελείται από δύο διαζευκτικούς κλάδους αναλόγως της τιμής της GuiMode:
      → RESIZABLE_TO_TRAILS: Εξάγεται η θέση που μεταφέρεται από το AISBroadcast και αποθηκεύεται σε δύο τοπικές μεταβλητές, για τις οποίες ελέγχεται το ενδεχόμενο να βρίσκονται έξω από τα όρια της theatre. Εάν αυτό συμβαίνει, το περιγεγραμμένο ορθογώνιο της εν λόγω περιοχής επεκτείνεται καταλλήλως ώστε να εφάπτεται και με την εν λόγω θέση, ενώ ακόμη πυροδοτείται η calculateScaleOriginOffset() για τον εκ

νέου υπολογισμό κλίμακας και λοιπών αποστάσεων της, επαυξημένης πια, προβαλλόμενης περιοχής.
→ RESIZABLE_TO_ICONS: Αρχικά καλείται η resetTheatreExtremes(), οπότε η προβαλλόμενη περιοχή περιορίζεται προς στιγμήν στα όρια των νησιών. Αμέσως μετά, για κάθε καταχώρηση του ships λαμβάνεται το αντικείμενο Ship<T>, και από το trail αυτού ένα αντίγραφο της πιο πρόσφατης θέσης του, που αποθηκεύεται σε ένα αντικείμενο Coordinates. Στη συνέχεια, εάν αυτή η θέση βρίσκεται εκτός των ορίων της προβαλλόμενης περιοχής (theatre), αυτή επεκτείνεται ώστε το περιγεγραμμένο ορθογώνιό της να εφάπτεται με τη θέση. Με το πέρας της επανάληψης, καλείται η calculateScaleOriginOffset() σε κάθε περίπτωση.
Οι διαφορετικές προσεγγίσεις προσαρμογής των ορίων της προβαλλόμενης περιοχής που πραγματοποιείται βάσει GuiMode από την μέθοδο της παρούσας ανάλυσης, είναι η μοναδική διαφοροποίηση που προκαλείται στην εκτέλεση του κώδικα του ResizableTerrain από τις τιμές της ρύθμισης αυτής.

- public void actionPerformed(ActionEvent e):
  Ο τύπος είναι υποχρεωμένος να υλοποιήσει τη μέθοδο με αυτό το signature καθώς υλοποιεί το interface java.awt.event.ActionListener. Τα γεγονότα που περιοδικά παράγει ο Timer που χρησιμοποιείται από τον ResizableTerrain, απλώς καλούν τη μέθοδο αυτή η οποία χειρίζεται το γεγονός του Timer. Κατά τη δημιουργία του Timer, δίνεται ως constructor argument μία αναφορά σε αντικείμενο που είναι ActionListener, δηλαδή πρακτικά υλοποιεί τη handler μέθοδο αυτή. Πράγματι δίνεται μία αναφορά στο ίδιο αντικείμενο με το keyword “this”. Το μόνο που συμβαίνει στον ορισμό της μεθόδου είναι μία κλήση στη repaint().

Προχωρώντας στην επεξήγηση της συνολικής λειτουργίας του τύπου, κατά τη δημιουργία ενός ResizableTerrain, αρχικοποιείται το Timer και το ships, ενώ το sea λαμβάνει τιμή από το constructor argument. Το flag noIslandsMode_firstBeep είναι αρχικά false. Το μέρος εκείνο της διεργασίας υπολογισμού κλίμακας και offsets που παραμένει αμετάβλητο, παρά την αλλαγή θέσης και έκτασης της προβαλλόμενης περιοχής πραγματοποιείται στον constructor: Υπολογίζεται το viewport αφαιρώντας τα πλαίσια παραθύρου και κατόπιν το canvas με την αφαίρεση των περιθωρίων. Στη συνέχεια προσδιορίζεται η σταθερή περιοχή που αναφέρεται ως sea, δηλαδή το αρχιπέλαγος. Γίνεται έλεγχος του αριθμού των νησιών του, και εάν αυτός είναι θετικός, από τα σύνορα του περιγεγραμμένου ορθογωνίου του λαμβάνουν τις αντίστοιχες τιμές οι μεταβλητές seaMinX, seaMinY, seaMaxX και seaMaxY. Σε διαφορετική περίπτωση αυτές μηδενίζονται, και το noIslandsMode_firstBeep τίθεται αληθές. Η κλήση της resetTheatreExtremes() εξασφαλίζει πως οι τιμές MinX, MinY, MaxX και MaxY (theatre) λαμβάνουν επίσης αρχική τιμή ώστε να δύναται να εκτελεσθεί το υπόλοιπο μέρος της διαδικασίας εύρεσης κλίμακας από την calculateScaleOriginOffset() που καλείται αμέσως μετά. Η τελευταία εντολή στον constructor του τύπου προκαλεί την έναρξη του Timer.

Ύστερα από κάθε κλήση της μεθόδου ResizableTerrain.beep() και αφού πραγματοποιηθούν σε αυτή τα δύο πρώτα, κοινά για τα JPanels μέρη, καλείται η updateTheatreExtremes() για την ανανέωση της προβαλλόμενης περιοχής η οποία τυπικά καλεί στο τέλος των εργασιών της και την calculateScaleOriginOffset() για τον υπολογισμό της κλίμακας και λοιπών παραμέτρων, εκ νέου.

Κάθε «χτύπος» του Timer, ανά broadcastInterval milliseconds, καλεί την actionPerformed() η οποία απλώς καλεί τη repaint(), έμμεσα την paintComponent(). Η μέθοδος σχεδίασης, αρχικά σχεδιάζει τα τύπου Island νησιά της τοπικής λίστας sea αφού πρώτα εφαρμόσει για κάθε ένα από αυτά τη διεργασία μετατροπής τους σε διανύσματα συντεταγμένων οθόνης, συμβατά με τις μεθόδους σχεδίασης της paintComponent(). Η αποτύπωσή τους στην οθόνη γίνεται με κλήση στην Graphics.fillPolygon() και κατόπιν ακολουθεί ολόκληρη η διεργασία εύρεσης θέσης τίτλου, περιλαμβάνοντας την εύρεση του μέσου κατά ύψος και πλάτος του πολυγώνου επί οθόνης, και επίσης λαμβάνοντας υπ' όψη τις διαστάσεις του caption. Μέσω της Graphics.drawString() εγγράφονται στην οθόνη τα ονόματά τους. Η paintComponent() συνεχίζει στα υπόλοιπα δύο κοινά μεταξύ JPanels τμήματα (προαιρετική χάραξη γραμμών μεταξύ των χωρίων στην οθόνη και σχεδίαση των πλοίων του ships με το ίχνος τους, βάσει ρυθμίσεων).

Από τις προπαρασκευαστικές εργασίες στη main(), είναι γνωστό πως εάν δεν υπάρχουν νησιά στο σενάριο, τότε είναι βέβαιο πως αρχικοποιείται ResizableTerrain και μάλιστα με GuiMode

RESIZABLE_TO_TRAILS. Από τα παραπάνω εξάγεται πως το flag noIslandsMode_firstBeep ορίζεται true στον constructor του τύπου, ενώ σε κάθε κλήση της updateTheatreExtremes() καθ' όλη τη διάρκεια μιας τέτοιας εκτέλεσης, θα τρέχει ο πρώτος κλάδος που αφορά την σχετική GuiMode τιμή. Στον αρχή του κλάδου αυτού υπάρχει και ένα ακόμη μπλοκ κώδικα το οποίο δεν παρουσιάστηκε νωρίτερα: Αυτό τρέχει μόνο όταν το noIslandsMode_firstBeep είναι αληθές, ενώ η τελευταία εντολή σε αυτό καθιστά το flag ψευδές. Με άλλα λόγια, όταν δεν υπάρχουν έγκυρα νησιά στα δεδομένα εισόδου, δημιουργείται ResizableTerrain με “trails” GuiMode και επίσης μόνο κατά την πρώτη φορά κλήσης της updateTheatreExtremes() εκτελείται πρόσθετος κώδικας. Αυτός μετατοπίζει το ανύπαρκτο, και σημειακό ήδη από τον constructor, αρχιπέλαγος στην ίδια θέση με αυτή του AISBroadcast που προκάλεσε την κλήση της μεθόδου. Αυτό συμβαίνει για αισθητικούς κυρίως λόγους, αποφεύγοντας έτσι την αναγκαστική περίληψη του αυθαίρετου σημείου που έπρεπε να αποδοθεί στο αρχιπέλαγος για λόγους αρχικοποίησης, εάν αυτό τελικά δεν βρίσκεται εντός της προβαλλόμενης περιοχής του χάρτη. Ο σχετικός σχολιασμός στο αρχείο ResizableTerrain.java παρέχει μερικές ακόμη λεπτομέρειες.

Οι κυριότερες διαφορές των δύο JPanels είναι αφ' ενός ο συνεχής ή μοναδικός προσδιορισμός της προβαλλόμενης περιοχής, που συνοδεύεται από τους απαραίτητους σχετικούς υπολογισμούς. Κατ' επέκταση, η απαραίτητη χρήση διαφορετικών τύπων σε κοινές διαδικασίες με χρήση γενικού προγραμματισμού. Αφ' ετέρου, διαφέρει η διαδικασία ανανέωσης της οθόνης. Σε αντίθεση με την προσέγγιση του FixedTerrain, το ResizableTerrain δεν καλεί αυτομάτως, ύστερα από κάθε λήψη AISBroadcast, τη repaint() για εκ νέου σχεδίαση της οθόνης. Αντιθέτως, σχεδιάζει περιοδικώς τα - ανανεωμένα ή μη - πλοία και στεριές. Η λύση αυτή χρησιμοποιήθηκε πειραματικά σε μία προσπάθεια περιστολής της δέσμευσης πόρων συστήματος, και δεν μοιάζει να προκαλεί ουσιαστική χρονική υστέρηση στην απόκριση επί οθόνης μεταξύ διαφορετικών πλοίων.

4.5: Εξομοίωση πλεύσης σε νήματα επεξεργασίας: Ο τύπος Lnav

Στην παρούσα ενότητα εξετάζεται η διαδικασία παραγωγής γνησίως εξομοιούμενων ταξιδιών, μία συνοπτική παρουσίαση των οποίων έγινε εντός του προηγούμενου κεφαλαίου. Ο υπεύθυνος τύπος δεδομένων είναι ο Lnav[70] που χάρη σε απλά μοντέλα φυσικής παράγει, με αφετηρία τα χαρακτηριστικά του πλοίου και της διαδρομής του, αρκετά αληθοφανή σειρά εκπομπών θέσεως σε πραγματικό χρόνο.

Πριν την παράθεση του τρόπου παραγωγής εκπομπών θέσης (LLEs) από τα νήματα επεξεργασίας γνησίως εξομοιούμενων ταξιδιών, καλύπτονται εδώ κάποιες προπαρασκευαστικές σχετικές έννοιες. Αυτές περιλαμβάνουν την εξομάλυνση των αλλαγών κατεύθυνσης στη διαδρομή του τύπου Itinerary, την εύρεση της μέγιστης ταχύτητας στις στροφές της εν λόγω πορείας με χρήση ενός απλού μοντέλου φυσικής και τελικώς την εξήγηση λίγων βοηθητικών μεθόδων και εννοιών του Lnav.

4.5.1: Η γεωμετρία των στροφών της διαδρομής

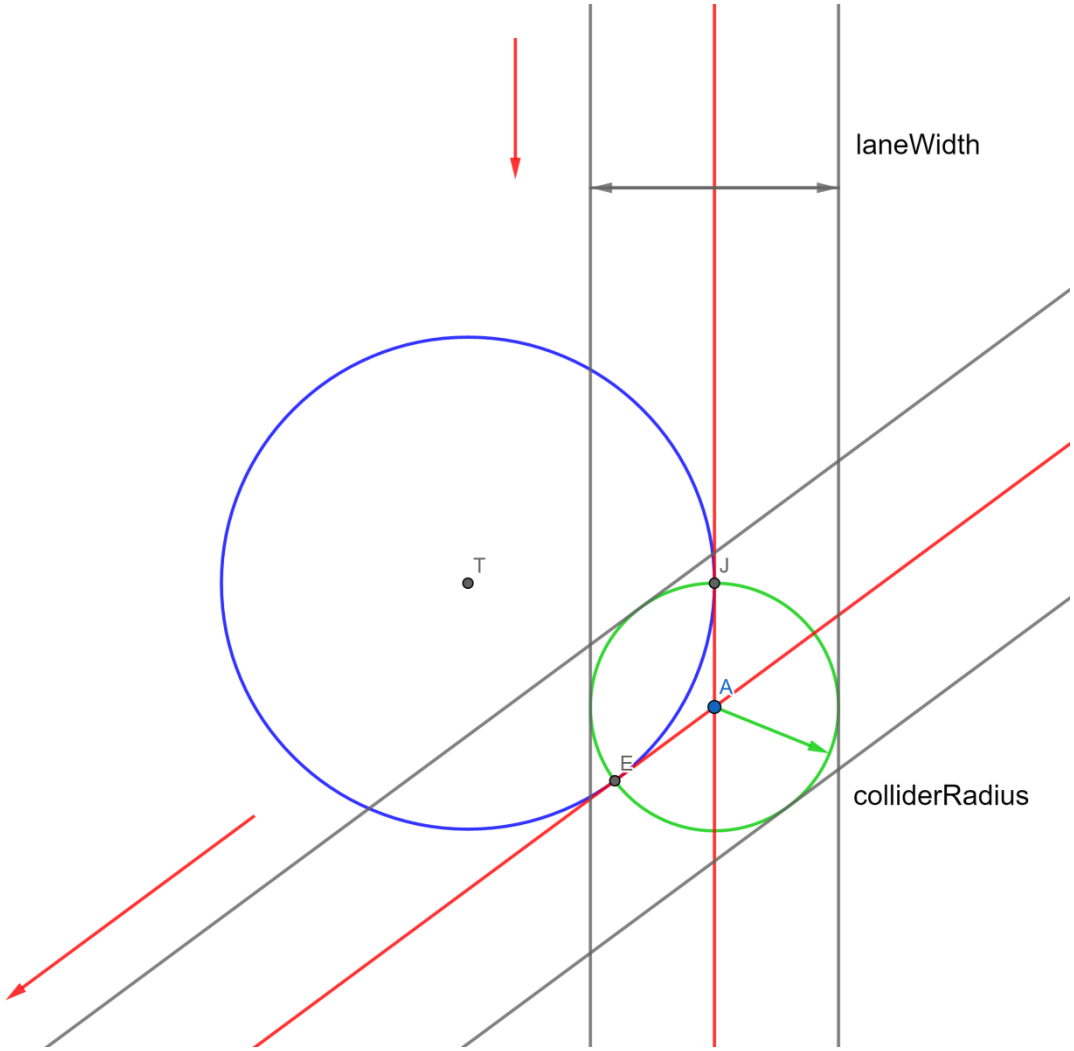
Όπως έχει υποστηριχθεί ήδη, συστατικό κάθε γνησίως εξομοιούμενου ταξιδιού είναι η διαδρομή του ως μία τεθλασμένη γραμμή επί του χάρτη. Αυτή ενθυλακώνεται σε ένα αντικείμενο Itinerary ως ακολουθία ευθυγράμμων τμημάτων (Legs), τα οποία με τη σειρά τους ορίζονται από δύο διαδοχικά Waypoints / κορυφές της τεθλασμένης.

Η πιστή και απαρέγκλιτη πλεύση επί της γραμμής αυτής, θα παρήγαγε πολύ απότομες στροφές αφού, ανεξαρτήτως γωνίας, αυτές είναι σημειακές (έχουν μηδενική ακτίνα). Σε προσπάθεια για μεγαλύτερη αληθοφάνεια, αντί της τροχιάς αυτής, κάθε πλοίο γνησίως εξομοιούμενου ταξιδιού ακολουθεί μία ελαφρώς τροποποιημένη διαδρομή στην οποία οι κορυφές μεταξύ των ευθυγράμμων τμημάτων (Legs) παρακάμπτονται από τόξα κύκλων που εφάπτονται στα δύο διαδοχικά τμήματα. Τα σημεία επαφής κάθε τέτοιου κύκλου στροφής με τα εμπλεκόμενα Legs, είναι αυτά που απέχουν μισό laneWidth από το Waypoint (κορυφή τεθλασμένης) της εκάστοτε στροφής, εκατέρωθεν. Η απόσταση αυτή, αναφερόμενη στον κώδικα της εφαρμογής ως colliderRadius[71], αποτελεί την ακτίνα ενός άλλου νοητού κύκλου, με κέντρο την κορυφή της στροφής. Το laneWidth αποτελεί σταθερή απόσταση για κάθε εκτέλεση, και σταθερό είναι κατ' επέκταση το μισό του, colliderRadius.

Για διευκόλυνση της ανάλυσης εισάγεται ένα παράδειγμα στροφής σε πορεία πλεύσης: Στο εικονιζόμενο παράδειγμα του Σχήματος 2, οι κόκκινες γραμμές παριστούν τις ευθείες που φέρουν τα δύο διαδοχικά Legs, με κοινό σημείο το Waypoint A. Τα βέλη δείχνουν την κατεύθυνση κίνησης του πλοίου, από το ένα Leg στο επόμενο. Η πορεία που ακολουθεί στην πραγματικότητα το σκάφος είναι η γραμμή του πρώτου Leg μέχρι το σημείο J, στη συνέχεια το τόξο JE και τελικώς πλέει και πάλι ευθύγραμμα από το σημείο E επί του επόμενου Leg, απομακρυνόμενο νοτιοδυτικά. Οι γκρι ευθείες, εκατέρωθεν κάθε φορέα, παράλληλες και ισαπέχουσες από αυτόν, παριστούν το νοητό υδάτινο «διάδρομο» (lane) που οριοθετεί την τεθλασμένη γραμμή, αντιπροσωπεύοντας κάποια ανοχή σε αποκλίσεις πορείας.

[70] Το όνομα προέρχεται από το Lateral Navigation, ένα συστατικό στο λογισμικό αυτόματων πιλότων.

[71] Collider, στην ονοματολογία ορισμένων εφαρμογών, είναι ο χώρος δύο ή τριών διαστάσεων, που προκαλεί κάποια ενέργεια (τυπικά κλήση σε κάποια listener μέθοδο) όταν κάποιο άλλο αντικείμενο παραβιάσει τα σύνορά του, πραγματοποιώντας δηλαδή μία σύγκρουση (collision). Από εκεί προέρχεται και το όνομα του κύκλου με κέντρο το waypoint της στροφής, η είσοδος στον οποίο σηματοδοτεί την αρχή της περιστροφικής κίνησης του πλοίου.



Σχήμα 3

Η διαδικασία υπολογισμού των στροφών κατά τη δημιουργία του αντικειμένου πορείας:
Κατά τα γνωστά, η δημιουργία της διαδρομής και η ενθυλάκωσή της σε αντικείμενο Itinerary ξεκινά από την ανάγνωση του διανύσματος συντεταγμένων από το σχετικό αρχείο εισαγωγής με χρήση του αρμόδιου parser. Αυτός πραγματοποιεί μόνο το μέρος των ελέγχων εγκυρότητας εκείνων που αφορούν τη μορφή του διανύσματος, αγνοώντας αυτούς που σχετίζονται με τη «συνολική εικόνα» της διαδρομής. Για το λόγο αυτό, ο parser με το πέρας των ελέγχων του, καλεί όχι τον constructor της Itinerary (που είναι ούτως ή άλλως, private), αλλά μία μέθοδο τάξης (static method) η οποία λαμβάνει λίστα των Waypoints. Εκείνη, μόνο αφού ολοκληρώσει επιτυχώς του υπόλοιπους, υψηλότερου επιπέδου, ελέγχους όπως πλεύση σε στεριά, καλεί τον constructor περνώντας σε αυτόν λίστα των Legs που προκύπτουν από τα Waypoints που δέχτηκε. Ο constructor της τάξης Itinerary πέρα από την ενθυλάκωση της - ελεγμένης και έγκυρης πια - διαδρομής σε ένα αντικείμενο, την εμπλουτίζει υπολογίζοντας πληθώρα πληροφοριών για κάθε στροφή της ώστε να είναι δυνατή η άμεση εκμετάλλευσή τους από το νήμα εξομοίωσης κατά την εκτέλεσή του. Έτσι το thread εκείνο δεν θα αναλάβει τους χρονοβόρους υπολογισμούς αυτούς που θα μπορούσαν να διαρκέσουν περισσότερο από την περίοδο εκπομπής, μειώνοντας την συχνότητα εκπομπών τους.

Οι πληροφορίες αυτές κάθε στροφής που θα επιτρέψουν στο πλοίο να κινηθεί επί της καμπύλης που περιγράφηκε, περικλείονται σε ένα αντικείμενο τύπου Turn. Η επεξήγηση της διαδικασίας προχωρά στον constructor της Itinerary όπου τα χαρακτηριστικά των καμπυλών υπολογίζονται, και το

προκύπτον Turn αποθηκεύεται ως πεδίο του δεύτερου και τελευταίου Waypoint (προορισμός του Leg / "B" στον κώδικα) για κάθε Leg, με εξαίρεση το τελευταίο ευθ. τμήμα. Καλύπτονται έτσι σειριακά όλες οι στροφές της τεθλασμένης, και αποφεύγονται τα άκρα αυτής (απόπλους και τελικός προορισμός του ταξιδιού) καθώς δεν αποτελούν στροφές. Η επεξήγηση εξακολουθεί να χρησιμοποιεί το παράδειγμα στροφής που απεικονίσθηκε, επεκτείνοντάς το παράλληλα με τις νέες έννοιες που εισάγονται. Για να πετύχει τα παραπάνω, ο constructor χρησιμοποιεί επαναληπτική δομή που επεξεργάζεται τα Legs της λίστας που έχει δεχτεί ως παράμετρο, ξεκινώντας από το πρώτο και παραλείποντας το τελευταίο. Κάθε κύκλος επανάληψης ανακτά επίσης και μία αναφορά του επόμενου Leg της λίστας, ώστε διαθέτοντας τα δύο προσκείμενα της στροφής Legs, να είναι σε θέση να υπολογίσει τα ζητούμενα χαρακτηριστικά της. Το «επόμενο» Leg της τεθλασμένης αποτελεί το «τρέχον» στον κύκλο που έπεται, κ.ο.κ.

Ανά κύκλο επανάληψης:

1. Ανακτάται το τρέχον (currentLeg) και το επόμενο (nextLeg) από τη λίστα.
2. Υπολογίζεται το relative bearing του επόμενου σε σχέση με το τρέχον Leg, συγκρίνοντας τα courses τους με τη μέθοδο Azimuth.relativeTo().
3. Το αζιμούθιο του προηγούμενου βήματος απλοποιείται χωριζόμενο σε δύο νέες μεταβλητές, ώστε να αντανακλά «προσημασμένη» (με φορά) στροφή. Το αζιμούθιο μετράται πάντοτε σύμφωνα με τη φορά των δεικτών του ρολογιού και έτσι δεν χρήζει επισήμανσης φοράς / κατεύθυνσης. Από την οπτική του πλοίου κατά την εξομοίωση, οι αριστερές στροφές, έχουν αζιμούθιο με τιμή μεγαλύτερη από 180 μοίρες. Από τέτοιες στροφές, αφαιρούνται 360 μοίρες και η νέα τιμή αποθηκεύεται ως turnAngleUnsigned ενώ η μεταβλητή turnDirection λαμβάνει την τιμή LEFT. Σε διαφορετική περίπτωση, η turnAngleUnsigned διατηρεί την τιμή του βήματος 2, με turnDirection RIGHT. Η enum τιμή του πεδίου turnDirection αποτελεί το «πρόσημο» (sign) που συνοδεύει την τιμή σε μοίρες, turnAngleUnsigned. Η γωνία που περιγράφουν τα νέα πεδία δείχνει την απόκλιση της πορείας μεταξύ των Legs, που στο Σχήμα 3 συμβολίζεται ως Φ. Η γωνία αυτή μπορεί να παριστά δεξιά ή αριστερή στροφή, και ο υπολογισμός της μαζί με φορά περιστροφής, ανταποκρίνεται στον τρόπο με τον οποίο τα πλοία στρέφουν στην πραγματικότητα, όπου ασφαλώς δεν περιορίζονται μόνο σε περιστροφές σύμφωνα με τη φορά των δεικτών του ρολογιού.
4. Υπολογίζεται η περιεχόμενη γωνία των ευθυγράμμων τμημάτων της στροφής, containedAngle, η οποία είναι παραπληρωματική[72] της turnAngleUnsigned. Στο Σχήμα 3, πρόκειται για τη γωνία EAJ.
5. Υπολογίζεται η ακτίνα του κύκλου περιστροφής (turnRadius), ως εξής:
   Τα τρίγωνα TJA και TEA του Σχήματος 4 είναι ίσα καθώς JA = AE = colliderRadius ως ακτίνες του ίδιου κύκλου, TE = TJ = turnRadius ως ακτίνες του κύκλου στροφής και TA κοινή πλευρά. Η ισότητα συνεπάγεται ομοιότητα τριγώνων και κατ' επέκταση οι γωνίες τους που ορίζονται από πλευρές ανάλογες (εν προκειμένω, ίσες) μεταξύ των τριγώνων, είναι επίσης ίσες. Συγκεκριμένα:
   Η TAE (Ψ) είναι ίση με τη JAT και η ETA με την ATJ. Εναλλακτικά, η TA διχοτομεί την EAJ (containedAngle), αλλά και την ETJ. Ακόμη, ίσες είναι και οι γωνίες TEA και TJA. Οι τελευταίες είναι ορθές, αφού οι κορυφές τους βρίσκονται επί του κύκλου στροφής και για κάθε μία οι πλευρές που την ορίζουν είναι αφ' ενός μία ακτίνα του κύκλου στροφής και αφ' ετέρου πλευρά ο φορέας της οποίας είναι εφαπτομένη του κύκλου στροφής με σημείο επαφής την κορυφή της γωνίας. Έτσι, η κοινή για τα παραπάνω ορθογώνια τρίγωνα, TA αποτελεί υποτείνουσά τους.
   Για τον υπολογισμό[73] επιλέχθηκε - αυθαιρέτως - το ορθογώνιο τρίγωνο TEA. Μία επίκεντρη γωνία κύκλου ισούται με τη γωνία μεταξύ των εφαπτόμενων ευθειών του κύκλου της, με σημεία επαφής τα άκρα του τόξου που ορίζει η συγκεκριμένη επίκεντρη γωνία. Από τις τέσσερις γωνίες που σχηματίζει η τομή των δύο εφαπτομένων, η ισότητα με την επίκεντρη ισχύει μόνο για τις δύο: αυτές που ορίζονται από ζεύγη ημιευθειών (με αρχή τους το σημείο

[72] Γωνίες με άθροισμα 180 μοίρες.

[73] Εναλλακτικός υπολογισμός, επί του τριγώνου TEA: Η πλευρά TE (turnRadius) ισούται με το γινόμενο AE (colliderRadius) επί εφαπτομένης της γωνίας TAE, η οποία συμβολίζεται με Ψ στο Σχήμα 4 και αναφέρεται ως halfContainedAngle.

τομής εφαπτομένων) στα οποία ακριβώς και μόνο μία φέρει κάποιο σημείο επαφής με τον κύκλο. Ως εκ τούτου, η γωνία Φ του σχήματος ισούται με την ETJ. Η γωνία ETA του τριγώνου TEA είναι μισή της ETJ. Άρα η ζητούμενη πλευρά TE (turnRadius) υπολογίζεται ως ο λόγος της AE προς την εφαπτομένη της γωνίας ETA.

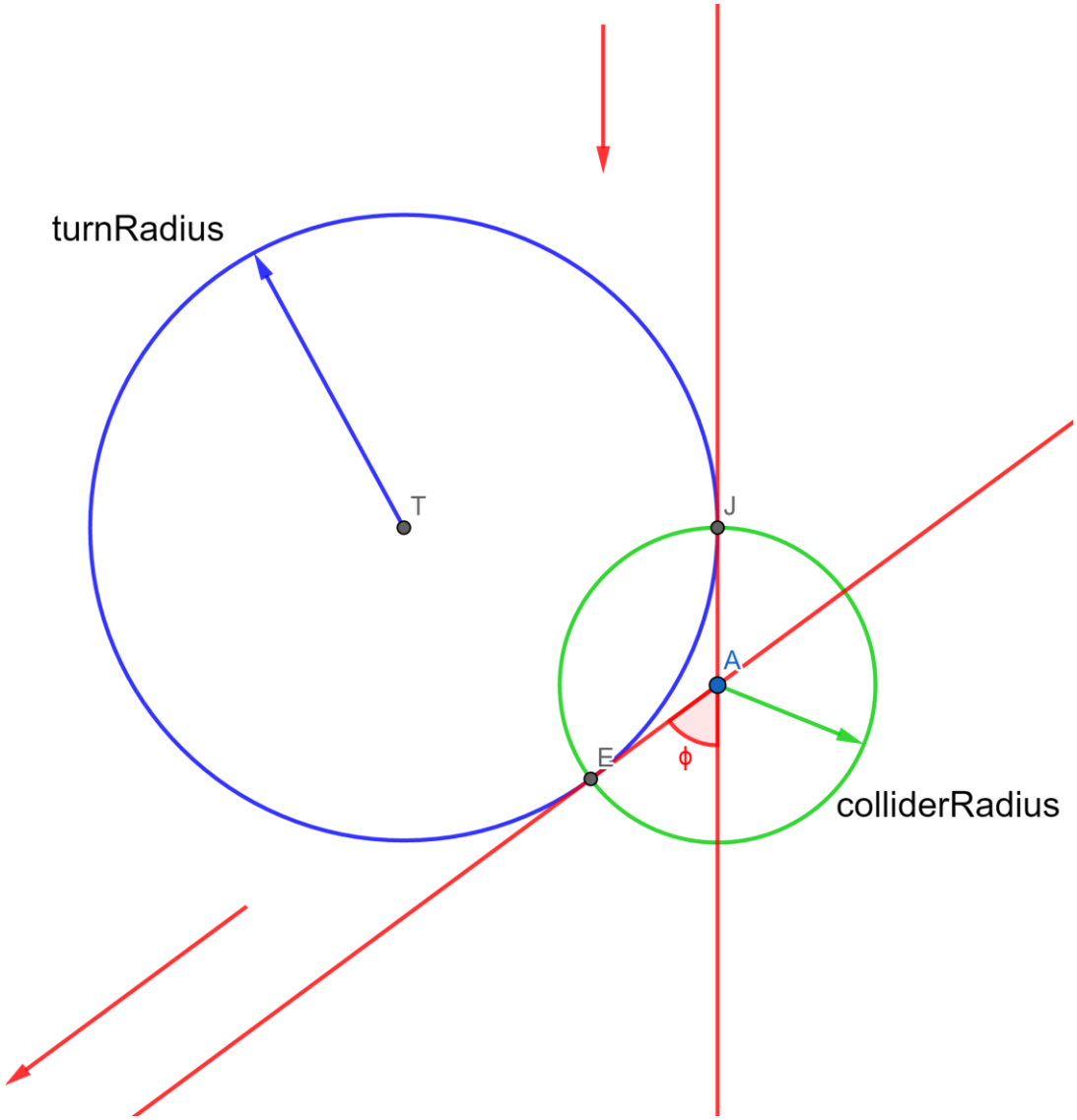
Το τρίγωνο TEA παρέχει επίσης τη σχέση που συνδέει την ακτίνα του κύκλου στροφής με τη γωνία κατά την οποία αλλάζει η κατεύθυνση του πλοίου στη στροφή (turnAngleUnsigned), δεδομένου ότι το μέγεθος colliderRadius = laneWidth/2 είναι σταθερό στο runtime. Ισχύει πως:

a. Γωνία ATJ = Φ/2 = turnAngleUnsigned/2
b. tan(ATJ) = (JA / TJ) <=>
   tan(ATJ) = (colliderRadius / turnRadius)

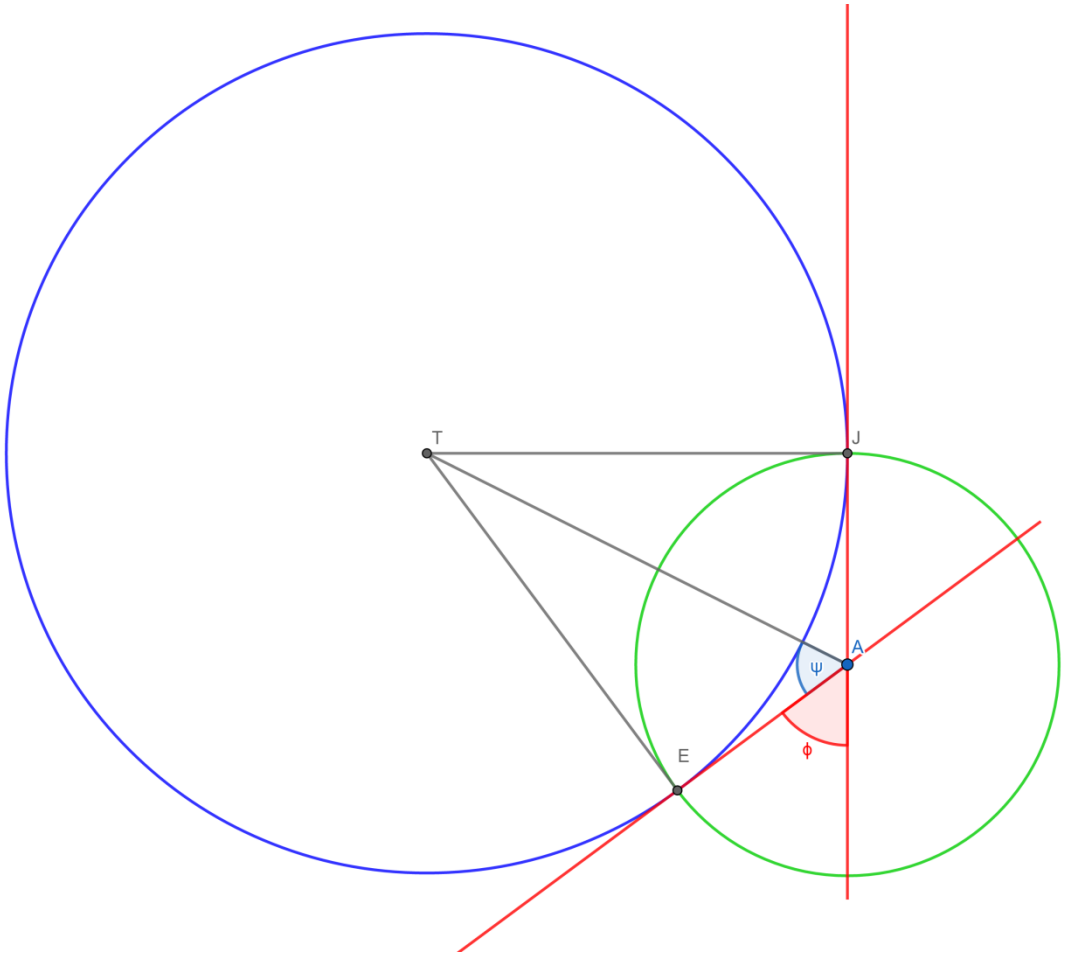
Άρα:
turnRadius = (colliderRadius / tan(Φ/2))

6. Υπολογίζεται το κέντρο του κύκλου περιστροφής, ως εξής:
Χρησιμοποιείται το Πυθαγόρειο θεώρημα επί του τριγώνου ETA για την εύρεση της TA. Στη συνέχεια δημιουργείται ένα νέο αντικείμενο Coordinates στη θέση του δεύτερου και τελευταίου Waypoint του Leg πριν τη στροφή (που αποτελεί άλλωστε την κορυφή της τεθλασμένης στην υπό επεξεργασία στροφή). Στο παράδειγμα, πρόκειται για το σημείο “A”. Δημιουργείται επίσης ένα αντικείμενο Azimuth, το οποίο λαμβάνει τιμή σύμφωνα με το course του ίδιου (προ στροφής), Leg. Για τη συνέχεια ελέγχεται η φορά της στροφής που υπολογίστηκε στο βήμα 3 και το Azimuth αντικείμενο περιστρέφεται, με τις μεθόδους Azimuth.rotateCounterClockwise() ή Azimuth.rotateClockwise() στην ίδια φορά και κατά τη γωνία (Φ + Ψ) του παραδείγματος, ή γενικά (turnAngleUnsigned + halfContainedAngle). Τελικώς το Coordinates αντικείμενο μετακινείται προς την κατεύθυνση του αντικειμένου Azimuth και σε απόσταση ίση με την υποτείνουσα του τριγώνου, TA, καταλήγοντας στο κέντρο του κύκλου περιστροφής. Η κίνηση αυτή πραγματοποιείται χάρη στη μέθοδο Coordinates.move().
7. Υπολογίζεται η θέση του σημείου που αποτελεί την είσοδο στη στροφή. Πρόκειται για το κοινό σημείο του Leg πριν από τη στροφή, βάσει της πλεύσης του σκάφους, με τον κύκλο στροφής. Με άλλα λόγια το «κατώφλι» της στροφής, που στον κώδικα της εφαρμογής αναφέρεται και ως apex[74], ενώ στο εικονιζόμενο παράδειγμα είναι το σημείο J. Για την εύρεσή του, δημιουργείται ένα νέο Coordinates στο πρώτο άκρο του Leg προ στροφής, δηλαδή στην προηγούμενη κορυφή της τεθλασμένης. Το σημείο αυτό μετακινείται, πάλι με χρήση της μεθόδου move(), προς το course του Leg, δηλαδή προς την υπό επεξεργασία στροφή, σε απόσταση όσο το μήκος του προ στροφής Leg, μειωμένο κατά colliderRadius. Η νέα θέση είναι το ζητούμενο σημείο.
8. Δημιουργείται το αντικείμενο στροφής από τις πληροφορίες που υπολογίστηκαν στα παραπάνω βήματα. Γίνεται κλήση στον constructor του Turn εκείνο που δέχεται 5 παραμέτρους, παραλείποντας το πεδίο speedCap, το οποίο όμως λαμβάνει μία αυθαίρετη αρνητική τιμή, προσωρινά. Τελικώς, μετά την επανάληψη, η τελευταία εντολή του Itinerary constructor αποθηκεύει την ενημερωμένη πλέον λίστα των Legs στο μοναδικό πεδίο της τάξης, ολοκληρώνοντας την ενθυλάκωση της διαδρομής.

[74] Ως apex εδώ εννοείται αυτό που πολλές άλλες πηγές αναφέρουν ως "turn-in, ενώ ως apex θεωρούν ένα σημείο κοντά στο μέσον του μήκους του τόξου.



Σχήμα 4



Σχήμα 5

4.5.2: Ο καθορισμός μέγιστης ταχύτητας στις στροφές του νήματος

Τα εν λόγω όρια ταχύτητας επί στροφών δεν αποτελούν θεσμοθετημένους περιορισμούς, η ισχύς των οποίων θα μπορούσε να επιβάλλεται από τη λιμενική αρχή δικαιοδοσίας του αρχιπελάγους. Πρόκειται καθαρά και μόνο για φυσικούς περιορισμούς και συγκεκριμένα για τη μέγιστη γραμμική ταχύτητα που δύναται να αναπτύξει σκάφος δεδομένων χαρακτηριστικών σε στροφή συγκεκριμένης ακτίνας, χωρίς να κινδυνεύσει η ευστάθειά του ή γενικότερα η ασφάλειά του από τον ελιγμό και επίσης χωρίς να παρεκκλίνει της πορείας του, διαγράφοντας παραδείγματος χάρη, τόξο μεγαλύτερης ακτίνας. Σε εξήγηση των παραπάνω αξίζει να επισημανθεί πως ο περιορισμός που μπορεί μία στροφή να επιβάλλει στην ταχύτητα ενός σκάφους είναι αποτέλεσμα της ακτίνας της και όχι του μήκους του τόξου της. Το μήκος αυτό δείχνει απλώς πόσο παρατεταμένη είναι η καμπή και κατ' επέκταση επηρεάζει το χρόνο που θα χρειαστεί το σκάφος για αυτή. Η ακτίνα αντίθετα, αντανακλά πόσο «κλειστή» είναι η στροφή. Ασφαλώς, η μείωση στην ταχύτητα που ίσως επιβληθεί λόγω αυτής, επίσης αυξάνει το χρόνο που θα χρειαστεί το σκάφος για την ολοκλήρωση της πλεύσης στο τόξο, όπως και στην περίπτωση του μήκους.

Για τον υπολογισμό των ορίων, χρησιμοποιούνται κανόνες Φυσικής για την κατάρτιση ενός βασικού μοντέλου το οποίο, αν και παραλείπει μερικές μεταβλητές, λαμβάνει υπ' όψη τις σημαντικότερες. Τα πλοία διατηρούν σε κάθε περίπτωση σταθερή ταχύτητα εντός της στροφής, έχοντας πραγματοποιήσει την ενδεχόμενη επιβράδυνσή τους στο ευθύγραμμο τμήμα πριν το τόξο. Έτσι, η κίνησή τους σε αυτό είναι ομαλή κυκλική. Σε αυτό το είδος κίνησης η γωνιακή επιτάχυνση είναι μηδενική, ενώ τα μεγέθη της γωνιακής αλλά και γραμμικής ταχύτητας παραμένουν σταθερά καθ' όλη την κίνηση. Αμετάβλητα παραμένουν επίσης η ακτίνα και το κέντρο του κύκλου περιστροφής. Η κεντρομόλος επιτάχυνση στην

ομαλή κυκλική κίνηση είναι ακόμη μία σταθερή ποσότητα και δίνεται από τον τύπο $ac = v^2 / r$, όπου: ac = κεντρομόλος επιτάχυνση, v = (γραμμική) ταχύτητα, r = ακτίνα περιστροφής. Πολλαπλασιάζοντας την εξίσωση με τη μάζα του κινούμενου σώματος[75] αυτή γίνεται $Fc = v^2 \times m / r$, όπου: m = μάζα, Fc = κεντρομόλος δύναμη[76]

Από την τελευταία εξίσωση, και με την αξιωματική παραδοχή πως η Fc είναι σταθερή, λαμβάνεται η ζητούμενη ταχύτητα - όριο της στροφής. Η σχέση ακόμη συνδέει την ακτίνα στροφής και τη μάζα του σκάφους, καθιστώντας σαφές πως από αυτές μόνο τις δύο ποσότητες εξαρτάται η ταχύτητα στα πλαίσια της παρούσας ανάλυσης. Σχετικά με την παραδοχή του μοντέλου, η μέγιστη κεντρομόλος δύναμη που μπορεί να αναπτυχθεί στον εγκάρσιο άξονα κάποιου σκάφους μάλλον ποικίλει ανά κατασκευή, αλλά πιθανότατα δεν παρουσιάζει μεγάλη διακύμανση μεταξύ σκαφών και ιδιαίτερα μεταξύ αυτών του ίδιου τύπου, με κοινά υδροδυναμικά χαρακτηριστικά. Η εξωγενής του υποδείγματος δύναμη αυτή ορίζεται καθολικά από το αρχείο ρυθμίσεων της εφαρμογής ως centripetalForce. Αφού η κεντρομόλος των ρυθμίσεων και κατ' επέκταση αυτή που εισάγεται στο μοντέλο είναι η μέγιστη ανεκτή, συνεπάγεται πως η εξαγόμενη ταχύτητα αντίστοιχα είναι η μέγιστη, το όριο ταχύτητας δηλαδή της στροφής. Σε ότι αφορά τη μάζα του πλοίου, φυσικά λογίζεται η συνολική, με όλα τα φορτία του σκάφους.

4.5.3: Η έννοια του άλματος / hop

Το άλμα αποτελεί το βασικό δομικό συστατικό της διαδικασίας εξομοίωσης και παραγωγής εκπομπών. Πρόκειται για σύνολο εντολών συχνά επαναλαμβανόμενων στον κώδικα της run(), καθώς διαφοροποιήσεις ανά τις εκδοχές του εντός του κώδικά της, δεν του επιτρέπουν επαναχρησιμοποίηση ως αυτοτελή μέθοδο. Τα παρόμοια χαρακτηριστικά μεταξύ των διαφόρων μορφών αλμάτων παρουσιάζονται συνοπτικά, ενώ εκείνα που είναι απολύτως κοινά έχουν πράγματι ανατεθεί σε τρείς μεθόδους ιδιωτικής πρόσβασης του Lnav, που παρουσιάζονται αργότερα.
Κατά τα φαινόμενα, τα αντικείμενα AISBroadcast που εκπέμπει ο Lnav στο περιβάλλον του, έκαστο αποτελούν στιγμιότυπα της θέσης του πλοίου για μία δεδομένη στιγμή της εξομοίωσης της πλεύσης του. Αυτές οι εκπομπές είναι και το μοναδικό μέσο που τα νήματα ταξιδιών διαθέτουν για την εξαγωγή πληροφοριών προς το περιβάλλον τους, στο ευρύτερο σύστημα της εφαρμογής. Όμως, μεταξύ δύο διαδοχικών εκπομπών του ίδιου νήματος, η κίνηση του πλοίου δεν είναι συνεχής, ή «αναλογική». Κατ' επέκταση, οι εκπομπές δεν αποτελούν στιγμιότυπα κάποιας τέτοιας κίνησης και δεν είναι δειγματοληπτικώς ανακτημένες. Αντίθετα, η κίνηση είναι «διακριτή», αποτελούμενη από στιγμιαίες μεταβολές στη θέση, κατεύθυνση και ταχύτητα του σκάφους ως αποτέλεσμα υπολογισμών βάσει αρκετών μεταβλητών του σεναρίου εξομοίωσης. Αυτοί οι υπολογισμοί μεταξύ διαδοχικών εκπομπών είναι το σύνολο εντολών του άλματος στη run() και λαμβάνουν χώρα ανά περίοδο εκπομπής (ρύθμιση broadcastInterval), με σκοπό να υπολογίσουν τις μεταβολές σε θέση, ταχύτητα και κατεύθυνση που θα σημείωνε το πλοίο εντός του χρονικού αυτού διαστήματος και να τις ενσωματώσουν στις προηγούμενες αντίστοιχες τιμές τους, δίνοντας την αίσθηση συνεχούς κίνησης που «ανακοινώνεται» ανά τακτά χρονικά διαστήματα από τον Lnav.
Η περιοδικότητα του άλματος οφείλεται στην αναστολή εκτέλεσης του νήματος που το «φιλοξενεί». Η αναστολή αυτή είναι και η τελευταία ενέργεια του συνόλου των εντολών του άλματος. Ο «ύπνος» του νήματος έχει διάρκεια όσο το broadcastInterval μειωμένο κατά το χρόνο που απαιτείται για την ολοκλήρωση των υπολοίπων υπολογισμών του, καθώς ενδέχεται να έχουν διάρκεια[77]. Η διάρκεια των υπολογισμών υπολογίζεται εύκολα καθώς αυτοί βρίσκονται μεταξύ δύο εντολών που λαμβάνουν την τρέχουσα ώρα του συστήματος και στη συνέχεια αφαιρούν την πρώτη μέτρηση από την τελευταία.

Σε ότι αφορά τον κοινό κώδικα των αλμάτων, αυτός έχει ανατεθεί στις μεθόδους:

- private Coordinates waveHit(Coordinates position):
  Ο τύπος Lnav προσομοιώνει, στα πλαίσια των αλμάτων, την επίδραση που έχουν τα καιρικά

[75] Εν προκειμένω, του πλοίου.
[76] Από το δεύτερο νόμο Κινηματικής του Νεύτωνα ισχύει πως $Fc = m \times ac$.
[77] Πολύ μικρή γενικώς, αν και μεγαλύτερη της, αμελητέας εν συγκρίσει, αντίστοιχης διαδικασίας του BroadcastPlayer.

φαινόμενα όπως κυματισμός και άνεμοι στην θέση του σκάφους. Η μέθοδος δέχεται τη θέση του πλοίου και στη συνέχεια λαμβάνει μία τυχαία τιμή δεκαδικού στο ίδιο εύρος με αυτό της καθολικής μεταβλητής turbulence, [0, 1]. Μόνο εάν η τελευταία είναι μεγαλύτερη, υπολογίζει τυχαία μία κατεύθυνση ανέμου ως ακέραιο αζιμούθιο και μία απόσταση ως γινόμενο της περιόδου εκπομπής, της turbulence και μίας ακόμη τυχαίας τιμής στο εύρος [0, 15]. Με κλήση της move() επί της παραμέτρου της, μετατοπίζει το αντικείμενο θέσης στην κατεύθυνση και απόσταση που υπολογίστηκαν. Τελικώς, και ανεξαρτήτως του αποτελέσματος της σύγκρισης στην αρχή της μεθόδου, επιστρέφεται το αντικείμενο που ελήφθη παραμετρικά, στην καλούσα.
Είναι εμφανές πως όσο μεγαλώνει η τιμή της turbulence, τόσο πιο αυξάνεται η δύναμη των κυμάτων, αλλά και η πιθανότητα εμφάνισής τους.

- private void broadcastAIS(Coordinates position, Azimuth heading, double speed):
  Ύστερα από την εκτέλεση των υπολογισμών θέσης, κατεύθυνσης και ταχύτητας, το trip thread χρησιμοποιεί την υπό εξέταση μέθοδο για να τις εξωτερικεύσει προς αποδέκτες εντός της εφαρμογής.
  Οι παραπάνω μεταβλητές παραλαμβάνονται ως παράμετροι από τη μέθοδο, η οποία δημιουργεί από αυτούς αντικείμενο AISBroadcast, ενσωματώνοντας ακόμη σε αυτό την τρέχουσα ημερομηνία / ώρα και αναφορά του Vessel. Από το αντικείμενο καλούνται οι μέθοδοι toConsole() για εμφάνιση των βασικών[78] πληροφοριών της εκπομπής στη γραμμή εντολών και toFile() για εγγραφή των πληροφοριών αυτών στο αρχείο εξόδου AIS Broadcasts.txt. Η τελευταία μέθοδος δέχεται αναφορά του αντίστοιχου File αντικειμένου, στο οποίο εγγράφει την καταχώρηση. Ακόμη, καλώντας μεθόδους των constructor arguments του Lnav, τα οποία έχουν ήδη αποθηκευτεί σε πεδία της τάξης, το αντικείμενο αποστέλλεται στη μηχανή επεξεργασίας γεγονότων της Esper, αλλά και στο υποσύστημα γραφικών gui.
- private void sleep(int milliseconds):
  Η μέθοδος δέχεται ως όρισμα τον ακέραιο αριθμό milliseconds για τον οποίο το καλόν νήμα επεξεργασίας σταματά τη λειτουργία του. Αυτό επιτυγχάνεται με κλήση στην Thread.sleep(), μεταβιβάζοντας σε αυτή την ίδια παράμετρο. Η Thread.sleep() απαιτεί το χειρισμό της μοναδικής δηλωμένης εξαίρεσής της. Καθώς πρόκειται για checked exception, Ο χειρισμός αυτός ελέγχεται από το μεταγλωττιστή, ο οποίος παράγει σφάλματα ελλείψει του. Έτσι λόγω των σχετικά πολλών γραμμών κώδικα που απαιτεί η κλήση αυτή μαζί με το σχετικό χειρισμό, και της κλήσης από κάθε τύπο άλματος της run(), η διαδικασία ανατίθεται στην παρούσα μέθοδο.

Οι εκδοχές των αλμάτων διαφέρουν μεταξύ τους κυρίως σε ότι αφορά ελιγμούς και χειρισμούς που απαιτούνται ανά τις φάσεις του ταξιδιού. Έτσι, τα κυριότερα σημεία διαφοροποίησης μεταξύ τους είναι ο τρόπος καθορισμού της γραμμικής ταχύτητας (επιβράδυνση, σταθερή ταχύτητα, επιτάχυνση ή ακινητοποίηση), της κατεύθυνσης του πλοίου (στροφές / ευθύγραμμη πορεία), αλλά και του τρόπου αντιμετώπισης των παρεκκλίσεων πορείας λόγω ανέμων / κυμάτων. Τα άλματα εξετάζονται ξεχωριστά στα πλαίσια της συνολικής διαδικασίας της μεθόδου run().

4.5.4: Η διαδικασία παραγωγής εκπομπών

Ο constructor της τάξης:
Στον constructor της τάξης Lnav όλες οι παράμετροι που αυτός δέχεται αποθηκεύονται σε αντίστοιχά τους ομώνυμα (και ασφαλώς ίδιου τύπου) πεδία της τάξης, ώστε να είναι καθολικώς προσβάσιμα (global) από όλες τις μεθόδους του τύπου. Οι παράμετροι tripsReady και lock, αποτελούν ένα μέσο επικοινωνίας μεταξύ των νημάτων επεξεργασίας ταξιδιών και εξετάζονται ξεχωριστά στην επόμενη ενότητα του κεφαλαίου.
Στη συνέχεια υπολογίζεται η συνολική μάζα του πλοίου ως το άθροισμα των lightshipWeight, ballast (έρμα), fuel (καύσιμα) και payload (ωφέλιμο φορτίο). Ο πρώτος προσθετέος λαμβάνεται από το

[78] Περιλαμβάνονται όλες οι πληροφορίες του AISBroadcast, με εξαίρεση εκείνες του Vessel, του οποίου εμφανίζεται μόνο το όνομα, πεδίο “name”.

αντικείμενο Vessel, ενώ οι τρεις τελευταίοι από το TripParameters. Ακόμη, λαμβάνεται η σταθερή κεντρομόλος δύναμη από τις ρυθμίσεις. Με χρήση επαναληπτικής δομής επεξεργάζονται όλες οι στροφές της διαδρομής και αναγιγνώσκεται η ακτίνα τους. Αναλόγως αυτής και με βάση το μοντέλο φυσικής που αναπτύχθηκε, υπολογίζεται η μέγιστη ταχύτητα για κάθε στροφή και τίθεται στο πεδίο speedCap (το οποίο μέχρι τη στιγμή αυτή είχε προσωρινή τιμή) του εκάστοτε αντικειμένου Turn, με χρήση της μεθόδου Turn.setSpeedCap().

Η μέθοδος Lnav.run():
Στη μέθοδο run() διατηρούνται για τις ανάγκες της διαδικασίας εξομοίωσης οι τιμές της τρέχουσας θέσης, ταχύτητας και κατεύθυνσης του σκάφους, στις μεταβλητές position, speed και heading, αντίστοιχα. Οι αλγόριθμοι εξομοίωσης μεταβάλλουν διαρκώς τις τιμές των παραπάνω μεταβλητών, οι αρχικές τιμές των οποίων καθορίζονται ως εξής:

- position: Αντιγράφοντας τις τιμές γεωγραφικού μήκους και πλάτους του Waypoint A του πρώτου Leg από το αντικείμενο Itinerary.
- speed: Από το αντικείμενο TripParameters με κλήση στην μέθοδο getInitialSpeed().
- heading: Αντιγράφοντας το course του πρώτου Leg της διαδρομής.

Η μέθοδος προχωρά σε μία επανάληψη που αφορά κάθε Leg της διαδρομής, επεξεργαζόμενη ένα μόνο σε κάθε κύκλο, ξεκινώντας από το πρώτο και συμπεριλαμβάνοντας έως και το τελευταίο. Κάθε κύκλος επανάληψης έχει δύο κύρια μέρη και ένα αρχικό, προπαρασκευαστικό. Τα δύο κύρια μέρη αφορούν κατά σειρά: την πλεύση στο ευθύγραμμο τμήμα του Leg και την πλεύση επί του τόξου στο τέλος του Leg.

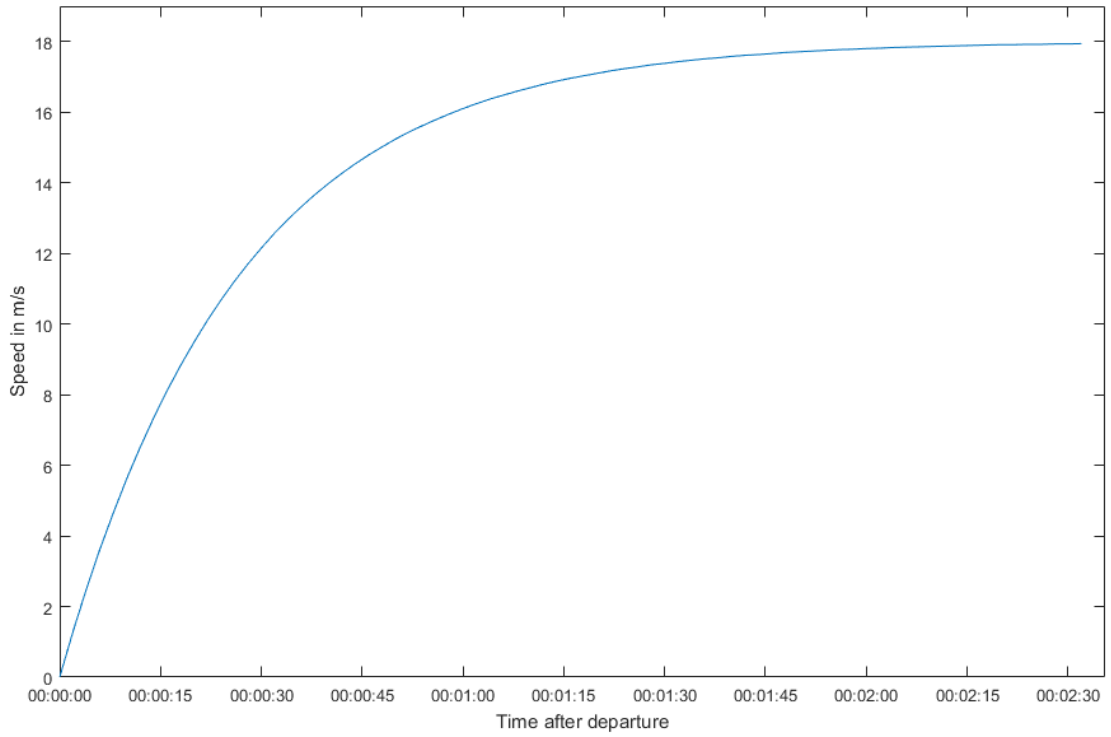
Στο σύντομο, προπαρασκευαστικό πρώτο τμήμα της επανάληψης, λαμβάνεται η αναφορά του Leg του συγκεκριμένου κύκλου, το course του, ο προορισμός του (το πέρας του, Waypoint “B”) και από αυτόν το αντικείμενο Turn που ενθυλακώνεται εκεί. Στη συνέχεια, ελέγχεται το ενδεχόμενο η εν λόγω αναφορά αντικειμένου στροφής να είναι null (χωρίς αντίκρισμα). Όπως έχει προηγουμένως περιγραφεί κατά τη διεργασία ενσωμάτωσης των στροφών στη διαδρομή από τον constructor της Itinerary, αυτό ισχύει πάντα για το τελευταίο Leg της διαδρομής και μόνο για αυτό. Η null τιμή θέτει αληθές το boolean flag lastLeg. Σε διαφορετική περίπτωση:

- Ανακτώνται και αποθηκεύονται οι λεπτομέρειες της στροφής, δηλαδή τα 6 πεδία του Turn, σε ομώνυμες μεταβλητές.
- Η γωνία αλλαγής πορείας, turnAngle (στο παράδειγμα: ΤΕJ = Φ), αποθηκεύεται επίσης σε νέα μεταβλητή με όνομα turnAngleRemainder. Η μεταβλητή αυτή χρησιμοποιείται από τα άλματα περιστροφής για τη διατήρηση του υπολοίπου, σε όρους επίκεντρης γωνίας, που μένει να καλυφθεί.
- Υπολογίζεται η περίμετρος του κύκλου στροφής.
- Υπολογίζεται το absolute bearing από το κέντρο του κύκλου προς το σημείο εισόδου στη στροφή (δηλαδή το σημείο όπου το τόξο του κύκλου στροφής εφάπτεται με το προ στροφής Leg, σημείο J του παραδείγματος) και αποθηκεύεται ως centralAngleAbsoluteAzm.

Η πρώτη εντολή του τμήματος είναι λήψη της τρέχουσας ώρας, η οποία αποθηκεύεται. Με την ολοκλήρωση των εργασιών του προπαρασκευαστικού τμήματος, λαμβάνεται εκ νέου η ώρα του συστήματος για τον υπολογισμό της χρονικής διάρκειάς του, η οποία αποθηκεύεται στην μεταβλητή legInitialDelay.

Για την ευθύγραμμη πλεύση επί του Leg, αρχικά υπολογίζεται η απόσταση μεταξύ της τρέχουσας θέσης και του σημείου εισόδου στη στροφή (το οποίο αναφέρεται και ως apex), distanceToApex. Το υπόλοιπο του παρόντος μέρους αποτελεί μία επαναληπτική δομή, κάθε κύκλος της οποίας είναι και ένα άλμα προς το apex. Οι επαναλήψεις συνεχίζονται όσο η απόσταση προς το σημείο εισόδου της στροφής είναι μεγαλύτερη ή ίση του μηδενός.

Εντός του άλματος, αρχικά αποφασίζεται εάν το πλοίο θα κινηθεί επιταχυνόμενο ή επιβραδυνόμενο. Σε ευθύγραμμη πορεία, το πλοίο πάντοτε βρίσκεται σε επιτάχυνση εκτός εάν η απόστασή του από το apex της στροφής είναι μικρότερη ή ίση από ένα προκαθορισμένο όριο απόστασης και επίσης η τρέχουσα ταχύτητά του ξεπερνά το μέγιστο της στροφής. Το όριο απόστασης, γνωστό ως απόσταση ελέγχου (ταχύτητας) είναι μήκος επί του Leg αμέσως πριν από το σημείο εισόδου στη στροφή. Στην περίπτωση της επιβράδυνσης, η ταχύτητα τίθεται ως το ποσοστό της απόστασης ελέγχου που απομένει να διανυθεί, επί την τελευταία τιμή ταχύτητας του σκάφους και ένας έλεγχος εξασφαλίζει πως η ταχύτητα δεν μειώνεται κάτω από το όριο της επικείμενης στροφής. Αντίθετα, στην περίπτωση της επιτάχυνσης, η ταχύτητα ορίζεται ως η προηγούμενη τιμή της επαυξημένη κατά την διαφορά της από τη μέγιστη δυνατή τιμή της ως ποσοστό της μέγιστης αυτής (τελικής) ταχύτητας, επί το συντελεστή επιτάχυνσης του σκάφους, επί τον αριθμό των δευτερολέπτων μεταξύ διαδοχικών εκπομπών. Η τελική ταχύτητα και ο συντελεστής επιτάχυνσης αποτελούν πεδία του αντικειμένου Vessel του πλοίου. Και στις δύο περιπτώσεις, η ταχύτητα παρουσιάζει ασυμπτωτική συμπεριφορά ως προς την ανεξάρτητη μεταβλητή της, που εντός των πλαισίων της εφαρμογής είναι η απόσταση και ο χρόνος για την επιβράδυνση και επιτάχυνση, αντιστοίχως.

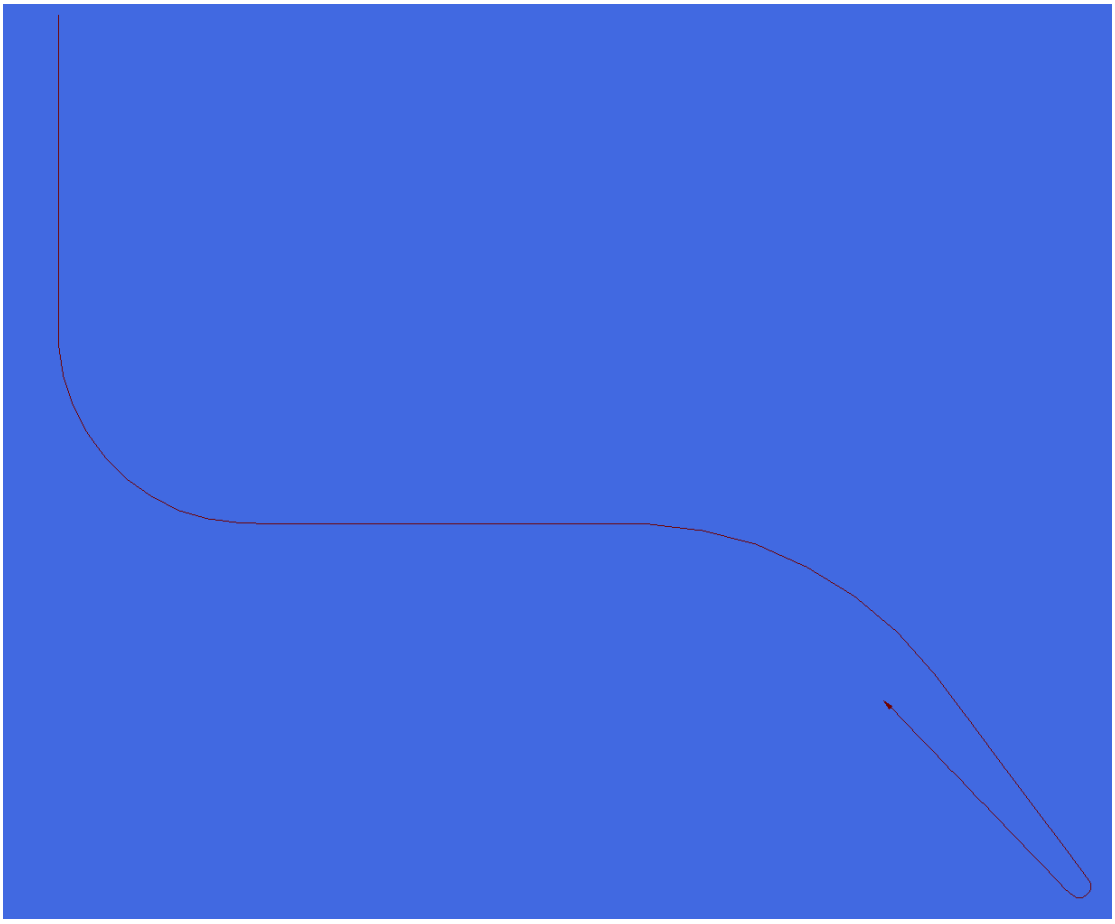


Γράφημα 1

Το Γράφημα 1 παρουσιάζει με ένα παράδειγμα τη μορφή της σχέσης μεταξύ χρόνου (μετά τον απόπλου σε δευτερόλεπτα) με την ταχύτητα (σε m/s). Πρόκειται για ευθύγραμμη κίνηση με αδιάλειπτη επιτάχυνση και περίοδο εκπομπής 2 δευτερολέπτων. Το πλοίο έχει συντελεστή επιτάχυνσης 0.65 και μέγιστη ταχύτητα 18 m/s. Τα διακριτά σημεία που αντιστοιχούν στις εκπομπές έχουν προσεγγιστεί από την καμπύλη του γραφήματος.

Στη συνέχεια υπολογίζεται το μήκος που διανύεται, πολλαπλασιάζοντας την ευρεθείσα, νέα ταχύτητα με την περίοδο εκπομπής σε δευτερόλεπτα. Όμως, το «βήμα» αυτό δεν ενσωματώνεται / προστίθεται στη θέση του σκάφους ακόμη.

Καλείται η waveHit() για προσομοίωση μετατόπισης λόγω κύματος, και ελέγχεται η απόκλιση που προκλήθηκε, ως η απόσταση της νέας θέσης από το ευθύγραμμο τμήμα, Leg. Εάν αυτή είναι μικρότερη μισού laneWidth, το πλοίο βρίσκεται εντός του διαδρόμου πλεύσης του, οπότε στρέφεται (αλλάζει heading) στο bearing του προορισμού του, Waypoint B.
Εάν όμως το πλοίο βρεθεί εκτός του διαδρόμου του, είτε ως αποτέλεσμα μίας πολύ έντονης μετατόπισης ή μερικών διαδοχικών ριπών προς την ίδια κατεύθυνση, τότε τα μέτρα που λαμβάνει για επαναφορά στην πορεία του είναι σαφώς πιο επιθετικά: Εάν η απόσταση του σκάφους από το Leg είναι μεγαλύτερη του βήματος που υπολογίστηκε νωρίτερα, το πλοίο στρέφεται κάθετα προς το ευθύγραμμο τμήμα αυτό, ώστε να πλησιάσει όσο δυνατόν περισσότερο. Εάν είναι μικρότερο, στρέφεται όσο χρειάζεται ώστε να καταλήξει ακριβώς επί του Leg, χωρίς δηλαδή να το διασχίσει. Το παραπάνω αποτέλεσμα είναι συνισταμένη δύο ενεργειών. Αρχικά, με χρήση νοητού ορθού τριγώνου[79], υπολογίζεται από την απόκλιση και το βήμα η οξεία γωνία μεταξύ του τελευταίου και του Leg, φυσικά χωρίς «πρόσημο». Εν συνεχεία, με εύρεση του relative bearing μεταξύ του Leg course και του bearing από το πλοίο στο Waypoint B, αναγνωρίζεται το ημιεπίπεδο στο οποίο βρίσκεται το πλοίο. Πρόκειται για τα ημιεπίπεδα τα οποία ο φορέας του Leg ορίζει, με άλλα λόγια εντοπίζεται εάν το σκάφος έχει παραπλεύσει προς τα αριστερά ή δεξιά. Με αυτόν τον τρόπο συμπεραίνεται η φορά της διορθωτικής περιστροφής, ώστε το πλοίο να βρεθεί ξανά επί της επιθυμητής διαδρομής του και πλησιέστερα στο πέρας του Leg.

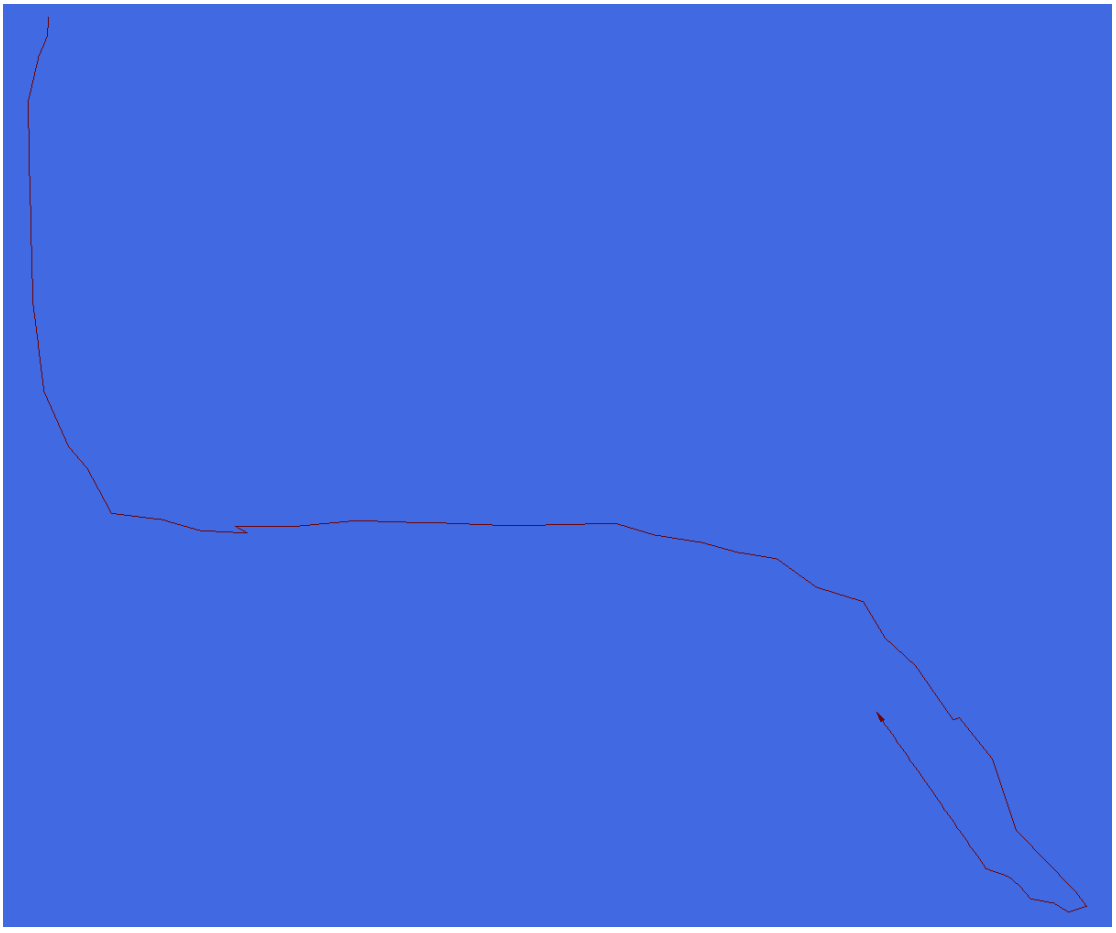


Στιγμιότυπο 1

Τα στιγμιότυπα εικονίζουν κίνηση πλοίου σε τροχιά που έχει εύρος γεωγραφικού μήκους 900 μέτρα και εύρος γεωγραφικού πλάτους 800 μ. Το laneWidth έχει ρυθμιστεί στα 300 μέτρα. Η μοναδική

[79] Τρίγωνο με σημείο τη θέση του πλοίου και πλευρές: step = υποτείνουσα, απόσταση από το Leg = μία κάθετη πλευρά.

διαφορά τους είναι η ρύθμιση turbulence, που έχει τεθεί στο 0 για το Στιγμιότυπο 1 και στο 0.53 για το Στιγμιότυπο 2.



Στιγμιότυπο 2

Καλείται η broadcastAIS() με τις πιο πρόσφατες τιμές διεύθυνσης, θέσης και ταχύτητας για παραγωγή της εκπομπής και διοχέτευσής της στα υπόλοιπα υποσυστήματα. Στη συνέχεια, το πλοίο κινείται με χρήση της Coordinates.move() προς την διορθωτική κατεύθυνση που υπολογίστηκε και σε απόσταση ίση με το ευρεθέν step (βήμα). Η μετακίνηση του πλοίου ύστερα από την διόρθωση της κατεύθυνσής του αλλά και την παραγωγή εκπομπής στο παρόν άλμα, έχει γίνει σκόπιμα: Κατά αυτόν τον τρόπο, μπορεί να γίνει αφ' ενός εκ των προτέρων ο έλεγχος που περιγράφηκε για τη διόρθωση πορείας, και αφ' ετέρου μπορεί να εξασφαλιστεί πως η επανάληψη θα τερματιστεί (βάσει της συνθήκης συνέχισής της) με όλες τις εκπομπές να τοποθετούνται εντός του ευθύγραμμου τμήματος του Leg, και όχι «υποστροφικά» χωρίς δηλαδή να δίνει την εντύπωση πως το πλοίο συνεχίζει την ευθύγραμμη κίνηση επί του τόξου στροφής, στο πρώτο έστω άλμα, προτού συμμορφωθεί. Τελικώς, υπολογίζεται εκ νέου η απόσταση από την είσοδο της στροφής.

Μία ακόμη ιδιαιτερότητα των αλμάτων αυτών είναι πως ο χρόνος απενεργοποίησης του thread συμπεριλαμβάνει και το legInitialDelay, δηλαδή το χρόνο που χρειάστηκε το προηγούμενο, προπαρασκευαστικό μέρος. Ασφαλώς αυτός πρέπει να ληφθεί υπ' όψη μόνο κατά την πρώτη επανάληψη, οπότε μετά την αφύπνιση η μεταβλητή αυτή μηδενίζεται.

Το τρίτο και τελευταίο μέρος της μεθόδου run() εξομοιώνει την πλεύση στο τέλος του Leg, δηλαδή την στροφή με σταθερή ταχύτητα προς το επόμενο ευθύγραμμο τμήμα, ή την ευθύγραμμη επιβραδυνόμενη κίνηση για πρόσδεση στον προορισμό του ταξιδιού. Έτσι, το μέρος αυτό λειτουργεί διαζευκτικά αναλόγως του flag lastLeg, το οποίο και εξετάζεται. Εάν η τιμή του είναι αληθής, πρόκειται για το τελευταίο Leg, οπότε ξεκινά επανάληψη αλμάτων που συνεχίζεται όσο η θέση του πλοίου απέχει από το Waypoint (το τελευταίο της διαδρομής), περισσότερο από μία εσωτερική σταθερά που αναπαριστά μία πολύ μικρή απόσταση. Με άλλα λόγια, η εξομοίωση συνεχίζεται όσο το πλοίο δεν βρίσκεται υπερβολικά κοντά στην ακριβή θέση του προορισμού. Εντός των αλμάτων αυτών:

1. Καλείται η broadcastAIS() για παραγωγή και δημοσίευση του στίγματος του πλοίου. Υπενθυμίζεται πως στο προηγούμενο μέρος η τελευταία μετακίνηση θέσης συνέβη μετά από την τελευταία παραγωγή / μετάδοση εκπομπής, με αποτέλεσμα το πλοίο να έχει ήδη περάσει εντός της περιμέτρου του “collider” κύκλου. Πλέει ήδη πέρα του ευθυγραμμου τμήματος για το οποίο ήταν υπεύθυνο το τμήμα εκείνο.
2. Υπολογίζεται εκ νέου η απόσταση από τον προορισμό.
3. Υπολογίζεται η ταχύτητα ως το γινόμενο της προηγούμενης τιμής της και του λόγου της απόστασης από τον προορισμό προς το ήμισυ του laneWidth (colliderRadius). Στη συνέχεια ελέγχεται εάν η ταχύτητα του πλοίου βρίσκεται εκτός του φάσματος μεταξύ κάτω - άνω ορίων του όπως αυτά ορίζονται στο αντίστοιχο vessel.csv. Εάν αυτό συμβαίνει, η ταχύτητα ορίζεται στο πλησιέστερό της όριο.
4. Υπολογίζεται εκ νέου το absolute bearing από τη θέση του πλοίου προς τον προορισμό και το heading του πλοίου αλλάζει αναλόγως.
5. Υπολογίζεται το βήμα ως το γινόμενο ταχύτητας και χρόνου μεταξύ εκπομπών σε δευτερόλεπτα.
6. Με κλήση στην Coordinates.move(), το πλοίο μετακινείται σε απόσταση του ευρεθέντος βήματος, στην κατεύθυνση του νέου heading.

Στην περίπτωση των Legs εκτός του τελευταίου, για τα οποία το lastLeg είναι ψευδές, πραγματοποιούνται άλματα περιστροφής:

Η ταχύτητα δεν μεταβάλλεται στις στροφές και έτσι δεν υπολογίζεται ξανά στα άλματα περιστροφής. Το βήμα (step), που έχει υπολογιστεί από το δεύτερο μέρος και φέρει τιμή που εξαρτάται από την τελευταία τιμή ταχύτητας που υπολόγισε το τελευταίο άλμα του μέρους εκείνου, παραμένει επίσης σταθερό. Από αυτό, υπολογίζεται η επίκεντρη γωνία του κύκλου στροφής που αντιστοιχεί σε μετακίνηση επί της περιμέτρου του ίδιου κύκλου για μήκος τόξου ίσο με step. Το αποτέλεσμα, angleStep, αποθηκεύεται σε μοίρες καθώς υπολογίζεται ως ο λόγος του βήματος επί 360 προς την περίμετρο.

Τα περιστροφικά άλματα υλοποιούν στα πλαίσια των εργασιών τους τη διεργασία περιστροφής γύρω από το κέντρο του κύκλου στροφής η οποία αναλαμβάνει την μετακίνηση του πλοίου επί του τόξου στροφής σε απόσταση[80] όσο step ενώ επίσης αναλαμβάνει την περιστροφή τους κατά την αντίστοιχη επίκεντρη γωνία, όπως περιγράφηκε προηγουμένως. Υπενθυμίζεται πως η αλλαγή[81] στο heading του πλοίου σε κάθε βήμα επί του τόξου ισούται με την επίκεντρη γωνία του κύκλου στροφής οι ακτίνες της οποίας ενώνουν έκαστη τις διαδοχικές θέσεις που είχε το πλοίο πριν και μετά το άλμα περιστροφής.

[80] Ως τμήμα περιμέτρου του κύκλου, όχι σε ευθεία χορδή.

[81] Εν προκειμένω, η γωνία μεταξύ εφαπτομένων του κύκλου στροφής, με σημεία επαφής διαδοχικές θέσεις του πλοίου.

Η προσέγγιση που ακολουθεί η εν λόγω διεργασία κάνει χρήση του absolute bearing από το κέντρο του κύκλου στροφής προς την εκάστοτε θέση του πλοίου στο τόξο, ως centralAngleAbsoluteAzm. Η νοητή πλευρά του αζιμουθίου που διαρκώς ενώνει τα δύο αυτά σημεία είναι η επιβατική ακτίνα της κίνησης και διαρκώς παρακολουθεί την θέση του πλοίου. Όπως έχει ήδη διατυπωθεί στο προπαρασκευαστικό μέρος, η αρχική θέση της επιβατικής ακτίνας είναι το σημείο εισόδου στη στροφή. Η διεργασία αποτελείται από τα εξής βήματα:

1. Ελέγχεται το turnDirection της στροφής (φορά περιστροφής: δεξιά / αριστερά).
2. Το centralAngleAbsoluteAzm μεταβάλλεται κατά angleStep και σύμφωνα[82] με τη φορά περιστροφής.
3. Το heading του πλοίου μεταβάλλεται κατά angleStep και σύμφωνα με τη φορά περιστροφής.
4. Δημιουργείται ένα νέο αντικείμενο Coordinates, ονόματι positionFinder, στην ίδια θέση με το κέντρο του κύκλου στροφής.
5. Το αντικείμενο του προηγούμενου βήματος μετακινείται σε απόσταση turnRadius και προς centralAngleAbsoluteAzm, έχοντας έτσι εντοπίσει τη νέα θέση του σκάφους στο τόξο.
6. Η θέση του positionFinder εγγράφεται στη θέση του πλοίου.

Το υπόλοιπο των εργασιών του μέρους αυτού εκτείνεται σε τρεις υπο-διαδικασίες, όπου κατά σειρά, υπολογίζεται το πρώτο άλμα, στη συνέχεια όλα τα ενδιάμεσα και τελικώς το τελευταίο άλμα περιστροφής. Ο λόγος που το πρώτο και το τελευταίο αντιμετωπίζονται ξεχωριστά είναι πως συνήθως τα βήματά τους, δηλαδή οι αποστάσεις που καλύπτονται κατά τα άλματα αυτά, αν και έχουν όπως σε όλα τα άλματα στροφής σταθερό μήκος, πολύ συχνά αυτά επικαλύπτουν μέρος του ευθυγράμμου τμήματος και μέρος του τόξου περιστροφής ταυτοχρόνως. Έτσι, στην περίπτωση του πρώτου άλματος, πρέπει να αφαιρεθεί το τμήμα του βήματος που διανύθηκε ευθύγραμμα μέχρι το σημείο εισόδου στη στροφή και το υπόλοιπό του να διανυθεί επί του τόξου. Αντίθετα, κατά το τελευταίο άλμα στροφής, θα πρέπει το υπόλοιπο μήκος βήματος εκτός τόξου στροφής να είναι ευθύγραμμο επί του επόμενου Leg.

Το πρώτο άλμα περιστροφής, η τροχιά του οποίου συνήθως θυμίζει το γράμμα “J” λόγω του ευθύγραμμου μέρους του και της καμπύλης του, αναλύεται στα συστατικά του. Για την εύρεση της εκπομπής που αντιστοιχεί σε αυτό το άλμα, αρκεί να βρεθεί η καμπύλη συνιστώσα της εν λόγω τροχιάς. Καθώς η πιο πρόσφατη μετακίνηση έχει γίνει κατά το προηγούμενο μέρος και είναι ευθύγραμμη (η μετακίνηση μετά την τελευταία παραγωγή εκπομπής), πρέπει να βρεθεί το τμήμα του προηγούμενου εκείνου βήματος που βρίσκεται πέραν του σημείου εισόδου στη στροφή και προς το Waypoint αυτής. Το ζητούμενο μήκος δίνεται από την αφαίρεση της απόστασης μεταξύ της τελευταίας θέσης και του εν λόγω Waypoint (που έχει ήδη υπολογιστεί από το προηγούμενο μέρος) από το colliderRadius. Πρόκειται για το μήκος του τόξου του πρώτου άλματος περιστροφής, firstCurvedComponent. Η επίκεντρη του κύκλου στροφής ακτίνα που αντιστοιχεί σε αυτό, firstAngledStep, υπολογίζεται με τον ίδιο τρόπο που απέδωσε την επίκεντρη γωνία του step. Οι περαιτέρω υπολογισμοί του παρόντος άλματος εκτελούνται μόνο εάν η firstAngledStep γωνία είναι μικρότερη από την συνολική επίκεντρη γωνία όλου του τόξου περιστροφής, turnAngleRemainder. Σε διαφορετική περίπτωση η επανάληψη του τρέχοντος Leg συνεχίζεται με την υπο-διαδικασία των ενδιάμεσων αλμάτων του τόξου. Από την εναπομείνουσα γωνία turnAngleRemainder αφαιρείται η γωνία που αντιστοιχεί στο firstAngleStep. Στη συνέχεια εκτελείται η διεργασία περιστροφής με τη διαφορά πως η γωνία κατά την οποία το πλοίο περιστρέφεται γύρω από το κέντρο περιστροφής turnPoint ακολουθώντας το τόξο στροφής είναι firstAngleStep και όχι angleStep. Τελικώς καλείται η waveHit() για προσομοίωση κύματος και στη συνέχεια η broadcastAIS() για δημοσίευση των αποτελεσμάτων στο περιβάλλον του τύπου.

[82] Από το τύπου Azimuth αντικείμενο καλείται η rotateClockWise() για δεξιές στροφές, αλλιώς η rotateCounterClockWise().

Η υπο-διεργασία που αφορά τα ενδιάμεσα άλματα είναι ουσιαστικά μία επαναληπτική δομή που εκτελείται όσο το υπόλοιπο της γωνίας στροφής turnAngleRemainder είναι μικρότερο του angleStep. Κάθε επανάληψή της πραγματοποιεί ένα από τα ενδιάμεσα περιστροφικά άλματα, σε σειρά σύμφωνη με την κίνηση του πλοίου. Σε κάθε επανάληψη:

1. Το υπόλοιπο turnAngleRemainder μειώνεται κατά angleStep.
2. Πραγματοποιείται η διεργασία περιστροφής γύρω από το κέντρο του κύκλου για γωνία angleStep.
3. Καλείται η waveHit() από τη νέα θέση του πλοίου.
4. Καλείται η broadcastAIS().

Η τελευταία υπο-διεργασία αφορά το τελευταίο περιστροφικό άλμα, που τυπικά μετακινεί το πλοίο σε τροχιά όμοια με αυτή του πρώτου, επίσης σχηματίζοντας μία καμπύλη “J” στο χάρτη. Το αντίστοιχο step του άλματος αναλύεται στα συστατικά του: Η καμπύλη συνιστώσα, lastAngleStep, είναι απλώς το υπόλοιπο της στροφής, turnAngleRemainder. Το μήκος του τόξου αυτού υπολογίζεται από την ισότητα των λόγων:

- μήκος τόξου lastAngleStep / περίμετρος κύκλου στροφής
- lastAngleStep (η γωνία είναι σε μοίρες) / 360

Από την αφαίρεση μήκος τόξου lastAngleStep από το step, προκύπτει το ευθύγραμμο τμήμα που εκτείνεται από το σημείο εξόδου της στροφής και επί του επόμενου Leg, lastStraightComponent. Στη συνέχεια:

1. Πραγματοποιείται η διεργασία περιστροφής γύρω από το κέντρο του κύκλου για τη γωνία lastAngleStep. Με το πέρας αυτής της ενέργειας, το πλοίο βρίσκεται τοποθετημένο στο σημείο εξόδου της στροφής, επί του επόμενου ευθύγραμμου τμήματος.
2. Το πλοίο μετακινείται προς το νέο του heading (το οποίο συμπίπτει πλέον με το course του επόμενου Leg) και σε απόσταση lastStraightComponent.
3. Καλείται η waveHit() από τη νέα θέση του πλοίου.
4. Καλείται η broadcastAIS().

Δύο σημεία ενδιαφέροντος:

1. Σε αντίθεση με την ευθύγραμμη πλεύση, δεν πραγματοποιούνται ελιγμοί διόρθωσης ύστερα από κλήσεις στη waveHit(). Αυτό διότι η επόμενη θέση του πλοίου υπολογίζεται, όχι με αφετηρία την προηγούμενη θέση του, αλλά το άλμα που ακολουθεί στην διαδικασία περιστροφής, τοποθετεί το σκάφος επακριβώς επί του τόξου στροφής, στην προβλεπόμενη από τη σταθερή ταχύτητα θέση, και με heading ανάλογο του ποσοστού του τόξου που έχει διανυθεί, όπως έχει τεκμηριωθεί από τη διεργασία περιστροφής γύρω από το κέντρο του κύκλου στροφής.
2. Το κριτήριο εκτέλεσης του πρώτου περιστροφικού άλματος καθώς και η συνθήκη τερματισμού της επανάληψης των ενδιάμεσων αλμάτων, βασίζονται σε συγκρίσεις μεταξύ του εκάστοτε υπολοίπου της γωνίας κατά την οποία το πλοίο πρέπει να περιστραφεί προς ολοκλήρωση της στροφής, και του τόξου ως μέρος αυτής της γωνίας που καλύπτεται σε κάθε άλμα. Αυτό σημαίνει πως εάν το step είναι αρκετά μεγάλο[83] και / ή η στροφή έχει μικρή ακτίνα, ενδέχεται να μην υπάρξουν ενδιάμεσα άλματα, ενώ σε πιο ακραίες περιπτώσεις ούτε το πρώτο άλμα. Σε κάθε περίπτωση η στροφή πραγματοποιείται χωρίς πρόβλημα.

Με τα παραπάνω ολοκληρώνεται ο ρόλος της run() και η διαδικασία εξομοίωσης πλεύσης από τον τύπο Lnav.

[83] Ως αποτέλεσμα υψηλής ταχύτητας / μεγάλης περιόδου εκπομπών.

4.6: Ταύτιση συχνότητας και φάσης των νημάτων παραγωγής εκπομπών

Στην προηγούμενη ενότητα έγινε αναφορά, μεταξύ άλλων, σε ένα σύστημα εξασφάλισης της μη μεταβολής της περιόδου εκπομπής από νήματα γνησίως εξομοιούμενων ταξιδιών. Αυτά εκτελούν σύνθετους, εν συγκρίσει με τα BroadcastPlayer, υπολογισμούς οι οποίοι ενδέχεται να διαρκούν αρκετά, ειδικά σε αργά συστήματα. Για το λόγο αυτό, η περίοδος απενεργοποίησης του νήματος πρέπει να λαμβάνει υπ' όψη τη διάρκεια των υπολογισμών αυτών. Διαφορετικά, ο χρόνος που μεσολαβεί μεταξύ εκπομπών αυξάνεται κατά την εν λόγω διάρκεια, πέραν της επιθυμητής παύσης (που ο χρήστης όρισε ως broadcastInterval) και συνεπώς η συχνότητα μειώνεται.
Κατά την εισαγωγή της έννοιας των αλμάτων, είχε διατυπωθεί πως οι εντολές τους ξεκινούν και τελειώνουν με τη λήψη της τρέχουσας ώρας συστήματος. Οι αντίστοιχες τιμές πριν και μετά τους υπολογισμούς αποθηκεύονται σε μεταβλητές τύπου long καθώς η τρέχουσα ώρα λαμβάνεται με κλήση σε μέθοδο που επιστρέφει την ημερομηνία και ώρα, ως τον αριθμό των milliseconds από τα μεσάνυκτα της 1ης Ιανουαρίου 1970. Η αφαίρεση της τελευταίας λήψης από την πρώτη δίνει τον αριθμό των sec × 10^{-3} που έχουν παρέλθει κατά την διάρκεια των υπολογισμών του εκάστοτε άλματος. Η αρχή και το τέλος ενός άλματος υλοποιούνται στην εφαρμογή ως εξής:

```
calculationStart = System.currentTimeMillis();

        ... υπολογισμοί άλματος ...

calculationEnd = System.currentTimeMillis();
timeTaken = (int)(calculationEnd - calculationStart);
if(timeTaken < BROADCAST_INTERVAL) sleep(BROADCAST_INTERVAL -
timeTaken);
```

Η διάρκεια απενεργοποίησης είναι ο χρόνος που μεσολαβεί μεταξύ εκπομπών βάσει ρυθμίσεων μειωμένος κατά τη διάρκεια του άλματος, εξασφαλίζοντας έτσι πως η συνολική χρονική διαφορά μεταξύ διαδοχικών εκπομπών είναι όσο η τιμή της ρύθμισης. Καθώς ο χρόνος «ύπνου» του νήματος δεν μπορεί να είναι αρνητικός, ελέγχεται πως η διάρκεια των υπολογισμών δεν υπερβαίνει την περίοδο των ρυθμίσεων. Σε περίπτωση που οι υπολογισμοί διαρκούν περισσότερο από το broadcastInterval, το thread δεν προβαίνει σε απενεργοποίηση, άλλωστε δεν[84] είναι δυνατό να δοθεί αρνητικός χρόνος στην sleep(). Αντί αυτού, απλώς συνεχίζει με το επόμενο άλμα. Δυστυχώς σε τέτοια περίπτωση υπάρχει καθυστέρηση στο χρονοδιάγραμμα των εκπομπών ίση με τη διαφορά των δύο συγκρινόμενων μεγεθών, ενώ η εφαρμογή δεν μπορεί να λάβει άλλα μέτρα για την εξασφάλιση συνεπούς συχνότητας καθώς το νήμα επεξεργασίας είναι ήδη διαρκώς ενεργό. Μία εξωσυστημική λύση θα ήταν η παροχή αυξημένης προτεραιότητας στα νήματα παραγωγής εκπομπών και στο JVM της sailAway γενικότερα. Ενώ υπάρχουν κάποιοι σχετικοί μηχανισμοί[85], αυτοί εξαρτώνται από το εκάστοτε λειτουργικό σύστημα το οποίο έχει τον τελικό λόγο στην πολιτική χρονοδιαμοιρασμού νημάτων (scheduling). Μια πιο απλή, αξιόπιστη αν και συμβιβαστική λύση (workaround) είναι να αυξηθεί το broadcastInterval. Παρόλο που έτσι η πλεύση γίνεται πιο «διακριτή» / σωρευτική και λιγότερο ομαλή, δίνεται περισσότερος χρόνος στο σύστημα να ολοκληρώσει τους υπολογισμούς του άλματος πριν χρειαστεί να πραγματοποιηθούν οι υπολογισμοί του επόμενου.

Οι παραπάνω κινήσεις συνθέτουν μία ικανοποιητική λύση στο θέμα της συνέπειας της συχνότητας εκπομπών, που αποτελεί και το σημαντικότερο ζήτημα συγχρονισμού τους. Μία άλλη απαίτηση που υπάρχει πέρα από αυτή της συχνότητας, είναι αυτή της ταυτόχρονης έναρξης / απόπλου των πλοίων, η οποία όμως δεν εξασφαλίζεται από την παραπάνω προσέγγιση. Η διαφορά φάσης εκπομπών μεταξύ νημάτων μπορεί να προκύψει από:

1. Διαφορά εργασιακού φόρτου μεταξύ ειδών ταξιδιών, σε ότι αφορά τις προπαρασκευαστικές εργασίες τους μέχρι την έναρξη της επαναληπτικής δομής που παράγει τις εκπομπές.

[84] Η Lnav.sleep() αναθέτει την απενεργοποίηση στην Thread.sleep(), η οποία εγείρει IllegalArgumentException εάν λάβει αρνητικές τιμές ως παραμέτρους.
[85] https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.html#setPriority-int-

2. Ζητήματα χρονοδιαμοιρασμού: οφείλονται στο λειτουργικό σύστημα από το οποίο τρέχει η εφαρμογή.
3. Περιορισμοί στους πόρους συστήματος.

Χωρίς να λαμβάνονται υπ' όψιν οι αιτίες της ασύγχρονης εκκίνησης των δρομολογίων, η λύση που προτιμήθηκε εξασφαλίζει αμελητέες αποκλίσεις στις χρονικές στιγμές απόπλου μεταξύ όλων των ταξιδιών του σεναρίου εξομοίωσης, ανεξαρτήτως τύπου. Πριν την παρουσίασή της, εξηγείται συνοπτικά το μέρος εκείνο του τεχνολογικού υποβάθρου στο οποίο βασίζεται ως υλοποίηση: Η Java υποστηρίζει πολυνηματικότητα εγγενώς, χωρίς να εξαρτάται από εξωτερικές βιβλιοθήκες. Στα πλαίσια συγγραφής multi-threaded εφαρμογών σε αυτή τη γλώσσα, και προς αποφυγή αλλοίωσης πληροφοριακού περιεχομένου πεδίων, ως αποτέλεσμα της ταυτόχρονης εγγραφής τους από πολλαπλά threads, χρησιμοποιούνται τεχνικές που καθιστούν τα τμήματα κώδικα με τις αντίστοιχες εντολές επεξεργασίας προσπελάσιμες από ένα μόνο thread σε κάθε δεδομένη στιγμή, εκμηδενίζοντας την πιθανότητα παράλληλης προσπέλασης ευαίσθητων πεδίων.
Μία δημοφιλής σχετική πρακτική είναι η χρήση "synchronized" μπλοκ κώδικα. Πρόκειται για δηλωμένες, συνεχείς περιοχές εντολών εντός των ορίων μεθόδων. Κάθε τέτοιο μπλοκ κώδικα είναι ρητά συσχετισμένο με ένα lock object, δηλαδή κάποιο αντικείμενο[86] που λειτουργεί ως mutex / «σκυτάλη». Πριν την εκτέλεσή του μπλοκ, το υποψήφιο thread υποχρεώνεται να δεσμεύσει το αντίστοιχο lock object. Εάν αυτό είναι διαθέσιμο, δεσμεύεται πράγματι από το εν λόγω νήμα, το οποίο προχωρά κατ' αποκλειστικότητα στην εκτέλεση του μπλοκ και αποδεσμεύει το αντικείμενο όταν ολοκληρώσει την εκτέλεση των εντολών σε αυτό. Στην περίπτωση που κάποιο νήμα προσπαθήσει να τρέξει κάποιο μπλοκ του οποίου το mutex είναι ήδη δεσμευμένο, θα πρέπει να περιμένει έως ότου αυτό αποδεσμευτεί. Όταν κάποιο νήμα «αποδεσμεύει» ένα lock, μπορεί εκείνο να δεσμευτεί στη συνέχεια από κάποιο άλλο «ανταγωνιστικό» νήμα το οποίο προτίθεται να εκτελέσει κάποιο από τα συσχετισμένα με αυτό μπλοκ. Μπορεί λοιπόν το ίδιο lock object να «κλειδώνει» περισσότερα του ενός μπλοκ, αλλά κάθε μπλοκ σχετίζεται με μόνο ένα lock.
Με τη χρήση αυτής της τεχνικής, το σύνολο των εργασιών εντός τέτοιων μπλοκ καθίσταται ατομικό[87], τουλάχιστον μεταξύ των threads της εφαρμογής όχι απαραίτητα μεταξύ όλων των νημάτων στον scheduler του συστήματος. Φυσικά αυτό δεν αποτελεί πρόβλημα: τα νήματα άλλων εφαρμογών δεν έχουν πρόσβαση στις διευθύνσεις μνήμης του JVM.
Μεταξύ τεχνικών συγχρονισμού των νημάτων, η παρούσα προσέγγιση διαθέτει ακόμη τη δυνατότητα κλήσης μεθόδων - υποτυπώδους - επικοινωνίας μεταξύ των threads που ανταγωνίζονται τη δέσμευση του ίδιου mutex. Συγκεκριμένα, καθώς όλα τα αντικείμενα στη Java μπορούν να αποτελέσουν mutexes στα πλαίσια της παρούσας τεχνικής, ο αρχαιότερος της κληρονομικής ιεραρχίας τύπος, java.lang.Object, ορίζει τις μεθόδους που κληρονομούνται από κάθε τάξη της γλώσσας:

- wait(): Σταματά την εκτέλεση του καλούντος νήματος, το απενεργοποιεί άμεσα και αποδεσμεύει το lock object.
- notify(): Ενεργοποιεί τυχαία ένα από τα απενεργοποιημένα νήματα που απενεργοποιήθηκαν ως αποτέλεσμα κλήσης τους στη wait() από το ίδιο lock object. Η κλήση της μεθόδου αυτής, αν και καθιστά επιλέξιμο από τον scheduler κάποιο από τα νήματα του lock, δεν σταματά το νήμα που την κάλεσε.
- notifyAll(): Η μοναδική διαφορά της με την notify() είναι πως αφορά όλα τα απενεργοποιημένα νήματα του ίδιου lock αντικειμένου.

Οι παραπάνω μέθοδοι καλούνται, εάν απαιτείται, σε περιπτώσεις όπου η επικοινωνία μεταξύ threads εξυπηρετεί τη διαδικασία του συγχρονισμού τους, και πάντοτε εντός του synchronized block και επίσης από το lock αυτού, από νήματα που το έχουν επιτυχώς δεσμεύσει. Κατόπιν της

[86] Μπορεί να είναι οποιοδήποτε reference type της Java, ακόμη και το java.lang.Object, ή αναφορά προς αντικείμενο που περιέχει τη μέθοδο με synchronized code block.
[87] Ατομική χαρακτηρίζεται μία διαδικασία που δεν μπορεί να επιμεριστεί σε βασικότερες υπο-διεργασίες, καθώς η εκτέλεσή της πραγματοποιείται σε ένα πολύ απλό και αδιάσπαστο βήμα. Καταχρηστικά, και στα πλαίσια του thread-safety, η έννοια χρησιμοποιείται και για διεργασίες οι οποίες αν και μπορούν τεχνικά να διακοπούν / επιμεριστούν, η συνέχισή τους ύστερα από παύση ή εκτέλεση των επιμέρους τμημάτων τους, αντίστοιχα, πραγματοποιείται από το ίδιο νήμα επεξεργασίας, αποφεύγοντας τα προβλήματα της ταυτόχρονης προσπέλασης δεδομένων.

«απενεργοποίησης» που προκαλεί η wait(), τα νήματα μπορούν και πάλι να ενεργοποιηθούν εάν συμβεί κάποιο από τα παρακάτω:

- Κλήση άλλου νήματος στην notifyAll() από το ίδιο lock object που απενεργοποίησε το νήμα νωρίτερα.
- Κλήση άλλου νήματος στην notify() από το ίδιο lock object που απενεργοποίησε το νήμα νωρίτερα, με το (τυχαία) επιλεγμένο προς ενεργοποίηση νήμα να είναι το ζητούμενο.
- Κλήση άλλου νήματος στην μέθοδο interrupt() του απενεργοποιημένου νήματος και κατάλληλο χειρισμό του γεγονότος αυτού, ως InterruptedException (ή με έλεγχο από τις interrupted() / isInterrupted(), για ενεργά νήματα μόνο).
- Εκπνοή της διάρκειας κατάστασης απενεργοποίησης (στην περίπτωση που έχει χρησιμοποιηθεί η υπερφορτωμένη εκδοχή της wait() που δέχεται παράμετρο timeout).

Προχωρώντας στην ανάλυση επί της μεθόδου main(), αρχικοποιείται εκεί μία λίστα παραμετρικού τύπου Thread, με σκοπό να συγκεντρώσει όλες τις αναφορές των νημάτων ταξιδιών. Ακόμη δημιουργείται ένα αντικείμενο, ονόματι lock, σε ρόλο mutex / «σκυτάλης» μεταξύ των νημάτων αυτών. Αρχικοποιούνται επίσης με τιμή μηδέν οι ακέραιοι lnavCount, banditCount και tripCount. Ο πρώτος αποτελεί μετρητή των συνολικών έγκυρων Lnav αντικειμένων που δημιουργούνται στο σενάριο εξομοίωσης, ο δεύτερος μετρά τα προδιαγεγραμμένα ταξίδια και ο τελευταίος αποθηκεύει το άθροισμα των δύο προηγούμενων. Μεγαλύτερο ενδιαφέρον στη διαδικασία συγχρονισμού παρουσιάζει μία ακόμη μεταβλητή, ονόματι tripsReady, τύπου java.util.concurrent.atomic.AtomicInteger με αρχική τιμή μηδέν. Πρόκειται για τύπο που ενθυλακώνει έναν ακέραιο αριθμό και παρέχει μεθόδους επεξεργασίας του που είναι ατομικές, καθιστώντας τον τύπο thread-safe.

```
LinkedList<Thread> trips = new LinkedList<Thread>();
Object lock = new Object();
int lnavCount = 0;
int banditCount = 0;
int tripCount = 0;
AtomicInteger tripsReady = new AtomicInteger(0);
```

Στη συνέχεια η main() προχωρά με την ανάγνωση πληροφοριών των ταξιδιών από τα αντίστοιχα αρχεία εισαγωγής και την επαλήθευση της ορθότητας των εισαγωγών. Στο τέλος έκαστων των διαδικασιών ανά τύπο ταξιδιού, δημιουργούνται τα αντικείμενα ταξιδιού Lnav / BroadcastPlayer τα οποία αμφότερα υλοποιούν το interface Runnable. Τα αντικείμενα αυτά δημιουργούνται ανώνυμα (χωρίς ονοματισμένο handle, “pointer”) και δίνονται ως ορίσματα constructor, έκαστο σε επίσης ένα ανώνυμο Thread αντικείμενο, το οποίο την ίδια στιγμή τοποθετείται στην λίστα trips. Η λίστα αυτή αποκτά σημασία καθώς είναι πλέον το μοναδικό μέσο επικοινωνίας με τα ανώνυμα αντικείμενα που περιέχει. Η δημιουργία ανώνυμων αντικειμένων δεν διευκολύνει χειρισμούς, αλλά είναι πρακτικά μονόδρομος σε προγράμματα που δημιουργούν μεταβλητό πλήθος τους ανά εκτέλεση. Με κάθε δημιουργία ταξιδιού αυξάνεται κατά ένα ο αντίστοιχος μετρητής, βάσει τύπου ταξιδιού. Στο τέλος των δύο διαδικασιών (γνησίως εξομοιούμενα και προδιαγεγραμμένα ταξίδια) οι μετρητές αθροίζονται για την εύρεση του συνόλου, tripCount, των διαφορετικών ταξιδιών / νημάτων επεξεργασίας της εφαρμογής πλέον του main thread.

Τη στιγμή της πρόθεσης αυτής, είναι βέβαιο πως ο κώδικας στους constructors των δύο τύπων και για όλα τα διαφορετικά ταξίδια έχει εκτελεστεί και μάλιστα με τη σειρά δημιουργίας τους, καθώς στο σημείο αυτό υπάρχει μόνο το main νήμα.

```
trips.add(new Thread(new Lnav(vessel, tripParameters, route, map,
engine, aisLogFile, tripsReady, lock)));
lnavCount++;

trips.add(new Thread(new BroadcastPlayer(vessel, sequence, map,
engine, aisLogFile, tripsReady, lock)));
banditCount++;
```

```
tripCount = lnavCount + banditCount;
```

Με διαπέραση όλων των στοιχείων της trips, καλείται η μέθοδος start() κάθε Thread / ταξιδιού σε αυτή, σειριακά. Η κλήση στην start() ενός Thread προκαλεί την εκτέλεση της run() του αντικειμένου αυτού σε ξεχωριστό[88] νήμα επεξεργασίας.

```
for(Thread t : trips){
      t.start();
}
```

Το πρόβλημα είναι πως τα νήματα ξεκινούν με τη σειρά να εκτελούν προπαρασκευαστικές πράξεις (δηλώσεις μεταβλητών, ανάκτηση αρχικών τιμών κ.λ.π.) που προηγούνται της κύριας επανάληψης που παράγει τις εκπομπές, με τα Lnav να εκτελούν σημαντικότερο όγκο τέτοιων εργασιών σε σχέση με τα BroadcastPlayer, και τα νήματα που κλήθηκαν πρώτα ανεξαρτήτως τύπου να έχουν ένα προβάδισμα. Η χρονική διαφορά είναι γενικώς ανεπαίσθητη, αλλά μπορεί να διευρυνθεί σε σενάρια με μεγαλύτερο αριθμό πλοίων.
Η λύση είναι το synchronized μπλοκ που τοποθετείται στη run() πριν την κύρια επανάληψη, αλλά μετά από τις εργασίες προετοιμασίας. Το μπλοκ είναι το ίδιο στους δύο τύπους:

```
synchronized(lock){
      tripsReady.incrementAndGet();
      try{
            lock.wait(25000);
      } catch(InterruptedException ie){
            ie.printStackTrace();
      }
}
```

Επιτρέπει σε κάθε ταξίδι, που τρέχει έκαστο σε ξεχωριστό νήμα, ανεξαρτήτως τύπου, να εκτελέσει τις προπαρασκευαστικές εργασίες του, ενώ με το πέρας αυτών και αφού κάθε νήμα εξασφαλίσει τη σκυτάλη, τρέχει ατομικά και χωρίς να διακοπεί από τα υπόλοιπα, το παραπάνω μπλοκ στο οποίο εκτελούνται με την παρακάτω σειρά οι ενέργειες:

1. Ο tripsReady αυξάνεται κατά ένα. Ο αριθμός αυτός συμβολίζει το πλήθος των νημάτων που έχουν ολοκληρώσει τις προπαρασκευαστικές τους εργασίες και είναι έτοιμα για παραγωγή εκπομπών.
2. Το thread απενεργοποιείται με κλήση στην wait() από το mutex, lock.

Υπενθυμίζεται πως τα tripsReady και lock έχουν δημιουργηθεί στη main() και αποτελούν τοπικές μεταβλητές της, αναφορές των οποίων έχουν περιέλθει στα αντικείμενα ταξιδιών ως constructor arguments, κατά τη δημιουργία τους. Ακόμη, ίσως η χρήση ενός AtomicInteger (tripsReady) δεν ήταν απολύτως επιβεβλημένη καθώς η τιμή του αλλάζει μόνο μέσα σε synchronized block, κλειδωμένο με κοινό για όλα τα νήματα lock...

Πίσω στη main(), και μετά από την μαζική εκκίνηση των νημάτων, το κύριο νήμα της εφαρμογής «κοιμάται» για ένα εικοστό του δευτερολέπτου, επαναλαμβανόμενα και όσες φορές χρειαστεί μέχρις ότου η τιμή του tripsReady φθάσει το σύνολο των δημιουργηθέντων νημάτων ταξιδιών, tripCount. Στη συνέχεια η μέθοδος εκτελεί το δικό της synchronized block, συσχετισμένο με το γνωστό lock object, η δέσμευση του οποίου συμβαίνει αυτομάτως και χωρίς αναμονή, αφού καθώς η συνθήκη της παραπάνω επανάληψης ικανοποιήθηκε, όλα τα trips έχουν απενεργοποιηθεί και απωλέσει το mutex. Εντός του μπλοκ, απλώς καλείται η notifyAll() του κοινού lock object.

[88] Απ' ευθείας κλήση στην run() προκαλεί την εκτέλεση του κώδικά της, αλλά στο ίδιο νήμα με την καλούσα της.

```
do{
	try{
		Thread.sleep(50);
	} catch(InterruptedException ie){
		ie.printStackTrace();
	}
} while(tripsReady.get() < tripCount);
synchronized(lock){
	lock.notifyAll();
}
```

Έτσι επαν-ενεργοποιούνται μαζικά όλα τα νήματα του σεναρίου που στο σύνολό τους είχαν ήδη ολοκληρώσει τις λοιπές αρχικές προαπαιτούμενες εργασίες και περίμεναν, αμέσως πριν τη φάση παραγωγής εκπομπών.

Τελικώς, ως συνέπεια της χρήσης πολυνηματικού κώδικα υπάρχει ο κίνδυνος κάποιο γονικό thread να ολοκληρώσει τις εργασίες του πριν από τα threads που το ίδιο δημιούργησε, και να τερματιστεί, «σκοτώνοντας» και τα νήματα εκείνα πριν ολοκληρωθούν. Στην περίπτωση της sailAway, το main thread δεν έχει πια άλλες εργασίες προς ολοκλήρωση, αλλά έχει μόλις δημιουργήσει ως νήματα - απογόνους του τα ταξίδια του σεναρίου εξομοίωσης, που βρίσκονται μόλις σε φάση απόπλου. Προς αποφυγή του πρόωρου τερματισμού της εφαρμογής, θα πρέπει η μέθοδος main() να «παύσει» τη λειτουργία της μέχρις ότου τα πλοία φθάσουν όλα στον προορισμό τους. Η αναμονή για την ολοκλήρωση ενός νήματος από το γονικό του γίνεται με κλήση της μεθόδου join() από το αντικείμενο του υπο-νήματος. Στην παρούσα εφαρμογή όπου τρέχουν πολλαπλά και ανώνυμα τέτοια νήματα, η συλλογή trips και το “for-each” loop της Java αποδεικνύουν και πάλι την αξία τους:

```
for(Thread t : trips){
	try{
		t.join();
	} catch(InterruptedException e){
		e.printStackTrace();
	}
}
```

Αξίζει να παρατηρηθεί πως ακόμη και χωρίς το παραπάνω μπλοκ κώδικα, η main() παραδόξως περιμένει την ολοκλήρωση όλων των ταξιδιών πριν τερματιστεί.

Κεφάλαιο 5: Αναγνώριση σύνθετων ναυσιπλοϊκών γεγονότων

Περιεχόμενα κεφαλαίου

5.1: Το υπόβαθρο λειτουργίας των ερωτημάτων EPL

5.1.1: Εκκίνηση και παραμετροποίηση της μηχανής αναγνώρισης σύνθετων γεγονότων

Πριν τους χειρισμούς που αφορούν αμιγώς τη μηχανή επεξεργασίας σύνθετων γεγονότων, υπενθυμίζεται πως οι απαραίτητες προϋποθέσεις της παροχής των .jar αρχείων της Esper και των εξαρτήσεών της, αλλά και της κατάδειξης της θέσης τους στο JVM είναι ήδη εκπληρωμένες, όπως έχει περιγραφεί. Ακόμη, καθώς η διαδικασία αναγνώρισης γεγονότων αποτελεί τον καθοριστικό[89] παράγοντα παραγωγής εξόδου από την εφαρμογή, περιγράφεται εδώ συνοπτικά η προετοιμασία των καταλόγων και αρχείων εξόδου: Ύστερα από διεργασίες στη main() που αφορούν την εύρεση της τρέχουσας ημερομηνίας / ώρας και μορφοποίησής τους προς ονοματοδοσία του φακέλου εκτέλεσης, δημιουργείται η δομή φακέλων υπό τον outputFiles. Εντός αυτού και υπό τον φάκελο εκτέλεσης, περιέχεται το AIS Broadcasts.txt όπου καταγράφονται όλα τα LLEs των νημάτων ταξιδιών. Στον ίδιο φάκελο δημιουργείται και ο υποφάκελος Event Reports όπου εγγράφονται τα αναγνωρισμένα HLEs, σε αρχεία αναφοράς ανά EPL ερώτημα. Τα αρχεία εκείνα εξετάζονται για κάθε ερώτημα, εξατομικευμένα.

```
String sep = System.getProperty("file.separator");
String runInstanceFolder = weekDay + " the " + monthDay +
monthDaySuffix + " of " + restOfDate;
File aisOutputPath = new File(".." + sep + ".." + sep + ".." + sep +
"outputFiles" + sep + runInstanceFolder);
File eventReportOutputPath = new File(aisOutputPath + sep + "Event
Reports");
eventReportOutputPath.mkdirs();
File aisLogFile = new File(aisOutputPath + sep + "AIS
Broadcasts.txt");
```

[89] Εάν φυσικά εξαιρεθούν βοηθητικές / δευτερεύουσες έξοδοι, όπως αποσφαλμάτωση στην κονσόλα και παράθυρο γραφικών.

Η διαδικασία αναγνώρισης σύνθετων γεγονότων με χρήση Esper στα πλαίσια μίας εφαρμογής Java, απαιτεί ένα instance της μηχανής αυτής, καθώς και παράγωγων αντικειμένων της που διεξάγουν τους απαιτούμενους χειρισμούς επί ερωτημάτων αλλά και λοιπές εργασίες παραμετροποίησης. Η αρχικοποίηση των οντοτήτων αυτών και η δήλωση του AISBroadcast ως τύπο δεδομένων που παριστά γεγονός, από την οπτική του Esper API:

```
EPServiceProvider esperEngine =
EPServiceProviderManager.getDefaultProvider();
EPAdministrator esperAdministrator = esperEngine.getEPAdministrator();
ConfigurationOperations esperConfiguration =
esperAdministrator.getConfiguration();
esperConfiguration.addEventType(AISBroadcast.class);
```

Πέραν της δυνατότητας προσπέλασης πεδίων των αντικειμένων AISBroadcast από τη μηχανή και παρά το μεγάλο εύρος των συναρτήσεων με το οποίο διατίθεται εξοπλισμένη η παρούσα διανομή της Esper, οι ανάγκες κάποιων ερωτημάτων δεν καλύπτονται. Αυτό γενικότερα είναι αναμενόμενο αποτέλεσμα του πολύ μεγάλου εύρους σεναρίων στα οποία το λογισμικό γεγονότων αυτό προορίζεται για χρήση, αλλά και των εγγενών, ιδιαίτερων χαρακτηριστικών κάθε εφαρμογής. Τη λύση παρέχουν μηχανισμοί εσωτερικοποίησης δεδομένων και διαδικασιών προερχόμενων από την περιβάλλουσα εφαρμογή, προς το EPL runtime. Στα πλαίσια των μηχανισμών αυτών, αξιοποιούνται δύο μέθοδοι από την sailAway, η addVariable() για χρήση μεταβλητών / πεδίων / βαθμωτών, και η addPlugInSingleRowFunction() για χρήση συναρτήσεων. Εξετάζοντας τις μεθόδους λεπτομερέστερα:

- ConfigurationOperations.addVariable()
  Η υπερφόρτωση[90] της μεθόδου δέχεται τρία ορίσματα, κατά σειρά από αριστερά:
  1. Το αναγνωριστικό με το οποίο η μεταβλητή αναφέρεται / καλείται εντός κώδικα EPL.
  2. Ο τύπος της μεταβλητής στο περιβάλλον της Java εφαρμογής, ως java.lang.Class αντικείμενο.
  3. Η αρχική τιμή της μεταβλητής.
- ConfigurationOperations.addPlugInSingleRowFunction()
  Η υπερφόρτωση[91] της μεθόδου δέχεται τρία ορίσματα, κατά σειρά από αριστερά:
  1. Το αναγνωριστικό με το οποίο η συνάρτηση αναφέρεται / καλείται εντός κώδικα EPL.
  2. Το πλήρες (fully-qualified) όνομα του τύπου όπου υλοποιείται η συνάρτηση στην εφαρμογή, ως java.lang.String.
  3. Το αναγνωριστικό της συνάρτησης όπως υλοποιείται στον τύπο της παραμέτρου (2). Η μέθοδος θα πρέπει να είναι ορισμένη ως public static.

Οι καλούμενες από τα EPL statements συναρτήσεις, δέχονται τις ίδιες παραμέτρους και επιστρέφουν το ίδιο τύπο με εκείνες που έχουν αντιστοιχισθεί από την addPlugInSingleRowFunction().

Οι μεταβλητές και συναρτήσεις που εσωτερικεύονται και καθίστανται διαθέσιμες στο EPL περιβάλλον της sailAway, εξετάζονται ξεχωριστά σε κάθε ερώτημα που εξαρτάται από αυτές, με εξαίρεση εκείνες που χρησιμοποιούνται από πολλαπλά ερωτήματα, οι οποίες παρουσιάζονται πριν από το σύνολο των ερωτημάτων της εφαρμογής.

[90] http://esper.espertech.com/release-6.1.0/esper-javadoc/com/espertech/esper/client/ConfigurationOperations.html#addVariable-java.lang.String-java.lang.Class-java.lang.Object-

[91] http://esper.espertech.com/release-6.1.0/esper-javadoc/com/espertech/esper/client/ConfigurationOperations.html#addPlugInSingleRowFunction-java.lang.String-java.lang.String-java.lang.String-

5.1.2: Συσχέτιση EPL - ερωτήματος – listener

Κάθε ερώτημα εκφράζεται σε κώδικα EPL, οποίος αποθηκεύεται σε αλφαριθμητικό (java.lang.String). Για να τεθεί σε ισχύ, απαιτείται κατ' ελάχιστο η εγγραφή / διοχέτευσή του στη μηχανή αναγνώρισης σύνθετων γεγονότων, η οποία θα το μετατρέψει στον αντίστοιχο τύπο δεδομένων και θα αναλάβει στη συνέχεια τον εντοπισμό των HLEs, κατά τον τρόπο που περιγράφουν τα EPL statements του αλφαριθμητικού. Φυσικά, υπάρχει και η πρόσθετη απαίτηση της παραγωγής κάποιου είδους απόκρισης σε συνέχεια αναγνώρισης ενός γεγονότος, κατά συγκεκριμένο και εξατομικευμένο τρόπο για κάθε ερώτημα. Όπως έχει ήδη αναφερθεί, αυτό επιτυγχάνεται με τη χρήση listeners οι οποίοι θα πρέπει επίσης να συσχετιστούν με το αντίστοιχο ερώτημα. Η καταχώρηση ενός ερωτήματος στην Esper γίνεται με την κλήση EPAdministrator.createEPL()[92]. Η μέθοδος δέχεται ως παράμετρο το αλφαριθμητικό του ερωτήματος EPL και πέρα από την καταχώρησή του στην μηχανή και ενεργοποίησή του, επιστρέφει το ερώτημα στον τύπο με τον οποίο ενθυλακώνεται στο περιβάλλον της Esper, EPStatement. Η συσχέτιση του ερωτήματος με τον listener γίνεται με κλήση στην μέθοδο EPStatement.addListener()[93], από το αντικείμενο του ερωτήματος. Τελικώς, η συσχέτιση μεταξύ listener και αρχείου εξόδου γίνεται με παροχή αναφοράς του τελευταίου ως constructor argument κατά τη δημιουργία του πρώτου. Οι παραπάνω συσχετίσεις - συνδέσεις είναι αμφιμονοσήμαντες και τα ακριβή βήματα υλοποίησής τους εξετάζονται σε κάθε ερώτημα. Θα πρέπει να σημειωθεί πως δεν συσχετίζεται κάθε EPL statement με κάποιο listener εάν αυτό αποτελεί κάποιο «ενδιάμεσο» / προπαρασκευαστικό χειρισμό για αποτελέσματα άλλων statements.

5.1.3: Παραγωγή αναφορών HLEs

Σε ότι αφορά την παραγωγή εξόδου από τους listeners, η εφαρμογή χρησιμοποιεί διαφορετικό τύπο δεδομένων για κάθε ένα - τελικό - ερώτημα EPL που έχει γραφεί στα πλαίσιά της. Οι τύποι αυτοί είναι τοπικώς ορισμένοι από την εφαρμογή, εξασφαλίζοντας ευελιξία στην μορφή των αναφορών που εγγράφονται στην έξοδο. Ασφαλώς, παρόλο που είναι τοπικώς εγγεγραμμένοι, οφείλουν να διαθέτουν υπόσταση «γνώριμη» στη μηχανή της Esper, χάριν αποστολής πληροφοριών σε αυτούς, σχετικών με αναγνωρισμένα γεγονότα. Το χάσμα γεφυρώνεται με την υλοποίηση του UpdateListener[94] από τους listeners, interface ορισμένου από την Esper. Με την πρόσθετη αυτή ιδιότητα, τα datatypes υποχρεούνται σε σαφή (concrete) ορισμό της μεθόδου με signature void update(EventBean[] newEvents, EventBean[] oldEvents). Την μέθοδο αυτή καλεί η μηχανή της Esper ύστερα από κάθε αναγνώριση του αντίστοιχου σύνθετου γεγονότος. Οι παράμετροι της μεθόδου είναι δύο EventBean arrays, interface αναπαράστασης γεγονότων. Αντικείμενα του είδους επιτρέπουν την ανάκτηση των ιδιοτήτων των αντιπροσωπευόμενων LLEs εκείνων που εμπλέκονται / συνθέτουν το HLE που περιγράφεται στο ερώτημα, για την ταυτοποίησή τους και τελικώς την εγγραφή τους σε ανθρωπίνως αναγνώσιμη μορφή. Η ανάκτηση αυτή τυπικά γίνεται χάρη στη μέθοδο EventBean.get(), ενώ αναφορά στο πηγαίο αντικείμενο παρέχεται με κλήση στην getUnderlying() του ίδιου τύπου. Στο array που αναφέρεται ως newEvents περιέχεται το σύνολο των γεγονότων που εισέρχονται στο παράθυρο[95] επεξεργασίας ερωτήματος Esper, τη στιγμή της κλήσης της μεθόδου, ενώ στο oldEvents όσα εξέρχονται του παραθύρου.

Ενώ κάθε sailAway listener γράφει έκαστος στο συσχετισμένο αρχείο του αναφορές μορφολογικά εξαρτώμενες από τη φύση του αντίστοιχου HLE του, η διαδικασία εγγραφής παραμένει κοινή μεταξύ των τύπων. Αυτή, δικαιολογεί την και ύπαρξη ενός κοινού κληρονομικού αφηρημένου προγόνου,

[92] http://esper.espertech.com/release-6.1.0/esper-javadoc/com/espertech/esper/client/EPAdministrator.html#createEPL-java.lang.String-
[93] Μέθοδος του EPStatement, που κληρονομήθηκε από το interface EPListenable: http://esper.espertech.com/release-6.1.0/esper-javadoc/com/espertech/esper/client/EPListenable.html#addListener-com.espertech.esper.client.UpdateListener-
[94] http://esper.espertech.com/release-6.1.0/esper-javadoc/com/espertech/esper/client/UpdateListener.html
[95] Για την έννοια του παραθύρου και λοιπών θεμελιωδών ζητημάτων επεξεργασίας δεδομένων από την Esper: http://esper.espertech.com/release-6.1.0/esper-reference/html_single/index.html#processingmodel

AISEventListener, και τη μεταφορά της σε αυτόν ως τη μέθοδο public synchronized void writeToOutputFile(String input).

Πριν την εστίαση στη μέθοδο αυτή, αξίζει μία αναφορά στα μέλη του AISEventListener: Πεδία:

- private File eventReport:
  Χρησιμεύει στην αποθήκευση αναφοράς του αρχείου εξόδου του συγκεκριμένου listener / ερωτήματος. Παρόλο που η τιμή του πεδίου προέρχεται από τους απογόνους - listeners, δεν καθίσταται κληρονομικώς διαθέσιμο σε αυτούς, αφού το eventReport χρησιμοποιείται μόνο από την AISEventListener.writeToOutputFile().
- protected String lineBreak:
  Αποθηκεύει τον χαρακτήρα αλλαγής γραμμής[96] του συστήματος στο οποίο τρέχει η εφαρμογή.

Ο μοναδικός constructor του AISEventListener, δέχεται ως όρισμα αναφορά του αρχείου εξόδου, η οποία έχει μεταβιβαστεί με constructor chaining από τον εκάστοτε concrete listener, χρησιμοποιώντας ένα super[97] statement. Από το όρισμα αυτό λαμβάνει τιμή το αντίστοιχο πεδίο, ενώ το lineBreak ορίζεται με την κλήση: System.getProperty("line.separator").

Η μέθοδος writeToOutputFile() δέχεται ως είσοδο αλφαριθμητικά προς εγγραφή στο αρχείο εξόδου (πεδίο AISEventListener.eventReport). Για την προσπέλαση του αρχείου χρησιμοποιεί ένα FileWriter instance, αρχικοποιημένο ώστε να κατευθύνει την έξοδο στο αρχείο σε append mode, διατηρώντας δηλαδή τις παλαιότερες εγγραφές σε αυτό ύστερα από νέες προσθήκες. Το FileWriter βρίσκεται ενθυλακωμένο σε ένα BufferedWriter αντικείμενο, εξασφαλίζοντας καλύτερες επιδόσεις και μειωμένες απαιτήσεις συστήματος. Για τη χρήση των παραπάνω τύπων προσπέλασης αρχείων και το χειρισμό των εξαιρέσεων των μεθόδων τους, χρησιμοποιείται ένα try-with-resources[98] statement όπως συμβαίνει και στο υποσύστημα parsers, αυτοματοποιώντας διαδικασίες όπως τερματισμό ρευμάτων από / προς αρχεία.

Στην τρέχουσα μορφή της εφαρμογής, μόνο το thread που τρέχει τη μηχανή Esper καλεί τους listeners, ακόμη και όταν αναγνωρίζονται πολλαπλά HLEs του ίδιου είδους, ταυτόχρονα. Παρόλα αυτά, για λόγους που καθιστούν την sailAway συμβατή με κάθε τροπή μελλοντικών αποφάσεων ανάπτυξης, επιλέχθηκε κάθε εγγραφή HLE instance σε αρχείο να αποτελεί ατομική / αδιάσπαστη ενέργεια.

Ο στόχος επιτυγχάνεται με δύο κινήσεις:

1. Δήλωση της μεθόδου ως synchronized. Η προσέγγιση αυτή χρησιμοποιεί ως mutex το ίδιο το αντικείμενο, «κλειδώνοντάς» το ολόκληρο. Καθώς στην παρούσα υλοποίηση η AISEventListener έχει μόνο μία μέθοδο, η λύση αυτή δεν προκαλεί καθυστερήσεις.
2. Χρήση του lineBreak στο String - παράμετρο της writeToOutputFile(), για εγγραφή αλφαριθμητικών που εκτείνονται σε πολλαπλές γραμμές, με μία μόνο κλήση.

Η μέθοδος είναι πλέον synchronized, και εάν η λειτουργία της διακοπεί πριν την ολοκλήρωσή της, μόνο το ίδιο νήμα που διακόπηκε σε αυτή θα συνεχίσει την εκτέλεσή της. Καθώς όμως υφίσταται και η απαίτηση για ατομικότητα / μη επιμερισμό της εγγραφής κάθε HLE instance στο σύνολό της, αυτή δεν θα πρέπει να εξαρτάται από πολλαπλές κλήσεις της writeToOutputFile(). Διαφορετικά θα μπορούσε, παραδείγματος χάρη, να γραφεί ένα μέρος της με μία κλήση, αλλά πριν μπορέσει να εξασφαλιστεί το mutex για τις επόμενες κλήσεις, να μεσολαβήσουν τμήματα άλλων instances στο αρχείο. Τελικώς τα γεγονότα θα καταγραφούν στο ακέραιό τους αλλά τα τμήματά τους δεν είναι εγγυημένο πως θα είναι διαδοχικά. Ασφαλώς τα αρχεία εξόδου θα ήταν δυσανάγνωστα στην καλύτερη περίπτωση. Η λύση που προτιμήθηκε είναι η χρήση του lineBreak χαρακτήρα ώστε οι πολλαπλών γραμμών εγγραφές να πραγματοποιούνται με την ίδια κλήση της μεθόδου, και όχι τμηματικά με πολυάριθμες.

[96] Carriage return / Line feed.
[97] https://docs.oracle.com/javase/tutorial/java/IandI/super.html
[98] https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html

5.1.4: Τερματισμός της Esper

Στο τέλος της μεθόδου main() και ύστερα από τη διαδικασία που αναγκάζει τη μέθοδο να περιμένει την ολοκλήρωση των trip threads, τερματίζεται η λειτουργία της μηχανής Esper με την κλήση:

```
engine.destroy();
```

5.1.5: Στοιχεία κοινής προσπέλασης, εσωτερικευμένα στο EPL περιβάλλον

Συνάρτηση “distance”:
Η συνάρτηση αντιστοιχεί στην public static Double calculateDistance(Coordinates a, Coordinates b) της τάξης Coordinates. Επιστρέφει την ευκλείδεια απόσταση δύο σημείων του καρτεσιανού επιπέδου, των ορισμάτων της a και b. Για την ακρίβεια υπολογίζει την απόλυτη τιμή της διαφοράς γεωγραφικού μήκους των δύο σημείων καθώς και την απόλυτη τιμή της διαφοράς γεωγραφικού πλάτους αυτών. Στη συνέχεια, υπολογίζει την τετραγωνική ρίζα του αθροίσματος των τετραγώνων των δύο παραπάνω διαφορών την οποία και επιστρέφει ως Double (boxed, όχι primitive). Με άλλα λόγια χρησιμοποιεί το Πυθαγόρειο θεώρημα για τον υπολογισμό της υποτείνουσας που ενώνει τα a, b. Σε περίπτωση που οποιοδήποτε από τα δύο ορίσματα είναι null, η calculateDistance() επιστρέφει επίσης null. Αυτός είναι και λόγος που ο τύπος επιστροφής δεν είναι double: Οι συναρτήσεις με primitive return type δεν μπορούν να επιστρέψουν κενό. Φυσικά το implicit auto-(un)boxing της Java εξασφαλίζει απροβλημάτιστη λειτουργία με consumer code τέτοιων συναρτήσεων.

Συνάρτηση “angle”:
Η συνάρτηση αντιστοιχεί στην public static Double angleFinder(Azimuth a, Azimuth b) της τάξης Azimuth. Επιστρέφει διαφορά αζιμουθίου σε μοίρες μεταξύ των ορισμάτων της a και b. Για την ακρίβεια υπολογίζει το relative bearing του δευτέρου από το πρώτο, με κλήση στην μέθοδο αντικειμένου Azimuth.relativeTo(). Εάν το ευρεθέν (σε μοίρες) ξεπερνά τις 180, τότε λαμβάνεται το υπόλοιπό του από τις 360. Αυτό ασφαλώς γίνεται καθώς τα πλοία μπορούν να περιστραφούν προς τις δύο φορές και όχι μόνο δεξιόστροφα, όπως μετράται ένα αζιμούθιο. Τελικώς, επιστρέφεται το αποτέλεσμα ως boxed Double. Σε περίπτωση που οποιοδήποτε από τα δύο ορίσματα είναι null, η angleFinder() επιστρέφει επίσης null.

Συνάρτηση “timeDifference”:
Η συνάρτηση αντιστοιχεί στην public static Double esper_findTimeDifference(Date lastDate, Date lastDate) της τάξης Application. Επιστρέφει χρονική διαφορά σε δευτερόλεπτα μεταξύ των ορισμάτων της lastDate και lastDate. Συγκεκριμένα, μετατρέπει σε milliseconds από την ώρα εκκίνησης χρόνου Unix (Μεσάνυκτα $1^{ης}$ Ιανουαρίου, 1970) τις αποθηκευμένες τιμές των ορισμάτων της, τύπου Date. Υπολογίζει τη διαφορά του νεότερου timestamp από το παλαιότερο, τη μετατρέπει σε δευτερόλεπτα και την επιστρέφει ως Double (boxed). Η μετατροπή σε δευτερόλεπτα είναι απαραίτητη για τη συμμόρφωση με τις μονάδες μέτρησης της εφαρμογής: οι αποστάσεις είναι σε μέτρα και οι ταχύτητες σε μέτρα ανά δευτερόλεπτο. Εάν οποιοδήποτε από τα δύο ορίσματα είναι null, η συνάρτηση εγκαταλείπει όλους τους υπολογισμούς της και επιστρέφει επίσης null.

Μεταβλητή “speedLimit”:
Η μεταβλητή αντιστοιχεί στο πεδίο “globalSpeedLimit” του αρχείου καθολικών ρυθμίσεων της εφαρμογής, κατόπιν ανάγνωσης και επιτυχούς επαλήθευσης εισαγωγών χρήστη, ή στην προεπιλεγμένη τιμή. Είναι τύπου double και εκφράζει το υποθετικό όριο ταχύτητας, σε μέτρα ανά δευτερόλεπτο, που επιβάλλουν οι λιμενικές Αρχές στο σενάριο εξομοίωσης.

5.2: Γεγονότα υψηλού επιπέδου

5.2.1: Reported Overspeed

Το ερώτημα έχει σκοπό τον εντοπισμό σκαφών που πλέουν με ταχύτητα υψηλότερη από το όριο ταχύτητας. Για την ακρίβεια, το ερώτημα έχει γραφεί για να ανιχνεύσει τις εκπομπές εκ του ρεύματος των LLEs εκείνες με αναφερόμενη ταχύτητα ανώτερη της τιμής του πεδίου globalSpeedLimit, εκ των καθολικών ρυθμίσεων της εφαρμογής. Σχετικά με το πεδίο αυτό, υπενθυμίζεται πως πρόκειται για το υποθετικό ανώτατο επιτρεπόμενο όριο ταχύτητας, βάσει θεσμών ή νόμων στα ύδατα του σεναρίου εξομοίωσης. Επίσης, το ερώτημα δεν λογίζει σκάφη των αρμόδιων Αρχών που αναλαμβάνουν αποστολές εκτάκτου ανάγκης και κοινωφελούς ρόλου. Έτσι, εξαιρούνται πλωτά και υποβρύχια μέσα του Πολεμικού Ναυτικού, του Λιμενικού Σώματος, σκάφη Έρευνας και Διάσωσης, πλωτά νοσοκομεία και σταθμοί παροχής ιατροφαρμακευτικής περίθαλψης. Με την υπόθεση πως τα ύδατα του σεναρίου είναι ελληνικά, υπάρχει και η πρόσθετη απαίτηση για τα εν λόγω σκάφη των παραπάνω ειδικών κατηγοριών να πλέουν υπό τη Σημαία, λόγω αποκλειστικότητας της δικαιοδοσίας δράσης στην περιοχή.

Ο κώδικας του ερωτήματος κάνει χρήση φίλτρων και περιορισμών για την εξαίρεση των κατάλληλων σκαφών, αλλά και την εφαρμογή του κριτηρίου ταχύτητας στα υπόλοιπα, αντίστοιχα.

```
String overspeedReportedEPL =
"SELECT vessel, speed " +
"FROM AISBroadcast((vessel.type NOT IN (" +
      "VesselType.MILITARY, " +
      "VesselType.LAWENFORCEMENT, " +
      "VesselType.SEARCHANDRESCUE, " +
      "VesselType.MEDICAL)) OR (vessel.flag NOT IN (Flag.GR))) " +
"WHERE speed > speedLimit";
```

Αρχικά επιβάλλεται φίλτρο στο ρεύμα των LLEs, δηλαδή επί των αντικειμένων AISBroadcast του FROM statement, κατά το οποίο θα πρέπει - διαζευκτικά - να ισχύουν τα παρακάτω:

1. Το εκπέμπον πλοίο δεν θα πρέπει να έχει κάποια από τις προαναφερθείσες χρήσεις (π.χ Φρεγάτα, περιπολικό Λιμενικού).
2. Η σημαία του πλοίου δεν θα πρέπει να είναι η Ελληνική.

Στο σημείο αυτό, θα πρέπει να διευκρινιστεί πως το φίλτρο περιγράφει τις εκπομπές που ΔΕΝ απορρίπτονται από την περεταίρω διαδικασία. Στη συνέχεια με το WHERE statement, επιλέγονται τα γεγονότα του παραθύρου επεξεργασίας με τιμή στο πεδίο speed (AISBroadcast.speed) μεγαλύτερη από την τιμή του speedLimit. Στα γεγονότα που έμειναν, τελικώς, επιλέγονται με το statement SELECT τα πεδία αυτών vessel και speed.

Το φίλτρο του ρεύματος μπορεί να μοιάζει πως εργάζεται στην απολύτως αντίθετη κατεύθυνση από την επιθυμητή, αλλά η περιγραφή που δόθηκε στην αρχή αφορά τις εκπομπές που πρέπει να εξαιρεθούν από την επεξεργασία. Καθώς όμως το φίλτρο δέχεται τον ορισμό των γεγονότων που επιτρέπει, θα πρέπει η έκφρασή του να μετατραπεί αναλόγως. Η λύση δίδεται από τα θεωρήματα προτασιακής λογικής / άλγεβρας Boole του De Morgan και συγκεκριμένα από το: NOT(P AND Q) <==> NOT(P) OR NOT(Q). Έτσι, τα δύο κριτήρια έχουν διαζευκτική εφαρμογή (OR) ενώ το περιεχόμενο κάθε ενός από αυτά αντικαθίσταται με τη λογική του άρνηση.
Παραμένοντας στα φίλτρα της Esper, ενώ αυτά είναι γενικώς ανταλλάξιμα με αντίστοιχους περιορισμούς που ακολουθούν WHERE predicates, υπάρχει μία λεπτή διαφορά, που ίσως διακρίνεται εμμέσως και από την επεξήγηση λειτουργίας του EPL κώδικα, νωρίτερα: τα φίλτρα προηγούνται στην επεξεργασία των WHERE περιορισμών. Ακόμη, και σημαντικότερα ίσως, όσα γεγονότα απορριφθούν από φίλτρα του είδους, δεν εισέρχονται καθόλου στο παράθυρο επεξεργασίας, σαν να μην υπήρξαν

ποτέ ως LLEs. Η διαφορά τους εξηγείται αναλυτικότερα στην αντίστοιχη ενότητα[99] της τεκμηρίωσης Esper.

Το αρχείο αναφοράς HLEs του ερωτήματος δημιουργείται ως εξής:

```
File overspeedReportedFile = new File(eventReportOutputPath + sep +
"Reported Overspeed Events.txt");
```

Η τιμή του speedLimit είναι επιλέξιμη από το χρήστη μέσω του αρχείου ρύθμισης, κατά τα γνωστά, ενώ επίσης εσωτερικεύεται στη μηχανή γεγονότων με την κλήση:

```
esperConfiguration.addVariable("speedLimit", Double.class, new
Double(AppConfiguration.getGlobalSpeedLimit()));
```

Για τη δημιουργία και ενεργοποίηση του ερωτήματος και του listener αυτού, αλλά και τη συσχέτισή τους με το αρχείο αναφοράς:

```
EPStatement overspeedReportedStatement =
esperAdministrator.createEPL(overspeedReportedEPL);
OverspeedReportedListener overspeedReportedListener = new
OverspeedReportedListener(overspeedReportedFile);
overspeedReportedStatement.addListener(overspeedReportedListener);
```

Σε ότι αφορά τους χειρισμούς στην OverspeedReportedListener.update(), εκείνη σε κάθε κλήση της απλώς ελέγχει το όρισμα newEvents και στην περίπτωση που αυτό δεν είναι κενό (null), τότε επαναληπτικά για κάθε EventBean του array αυτού: Ανακτά MMSI, όνομα πλοίου, σημαία, είδος και αναφερόμενη ταχύτητα. Στη συνέχεια, με κλήση στην writeToOutputFile(), εγγράφει στο overspeedReportedFile αναφορές με τις καταχωρήσεις αυτές να τοποθετούνται τη μία κάτω από την άλλη, σε αύξουσα χρονική σειρά. Ένα απόσπασμα από αρχείο Reported Overspeed Events.txt, όπου φαίνεται η μορφή δύο διαδοχικών εγγραφών:

```
EVENT: Reported speed is above the legal limit of 15.0 m/s.
	Vessel:
		MMSI: 239737000
		Name: BlueStar2
		Flag: GR
		Type: PASSENGER
	Reported speed: 16.0 m/s.

EVENT: Reported speed is above the legal limit of 15.0 m/s.
	Vessel:
		MMSI: 200000004
		Name: Barbaros
		Flag: TR
		Type: MILITARY
	Reported speed: 15.75 m/s.
```

[99] 3.4. Filters and Where-clauses: http://esper.espertech.com/release-6.1.0/esper-reference/html_single/#processingmodel_filter βλ. σχήματα.

5.2.2: Calculated Overspeed

Πρόκειται για ακόμη ένα ερώτημα με σκοπό τον εντοπισμό σκαφών που πλέουν με ταχύτητα υψηλότερη από το θεσμικό / νόμιμο όριο ταχύτητας. Συγκεκριμένα, το ερώτημα έχει γραφεί για να ανιχνεύσει τις εκπομπές εκ του ρεύματος των LLEs εκείνες, με ταχύτητα ανώτερη της τιμής του πεδίου globalSpeedLimit. Η υλοποίηση του παρόντος μπορεί να μοιάζει περιττή, αλλά η χρήση του ερωτήματος δεν αποτελεί πλεονασμό: Σε σχέση με το ερώτημα Reported Overspeed, διαφοροποιείται εκτενώς, αφού δεν εφαρμόζει κανένα είδος περιορισμού σε τύπο πλοίου αλλά ούτε στη σημαία. Επίσης, η σημαντικότερη διαφορά είναι ο τρόπος με τον οποίο συμπεραίνεται η ταχύτητα κάθε σκάφους τη στιγμή της λήψης μίας δεδομένης εκπομπής του. Το παρόν ερώτημα αγνοεί εντελώς την αναφερόμενη ταχύτητα (πεδίο AISBroadcast.speed) στην οποία το Reported Overspeed βασίζεται για την λειτουργία του, και αντίθετα υπολογίζει την ταχύτητα μεταξύ δύο διαδοχικών εκπομπών κάθε πλοίου από τη διαφορά των χρονικών στιγμών εκπομπής και από την απόσταση που έχει διανυθεί στο διάστημα αυτό.

Ο κώδικας του ερωτήματος μοιράζεται σε πολλαπλά EPL statements που το καθένα έχει προπαρασκευαστικό χαρακτήρα για τη λειτουργία εκείνων που έπονται. Έτσι, η συγκριτικά σύνθετη λειτουργία του ερωτήματος επιμερίζεται σε στάδια που εκτελούν μέρος των υπολογισμών. Η σημαντικότερη από αυτές τις «ενδιάμεσες» λειτουργίες είναι η παραγωγή ενός ρεύματος “middle-level” γεγονότων που αντιπροσωπεύουν τα διαφορικά απόστασης και χρόνου μεταξύ διαδοχικών AISBroadcasts του ίδιου σκάφους.

```
String broadcastDeltaEPL =
"CREATE SCHEMA DeltaAIS(mmsi long, name String, distanceCovered
double, angleRotated double, timeTaken double)";

String deltaWindowEPL =
"CREATE WINDOW DeltaEvents.win:length(3) AS DeltaAIS";

String insertToDeltaWinEPL =
"INSERT INTO DeltaEvents " +
"SELECT " +
"     vessel.mmsi AS mmsi, " +
"     vessel.name AS name, " +
"     distance(position, prev(position)) AS distanceCovered, " +
"     angle(heading, prev(heading)) AS angleRotated, " +
"     timeDifference(timeStamp, prev(timeStamp)) AS timeTaken " +
"FROM AISBroadcast#groupwin(vessel.mmsi)#length(2)";

String overspeedCalculatedEPL =
"SELECT mmsi, name, distanceCovered, timeTaken, (distanceCovered /
timeTaken) AS calculatedSpeed " +
"FROM DeltaEvents " +
"WHERE (distanceCovered / timeTaken) > speedLimit";
```

Στη συνέχεια παρατίθεται η σειριακή ανάλυση του κώδικα, όπως βρίσκεται οργανωμένος σε ονοματισμένα αλφαριθμητικά όπου αποθηκεύονται τα statements:

broadcastDeltaEPL:
Με χρήση του clause CREATE SCHEMA, δημιουργείται ο νέος τύπος δεδομένων DeltaAIS για δήλωση στην μηχανή γεγονότων της Esper. Εντός της παρένθεσης βρίσκονται, χωρισμένα με κόμματα, τα πεδία του ως λίστα. Το κάθε ένα δηλώνεται πρώτα με το αναγνωριστικό του και στη συνέχεια με datatype. Τα instances του DeltaAIS αφορούν έκαστο ένα συγκεκριμένο σκάφος και τις μετρήσεις χωρικής απόστασης και χρόνου μεταξύ δύο διαφορετικών εκπομπών AISBroadcast αυτού του σκάφους. Ο τύπος έχει πεδία:

1. mmsi (long): Το mmsi του πλοίου.
2. name (String): Το όνομά του.

3. distanceCovered (double): Η απόσταση μεταξύ των αναφερόμενων θέσεων από δύο εκπομπές του ίδιου σκάφους.
4. angleRotated (double): Η διαφορά στον προσανατολισμό (heading) μεταξύ των αναφερόμενων εκπομπών του σκάφους.
5. timeTaken (double) Ο χρόνος που έχει παρέλθει μεταξύ των ίδιων εκπομπών που αφορούν τη μέτρηση του πεδίου (3), ως διαφορά των timeStamp πεδίων σε αυτές.

Το clause CREATE SCHEMA επιτρέπει τη χρήση πρόσθετων τύπων δεδομένων για τις ανάγκες των ερωτημάτων, καθιστώντας δυνατή τη σύνθεση πολύπλοκων μορφών από τα διάφορα LLE types. Από την οπτική της Esper, οι τύποι δεδομένων που δημιουργούνται κατά τον τρόπο αυτό είναι ισότιμοι με αυτούς που προέρχονται από την περιβάλλουσα Java εφαρμογή και εσωτερικεύονται με κλήση στην ConfigurationOperations.addEventType().

deltaWindowEPL:
Το clause CREATE WINDOW δημιουργεί ένα νέο παράθυρο επεξεργασίας κάποιου ρεύματος συγκεκριμένου τύπου γεγονότων, η δήλωση του οποίου ακολουθεί τη δεσμευμένη λέξη AS. Εν προκειμένω, αποτελείται από τα διαφορικά χρόνου, γωνίας και απόστασης ως αντικείμενα DeltaAIS. Η δήλωση του ονόματος του παραθύρου, σε αυτήν την περίπτωση DeltaEvents, περιλαμβάνει και την εφαρμογή μίας εκ των προκαθορισμένων προβολών[100] επιβολής περιορισμών εύρους ή περαιτέρω επεξεργασίας των γεγονότων παραθύρου. Η .win:length() απλώς περιορίζει το εύρος του παραθύρου σε πλήθος περιεχόμενων γεγονότων για εξοικονόμηση πόρων. Άλλωστε, το τελευταίο statement EPL κώδικα που διαβάζει από αυτό το παράθυρο, εντοπίζει γεγονότα κατά την είσοδό τους σε αυτό και δεν επεξεργάζεται εκείνα που ήδη βρίσκονται στο εύρος του. Κατόπιν δημιουργίας, το παράθυρο επεξεργασίας είναι κενό.

insertToDeltaWinEPL:
Το statement αυτό είναι υπεύθυνο για την κατασκευή και εισαγωγή γεγονότων DeltaAIS στο ρεύμα DeltaEvents. Επιτυγχάνει δε αυτό με χρήση ενός INSERT INTO clause το οποίο ακολουθείται από το αναγνωριστικό του ρεύματος - προορισμού. Ύστερα από αυτά, εν είδει subquery, ο ορισμός των προς εισαγωγή στοιχείων: Η προέλευση των γεγονότων καθορίζεται από το FROM clause, με πηγή το ρεύμα των γνωστών LLEs, AISBroadcast, υπό την επίδραση όμως δύο διαφορετικών views, .std:groupwin[101] και .win:length, με ορίσματα vessel.mmsi και 2, αντίστοιχα. Ο συνδυασμός τους προκαλεί τη διαμέριση του ρεύματος AISBroadcast σε επιμέρους ρεύματα / παράθυρα, κάθε ένα από τα οποία περιέχει τις δύο πιο πρόσφατες εκπομπές πλοίου με το ίδιο mmsi, αποκλειστικά. Η ομαδοποίηση ανά κάποιο δεδομένο χαρακτηριστικό είναι όμοια με αυτή του clause GROUP BY, ενώ επίσης η Esper διαθέτει πρόσθετους μηχανισμούς ομαδοποίησης ή δημιουργίας φιλτραρισμένων προβολών γενικότερα όπως τα Contexts[102]. Αξίζει να σημειωθεί πως μεταξύ τους δεν είναι ανταλλάξιμοι: Έχοντας αρκετές διαφορές στον τρόπο λειτουργίας τους, ενδείκνυνται για διαφορετικές εργασίες. Τα GROUP BY και .std:groupwin διαφοροποιούνται αρκετά ώστε να μην συμπεριφέρονται κατά τον ίδιο τρόπο ακόμη και σε ένα όχι ιδιαίτερα πολύπλοκο statement όπως το insertToDeltaWinEPL: Στην περίπτωση χρήσης GROUP BY, το αρχικό ρεύμα δεν θα διαμεριζόταν, θα εξακολουθούσε να φέρει εκπομπές από όλα τα πλοία του σεναρίου, απλώς θα μειωνόταν στο εύρος των δύο αντικειμένων, υπό την επίδραση του #length(2) view, ή εκφρασμένο εναλλακτικά, .win:length(2). Το GROUP BY επιδρά επιμεριστικά μόνο όταν χρησιμοποιείται σε συνδυασμό με συγκεντρωτικές (aggregate) συναρτήσεις, καλούμενες στα clauses SELECT και/ή HAVING. Εάν το ερώτημα, πέρα από την επιμέριση των παραθύρων, περιλάμβανε και τέτοιες συναρτήσεις, θα έπρεπε να χρησιμοποιηθούν και τα δύο εργαλεία στο ίδιο statement. Σε τέτοιες περιπτώσεις βέβαια, ίσως η καλύτερη λύση είναι η χρήση Context που έχει καθολική και λιγότερο στοχευμένη επίδραση. Ο πίνακας “Grouping Options”[103] της τεκμηρίωσης Esper παρέχει μία συνοπτική αλλά περιεκτική εικόνα της επίδρασης των παραπάνω εργαλείων.

[100] EPL Reference: Views http://esper.espertech.com/release-6.1.0/esper-reference/html_single/#epl-views
[101] 14.4.2. Grouped Data Window (groupwin or std:groupwin) http://esper.espertech.com/release-6.1.0/esper-reference/html_single/#view-std-groupwin
[102] 4. Context and Context Partitions http://esper.espertech.com/release-6.1.0/esper-reference/html_single/#context
[103] 5.6.8. Comparing Keyed Segmented Context, the Group By clause and the std:groupwin view http://esper.espertech.com/release-6.1.0/esper-reference/html_single/#epl-group-by-versus-view

Σε κάθε επιμερισμένο παράθυρο (οι δύο πιο πρόσφατες εκπομπές του ίδιου πλοίου) με χρήση του SELECT clause, επιλέγονται vessel.mmsi και vessel.name, προφανώς ίδια και στις δύο εκπομπές, με aliases mmsi και name, αντίστοιχα. Η αλλαγή ονόματος εντός SELECT και άλλων clauses γίνεται με την δεσμευμένη λέξη AS. Σημαντικότερα, εντός του SELECT επιλέγονται ακόμη τα διαφορικά:

- Με alias “distanceCovered”, το αποτέλεσμα που επιστρέφει η single-row-function distance με ορίσματα:
    - position
    - prev(position)
- Με alias “angleRotated”, το αποτέλεσμα που επιστρέφει η single-row-function angle με ορίσματα:
    - heading
    - prev(heading)
- Με alias “timeTaken”, το αποτέλεσμα που επιστρέφει η single-row-function timeDifference με ορίσματα:
    - timestamp
    - prev(timeStamp)

Τα πεδία position, heading και timeStamp είναι τα γνωστά πεδία του τύπου AISBroadcast της sailAway, ενώ από την οπτική της Esper, ασφαλώς πρόκειται για τα πιο πρόσφατα του κάθε επιμερισμένου παραθύρου AISBroadcast#groupwin κάθε πλοίου του σεναρίου. Γενικώς τα απλά μη συγκεντρωτικά ερωτήματα επί ρευμάτων, επιστρέφουν κάθε φορά το πιο πρόσφατο γεγονός του αντίστοιχου παραθύρου, τη στιγμή δηλαδή που εισέρχεται στο παράθυρο εκείνο, ενώ κατ' επέκταση οι εισαγωγές / εξαγωγές γεγονότων σε/από αυτά γίνονται εν είδει ουράς lifo. Τα αποστελλόμενα προς listeners αποτελέσματα επίσης προκαλούν κλήση τους για κάθε ένα γεγονός του ρεύματος, την στιγμή της εισαγωγής του στο αντίστοιχο παράθυρο (insert stream, istream)[104]. Έτσι, τα πεδία που δίδονται ως ορίσματα στις παραπάνω συναρτήσεις είναι αυτά των πιο πρόσφατων γεγονότων στο παράθυρό τους. Η συνάρτηση prev(), υπερφόρτωση της οποίας δέχεται δύο ορίσματα, εκ των οποίων ένα είναι n-offset και το άλλο το επιθυμητό property κάποιου γεγονότος, επιστρέφει το ζητούμενο πεδίο του γεγονότος n-οστής αύξουσας παλαιότητας, χωρίς να λογίζει το πιο πρόσφατο. Άρα η κλήση prev(1, position) που αντιστοιχεί στην υπερφόρτωση prev(position), επιστρέφει το πεδίο position του 2ου πιο πρόσφατου γεγονότος του παραθύρου στο οποίο εφαρμόζεται.

overspeedCalculatedEPL:
Το τελευταίο statement διαβάζει (γραμμή FROM) το ρεύμα DeltaEvents και από τα αντικείμενά του επιλέγει με χρήση του WHERE clause, εκείνα με λόγο distanceCovered/timeTaken μεγαλύτερο της τιμής του speedLimit. Το πηλίκο αυτό αποτελεί και τον υπολογισμό της (μέσης) ταχύτητας μεταξύ των δύο εκπομπών, ως λόγος των διαφορικών μετατόπισης και χρόνου. Από τα αντικείμενα που δεν θα απορριφθούν, επιλέγονται όλα τους τα πεδία με χρήση SELECT, εξαιρούμενου του angleRotated, ενώ ακόμη προστίθεται ad-hoc ο παραπάνω λόγος της ταχύτητας με alias “calculatedSpeed”. Τα αποτελέσματα του τελικού αυτού statement κατευθύνονται προς το listener του παρόντος ερωτήματος.

Το αρχείο αναφοράς HLEs του ερωτήματος δημιουργείται ως εξής:

```
File overspeedCalculatedFile = new File(eventReportOutputPath + sep + "Calculated Overspeed Events.txt");
```

Η ίδια κλήση που παρουσιάστηκε για τις ανάγκες του προηγούμενου ερωτήματος, `esperConfiguration.addVariable("speedLimit", Double.class, new Double(AppConfiguration.getGlobalSpeedLimit()))`, προφανώς εξυπηρετεί και τη λειτουργία του παρόντος, εξασφαλίζοντας πρόσβαση στην τιμή του ορίου ταχύτητας από τον EPL κώδικα. Ακόμη, το αποθηκευμένο στο αλφαριθμητικό insertToDeltaWinEPL statement, καλεί τις

[104] Φυσικά, τα ερωτήματα μπορούν να παραμετροποιηθούν με διάφορους τρόπους ώστε να επιστρέφουν γεγονότα σε προκαθορισμένα χρονικά πλαίσια, ομαδικώς, ή ακόμη και από το remove stream κάποιου παραθύρου, δηλαδή κατά την απόρριψή τους από το παράθυρο επεξεργασίας τους. Το κεφάλαιο 3 (Processing Model) και η ενότητα 5.7 (Stabilizing and Controlling Output: the Output Clause) παρέχουν πληροφόρηση για αυτές τις επιλογές.

συναρτήσεις distance, angle και timeDifference για υπολογισμό απόστασης και για υπολογισμό χρονικής διάρκειας αντίστοιχα. Αυτές εσωτερικοποιούνται με τις κλήσεις:

```
esperConfiguration.addPlugInSingleRowFunction("distance",
Coordinates.class.getName(), "calculateDistance");
esperConfiguration.addPlugInSingleRowFunction("angle",
Azimuth.class.getName(), "angleFinder");
esperConfiguration.addPlugInSingleRowFunction("timeDifference",
Application.class.getName(), "esper_findTimeDifference");
```

Ο υπολογισμός της διαφοράς του προσανατολισμού διαδοχικών εκπομπών πλοίου, angleRotated, δεν απαιτείται για τη λειτουργία του παρόντος ερωτήματος. Όμως το παράθυρο DeltaEvents αποτελεί ενδιάμεσο βήμα για την εύρεση και κάποιου άλλου HLE, το οποίο συγκρίνει τα headings κάθε πλοίου από διαφορετικές χρονικές στιγμές, επίσης μέσω DeltaAIS instances. Αξίζει ακόμη να σημειωθεί πως ενώ οι κλήσεις στην prev() επί των δεύτερων ορισμάτων των single-row-functions απόστασης, ταχύτητας και χρόνου του statement insertToDeltaWinEPL υπονοούν την παραγωγή των αντίστοιχων διαφορικών από διαδοχικές εκπομπές, θα μπορούσαν να ληφθούν και διαφορές ιδιοτήτων από εκπομπές που απέχουν περισσότερο χρονικά. Σε κάθε περίπτωση θα πρέπει οι ιδιότητες να προέρχονται από εκπομπές της ίδιας χρονικής απόστασης για τις τρεις συναρτήσεις, ώστε λόγοι όπως (distanceCovered/timeTaken) ή (angleRotated/timeTaken)[105] να παράγουν χρήσιμα αποτελέσματα, τα οποία φυσικά αντιπροσωπεύουν τη σχετική μέση τιμή κατά το χρονικό εύρος των εμπλεκόμενων εκπομπών. Μία τέτοια προσέγγιση παρουσιάζει ενδιαφέρον καθώς θα προκαλούσε μια «ελαστικότερη» αναγνώριση σύνθετων γεγονότων, σε υψηλότερο επίπεδο. Στην περίπτωση της ταχύτητας για παράδειγμα, θα επέτρεπε μικρής διάρκειας και έντασης spikes άνω του ορίου ταχύτητος, σε ένα ευρύτερο χρονικά «παράθυρο», σε αντίθεση με την παρούσα προσέγγιση «μηδενικής ανοχής».

Η λειτουργία του ερωτήματος, ασφαλώς απαιτεί την δημιουργία και καταχώρηση των EPL statements που παρουσιάστηκαν νωρίτερα, αποθηκευμένα σε Strings. Αυτά ενεργοποιούνται στην ίδια σειρά που περιγράφηκαν, αφού οι αντίστοιχες κλήσεις createEPL() βρίσκονται στην ίδια διάταξη:

```
EPStatement broadcastDeltaStatement =
esperAdministrator.createEPL(broadcastDeltaEPL);
EPStatement deltaWindowStatement =
esperAdministrator.createEPL(deltaWindowEPL);
EPStatement insertToDeltaWinStatement =
esperAdministrator.createEPL(insertToDeltaWinEPL);
EPStatement overspeedCalculatedStatement =
esperAdministrator.createEPL(overspeedCalculatedEPL);
```

Για τα ενδιάμεσα ερωτήματα ασφαλώς δεν απαιτείται έξοδος των αποτελεσμάτων τους. Η συσχέτιση του τελικού με τον δικό του listener και με το αρχείο αναφοράς:

```
OverspeedCalculatedListener overspeedCalculatedListener = new
OverspeedCalculatedListener(overspeedCalculatedFile);
overspeedCalculatedStatement.addListener(overspeedCalculatedListener);
```

Σε ότι αφορά τους χειρισμούς στην OverspeedCalculatedListener.update(), εκείνη σε κάθε κλήση της απλώς ελέγχει το όρισμα newEvents και στην περίπτωση που αυτό δεν είναι κενό (null), τότε επαναληπτικά για κάθε EventBean του array αυτού: Ανακτά MMSI, όνομα πλοίου, διαφορικά απόστασης και χρόνου, αλλά και το λόγο τους ως ταχύτητα. Στη συνέχεια, με κλήση στην writeToOutputFile(), εγγράφει στο overspeedCalculatedFile αναφορές με τις καταχωρήσεις αυτές να τοποθετούνται τη μία κάτω από την άλλη, σε αύξουσα χρονική σειρά. Ένα απόσπασμα από αρχείο Calculated Overspeed Events.txt, όπου φαίνεται η μορφή δύο διαδοχικών εγγραφών:

[105] Ρυθμός περιστροφής, χρησιμοποιείται στο Sharp Turn HLE.

```
EVENT: Calculated speed is above the legal limit of 4.0 m/s.
	Vessel:
		MMSI: 200000001
		Name: Gavdos
	Distance Covered: 4.47213595499958 meters.
	Time Taken: 0.532 seconds.
	Calculated speed: 8.406270592104473 m/s.

EVENT: Calculated speed is above the legal limit of 4.0 m/s.
	Vessel:
		MMSI: 239923000
		Name: BlueStarNaxos
	Distance Covered: 2.23606797749979 meters.
	Time Taken: 0.543 seconds.
	Calculated speed: 4.117988908839392 m/s.
```

Σε επέκταση της προσπάθειας του παρόντος ερωτήματος για ανεξαρτησία από τις αναφερόμενες πληροφορίες των AISBroadcast, τα timeStamp πεδία των εκπομπών αυτών θα μπορούσαν επίσης να αγνοηθούν, υπέρ χρήσης του χρόνου λήψης της εκπομπής από τη μηχανή, με κλήση της παρεχόμενης από την Esper συνάρτησης current_timestamp()[106], που επιστρέφει την τρέχουσα ώρα συστήματος κατά την αποτίμηση του ερωτήματος από τη μηχανή γεγονότων (με την αποδοχή ενός μικρού latency). Φυσικά, ο υπολογισμός της ταχύτητας εξαρτάται και από τις εκάστοτε χωρικές θέσεις των εκπομπών, οι οποίες δεν μπορούν από την παρούσα υλοποίηση να εκτιμηθούν. Παρόμοιες ενέργειες μπορούν να θωρακίσουν μερικώς μόνο την ορθή αναγνώριση γεγονότων από ψευδείς αναφορές, καθώς η βάση αναγνώρισης παραμένει το AISBroadcast ως LLE.

[106] http://esper.espertech.com/release-6.1.0/esper-reference/html/functionreference.html#epl-single-row-function-ref-currenttime

5.2.3: Sharp Turn

Το ερώτημα EPL έχει σκοπό τον εντοπισμό σκαφών που αλλάζουν κατεύθυνση απότομα. Για την ακρίβεια, το ερώτημα έχει γραφεί για να ανιχνεύσει τα σκάφη που στρίβουν σε μία καμπή του δρομολογίου τους, με αποτέλεσμα να περιστρέφονται επίσης γύρω από τον κατακόρυφο άξονά τους με ρυθμό περιστροφής (γωνιακή ταχύτητα) μεγαλύτερο ενός καθορισμένου ορίου.

Ο κώδικας του ερωτήματος, που κάνει χρήση απλών EPL clauses, εξαρτάται από παράθυρο «ενδιάμεσων» γεγονότων που χρησιμοποιήθηκε σε προηγούμενο ερώτημα και υπολογίζει το λόγο δύο ιδιοτήτων.

```
String sharpTurnEPL =
"SELECT mmsi, name, angleRotated, timeTaken, (angleRotated /
timeTaken) AS rateOfTurn " +
"FROM DeltaEvents " +
"WHERE (angleRotated / timeTaken) > 40"; // angle in degrees, time in
seconds.
```

Το FROM clause του sharpTurnEPL καθορίζει την προέλευση των προς επεξεργασία γεγονότων του ερωτήματος. Αυτά είναι τύπου DeltaAIS, προερχόμενα από το γνωστό παράθυρο DeltaEvents. Η περαιτέρω επιλογή / φιλτράρισμά τους γίνεται από ένα WHERE clause, το οποίο απορρίπτει εκείνα όπου ο λόγος των πεδίων τους angleRotated προς timeTaken είναι μικρότερος από το αναγραφόμενο στο statement όριο γωνιακής ταχύτητας. Το όριο αυτό μετράται σε μοίρες ανά δευτερόλεπτο, σε συμφωνία με τις μονάδες μέτρησης που βρίσκονται τα μεγέθη του κλάσματος. Τα DeltaAIS με ρυθμό περιστροφής άνω του ορίου επιλέγονται, ενώ με τη χρήση μίας πρότασης SELECT επιλέγονται όλα τους τα πεδία, εξαιρούμενου του distanceCovered, ενώ ακόμη προστίθεται λόγος της γωνιακής ταχύτητας όπως ακριβώς υπολογίστηκε και από το WHERE clause, με alias “rateOfTurn”.

Ο κώδικας που παρατίθεται στη συνέχεια αποτελεί μία εναλλακτική προσέγγιση για την αναγνώριση των ίδιων HLEs, που κάνει χρήση του clause MATCH_RECOGNIZE, ενός μηχανισμού αναγνώρισης ακολουθιών γεγονότων που θα μπορούσε να παραλληλιστεί με τις κανονικές εκφράσεις (regular expressions) αλφαριθμητικών. Η λειτουργία του clause αυτού καθώς και των σχετικών με αυτό δεσμευμένων λέξεων EPL, εξετάζεται στο επόμενο ερώτημα όπου και αποτελεί την κύρια προσέγγιση για εκείνο το πρόβλημα.

```
String sharpTurnEPL =
"SELECT * " +
"FROM AISBroadcast " +
"MATCH_RECOGNIZE(" +
"PARTITION BY vessel.mmsi " +
"MEASURES A AS a, B AS b " +
"PATTERN (A B) " +
"DEFINE" +
" B AS (B.speed * angle(B.heading, A.heading) > 60))";*/ // angle in
degrees, speed in m/s.
```

Η λειτουργία του εναλλακτικού statement δεν εξαρτάται από το ρεύμα DeltaEvents του προηγούμενου ερωτήματος, ενώ για την σωστή λειτουργία του απαιτεί αλλαγές στον listener. Ακόμη, εφ' όσον δεν έχει πρόσβαση σε DeltaAIS γεγονότα, εξαρτάται από την συνάρτηση “angle”, που δηλώνεται ως:
`esperConfiguration.addPlugInSingleRowFunction("angle", Azimuth.class.getName(), "angleFinder");`, για τον υπολογισμό των διαφοράς προσανατολισμού.

Το αρχείο αναφοράς HLEs του ερωτήματος δημιουργείται ως εξής:

```
File sharpTurnFile = new File(eventReportOutputPath + sep + "Sharp
Turn Events.txt");
```

Η δημιουργία και ενεργοποίηση του ερωτήματος και του listener αυτού, αλλά και η συσχέτισή τους με το αρχείο αναφοράς, πραγματοποιείται με τις κλήσεις API:

```
EPStatement sharpTurnStatement =
esperAdministrator.createEPL(sharpTurnEPL);
SharpTurnListener sharpTurnListener = new
SharpTurnListener(sharpTurnFile);
sharpTurnStatement.addListener(sharpTurnListener);
```

Σε ότι αφορά τους χειρισμούς στην SharpTurnListener.update(), εκείνη σε κάθε κλήση της απλώς ελέγχει το όρισμα newEvents και στην περίπτωση που αυτό δεν είναι κενό (null), τότε επαναληπτικά για κάθε EventBean του array αυτού: Ανακτά MMSI, όνομα πλοίου, διαφορικά γωνίας και χρόνου, αλλά και το λόγο τους ως γωνιακή ταχύτητα. Στη συνέχεια, με κλήση στην writeToOutputFile(), εγγράφει στο sharpTurnFile αναφορές με τις καταχωρήσεις αυτές να τοποθετούνται τη μία κάτω από την άλλη, σε αύξουσα χρονική σειρά. Ένα απόσπασμα από αρχείο Sharp Turn Events.txt, όπου φαίνεται η μορφή μίας εγγραφής:

```
EVENT: Sharp turn taken.
	Vessel:
		MMSI: 239737000
		Name: BlueStar2
	Rotation: 135.0 degrees.
	Time Taken: 2.319 seconds.
	Rate of Turn: 58.214747736093145 degrees per second.
```

5.2.4: Package Delivery

Το ερώτημα αυτό σκοπεύει στον εντοπισμό μετάδοσης αντικειμένων μεταξύ σκαφών εν πλω. Συγκεκριμένα, εντοπίζει την αδικαιολόγητη στάση δύο πλοίων κατά τη διάρκεια του ταξιδιού τους, σε κοντινές τοποθεσίες και με μικρή χρονική υστέρηση μεταξύ των στάσεων. Στην πράξη, μία τέτοια κατάσταση μαρτυρά την ανταλλαγή ενός υλικού πλωτού αντικειμένου (ή ενός που κατέστη πλωτό / αδιάβροχο χάρη σε διάφορα υλικά συσκευασίας / περιτυλίγματα), από το πλοίο που σταματά πρώτο και το εναποθέτει, προς το πλοίο που προσέρχεται αργότερα κοντά στο σημείο, σταματά εκεί και συλλέγει το αντικείμενο από την επιφάνεια. Ασφαλώς η ανταλλαγή υλικών αντικειμένων καθ' αυτή δεν αποτελεί παράβαση, αλλά η φύση αυτής της μεταβίβασης που ενώ μακριά από στεριά είναι επίπονη και δύσκολη αλλά ωστόσο δεν γίνεται εύκολα αντιληπτή, εγείρει υποψίες για την νομιμότητά της. Τέτοιες θαλάσσιες δοσοληψίες μπορεί να αποτελούν μέσα φυγάδευσης ή διακίνησης: ναρκωτικών ουσιών, όπλων, ανθρώπων ή στην πιο «αθώα» περίπτωση, νόμιμων καταναλωτικών αγαθών από και προς το εξωτερικό, που όμως δεν εκτελωνίστηκαν και φορολογήθηκαν κατά τα προβλεπόμενα. Σε κάθε περίπτωση, δεν πρόκειται για κάποιο ακούσιο λάθος, όπως μία μικρή υπέρβαση ορίου ταχύτητας, αλλά μία εκούσια κακόβουλη πράξη.

Όπως και με άλλα ερωτήματα, ο κώδικας του παρόντος μοιράζεται σε πολλαπλά EPL statements προπαρασκευαστικού / βοηθητικού χαρακτήρα για χρήση από εκείνα που ακολουθούν. Έτσι, η λειτουργία του ερωτήματος επιμερίζεται σε στάδια που εκτελούν μέρος των υπολογισμών. Η σημαντικότερη από αυτές τις «ενδιάμεσες» λειτουργίες είναι η παραγωγή ενός ρεύματος “middle-level” γεγονότων που αντιπροσωπεύουν την στάση ενός πλοίου πριν τον προορισμό του και τον εν συνεχεία απόπλου του από το σημείο.

```
String SimpleBroadcastEPL =
"CREATE SCHEMA SimpleBroadcast(time Date, velocity double, ship
Vessel, site Coordinates)";

String stoppagesWindowEPL =
"CREATE WINDOW Stoppages#length(25) AS SimpleBroadcast";

String insertToStopWinEPL =
"INSERT INTO Stoppages" +
" SELECT" +
" b.lastOf().timeStamp AS time," +
" b.lastOf().speed AS velocity," +
" cast(b.lastOf().vessel, Vessel) AS ship," +
" cast(b.lastOf().position, Coordinates) AS site " +
"FROM AISBroadcast.win:time(900 sec) " +
"MATCH_RECOGNIZE(" +
"PARTITION BY (vessel.mmsi) " +
"MEASURES A AS a, B AS b " +
"PATTERN (A{4,} B+ A{4,}) " +
"DEFINE" +
" A AS (A.speed > A.vessel.minimumSpeed)," +
" B AS (B.speed <= B.vessel.minimumSpeed))";
```

```
String packageDeliveryEPL =
"SELECT " +
"     a('time') AS timeOfDispense, " +
"     c('time') AS timeOfRecovery, " +
"     a('ship') AS dispenserVessel, " +
"     c('ship') AS collectorVessel, " +
"     a('site') AS siteOfDispense, " +
"     c('site') AS siteOfRecovery, " +
"     distance(cast(a('site'), Coordinates), cast(c('site'),
Coordinates)) AS dist " +
"FROM Stoppages " +
"MATCH_RECOGNIZE(" +
"MEASURES A AS a, C AS c " +
"PATTERN (A B* C) " +
"DEFINE" +
" C AS (distance(C.site, A.site) <= 100) AND" +
" (timeDifference(C.time, A.time) <= 600) AND" +
" (C.ship.mmsi != A.ship.mmsi))";
```

Ο κώδικας, όπως βρίσκεται οργανωμένος στα ονοματισμένα αλφαριθμητικά των statements:

SimpleBroadcastEPL:
Με CREATE SCHEMA clause, δημιουργείται ο νέος τύπος δεδομένων SimpleBroadcast και δηλώνεται στο Esper runtime. Τα instances αυτού αφορούν έκαστο μία απλοποιημένη εκδοχή συγκεκριμένης εκπομπής AIS, περιέχοντας μόνο πεδία ενδιαφέροντος κατά τους υπολογισμούς των επόμενων statements. Για τους χειρισμούς εκείνους, αντιπροσωπεύουν μία από τις εκπομπές του πλοίου κατά τη διάρκεια της ανεξήγητης στάσης του, ενώ σε επόμενο statement αποσαφηνίζεται για ποιά ακριβώς από αυτές πρόκειται. Κάθε πεδίο του τύπου δηλώνεται με το αναγνωριστικό του και με datatype του:

1. time (java.util.Date): Η χρονική στιγμή της εκπομπής.
2. velocity (double): Η ταχύτητα[107] κατά την εκπομπή αυτή.
3. ship (Vessel): Το εκπέμπον πλοίο.
4. site (Coordinates): Η γεωγραφική θέση της στάσης.

stoppagesWindowEPL:
Το clause CREATE WINDOW δημιουργεί ένα νέο παράθυρο επεξεργασίας κάποιου ρεύματος συγκεκριμένου τύπου γεγονότων, η δήλωση του οποίου ακολουθεί τη δεσμευμένη λέξη AS. Εν προκειμένω, αποτελείται από απλοποιημένες εκπομπές AIS ως αντικείμενα SimpleBroadcast. Η δήλωση του ονόματος του παραθύρου, Stoppages, περιλαμβάνει και την εφαρμογή μίας επιβολής περιορισμού εύρους για εξοικονόμηση πόρων. Άλλωστε, μόνο οι πιο πρόσφατες στάσεις πλοίων απαιτούνται για τον εντοπισμό του HLE, λόγω της χρονικής εγγύτητάς τους.

insertToStopWinEPL:
Το statement αυτό είναι υπεύθυνο για την κατασκευή και εισαγωγή γεγονότων SimpleBroadcast στο ρεύμα Stoppages. Επιτυγχάνει δε αυτό με χρήση ενός INSERT INTO clause το οποίο ακολουθείται από το αναγνωριστικό του ρεύματος - προορισμού. Ύστερα από αυτά, εν είδει subquery, ο ορισμός των προς εισαγωγή στοιχείων: Η προέλευση των γεγονότων καθορίζεται από το FROM clause, με πηγή ένα ιδιαίτερο υποσύνολο γεγονότων προερχόμενων από το ρεύμα των γνωστών LLEs, AISBroadcast. Το υποσύνολο αυτό περιγράφεται από την συνδυασμένη επίδραση των:

1. view .win:time, με όρισμα τα 900 δευτερόλεπτα, επί του ρεύματος των AISBroadcast.
2. clause MATCH_RECOGNIZE[108] και των υπόλοιπων δεσμευμένων λέξεων, μετά από αυτό και μέχρι το τέλος του statement, που καθορίζουν λεπτομερώς τον τρόπο λειτουργίας του.

Το MATCH_RECOGNIZE αποτελεί μηχανισμό αναγνώρισης μοτίβων σε ακολουθίες γεγονότων, όμοιο με αυτό των κανονικών εκφράσεων λεκτικής αναγνώρισης συμβολοσειρών. Χρησιμοποιείται στην Esper

[107] Ενώ πρόκειται για στάση, η ταχύτητα δεν είναι απαραίτητο να είναι μηδενική, αλλά απλώς μικρότερη ενός πολύ χαμηλού ορίου. Βλέπε statement insertToStopWinEPL.
[108] Match Recognize: http://esper.espertech.com/release-6.1.0/esper-reference/html_single/index.html#match-recognize

μόνο μετά από ένα FROM clause, λαμβάνοντας έτσι το ρεύμα των γεγονότων επί των οποίων θα πραγματοποιήσει «ταίριασμα». Εάν το ρεύμα αυτό περιορίζεται από views ή φίλτρα, η επίδραση του MATCH_RECOGNIZE προσαρμόζεται αναλόγως. Εν προκειμένω, η αναγνώρισή του συμβαίνει μόνο στα AISBroadcast παλαιότητας έως 15 λεπτών. Η μορφή του μοτίβου, καθώς και διάφορες επιλογές σχετικές με την διαδικασία αναγνώρισης, ορίζονται συμπληρωματικά clauses του MATCH_RECOGNIZE, ως εξής:

- PARTITION BY: Προαιρετικό switch που εφαρμόζει το matching ξεχωριστά ανά τις διάφορες τιμές του ορίσματος / πεδίου του γεγονότος προέλευσης που δίδεται μετά το clause. Είναι το ανάλογο του groupwin[109] view για τον υπό εξέταση μηχανισμό. Εν προκειμένω, η λειτουργία επιμερίζεται σε παράθυρα όπου το κάθε ένα έχει μοναδικό vessel.mmsi, με άλλα λόγια η αναγνώριση (όπως περιγράφεται στα PATTERN και DEFINE) πραγματοποιείται ξεχωριστά για κάθε πλοίο του σεναρίου εξομοίωσης.
- PATTERN: Ορίζει το μοτίβο ως ακολουθία γεγονότων, κατ' αναλογία με μία κανονική έκφραση αλφαριθμητικού, κάνοντας χρήση των ίδιων ως επί το πλείστον modifiers / quantifiers. Στο εν λόγω παράδειγμα, για να αναγνωριστεί μία ακολουθία θα πρέπει να αποτελείται από τέσσερα ή περισσότερα AISBroadcast χαρακτηρισμένα ως A, ακολουθούμενα από ένα ή περισσότερα AISBroadcast τύπου B, ακολουθούμενα από ακόμη τέσσερα ή περισσότερα AISBroadcast του είδους A. Προφανώς, καθώς ένα γεγονός ως Java POJO ή EPL SCHEMA είναι πολύ πιο σύνθετο από απλούς χαρακτήρες ή κατηγορίες χαρακτήρων έχοντας πολλές ιδιότητες, χαρακτηρισμοί όπως οι A, B που χρησιμοποιήθηκαν εδώ αποτελούν απλώς placeholders για περαιτέρω αποσαφήνιση σε άλλο, συμπληρωματικό clause.
- DEFINE: Πρόκειται για προαιρετικό clause ύστερα από το οποίο πραγματοποιείται με χρήση μίας ή πολλαπλών boolean εκφράσεων, η αποσαφήνιση των μεταβλητών που χρησιμοποιήθηκαν στο PATTERN. Διάφοροι τελεστές όπως τομή και ένωση συνθέτουν τους επιθυμητούς περιορισμούς μεταξύ εκφράσεων, ενώ τελεστές ποσοτικοί, λογικοί, σχέσεις εγκλεισμού κ.α., ολοκληρώνουν κάθε έκφραση. Εν προκειμένω, A είναι εκπομπές AIS με αναφερόμενη ταχύτητα μεγαλύτερη από την ελάχιστη του εκπέμποντος Vessel, ενώ τα B είναι εκπομπές με την τιμή του ίδιου πεδίου να βρίσκεται υπό ή με ίση τιμή του δηλωθέντος αυτού ορίου του αντίστοιχου vessel.csv.
- MEASURES: Το υποχρεωτικό αυτό clause αποτελεί ένα μέσο αμφιμονοσήμαντης αντιστοίχησης μεταξύ placeholders και ονομάτων από το εσωτερικό του MATCH_RECOGNIZE “subquery” προς το περιβάλλον αυτού, που είναι το υπόλοιπο EPL statement (π.χ. SELECT). Η αντιστοίχηση γίνεται με τη δεσμευμένη λέξη AS, με τα placeholders να λαμβάνουν aliases τα αντίστοιχα πεζά γράμματα.

Η ακριβής χρήση των παραπάνω clauses δεν εξαντλείται από τα ερωτήματα αυτής της εφαρμογής. Ακόμη, υπάρχουν πρόσθετα, προαιρετικά clauses που μεταβάλλουν τη συμπεριφορά του μηχανισμού MATCH_RECOGNIZE όπως ο χειρισμός επικαλυπτόμενων ή μερικών matches, η χρήση χρονικών διαστημάτων παραγωγής εξόδου κ.α.. Στην τεκμηρίωση Esper αναλύονται εκτενώς όλες οι δυνατότητες του μηχανισμού, συνολικά και ανά clause.

Λόγω της χρήσης της άπληστης εκδοχής του quantifier «ένα ή περισσότερα» στο PATTERN clause, ταιριάζονται όλα τα διαθέσιμα B γεγονότα της ακολουθίας, με αποτέλεσμα τα B, b να ενθυλακώνονται ως collections, ακόμη και εάν υπήρξε μόλις ένα B γεγονός στο match. Η χρήση της άπληστης προσέγγισης είναι εσκεμμένη, καθώς είναι βάσιμο να υποτεθεί πως η πραγματική στιγμή της απόθεσης του αντικειμένου στο νερό ή της περισυλλογής του από αυτό συμβαίνει μόνο ύστερα από πιθανές αποτυχημένες απόπειρες ή τη σχετική προετοιμασία. Συνεπώς η χρήση της τελευταίας εκπομπής κατά τη στάση προσδίδει μεγαλύτερη ακρίβεια και αληθοφάνεια στο ερώτημα. Για την ολοκλήρωση της εισαγωγής των matches στο ρεύμα Stoppages, απαιτείται η επιλογή πεδίων από τα γεγονότα εκείνα. Προφανώς θα πρέπει τα πεδία να είναι συμβατά κατά όνομα και τύπο με τη σειρά των δηλωθέντων πεδίων του τύπου SimpleBroadcast. Έτσι, το SELECT clause επιλέγει:

- Με alias “time”, το πεδίο “timestamp”.
- Με alias “velocity”, το πεδίο “speed”.

[109] Παράδειγμα του view αυτού επεξηγήθηκε για το Calculated Overspeed HLE.

- Με alias “ship”, το πεδίο “vessel”, κατόπιν αλλαγής[110] τύπου σε Vessel με κλήση στην συνάρτηση cast()[111].
- Με alias “site”, το πεδίο “position”, κατόπιν αλλαγής τύπου σε Coordinates με κλήση στην συνάρτηση cast().

Κάθε πεδίο από τα παραπάνω προέρχεται από από το τελευταίο στοιχείο του collection b, χάρη σε κλήση στην συνάρτηση lastOf()[112].

packageDeliveryEPL:
Έχοντας στη διάθεσή του από τα προηγούμενα statements ένα ρεύμα με απλοποιημένες εκπομπές πλοίων που αντιπροσωπεύουν την τελευταία στιγμή μίας αδικαιολόγητης στάσης τους πριν τον προορισμό τους, το packageDeliveryEPL προχωρά στην εύρεση στάσεων από διαφορετικά πλοία με χωροχρονική εγγύτητα. Το εγχείρημά του υλοποιείται επίσης με το μηχανισμό matching του προπαρασκευαστικού statement MATCH_RECOGNIZE, το οποίο εμφανίζεται μετά από το FROM clause στο οποίο ως προέλευση των γεγονότων εισόδου έχει ασφαλώς οριστεί το ρεύμα Stoppages. Το μοτίβο προς εύρεση ορίζεται ως ένα SimpleBroadcast A είδους, ακολουθούμενο από οποιοδήποτε αριθμό B SimpleBroadcast γεγονότων, ακολουθούμενων από ένα SimpleBroadcast C είδους. Με άλλα λόγια πρόκειται για δύο στάσεις A, C με καμία, μία ή πολλές άλλες στάσεις μεταξύ τους στον χρονικό ορίζοντα. Η αποσαφήνιση που έπεται στο DEFINE clause, επιδεικνύει μία πολύ σημαντική δυνατότητα του μηχανισμού: να ορίζει σχέσεις μεταξύ των ιδιοτήτων των placeholders του μοτίβου. Εν προκειμένω, οι εκπομπές θα πρέπει να προέρχονται από διαφορετικά πλοία, οι θέσεις τους να απέχουν 100 μέτρα ή λιγότερο και έχουν συμβεί αμφότερες εντός 10 λεπτών, με το A πρώτο. Προφανώς το A αντιπροσωπεύει τη στάση του πλοίου που εναποθέτει το αντικείμενο, το C τη στάση εκείνου που το συλλέγει και το B ενδεχόμενες άλλες στάσεις πλοίων που όμως δεν εμπλέκονται σε αυτή τη δοσοληψία. Το MEASURES clause αντιστοιχίζει τα A και C με τα μικρά τους γράμματα, και από τα ζεύγη των στάσεων που καλύπτουν τα κριτήρια, το SELECT clause επιλέγει για αποστολή στον listener του ερωτήματος τα πεδία:

- Με alias “timeOfDispense”, το πεδίο a.time
- Με alias “timeOfRecovery”, το πεδίο c.time
- Με alias “dispenserVessel”, το πεδίο a.ship
- Με alias “collectorVessel”, το πεδίο c.ship
- Με alias “siteOfDispense”, το πεδίο a.site
- Με alias “siteOfRecovery”, το πεδίο c.site
- Με alias “dist”, η τιμή που επιστρέφει η distance με ορίσματα τις θέσεις των δύο στάσεων, αφού πρώτα αυτές επανέλθουν στον ακριβή τύπο δεδομένων τους, Coordinates, με τις ανάλογες κλήσεις στην cast().

Η χρονική διαφορά δεν διοχετεύεται στον listener αλλά υπολογίζεται εκεί εκ νέου από τα a.time και c.time. Οι χαρακτήρες διαφυγής για πρόσβαση στα επιμέρους πεδία των placeholder aliases του MATCH_RECOGNIZE που χρησιμοποιεί το SELECT clause, δηλ. μονά εισαγωγικά σε παρενθέσεις, είναι αναγκαίοι λόγω της μορφή στην οποία η μηχανή αναγνώρισης ενθυλακώνει τα αποτελέσματα μηχανισμών όπως ο παραπάνω: mapped keys[113].

[110] Το typecasting, αν και φαινομενικά περιττό, είναι απαραίτητο σε reference types ύστερα από διάφορους EPL χειρισμούς, συμπεριλαμβανομένου του MATCH_RECOGNIZE. Αυτοί πιθανότατα χρησιμοποιούν generic κώδικα λόγω της πληθώρας των τύπων που καλούνται να επεξεργαστούν. Καθώς η Esper είναι γραμμένη σε Java, ο γενικός προγραμματισμός στη γλώσσα αυτή επιτυγχάνεται με upcasting σχεδόν πάντοτε στο java.lang.Object, προκαλώντας στην περίπτωση του Package Delivery, ασυμφωνία τύπων.
[111] http://esper.espertech.com/release-6.1.0/esper-reference/html_single/index.html#epl-single-row-function-cast
[112] http://esper.espertech.com/release-6.1.0/esper-reference/html_single/index.html#enumeration-method-lastof
[113] 2.2. Event Properties: http://esper.espertech.com/release-6.1.0/esper-reference/html_single/index.html#eventrep-properties

Ο κώδικας που παρατίθεται στη συνέχεια αποτελεί μία εναλλακτική προσέγγιση για την αναγνώριση Package Delivery events, επίσης με χρήση του clause MATCH_RECOGNIZE. Για την σωστή λειτουργία του απαιτεί προσαρμοσμένο listener. Η παρακάτω λύση δεν εξαρτάται από ενδιάμεσα βήματα / ρεύματα, αλλά εντοπίζει τα υψηλού επιπέδου γεγονότα με ένα statement. Ένα μειονέκτημά της είναι πως οι στάσεις των πλοίων ορίζονται απλώς ως εκπομπές με ταχύτητα κάτω του ορίου vessel.minimumSpeed του εκάστοτε σκάφους, αλλά χωρίς αυτή απαραίτητα να βρίσκεται χρονικά μεταξύ άλλων εκπομπών του σκάφους με ταχύτητα ανώτερη του ιδίου ορίου. Συνεπώς είναι ιδιαίτερα επιρρεπής σε false positives σε περιοχές υψηλής πυκνότητας σκαφών και χαμηλής ταχύτητας, που όμως δεν αποτελούν ενδιάμεσα τμήματα ταξιδιών, παρά αρχή / τέλος αυτών. Χαρακτηριστικά παραδείγματα είναι τα λιμάνια και τα αγκυροβόλια.

```
String packageDeliveryEPL =
"SELECT * " +
"FROM AISBroadcast.win:time(900 sec) " +
"MATCH_RECOGNIZE(" +
"MEASURES A AS a, B AS b, C AS c " +
"ALL MATCHES " +
"PATTERN (A B* C) " +
"DEFINE" +
" A AS (A.speed <= A.vessel.minimumSpeed)," +
" C AS (C.speed <= C.vessel.minimumSpeed) AND" +
" (distance(A.position, C.position) <= 100) AND" +
" (C.timeStamp.getTime() - A.timeStamp.getTime() <= 600000) AND" +
" (C.vessel.mmsi != A.vessel.mmsi))";
```

Οι συναρτήσεις που καλούνται από τα statements, δηλώνονται:

```
esperConfiguration.addPlugInSingleRowFunction("distance",
Coordinates.class.getName(), "calculateDistance");
esperConfiguration.addPlugInSingleRowFunction("timeDifference",
Application.class.getName(), "esper_findTimeDifference");
```

Το αρχείο αναφοράς HLEs του ερωτήματος δημιουργείται ως εξής:

```
File packageDeliveryFile = new File(eventReportOutputPath + sep +
"Package Delivery Events.txt");
```

Η δημιουργία και ενεργοποίηση του ερωτήματος και του listener αυτού, αλλά και η συσχέτισή τους με το αρχείο αναφοράς, πραγματοποιείται με τις κλήσεις API:

```
EPStatement simpleBroadcastStatement =
esperAdministrator.createEPL(SimpleBroadcastEPL);
EPStatement stoppagesWindowStatement =
esperAdministrator.createEPL(stoppagesWindowEPL);
EPStatement insertToStopWinStatement =
esperAdministrator.createEPL(insertToStopWinEPL);
EPStatement packageDeliveryStatement =
esperAdministrator.createEPL(packageDeliveryEPL);
PackageDeliveryListener packageDeliveryListener = new
PackageDeliveryListener(packageDeliveryFile);
packageDeliveryStatement.addListener(packageDeliveryListener);
```

Σχετικά με τους χειρισμούς στην PackageDeliveryListener.update(), εκείνη σε κάθε κλήση της απλώς ελέγχει το όρισμα newEvents και στην περίπτωση που αυτό δεν είναι κενό (null), τότε επαναληπτικά για κάθε EventBean του array αυτού: Ανακτά όλα τα πεδία που παρέχονται από το SELECT clause του statement packageDeliveryEPL, ενώ με κλήσεις στους Vessel getters ανακτά επίσης ονόματα και MMSI των συναλλασσόμενων πλοίων. Από τα δύο timestamps υπολογίζει τη χρονική διαφορά και τη μετατρέπει σε δευτερόλεπτα. Στη συνέχεια, με κλήση στην writeToOutputFile(), εγγράφει στο packageDeliveryFile αναφορές με τις καταχωρήσεις αυτές να τοποθετούνται τη μία κάτω από την άλλη, σε αύξουσα χρονική σειρά. Ένα απόσπασμα από αρχείο Package Delivery Events.txt, όπου φαίνεται η μορφή μίας καταχώρησης:

```
EVENT: Package Delivery: Vessel BlueStarNaxos discarded a package,
subsequently collected by vessel FestosPalace, 3.549 seconds later, 1.0
meters away.
      Dispense details:
            Vessel:
                  MMSI: 239923000
                  Name: BlueStarNaxos
            Time: Fri Jul 27 03:42:38 EEST 2018
            Location: LON: 26.0 LAT: 7.0
      Recovery details:
            Vessel:
                  MMSI: 237611000
                  Name: FestosPalace
            Time: Fri Jul 27 03:42:42 EEST 2018
            Location: LON: 27.0 LAT: 7.0
```

Αξίζει να σημειωθεί, πως ο τρόπος συγγραφής του ερωτήματος το καθιστά ικανό να εντοπίσει ανταλλαγές πακέτων μεταξύ πλοίων «από κουπαστή σε κουπαστή», δηλαδή χωρίς απόθεση του δέματος στην επιφάνεια του νερού, αλλά με απ' ευθείας μεταβίβαση από το ένα πλοίο στο άλλο, εάν φυσικά οι ταχύτητές τους είναι αρκετά χαμηλές, ώστε να ικανοποιηθεί το αντίστοιχο κριτήριο του EPL κώδικα. Καθώς αυτή η μορφή μεταφοράς απαιτεί την ταυτόχρονη παρουσία των σκαφών σε πολύ κοντινές θέσεις, ενδέχεται να υπάρξει false positive άλλου HLE της εφαρμογής που αφορά σύγκρουση σκαφών.

Σε ότι αφορά Package Delivery false positives, υπάρχει ένας πρόσθετος έλεγχος που θα μπορούσε να επιβληθεί, βελτιώνοντας την ακρίβεια της επιλεγμένης και περισσότερο της εναλλακτικής προσέγγισης. Αυτός είναι ο έλεγχος της απόστασης από την πλησιέστερη στεριά. Περιοχές με πολύ υψηλή πυκνότητα πλοίων και χαμηλές ταχύτητες, όπως λιμένες, θα πρέπει σχεδόν πάντοτε να εξαιρούνται λόγω των false positives που εγείρονται εκεί, χωρίς να αντιστοιχούν σε πραγματική μεταβίβαση πακέτου, αλλά προκαλούνται από χειρισμούς πρόσδεσης στην στεριά, δίπλα σε άλλα σκάφη. Άλλωστε δεν υπάρχει κίνητρο οποιασδήποτε ύποπτης δοσοληψίας εκεί, καθώς είναι εύκολα ορατή από τη στεριά, από τα υπόλοιπα αγκυροβολημένα σκάφη και επίσης από δυνάμεις του Λιμενικού Σώματος που έχει σχεδόν διαρκή παρουσία εκεί. Η εφαρμογή διαθέτει τα μέσα προσδιορισμού αποστάσεων σημείων από στεριές (ως πολύγωνα) με χρήση αλγορίθμων που παρουσιάζονται σε άλλο ερώτημα. Ο λόγος που το Package Delivery δεν εμπλουτίστηκε από την εκμετάλλευση αυτών, είναι η υψηλή πολυπλοκότητά τους που επιβάλλει τη φειδωλή κλήση τους, ιδιαίτερα σε σενάρια εξομοίωσης με πολλά νησιά.

5.2.5: Imminent Collision

Το παρόν ερώτημα σκοπεύει στην πρόβλεψη χωρικής προσέγγισης δύο σκαφών εν πλω, αρκετά κοντινής ώστε να ενέχει τον κίνδυνο της μεταξύ τους σύγκρουσης. Επιτυγχάνει δε το στόχο αυτό, εντοπίζοντας στο ρεύμα των εκπομπών AIS ταυτόχρονες (ή σχεδόν χρονικά ταυτισμένες) εκπομπές υψηλής χωρικής εγγύτητας, προερχόμενες από διαφορετικά πλοία. Ως πρόσθετη απαίτηση, οι εκτιμήσεις για τις αμέσως επόμενες (μελλοντικές) θέσεις αυτών των πλοίων θα πρέπει να συγκλίνουν χωρικά, αυξάνοντας έτσι την αναμενόμενη εγγύτητά τους και μαζί με αυτήν, την πιθανότητα πρόσκρουσης. Οι εκτιμήσεις μελλοντικών θέσεων γίνονται σε βάθος δύο εκπομπών, με τις συγκρίσεις εγγύτητας να πραγματοποιούνται πάντοτε για την ίδια χρονική στιγμή, μελλοντική ή μη, μεταξύ των δύο σκαφών.

```
String imminentCollisionEPL =
"SELECT *, current_timestamp() AS time " +
"FROM AISBroadcast.win:length(250) " +
"MATCH_RECOGNIZE(" +
"MEASURES A AS a, B AS b, C AS c " +
"PATTERN (A B* C) " +
"DEFINE" +
" C AS (C.vessel.mmsi != A.vessel.mmsi) AND" +
" (Math.abs(C.timeStamp.getTime() - A.timeStamp.getTime()) <
broadcastInterval) AND " +
" (distance(C.position, A.position) < 200) AND" +
" ((distance(futurePosition(1, C), futurePosition(1, A)) < 70) OR" +
" (distance(futurePosition(2, C), futurePosition(2, A)) < 70)))";
```

Η προέλευση των LLEs του ερωτήματος, όπως καθορίζεται από το FROM clause, είναι το ρεύμα των AISBroadcast γεγονότων, για την ακρίβεια τα 250 πιο πρόσφατα από αυτά ανά δεδομένη στιγμή, χάρη στην επίδραση του view .win:length. Σε αυτά εντοπίζονται, από το μηχανισμό αναγνώρισης μοτίβων MATCH_RECOGNIZE, ακολουθίες αποτελούμενες από ένα AISBroadcast A είδους, ακολουθούμενο από οποιοδήποτε αριθμό B AISBroadcast γεγονότων, ακολουθούμενων από ένα AISBroadcast C είδους, όπως ορίζει το PATTERN clause. Με άλλα λόγια αναζητούνται, δύο εκπομπές A, C με καμία, μία ή πολλές άλλες μεταξύ τους, χρονικά. Σύμφωνα με την αποσαφήνιση που έπεται του DEFINE clause, στο πρώτο και τελευταίο γεγονός πρέπει να ισχύουν όλοι οι παρακάτω περιορισμοί:

- Τα πεδία vessel.mmsi πρέπει να είναι διαφορετικά.
- Τα πεδία timeStamp πρέπει να υποδηλώνουν ταυτόχρονα γεγονότα. Όπως έχει εξηγηθεί σε άλλο κεφάλαιο, η διαδικασία παραγωγής εκπομπών από την εφαρμογή λαμβάνει μέτρα για την διατήρηση του ρυθμού παραγωγής τους σε 1000/broadcastInterval ανά δευτερόλεπτο και με ελάχιστη ή καθόλου διαφορά φάσης (latency, προπορεία / βραδυπορεία μεταξύ εκπομπών). Καθώς όμως αμελητέες διαφορές μπορεί να εμφανιστούν, ο περιορισμός τις αποδέχεται, αρκεί φυσικά αυτές να μην ξεπερνούν το broadcastInterval, οπότε και θεωρούνται ετεροχρονισμένες.
- Η απόσταση, όπως προκύπτει από κλήση στην συνάρτηση distance με ορίσματα τα πεδία position των A και C, πρέπει να είναι μικρότερη του αναγραφόμενου ορίου (σε μέτρα).
- Η μελλοντική απόσταση, μεταξύ του επόμενου ή μεθεπόμενου ζεύγους εκπομπών από τα πλοία των εκπομπών A και C, πρέπει να είναι μικρότερη των αντίστοιχων ορίων σε μέτρα που αναφέρονται στο EPL statement. Η μελλοντική απόσταση υπολογίζεται με κλήση στην συνάρτηση futurePosition()

Με το wildcard «*» του SELECT clause, επιλέγονται αυτούσιες οι σειρές γεγονότων AISBroadcast που ταίριαξαν, ασφαλώς με τα γεγονότα της σειράς αυτής να αντιπροσωπεύονται στον listener από τα aliases που έλαβαν στο MEASURES clause, που δεν είναι άλλα από τα αντίστοιχα lower case γράμματα που εμφανίζονται στον ορισμό του μοτίβου. Ακόμη, επιλέγεται με alias “time” η ημερομηνία / ώρα ως UNIX time κατά την αποτίμηση του statement από τη μηχανή της Esper, χάρη σε κλήση στην συνάρτηση current_timestamp().

Παρατίθεται στη συνέχεια μία εναλλακτική επίλυση του προβλήματος, που χρησιμοποιεί ένα self-join του ρεύματος των εκπομπών, ενώ εφαρμόζει τα ίδια κριτήρια εύρεσης του σύνθετου γεγονότος. Καθώς υπάρχουν διαφοροποιήσεις στο SELECT τμήμα, το statement απαιτεί τις ανάλογες προσαρμογές στον listener για τη σωστή λειτουργία του:

```
String imminentCollisionEPL =
"SELECT * " +
"FROM AISBroadcast a1, AISBroadcast a2 " +
"WHERE " +
"(a1.vessel.mmsi != a2.vessel.mmsi) AND " +
"(Math.abs(a1.timeStamp.getTime() - a2.timeStamp.getTime()) <
broadcastInterval) AND " + "(distance(a1.position, a2.position) < 200)
AND " +
"((distance(futurePosition(1, a1), futurePosition(1, a2) < 70)) OR " +
"(distance(futurePosition(2, a1), futurePosition(2, a2) < 70)))";
```

Η καθολική μεταβλητή καθορισμού του διαστήματος μεταξύ διαδοχικών εκπομπών από τα πλοία, broadcastInterval, εσωτερικεύεται στον κώδικα EPL με την κλήση:

```
esperConfiguration.addVariable("broadcastInterval", Integer.class, new
Integer(AppConfiguration.getBroadcastInterval()));
```

Οι συναρτήσεις που καλούνται από τα statements, δηλώνονται με τις κλήσεις API:

```
esperConfiguration.addPlugInSingleRowFunction("distance",
Coordinates.class.getName(), "calculateDistance");
esperConfiguration.addPlugInSingleRowFunction("futurePosition",
Application.class.getName(), "esper_predictFuturePosition");
```

Η συνάρτηση futurePosition επιστρέφει την εκτιμώμενη μελλοντική θέση ενός πλοίου βάσει μίας εκπομπής του, τυπικά της πιο πρόσφατης. Δέχεται ως ορίσματα έναν ακέραιο που αναπαριστά αριθμό μελλοντικών περιόδων και τελευταίο ένα αντικείμενο AISBroadcast. Το πρώτο όρισμα καθορίζει το χρονικό βάθος της μελλοντικής πρόβλεψης. Παραδείγματος χάρη, η τιμή 1 αφορά την επόμενη θέση, 2 τη μεθεπόμενη, κ.ο.κ. Στην περίπτωση που η παραμετρική τιμή είναι μικρότερη του 1, χρησιμοποιείται η ελάχιστη αυτή τιμή παράγοντας πρόβλεψη για την αμέσως επόμενη θέση, ύστερα από ένα προειδοποιητικό μήνυμα. Από το δοθέν AISBroadcast εξάγονται και αποθηκεύονται τοπικά η γεωγραφική θέση, το heading και η ταχύτητα. Δημιουργείται ένα νέο Coordinates αντικείμενο προς επιστροφή, το οποίο τοποθετείται στην ίδια θέση με αυτή της εκπομπής - παραμέτρου. Από το γινόμενο της ταχύτητας της εκπομπής με το broadcastInterval (κατόπιν μετατροπής αυτού σε δευτερόλεπτα), εξάγεται η απόσταση που το πλοίο θα διανύσει στο επόμενο άλμα του, ενώ πολλαπλασιάζοντας την απόσταση αυτή με τον αριθμό περιόδων, παράγεται η πρόβλεψη της συνολικής απόστασης. Με κλήση στην instance method Coordinates.move() από το νέο αντικείμενο θέσης, με ορίσματα τον προσανατολισμό και τη συνολική απόσταση, το αντικείμενο αυτό αλλάζει συντεταγμένες αναλόγως, μετακινούμενο κατά την απόσταση αυτή μακριά από την αρχική του θέση, προς την κατεύθυνση του αζιμουθίου του πρώτου ορίσματος της κλήσης. Καθώς οι προβλέψεις βασίζονται μόνο σε υφιστάμενα heading και ταχύτητα, χάνουν αξιοπιστία όσο πιο μακρινές είναι οι μελλοντικές περίοδοι, αλλά είναι αρκετά ακριβείς για κοντινές αφού τα πλοία είναι γενικώς αδρανή και δεν μεταβάλλουν προσανατολισμό και ταχύτητα με ιδιαίτερη ευκολία. Ακόμη, χαμηλό broadcastInterval σημαίνει συγκριτικά ασφαλέστερες προβλέψεις για το ίδιο χρονικό βάθος σε περιόδους.

Το αρχείο αναφοράς HLEs του ερωτήματος δημιουργείται ως εξής:

```
File imminentCollisionFile = new File(eventReportOutputPath + sep +
"Imminent Collision Events.txt");
```

Η δημιουργία και ενεργοποίηση του ερωτήματος και του listener αυτού, αλλά και η συσχέτισή τους με το αρχείο αναφοράς, πραγματοποιείται με τις κλήσεις API:

```
EPStatement imminentCollisionStatement =
esperAdministrator.createEPL(imminentCollisionEPL);
ImminentCollisionListener imminentCollisionListener = new
ImminentCollisionListener(imminentCollisionFile);
imminentCollisionStatement.addListener(imminentCollisionListener);
```

Σε ότι αφορά τη διαδικασία παραγωγής αναφορών εξόδου από την μέθοδο update() του ImminentCollisionListener, σε κάθε κλήση της εκείνη απλώς ελέγχει το όρισμα newEvents και στην περίπτωση που αυτό δεν είναι null, τότε για κάθε EventBean του array αυτού ανακτά το πεδίο time, δηλαδή το αποτέλεσμα της current_timestamp(). Ο λόγος για τον οποίο δεν χρησιμοποιείται το timeStamp ενός από τα πλοία, είναι η ενδεχόμενη μικρή ασυμφωνία μεταξύ τους. Το πεδίο αυτό, καθώς αντιπροσωπεύει milliseconds από την αρχή του UNIX time, χρησιμοποιείται ως constructor argument δημιουργίας ενός java.util.Date που προτιμάται καθώς είναι σαφώς πιο εύκολα κατανοητό από τον αναγνώστη. Με κλήσεις για ανάκτηση και άλλων πεδίων, αντλούνται τα αντικείμενα position, heading και vessel των εκπομπών a και c. Τα πεδία πλοίων παρέχουν όνομα και MMSI, ενώ τα αντικείμενα προσανατολισμού τα αζιμούθια ως δεκαδικούς. Στη συνέχεια, η writeToOutputFile(), εγγράφει στο imminentCollisionFile αναφορές με τις καταχωρήσεις αυτές να τοποθετούνται, σε αύξουσα χρονική σειρά. Ένα απόσπασμα από αρχείο Imminent Collision Events.txt, όπου φαίνεται η μορφή μίας εγγραφής:

```
EVENT: Imminent Collision of vessels. Alert received at Fri Aug 10 08:43:18
EEST 2018
At 08:43:18, vessel of:
       MMSI: 237611000
       Name: FestosPalace
       Reported:
       Position: LON: 27.0 LAT: 7.0
       Heading: 180.0 degrees.
       Speed: 10.0 m/s.
At 08:43:18, vessel of:
       MMSI: 239923000
       Name: BlueStarNaxos
       Reported:
       Position: LON: 26.0 LAT: 7.0
       Heading: 180.0 degrees.
       Speed: 20.0 m/s.
```

5.2.6: Amphibious Assault

Το ερώτημα σκοπεύει στην ειδοποίηση εγγύτητας ξένων πολεμικών πλοίων σε στεριά, σε προσπάθεια έγκαιρης αντίδρασης στην περίπτωση απόβασης ή πριν από αυτή. Η υλοποίηση χρησιμοποιεί τον ορισμό του τύπου δεδομένων SimpleBroadcast, που δηλώθηκε για τις ανάγκες του ερωτήματος Package Delivery. Το απλοποιημένο αυτό είδος εκπομπών AIS δεν αντιπροσωπεύει εδώ στάσεις, αλλά γενικότερα όλες τις εκπομπές, που προέρχονται ωστόσο από πολεμικά σκάφη ξένης σημαίας. Οι απλοποιημένες αυτές εκπομπές των εν δυνάμει εισβολέων μεταφέρονται σε ξεχωριστό ρεύμα γεγονότων στο οποίο πραγματοποιείται άλλο στάδιο αξιολόγησης με βάση την απόστασή τους από την πλησιέστερη στεριά. Ο διαχωρισμός αυτός επιτρέπει τον περιορισμό της έκτασης του σχετικά χρονοβόρου και πολύπλοκου ελέγχου εγγύτητας.

```
String foreignNavyWindowEPL =
"CREATE WINDOW ForeignMilitaryBroadcasts.win:length(50) AS
SimpleBroadcast";

String foreignNavyBeepsEPL =
"INSERT INTO ForeignMilitaryBroadcasts " +
"SELECT" +
" timeStamp AS time," +
" speed AS velocity," +
" cast(vessel, Vessel) AS ship," +
" cast(position, Coordinates) AS site " +
"FROM AISBroadcast#length(10) " +
"WHERE ((vessel.flag != Flag.GR) AND (vessel.type =
VesselType.MILITARY))";

String amphibiousAssaultEPL =
"SELECT time, ship, site, landDistance(site, sea) AS ld " +
"FROM ForeignMilitaryBroadcasts " +
"HAVING landDistance(site, sea) < 10";
```

Η αναγνώριση του HLE γίνεται με τον κώδικα EPL οργανωμένο σειριακά στα αλφαριθμητικά των statements:

foreignNavyWindowEPL:
Με χρήση του CREATE WINDOW δημιουργείται ένα κενό αρχικά παράθυρο επεξεργασίας για γεγονότα τύπου SimpleBroadcast. Ο ορισμός του τύπου, όπως παρουσιάστηκε στο HLE Package Delivery, περιλαμβάνει: τη στιγμή της εκπομπής (πεδίο time), ταχύτητα (velocity), το εκπέμπον πλοίο (ship) και τη γεωγραφική θέση του (site). Το παράθυρο έχει όνομα ForeignMilitaryBroadcasts, ενώ το view .win:length() με όρισμα 50 περιορίζει για λόγους εξοικονόμησης πόρων τη χωρητικότητά του. Σε αντίθεση με το ερώτημα στο οποίο εμφανίστηκαν νωρίτερα, τα instances του τύπου σε αυτό το παράθυρο προορίζονται να αντιπροσωπεύσουν εκπομπές ξένων / εχθρικών πολεμικών πλωτών μέσων, και όχι ακινητοποιήσεις. Το όνομα του τύπου άλλωστε είναι αρκετά περιγραφικό και υπονοεί την ευελιξία χρήσης του.

foreignNavyBeepsEPL:
Το statement αυτό είναι υπεύθυνο για το πρώτο στάδιο αναγνώρισης του HLE, εντοπίζοντας ξένα σκάφη, μετουσιώνοντάς τα σε SimpleBroadcast και προωθώντας τα εντός του παραθύρου ForeignMilitaryBroadcasts με χρήση INSERT INTO, για πρόσθετη επεξεργασία από άλλα statements. Το "subquery" που ορίζει τα αντικείμενα εισαγωγής έχει προέλευση γεγονότων, σύμφωνα με το FROM clause, το ρεύμα των LLEs AISBroadcast. Από εκείνα, επιλέγονται από το WHERE clause ξένα πολεμικά πλοία χάρη στα κριτήρια που αυτό επιβάλει στα πεδία vessel.flag και vessel.type. Αυτά θα πρέπει να έχουν τιμές: διαφορετική σημαία του Flag.GR (ελληνική) και τύπο σκάφους VesselType.MILITARY (πολεμικό), αντιστοίχως. Οι περιορισμοί αυτοί θα πρέπει να συναληθεύουν (λέξη AND). Τα γεγονότα που ικανοποιούν την περιγραφή του WHERE, υπόκεινται στην κατάλληλη μορφοποίηση ώστε να είναι συμβατά με τη δήλωση τύπου του παραθύρου για το οποίο προορίζονται. Το έργο αυτό αναλαμβάνει το SELECT clause του statement, το οποίο επιλέγει μόνο τα 4 πεδία του AISBroadcast που χρησιμοποιούνται στον SimpleBroadcast με τα aliases του τύπου-προορισμού. Για ακόμη μία φορά η συνάρτηση cast() επαναφέρει τα reference type πεδία ship και site στον ακριβή τους τύπο δεδομένων.

amphibiousAssaultEPL:
Η αναγνώριση της επικείμενης απόβασης ολοκληρώνεται στο τελευταίο statement, το οποίο αντλώντας με το FROM clause του τις εκπομπές ξένων πολεμικών σκαφών εκ του παραθύρου ForeignMilitaryBroadcasts, επιλέγει όλα τους τα πεδία, εκτός αυτού της ταχύτητας και επίσης με την προσθήκη ενός νέου πεδίου με alias "ld" ως το αποτέλεσμα κλήσης στη συνάρτηση landDistance με ορίσματα τη θέση της εκάστοτε εκπομπής και του συνόλου των νησιών του σεναρίου (sea). Η συνάρτηση επιστρέφει δεκαδικό διπλής ακριβείας που αντιπροσωπεύει την ελάχιστη απόσταση του εκπέμψαντος από τις στεριές εντός του αρχιπελάγους, sea. Δεν αποστέλλονται όλα τα instances αυτά προς το listener του ερωτήματος, παρά μόνο εκείνα με την τιμή του πεδίου τους ld μικρότερη από το αναγραφόμενο όριο. Αυτός ο έλεγχος τιμής δεν πραγματοποιείται σε κάποιο WHERE clause, αλλά στο HAVING του ερωτήματος. Ο λόγος είναι πως το πρώτο γενικώς χρησιμοποιείται για να θέσει κριτήρια και περιορισμούς σε πεδία των εγγραφών, ενώ το HAVING δύναται να αξιολογήσει τιμές συναρτήσεων, συγκεντρωτικών ή μη, που υπολογίζονται επί των πεδίων, εκφράσεις, decorated streams, και εν γένει λειτουργεί σε μεταγενέστερο στάδιο επεξεργασίας των γεγονότων. Στην περίπτωση των γεγονότων προέλευσης του παρόντος, το πεδίο ld δεν αποτελεί πηγαία, αρχική πληροφόρηση, αλλά εξάγεται ύστερα από κλήση σε συνάρτηση.

Η τιμή του αρχιπελάγους sea, δηλαδή της λίστας όλων των νησιών του σεναρίου, εσωτερικεύεται στη μηχανή γεγονότων με την κλήση:

```
esperConfiguration.addVariable("sea", Sea.class, sea);
```

Η συνάρτηση landDistance που καλείται από το τελευταίο statement του ερωτήματος, δηλώνεται με:

```
esperConfiguration.addPlugInSingleRowFunction("landDistance",
Application.class.getName(), "esper_findDistanceFromLand");
```

Επί του ζητήματος του προσδιορισμού της εγγύτητας μίας θέσης στη στεριά, η παραπάνω δήλωση συνάρτησης καθιστά σαφές πως οι κλήσεις της landDistance(Coordinates, Sea) από κώδικα EPL, αποτέλεσμα έχουν την κλήση της public static double Application.esper_findDistanceFromLand(Coordinates, Sea) η οποία διαθέτει τα μέσα υπολογισμού της απόστασης αυτής. Κατά την εκεί διαδικασία, λαμβάνονται από το αντικείμενο sea τα νησιά ως διασυνδεδεμένη λίστα και επαναληπτικά, σε κάθε ένα από αυτά καλείται η μέθοδος αντικειμένου distanceFrom(), με παράμετρο τη δοθείσα θέση. Οι αποστάσεις από κάθε νησί συγκρίνονται και επιστρέφεται η ελάχιστη. Κατά απολύτως ανάλογο τρόπο, ύστερα από κλήση της Island.distanceFrom() κάθε νησιού, λαμβάνονται οι ακτές αυτού (αντικείμενα Shore), και για κάθε μία από αυτές, καλείται η - ομώνυμη - instance μέθοδος LinearSegment.distanceFrom() που ως όρισμα παραλαμβάνει το ίδιο Coordinates reference για τη δική της σύγκριση. Η μέθοδος εκείνη υπολογίζει την απόσταση του Coordinates σημείου από το ευθύγραμμο τμήμα σεβόμενη τα άκρα του: δεν το αντιμετωπίζει ως ευθεία. Η λειτουργία της βασίζεται στη χρήση προβολών διανυσμάτων και η επίλυση περιγράφεται αναλυτικά στο σχολιασμό[114] της LinearSegment.distanceFrom().

[114] Αρχείο sailAway\application\dataTypes\lowLevelTypes\src\LinearSegment.java.

Το αρχείο αναφοράς HLEs του ερωτήματος δημιουργείται ως εξής:

```
File amphibiousAssaultFile = new File(eventReportOutputPath + sep +
"Amphibious Assault Events.txt");
```

Η δημιουργία και ενεργοποίηση του ερωτήματος και του listener αυτού, αλλά και η συσχέτισή τους με το αρχείο αναφοράς, πραγματοποιείται με τις κλήσεις API:

```
EPStatement foreignNavyWindowStatement =
esperAdministrator.createEPL(foreignNavyWindowEPL);
EPStatement foreignNavyBeepsStatement =
esperAdministrator.createEPL(foreignNavyBeepsEPL);
EPStatement amphibiousAssaultStatement =
esperAdministrator.createEPL(amphibiousAssaultEPL);
AmphibiousAssaultListener amphibiousAssaultListener = new
AmphibiousAssaultListener(amphibiousAssaultFile);
amphibiousAssaultStatement.addListener(amphibiousAssaultListener);
```

Κατά την παραγωγή αναφορών εξόδου μέσω της AmphibiousAssaultListener.update(), εκείνη ύστερα από κάθε κλήση της ελέγχει την παράμετρο newEvents και εάν αυτή δεν είναι null, επαναληπτικά για κάθε EventBean του array αυτού (που ενθυλακώνει ένα decorated SimpleBroadcast) ανακτά όλα τα διαθέσιμα πεδία: time, ship, site και την ελάχιστη απόσταση από στεριά, ως "Id". Από το πεδίο ship εξάγονται ακόμη όνομα και MMSI. Στη συνέχεια, η writeToOutputFile(), ενημερώνει το amphibiousAssaultFile με αναφορές, καταχωρήσεις των οποίων εγγράφονται σε αύξουσα χρονική σειρά. Ενδεικτικά, παρατίθεται τμήμα από αρχείο Amphibious Assault Events.txt, όπου φαίνεται η μορφή μίας εγγραφής:

```
EVENT: Possible imminent amphibious assault by a foreign Navy, due to their
unauthorized sail within National waters.
Details:
	Date and time: Mon Jul 30 02:58:47 EEST 2018
	Offending foreign military vessel:
		MMSI: 200000004
		Name: Barbaros
	Position: LON: -8.0 LAT: 3.0
	Distance from nearest landmass: 8.54400374531753
```

Το ερώτημα αυτό μπορεί να δεχθεί μερικές επεκτάσεις ώστε να επιστρέφει το όνομα του νησιού που αντιστοιχεί στην ελάχιστη απόσταση κάποιας αναφοράς. Ακόμη, με αρκετές τροποποιήσεις, θα μπορούσε να δέχεται μόνο κάποιο υποσύνολο των νησιών του σεναρίου εξομοίωσης, καθώς στη παρούσα προσέγγιση γίνεται η σιωπηρή υπόθεση πως δεν υπάρχουν ξένα νησιά στο σενάριο. Ακόμη, μία απλούστερη εκδοχή του, απαλλαγμένη από έλεγχο σημαίας και τύπου, και επίσης με μικρότερα όρια εγγύτητας σε νησιά, θα λειτουργούσε ως μηχανισμός εύρεσης HLEs όπως «Επικείμενη πρόσκρουση σε στεριά».

5.3: Σχολιασμός επί της αναγνώρισης σύνθετων ναυτιλιακών γεγονότων από εκπομπές AIS

Ύστερα από την παρουσίαση της διαδικασίας εντοπισμού μερικών ναυτιλιακών γεγονότων υψηλού επιπέδου - και ενδιαφέροντος - από ρεύμα απλοϊκών συμβάντων χαμηλής αξίας στην παρούσα εφαρμογή, κρίνεται σκόπιμη η τοποθέτηση ορισμένων θεμάτων που μπορεί να επιδράσουν δυσμενώς στην αξιοπιστία της διαδικασίας αυτής. Τα ζητήματα αφορούν όχι μόνο το ρεύμα AISBroadcast και τη μηχανή της Esper, αλλά γενικότερα τη φύση των εκπομπών AIS από πλοία ως LLEs και οποιαδήποτε τεχνολογία αναγνώρισης HLEs από αυτά.

Εξετάζοντας στις εκπομπές AIS ως πηγή αναγνώρισης γεγονότων ενδιαφέροντος, θα πρέπει να ληφθούν υπόψη τα εξής:

Η κάλυψη του στόλου ανά τις θάλασσες του κόσμου από το δίκτυο μετάδοσης των εκπομπών αυτών δεν είναι αυτονόητη ή δεδομένη. Καθώς τα ηλεκτρομαγνητικά σήματα εκτίθενται σε διάφορα είδη αλλοιώσεων όσο η απόσταση που διανύουν αυξάνεται, πέραν κάποιας απόστασης καθίστανται μη αναγνώσιμα ή ασθενή. Το πρόβλημα αυτό δεν υπάρχει κοντά σε ηπειρωτικές στεριές ή αρχιπελάγη όπου βρίσκονται σταθμοί λήψης των εκπομπών αυτών και μεταφοράς τους δικτυακά. Για περιοχές ανεπαρκούς κάλυψης όπως τμήματα ωκεανών, χρησιμοποιείται η τεχνολογία satellite-AIS, S-AIS, για λήψη εκπομπών από τεχνητούς δορυφόρους. Σε κάθε περίπτωση όμως ενδέχεται να υπάρχουν κινήσεις πλοίων που δεν αποτυπώνονται στο ρεύμα των εκπομπών.

Οι πληροφορίες των πεδίων AIS μπορεί να υπόκεινται σε σφάλματα μετρήσεων ή ανακρίβειες, καθώς ο εξοπλισμός AIS αλλά και οι αισθητήρες από όπου πηγάζουν οι διάφορες μετρήσεις κάθε σκάφους μπορεί να διαφέρουν σε ακρίβεια, ποιότητα κατασκευής κ.λ.π.

Οι εκπομπές ίσως καταφθάνουν στο σύστημα αναγνώρισης σύνθετων γεγονότων με διακύμανση στην καθυστέρησή τους (μεταβλητό latency). Μικρές διακυμάνσεις μπορούν να αντιμετωπιστούν με μηχανισμό ο οποίος θα τις διατάσσει στην σωστή χρονική σειρά εντός κάποιου buffer, αρκεί τα timestamps των εκπομπών να είναι αληθή.

Οι εκπομπές AIS ενδέχεται να δεχτούν εκούσια χειραγώγηση είτε με απενεργοποίηση του ενσωματωμένου στα σκάφη εξοπλισμό, όπως παρατήρησαν και οι Alessandrini A., Alvarez M. και λοιποί (2016), ή με προγραμματισμό αυτού ώστε να μεταδίδει δεδομένα διαφορετικά όσων συλλέγονται από τους αισθητήρες του πλοίου, αν πρόκειται για δυναμικά πεδία όπως heading, ή διαφορετικά από τα ορθά στατικά δεδομένα του ταξιδιού / session (όπως MMSI). Τυπικά αυτή η συμπεριφορά εξηγείται από την προσπάθεια των εμπλεκομένων να αποκρύψουν τις παραβατικές ενέργειές τους. Μία στρατηγική αντιμετώπισης αυτού του προβλήματος, είναι η συμπληρωματική χρήση LLEs διαφορετικής φύσης, ανεξάρτητης προέλευσης, και εκτός πρόσβασης πληρωμάτων. Εκείνα τα γεγονότα χαμηλού επιπέδου θα μπορούσαν μερικώς έστω να επαληθεύσουν δεδομένα AIS και να εντοπίσουν αποκλίσεις. Ακόμη, θα χρησίμευαν στην επέκταση της δυνατότητας αναγνώρισης σύνθετων γεγονότων του συστήματος. Παράδειγμα τέτοιων LLEs είναι τα στίγματα marine radar.
Η επιστημονική βιβλιογραφία διαθέτει άρθρα που αναγνωρίζουν τις αδυναμίες αυτές του συστήματος AIS. Ενδεικτικά, οι Ray C., Iphar C. και Napoli A. (2016), περιγράφουν τεχνικές αποκάλυψης ψευδών εκπομπών με ενδελεχή εξέταση παραμέτρων του φυσικού επιπέδου των μεταδόσεων, όπως συχνότητα, εύρος ζώνης, ισχύς του σήματος συναρτήσει της αναφερόμενης απόστασης, χρονικές υστερήσεις κ.α. Εκμεταλλευόμενοι διαφορετική μεθοδολογία, οι Katsileris F., Braca P. και Coraluppi S. (2013), τονίζουν τη σημασία της διαποικίλησης ειδών γεγονότων εισόδου και προτάσσουν το ναυτικό radar ως το πιο δόκιμο μέσο επαλήθευσης της εγκυρότητας των μηνυμάτων AIS. Ακόμη, οι Fischer Y. και Bauer A. (2010), επεκτείνουν την ίδια προσέγγιση με τη στρατηγική της σύντηξης ακόμη περισσότερων πηγών μετρήσεων σε ένα ενοποιημένο τύπο LLE, αντικειμενοστραφούς δομής και αναπαράστασης. Οι αισθητήριες προελεύσεις των μηνυμάτων στην εν λόγω επινόηση, πέραν των μηνυμάτων AIS και στιγμάτων radar, περιλαμβάνουν εικόνες και video από επίγειους και ιπτάμενους σταθμούς όπως μη επανδρωμένα αεροσκάφη.

Σε γενικότερη επέκταση της λογικής αυτής, πέραν των πολλαπλών ειδών γεγονότων εισόδου στο δεδομένο σύστημα αναγνώρισης, μπορούν να χρησιμοποιηθούν και διαφορετικά, ανεξάρτητα συστήματα αναγνώρισης σύνθετων καταστάσεων προς επαλήθευση και σύγκριση των ευρημάτων του παρόντος. Παραδείγματος χάρη, στην περίπτωση της μεταβίβασης δέματος μετξύ πλοίων στην ανοικτή θάλασσα, ένα σύστημα αναγνώρισης φόρτω-εκφόρτωσης ύποπτων αντικειμένων, τροφοδοτούμενο από εικόνες κλειστού κυκλώματος τηλεόρασης σε λιμένες, μπορεί να επιβεβαιώσει ή να διαψεύσει Package Delivery HLE instances. Οι τεχνικές αναγνώρισης σύνθετων γεγονότων από βίντεο έχουν επιτυχώς εφαρμοστεί, ενώ υπάρχουν και σχετικές επιστημονικές δημοσιεύσεις, όπως εκείνη των Artikis A., Skarlatidis A. και Paliouras G. (2010), με τίτλο “Behaviour Recognition from video content: A logic programming approach”.

Οι προαναφερθείσες παθογένειες των εκπομπών AIS δεν αποτελούν τη μοναδική πηγή σφαλμάτων στην αναγνώριση σύνθετων γεγονότων, καθώς η διαδικασία αναγνώρισης καθ' αυτή, συχνά έχει αδύναμα σημεία. Τα εν λόγω σφάλματα διακρίνονται σε γεγονότα που αναγνωρίζονται χωρίς ωστόσο να ανταποκρίνονται σε μία υπαρκτή κατάσταση (false positives), και σε εκείνα που διαφεύγουν αναγνώρισης ενώ έχουν συμβεί (false negatives).

Στην προσπάθεια περιορισμού των συνεπειών των σφαλμάτων ενός συστήματος αναγνώρισης σύνθετων γεγονότων και εν προκειμένω του έργου που αναλαμβάνουν τα ερωτήματα EPL της sailAway, ο χρήστης δεν θα πρέπει να επαφίεται μόνο στον τίτλο των ερωτημάτων και στον σκοπό τους, ή με άλλα λόγια, στην περιφραστική ανθρωπίνως αναγνωρίσιμη περιγραφή τους. Προ αποδοχής των HLEs που εντοπίζονται από τη μηχανή, θα πρέπει να έχει εδραιωθεί μία βασική κατανόηση του κώδικα των ερωτημάτων EPL (αυτός είναι άλλωστε και ο σκοπός των προηγούμενων ενοτήτων). Τα ερωτήματα είναι απλώς ορισμοί σύνθετων γεγονότων εκφρασμένοι ως σχέσεις μεταξύ απλούστερων μονάδων, των υπογεγονότων / LLEs. Συνεπώς η ικανότητα ενός ερωτήματος να εντοπίζει κάποιο υψηλότερου επιπέδου γεγονός άνευ σφαλμάτων, εξαρτάται από την «ευστοχία» της έκφρασής του σε όρους απλούστερων γεγονότων. Ο κώδικας αποκαλύπτει ο ίδιος τις ενδεχόμενες αδυναμίες του. Για παράδειγμα, το ερώτημα Package Delivery, ελέγχει για δύο πλοία που σταμάτησαν κοντά στο ίδιο σημείο σε κοντινές χρονικές στιγμές και μόνο για αυτό, καθώς η εφαρμογή είναι περιορισμένη στη χρήση AIS ως γεγονότα εισόδου. Ο κώδικας EPL δεν μπορεί πέρα από τις εκπομπές των πλοίων να «δει» το δέμα, το περιεχόμενό του ή τη μεταφορά του από το ένα κατάστρωμα στο άλλο.

Ιδανικά, θα πρέπει να υπάρχουν καθόλου ή ελάχιστα σφάλματα. Ένα false positive θα ήταν μάλλον προτιμότερο από ένα false negative, αλλά αυτό εξαρτάται επίσης από τη φύση του εκάστοτε HLE, τη σοβαρότητα της κατάστασης που αντιπροσωπεύει, αλλά και την πολιτική των Αρχών που ενδέχεται να χρησιμοποιούν κάποιο όμοιο σύστημα θαλάσσιας εποπτείας, π.χ. πολιτική «μηδενικής ανοχής» στις παραβάσεις.

Παραμένοντας στο παράδειγμα του Package Delivery, πρόκειται κατά την υποκειμενική άποψη του συγγραφέως, για σοβαρό συμβάν που θα πρέπει να οδηγεί σε έλεγχο των σκαφών έστω προληπτικά και με το ενδεχόμενο να είναι εν τέλει περιττός. Αντίθετα, false negative ενός Reported Overspeed (δηλαδή αυτό που δεν εντοπίστηκε) δεν αποτελεί εξαιρετικά ανησυχητικό γεγονός και ιδιαίτερα στην ανοικτή θάλασσα. Ένα επίσης σοβαρό γεγονός είναι η επικείμενη σύγκρουση μεταξύ σκαφών. Εδώ θα προτιμούσαμε να ανεχτούμε μερικά false positives εάν η συγγραφή του ερωτήματος εξασφάλιζε ελάχιστα ή καθόλου false negatives.

Εφιστώντας την προσοχή του αναγνώστη στους κινδύνους που εμπεριέχουν τα παραπάνω ζητήματα, εκείνος θα έρθει σε θέση, είτε ως χρήστης να λαμβάνει τα αποτελέσματα τέτοιων μηχανισμών με την απαραίτητη δόση δυσπιστίας, εργαζόμενος προς πρόσθετη διερεύνηση, ή ως μηχανικός να εργάζεται για υψηλότερη ακρίβεια αναγνώρισης.

Κεφάλαιο 6: Επίλογος

Περιεχόμενα κεφαλαίου

6.1: Περαιτέρω ανάπτυξη

Οι όροι "βελτιώσεις" και "μελλοντικές επεκτάσεις" ίσως γίνονται αντιληπτές με ένα βαθμό εννοιολογικής επικάλυψης, ιδιαίτερα με την διαπίστωση πως οι ενέργειες που περιγράφονται από αμφότερους μπορούν μόνο μελλοντικά να πραγματοποιηθούν, δεδομένης της παρούσας μορφής της εργασίας. Ακόμη, και οι δύο αφορούν αλλαγές προς μία «καλύτερη» κατάσταση. Για αποσαφήνιση της σημασίας τους στα πλαίσια της εφαρμογής sailAway, δίνονται εδώ οι δύο επεξηγήσεις:

- Βελτιώσεις της εφαρμογής είναι οι ενέργειες εκείνες που εξυπηρετούν τη λειτουργία του λογισμικού (όπως μείωση της πολυπλοκότητας / ακριβέστερη εξομοίωση) και/ή την εμπειρία των χρηστών του, με τον περιορισμό πως οι ρόλοι και λειτουργίες της εφαρμογής δεν μεταβάλλονται. Έτσι, ενώ μπορεί να υπάρξει πληθώρα βελτιώσεων, σε κάθε περίπτωση το αποτέλεσμα όλων αυτών επί της παρούσας υλοποίησης θα εξακολουθεί γενικώς να περιορίζεται σε όσα παρουσιάστηκαν εκτενώς στα Κεφάλαια 4 και 5.
- Οι μελλοντικές επεκτάσεις είναι προτάσεις εισαγωγής πρόσθετης λειτουργικότητας, όπως δυνατότητες αναγνώρισης γεγονότων με εναλλακτικές τεχνικές, χρήση νέων τύπων ροών ανάδρασης προς το χρήστη για τα διάφορα στάδια λειτουργίας, όπως αξιοποίηση του περιβάλλοντος οπτικοποίησης για αμφίδρομη επικοινωνία και εισαγωγή δεδομένων.

Ύστερα από ενέργειες που εμπίπτουν σε οποιαδήποτε από τις παραπάνω κατηγορίες, η sailAway θα πρέπει να εξακολουθεί να είναι, όπως αναγράφεται στην Περίληψη και Εισαγωγή, μία εφαρμογή αναγνώρισης σύνθετων ναυτιλιακών γεγονότων από μία διαδικασία εξομοίωσης θαλάσσιας κυκλοφορίας και παραγωγής εκπομπών AIS.

6.1.1: Βελτιώσεις

Αλγόριθμοι εξομοίωσης πλεύσης:
Το παρόν μοντέλο φυσικής που χρησιμοποιείται για την παραγωγή εκπομπών AIS γνησίως εξομοιούμενων ταξιδιών έχει κάποιο περιθώριο βελτίωσης. Αυτό συγκεκριμένα εντοπίζεται στα άλματα ευθύγραμμης πλεύσης, κατά τα οποία τα πλοία μεταβάλλουν την ταχύτητά τους (υπενθυμίζεται πως η πλεύση σε τόξα γίνεται με σταθερή γραμμική ταχύτητα).
Επί της παρούσας υλοποίησης, η επιτάχυνση, αποτελεί αλγόριθμο ο βασιζόμενο σε τμηματικούς υπολογισμούς (τα "άλματα" του Κεφ. 4) που χρησιμοποιεί αντί συνάρτησης ταχύτητας, συνάρτηση που υπολογίζει διαφορά ταχύτητας. Αυτή η διαφορά μεταξύ εκπομπών εξαρτάται από την τελευταία υπολογισμένη τιμή της ταχύτητας και από την διαφορά της από τη μέγιστη δυνατή τιμή της (πεδίο maximumSpeed). Ο υπολογισμός αυτός εξαρτάται ακόμη και από τον συντελεστή επιτάχυνσης του σκάφους (πεδίο acceleration). Στην περίπτωση της επιβράδυνσης δε, στην τρέχουσα μορφή της, η ταχύτητα ελέγχεται μόνο ύστερα από κάποιο σημείο συγκεκριμένης εγγύτητας στην είσοδο της στροφής, και σε κάθε μετέπειτα άλμα προς αυτή. Αν υπερβαίνει το όριο της στροφής, μειώνεται. Διαφορετικά παραμένει αμετάβλητη για τη συγκεκριμένη επανάληψη, γεγονός που προδίδει καθυστερήσεις (μικρές γενικώς) άφιξης στον προορισμό. Ακόμη, οι τμηματικές μειώσεις ταχύτητας αυτές είναι συνάρτηση της απόστασης που απομένει ύστερα από κάθε άλμα προς το σημείο εισόδου

στο τόξο, μεταβλητή που δεν σχετίζεται με τις ικανότητες επιβράδυνσης ενός σκάφους (το πεδίο deceleration δεν συμμετέχει στον υπολογισμό).

Σε μία ρεαλιστικότερη υλοποίηση, η επιτάχυνση και η επιβράδυνση, θα πρέπει να αποτελούν συναρτήσεις ταχύτητας όπου ο χρόνος, ή εναλλακτικά η μετατόπιση, θα υπολογίζεται σε ρόλο ανεξάρτητης μεταβλητής. Η ακριβής μορφή των συναρτήσεων θα πρέπει να εξαρτάται από τους συντελεστές επιτάχυνσης και επιβράδυνσης καθώς και από την ελάχιστη και μέγιστη ταχύτητα, κατά τρόπο όχι πολύ διαφορετικό από αυτόν της τρέχουσας υλοποίησης. Ταυτόχρονα, θα πρέπει τουλάχιστον η συνάρτηση της επιτάχυνσης να παρουσιάζει ασυμπτωτική συμπεριφορά, όπως ήδη ισχύει, ενώ η συνάρτηση της επιβράδυνσης μπορεί να έχει πιο απλή μορφή, ακόμη και γραμμική. Επιπλέον, εξακολουθεί να υπάρχει η απαίτηση για αποκλειστικά επιταχυνόμενη ή επιβραδυνόμενη κίνηση στα ευθύγραμμα τμήματα της διαδρομής. Στην πραγματικότητα βέβαια υπάρχουν και περιπτώσεις όπου τα πλοία κινούνται ευθύγραμμα με σταθερή - πρακτικά - ταχύτητα, έχοντας αναπτύξει την μέγιστη δυνατή και διατηρώντας τη. Επιπροσθέτως, η κίνηση στο ευθύγραμμο τμήμα μεταξύ δύο τόξων θα πρέπει να καταλαμβάνει τον ελάχιστο δυνατό χρόνο, δεδομένων των χαρακτηριστικών κάθε σκάφους και της διαδρομής του.

Οι παραπάνω απαιτήσεις σημαίνουν αφ' ενός πως το σημείο ελέγχου ταχύτητας πριν την είσοδο της στροφής δεν θα είναι πια μία προκαθορισμένη απόσταση από την εν λόγω αρχή του τόξου, αλλά θα εξαρτάται από τις εκάστοτε συνθήκες. Ακόμη, η φάση της ευθύγραμμης κίνησης θα αποτελείται το πολύ από δύο φάσεις με την πρώτη να είναι επιταχυνόμενη κίνηση, ακολουθούμενη από επιβράδυνση. Αναλόγως συνθηκών (βλ. παρακάτω), ενδέχεται να υπάρχει μόνο μία από τις δύο.

Αν το ευθύγραμμο τμήμα μεταξύ τόξων είναι το AB, με γνωστές τις συναρτήσεις μεταβολής ταχύτητας αλλά και τις τιμές ταχύτητας στα A και B (ως όρια των προσκείμενων τόξων), τότε η διαδικασία υπολογισμού του σημείου μεταξύ A και B από το οποίο θα πρέπει να διακοπεί η επιτάχυνση και να αρχίσει άμεσα η επιβράδυνση, μοιάζει με ένα πρόβλημα βελτιστοποίησης υπό περιορισμούς.

Σε επέκταση των παραπάνω αλλαγών, θα πρέπει να προστεθεί πλεονάζουσα λειτουργικότητα στη διαδικασία υπολογισμού ορίων ταχύτητας στις στροφές. Στην υφιστάμενη υλοποίηση, κάθε πλοίο είναι σε θέση να απολέσει το πλεόνασμα ταχύτητάς του ώστε να συμμορφωθεί με το όριο ταχύτητας της στροφής εν όψει, εντός ενός σταθερού μήκους πριν από την είσοδο σε αυτή. Η μεταβολή αυτή συμβαίνει ανεξαρτήτως των τεχνικών δυνατοτήτων του πλοίου και του ύψους αυτού του πλεονάσματος. Έτσι ενδέχεται να εμφανιστούν αυθαίρετες διαφορές ταχύτητας που δεν αντανακλούν ρεαλιστικές δυνατότητες επιβράδυνσης. Κατά τη βελτιωμένη όμως υλοποίηση που προτείνεται, τα μεταβλητά, από πλευράς τοποθέτησης, σημεία μετάβασης από επιτάχυνση σε επιβράδυνση, σε συνδυασμό με στροφές που μπορεί να βρίσκονται σε μεγάλη χωρική εγγύτητα, αποκαλύπτουν την αδυναμία της τρέχουσας «μυωπικής» διαδικασίας υπολογισμού ορίων, καθώς αυτή λογίζει μεμονωμένα την εκάστοτε στροφή. Χαρακτηριστικό παράδειγμα που αναδεικνύει το πρόβλημα είναι ένα ζεύγος διαδοχικών στροφών με την πρώτη να έχει εξαιρετικά μεγάλη ακτίνα στροφής, και τη δεύτερη εξαιρετικά μικρή ενώ ακόμη το ενδιάμεσο ευθύγραμμο τμήμα που χωρίζει τα δύο τόξα είναι ελάχιστα μεγαλύτερο από το laneWidth. Εάν ακόμη υποτεθεί πως πριν από αυτό το σύστημα στροφών της διαδρομής υπάρχει ένα μακρύ ευθύγραμμο τμήμα στο οποίο πλέει σκάφος ικανό να αναπτύξει μεγάλη ταχύτητα ενώ έχει συμβατικές δυνατότητες επιβράδυνσης, τότε εύκολα μπορεί κάποιος να προβλέψει πως το πλοίο θα ξεπεράσει κατά πολύ το όριο ταχύτητας της δεύτερης στροφής, καθώς δεν θα προλάβει να επιβραδύνει επαρκώς στο ενδιάμεσο τμήμα.

Η λύση σε αυτήν την περίπτωση μοιάζει να είναι ένας αλγόριθμος που λειτουργεί συμπληρωματικά της υπάρχουσας διαδικασίας, και μετά από αυτή (ως δεύτερο «πέρασμα»). Αφού δηλαδή υπολογιστούν τα όρια για κάθε στροφή αγνοώντας τη συνολική διαδρομή (πρώτο «πέρασμα»), θα πρέπει να εκτελεστεί ενός είδους προς τα πίσω επαγωγής: Ξεκινώντας από την τελευταία στροφή της διαδρομής και διαβάζοντας το όριό της, ο αλγόριθμος λαμβάνοντας ακόμα υπ' όψιν τη συνάρτηση επιβράδυνσης του συγκεκριμένου πλοίου του ταξιδιού, υπολογίζει τη μέγιστη ταχύτητα από την έξοδο της προτελευταίας στροφής από την οποία το πλοίο μπορεί να επιβραδύνει ώστε να μην παραβιάσει το όριο της τελευταίας. Εάν η ταχύτητα αυτή είναι μικρότερη του τρέχοντος ορίου της προτελευταίας, το όριο αντικαθίσταται από την υπολογισμένη τιμή. Με αφετηρία το νέο όριο, είτε αυτό έχει αλλάξει ή όχι, εξετάζεται το όριο της τρίτης από το τέλος στροφής, και ούτω καθ' εξής.

Εάν το όριο κάποιας στροφής αλλάξει από αυτή τη διαδικασία, είναι προφανές πως το ευθύγραμμο τμήμα που έπεται αυτής θα αναλωθεί εξ' ολοκλήρου σε ένα μόνο είδος κίνησης, και αυτή θα είναι μία επιβράδυνση. Αντίθετα, σε περιπτώσεις όπου το όριο ταχύτητας μίας στροφής είναι πολύ υψηλό και δεν υπάρχουν κλειστές και/ή κοντινές στροφές μετά από αυτή, ενδέχεται το ευθύγραμμο τμήμα που προηγείται αυτής να αναλωθεί όλο στη φάση επιτάχυνσης, ειδικά εάν το πλοίο δεν είχε αρκετό χώρο να το φθάσει, ή υπολείπεται σε μέγιστη ταχύτητα κ.α.

Η παραπάνω πρόταση, αν και προσφέρει ακρίβεια και ντετερμινισμό, ίσως δεν συνάδει με τον πραγματικό βαθμό προετοιμασίας του πληρώματος που ασχολείται με την πλοήγηση κάποιου σκάφους. Οι διαδικασίες αυτές μοιάζουν περισσότερο με «σχέδιο πτήσης» παρά με «σχέδιο πλεύσης», αντανακλώντας τις διαφορετικές ανοχές στα μέσα μεταφοράς. Ίσως μία πιο εφαρμοσμένη πρακτική να περιλαμβάνει κάποια υλοποίηση ασαφούς / προμπαμπιλιστικής λογικής που φυσικά σχεδόν τελείως αναιρεί την ανάλυση των παραγράφων που προηγήθηκαν.

Επίδραση της μάζας στις επιδόσεις:
Σε επέκταση της προσπάθειας για ένα ρεαλιστικό μοντέλο πλεύσης, ίσως η συνολική μάζα κάποιου σκάφους (lightshipWeight + μεταβλητά φορτία) θα πρέπει να αντανακλά, πέρα από την μέγιστη ταχύτητα με την οποία μπορεί να πλεύσει σε τόξο δεδομένης ακτίνας, την ικανότητά του να αυξομειώνει την ταχύτητά του. Αυτό θα μπορούσε να υλοποιηθεί τοποθετώντας τη συνολική μάζα ως κάποιας μορφής παράμετρο στις συναρτήσεις επιβράδυνσης / επιτάχυνσης.

Άρση περιορισμών διαδρομής:
Επί της υφιστάμενης υλοποίησης, η εφαρμογή δεν επιτρέπει σε γνησίως εξομοιούμενα ταξίδια να έχουν δρομολόγια (αντικείμενα Itinerary) με Legs βραχύτερα του πλάτους του διαδρόμου πλεύσης (ρύθμιση laneWidth), για λόγους που σχετίζονται με τον ομαλό υπολογισμό των τόξων στροφής. Ως συνέπεια του περιορισμού τα δρομολόγια αυτά δεν δύνανται να αποτυπώσουν ελιγμούς υψηλής ακριβείας, όπως θα απαιτούσε η κίνηση σε μικρά ποτάμια, κάποια λιμάνια και ισθμούς.
Σε αντιμετώπιση του περιορισμού, δεν είναι ανάγκη να μεταβληθεί το μοντέλο υπολογισμού των τόξων στροφής όπως παρουσιάστηκε σε προηγούμενο κεφάλαιο. Αντίθετα, το laneWidth θα μπορούσε να μην αποτελεί καθολική ρύθμιση, αλλά μία από τις παραμέτρους πλεύσης (στο parameters.csv). Ακόμη, αφού η ρύθμιση επηρεάζει άμεσα το colliderRadius για τον υπολογισμό στροφών, αντί για σταθερή ποσότητα, η τιμή της μπορεί να μειωθεί όσο χρειαστεί στα άκρα των Legs εκείνων που είναι μικρότερα από το πλάτος διαδρόμου, ώστε οι γεωμετρικοί υπολογισμοί των τόξων εκατέρωθεν να παραμένουν ανεξάρτητοι μεταξύ τους. Με άλλα λόγια το laneWidth δε θα ήταν ένα απόλυτο μέγεθος, απλώς το μέγιστο δυνατό της συγκεκριμένης διαδρομής. Κατόπιν αυτής της αλλαγής, το LegTooShortException μπορεί να καταργηθεί, φυσικά μαζί με τον έλεγχο στον κώδικα που προκαλεί την έγερσή του.

6.1.2: Μελλοντικές επεκτάσεις

Θέματα αλληλεπίδρασης ανθρώπου – υπολογιστή:

- Εισαγωγή δεδομένων με γραφικό περιβάλλον:
  Σε ότι αφορά την εισαγωγή δεδομένων από πλευράς χρήστη πριν την έναρξη της εξομοίωσης, θα μπορούσε να προστεθεί λειτουργικότητα που θα κατευθύνει τις εισαγωγές του χρήστη με χρήση συσκευών κατάδειξης (όπως ποντίκι) σε παράθυρο γραφικών, για νησιά και διαδρομές. Ουσιαστικά θα πρόκειται για ένα μέσο παραγωγής διανυσμάτων από συντεταγμένες στο επίπεδο. Τα διανύσματα θα μπορούσαν να αποθηκευτούν σε αρχεία και μαζί με τα υφιστάμενα αρχεία του σεναρίου εξομοίωσης να οδηγηθούν στην διαδικασία ενσωμάτωσης εισαγωγών.
- Ένδειξη κλίμακας:
  Ένα ευθύγραμμο τμήμα σε κάποια γωνία του χάρτη / παραθύρου γραφικών (τυπικά κάτω δεξιά), συνοδευόμενο από την ένδειξη του μήκους του, σαν να ήταν υπαρκτή απόσταση της προβαλλόμενης περιοχής βοηθά το χρήστη να αντιληφθεί το βαθμό σμίκρυνσης του εικονιζόμενου πεδίου, συγκρίνοντας την ένδειξη αυτή με το εικονιζόμενο μήκος του τμήματος στην οθόνη ως σειρά εικονοστοιχείων.

- Ετικέτες πλοίων:
  Ετικέτες αναγράφουσες του ονόματος κάθε πλοίου (ή το MMSI αυτού) και σε κινούμενες σε εγγύτητα των συμβόλων (τρίγωνα / κύκλους) πλοίων στο χάρτη της εφαρμογής, οπτικοποιούν με μεγαλύτερη σαφήνεια το σενάριο εξομοίωσης. Τα captions αυτά μπορούν να είναι όμοια με τα titles των νησιών, και θα μπορούσαν να έχουν το ίδιο χρώμα που ανατίθεται τυχαία στο σύμβολο και στο trail του εικονιζόμενου πλοίου. Μία εναλλακτική προσέγγιση θα ήταν η χρήση mouse-over tooltips με περισσότερες πληροφορίες.
- Δυνατότητα αλλαγής μεγέθους του παραθύρου γραφικών:
  Επί της παρούσας μορφής του υποσυστήματος γραφικών δεν υπάρχει η δυνατότητα αυτή, καθώς λαμβάνονται οι διαστάσεις της οθόνης του συστήματος, το παράθυρο μεγιστοποιείται και απενεργοποιούνται τα στοιχεία ελέγχου σε αυτό που επιτρέπουν ορισμό διαφορετικού μήκους και πλάτους. Οι τεχνικές αλλαγές που απαιτούνται προς την επέκταση αυτή, με κυριότερες την εύρεση των εσωτερικών διαστάσεων παραθύρου και τον επάν-υπολογισμό της κλίμακας, παρουσιάζονται συνοπτικά στο σχολιασμό του αρχείου πηγαίου κώδικα \sailAway\application\dataTypes\gui\src\Terrain.java. Πρόκειται ως επί το πλείστον για γνώριμες διεργασίες που έχουν ήδη περιγραφεί στην ανάλογη ενότητα, οργανωμένες και συσχετισμένες με διαφορετικό τρόπο.
- Έλεγχος κύλισης και εστίασης:
  Σε επέκταση της προηγούμενης πρότασης, ίσως θα ήταν χρήσιμη η προσθήκη κουμπιών / στοιχείων ελέγχου (controls) που θα επέτρεπαν την αύξηση και μείωση της κλίμακας προβολής (zoom), ώστε ο χρήστης να είναι σε θέση να εστιάσει σε συγκεκριμένα νησιά και τροχιές του σεναρίου. Συμπληρωματικά, θα πρέπει να δοθούν και τα στοιχεία ελέγχου που θα επιτρέψουν την κύλιση στους δύο άξονες (pan). Ασφαλώς τα στοιχεία αυτά θα πρέπει να συνοδεύει και η απαραίτητη λειτουργικότητα στα JPanels.

<u>Θέματα Αναγνώρισης Σύνθετων Γεγονότων</u>:

- Ειδοποιήσεις HLEs επί παραθύρου γραφικών:
  Η παρούσα πρόταση περιλαμβάνει τη χρήση ειδοποιήσεων επί του χάρτη γραφικών της sailAway σε πραγματικό χρόνο για τα γεγονότα υψηλού επιπέδου τη στιγμή που αναγνωρίζονται. Οι ειδοποιήσεις θα πρέπει να είναι συνοπτικές αλλά ουσιαστικές: Το πληροφοριακό περιεχόμενο μίας τέτοιας στιγμιαίας ειδοποίησης σίγουρα δεν μπορεί να είναι τόσο λεπτομερές όσο αυτό στην γραπτή αναφορά που παράγεται, αλλά προσφέρει στο χρήστη χώρο-χρονική σύνδεση με τις οπτικοποιημένες εκπομπές του σεναρίου στο σύνολό του. Ακόμη, στο περιθώριο του παραθύρου (ή σε άλλο JFrame) θα μπορούσαν να εμφανίζονται συγκεντρωτικά δεδομένα, όπως ο συνολικός αριθμός περιστατικών ανά συμβάν υψηλού επιπέδου, τα MMSI με το μεγαλύτερο αριθμό παραβάσεων, κ.α.
  Η εν λόγω επέκταση απαιτεί αλλαγές ως επί το πλείστον στο υποσύστημα γραφικών παρά στην εγκατάσταση και παραμετροποίηση της μηχανής επεξεργασίας γεγονότων της εφαρμογής. Ωστόσο κατηγοριοποιείται στις επεκτάσεις των διεργασιών αναγνώρισης σύνθετων γεγονότων λόγω της αξίας που έχει προς την κατεύθυνση αυτή, προσφέροντας άμεση ειδοποίηση στο χρήστη και σε ζωντανό χρόνο για κάποιο γεγονός που έχει μόλις συμβεί.
- Ανάγνωση πραγματικών εκπομπών AIS:
  Με τη συγγραφή τάξεων σε ρόλο data adapters, την διοχέτευση ρεύματος (ή αναπαραγωγή ιστορικής σειράς) εκπομπών AIS από πραγματική κυκλοφορία πλοίων, η εφαρμογή θα μπορούσε να ανιχνεύσει παρόμοιες καταστάσεις με αυτές που εντοπίζει στην τρέχουσα μορφή της, πέραν υποθετικών σεναρίων. Η πραγματική έκταση των αλλαγών που θα απαιτηθούν, είναι δύσκολο να προβλεφθεί. Για τη συμμόρφωση στον περιορισμό που απαιτεί τη διατήρηση των λειτουργιών που η εφαρμογή επιτελεί στην παρούσα μορφή της, ο διπλός ρόλος που θα πρέπει να αποκτήσει σε μία τέτοια εξέλιξη, μάλλον αυξάνει την έκταση των μετατροπών.

- Εναλλακτικές τεχνικές αναγνώρισης σύνθετων γεγονότων:
  Η πρόταση αυτή, παρόλο που δεν αλλάζει τη σκοπιμότητα και το εύρος χρήσης της εφαρμογής, θα ήταν σε θέση να επιφέρει ριζικές αλλαγές σε αυτή, με τη σημαντικότερη την πλήρη εγκατάλειψη της Esper από τη διαδικασία αναγνώρισης σύνθετων γεγονότων. Ο λόγος είναι πως το σύστημα αυτό δεν είναι σε θέση να χρησιμοποιήσει επαγωγικές ή απαγωγικές μεθόδους για επεξεργασία γεγονότων, αλλά λαμβάνει LLEs (γεγονότα - αίτια) εξωγενώς, όπως συμβαίνει και με τις περιγραφές των γεγονότων ενδιαφέροντος (κανόνες), εκφρασμένες στη γλώσσα EPL. Ακόμη, δε διαθέτει μηχανισμούς δυναμικής αναπροσαρμογής κατά το runtime και δε δύναται να «μάθει» από τα γεγονότα εισόδου. Εναλλακτικές τεχνολογίες αναγνώρισης γεγονότων με δυνατότητες εύρεσης ή μετουσίωσης των κανόνων περιγραφής HLEs περιλαμβάνουν, ενδεικτικά: επαγωγικό λογικό προγραμματισμό, γραφικά μοντέλα αβεβαιότητας, διάφορες τεχνολογίες στη σφαίρα της μηχανικής μάθησης και άλλα, όπως έχει δείξει η επιστημονική κοινότητα στη σχετική βιβλιογραφία.

6.2: Σύνοψη / Συμπεράσματα

Στις σελίδες της εργασίας παρουσιάστηκε μία εφαρμογή εξομοίωσης ναυτιλιακής κίνησης και αναγνώρισης σύνθετων γεγονότων σε αυτή. Επεξηγήθηκε εκτενώς η διαδικασία λήψης δεδομένων εισόδου και οι προσαρτημένοι μηχανισμοί επαλήθευσης και ενσωμάτωσης των πληροφοριών στην εφαρμογή, σε συνδυασμό με τεχνικές ανάδρασης. Αναλύθηκε μία διττή προσέγγιση στην παραγωγή εκπομπών θέσεων ως μηνύματα του συστήματος AIS: με πραγματοποίηση εξομοίωσης και, συμπληρωματικά, με αναπαραγωγή προ-επεξεργασμένων σειρών. Παρουσιάστηκε η διαδικασία οπτικοποίησης και καταγραφής της εξόδου αυτής, αλλά και του συγχρονισμού των εκπομπών.
Με αξιοποίηση του δημοφιλούς συστήματος επεξεργασίας γεγονότων Esper CEP, αναπτύχθηκαν μονάδες αναγνώρισης γεγονότων υψηλού επιπέδου ιδιαίτερης αξίας για τη θαλάσσια πλοήγηση, έκαστη προσανατολισμένη σε κάποιο διαφορετικό είδος. Το κυριότερο κριτήριο επιλογής των ειδών αυτών υπήρξε η ασφάλεια μετακινήσεων. Η εργασία απαριθμεί και περιγράφει έξι είδη, αλλά στην πραγματικότητα ο αριθμός τους προσδιορίζεται από τον τρόπο που ο αναγνώστης αντιλαμβάνεται την αξία κάποιων καταστάσεων σε υδάτινες διαδρομές. Έτσι, εναλλακτικές προσεγγίσεις μπορεί να αγνοηθούν ως περιττές, ενώ αντίθετα «ενδιάμεσου» επιπέδου γεγονότα και προπαρασκευαστικές ενέργειες ίσως θεωρηθούν αρκετά ενδιαφέροντα ως αυτοτελείς οντότητες. Σε κάθε περίπτωση, υπήρξε ανάλυση όλων των βημάτων αναγνώρισης ανά τύπο HLE ενώ επίσης εκφράστηκαν σχόλια για πιθανά σφάλματα αναγνώρισης καθώς και οι ενδεχόμενες συνέπειές τους. Σε γενίκευση επί του συνολικού εγχειρήματος, τοποθετήθηκαν ιδέες για μελλοντικές κατευθύνσεις βελτίωσης και επέκτασης.
Πρόσθετα, εξετάστηκαν συνεισφορές - ορόσημα από την διεθνή επιστημονική κοινότητα στο χώρο της Αναγνώρισης Σύνθετων Γεγονότων στη Ναυσιπλοΐα και η ανάλυση προχώρησε σε κάποιες συγκρίσεις της βιβλιογραφίας με την παρούσα εργασία, σε μία προσπάθεια προσδιορισμού σχετικής εννοιολογικής τοποθέτησης αλλά και διαφοροποιήσεων.

Τα συμπεράσματα στα οποία καταλήγει κάποιος από την χρήση και την ανάπτυξη της εφαρμογής λογισμικού, αλλά και από όσα παρατίθενται στο κείμενο που τη συνοδεύει, εντάσσονται σε δύο κύριους άξονες:

1. Η αναγνώριση σύνθετων γεγονότων μπορεί να αποτελέσει εργαλείο ανεκτίμητο στην προσπάθεια εντοπισμού σημαντικών καταστάσεων μέσα σε ένα κατακλυσμικό όγκο πρωτογενών δεδομένων ιδιαίτερα δε εάν οι ζητούμενες καταστάσεις διέπονται από πολύπλοκες σχέσεις μεταξύ των πρωτογενών συμβάντων. Στην παρούσα εφαρμογή, με αφετηρία ναυτιλιακά μηνύματα, κατέστη προφανές πως πίσω από τον καταιγιστικό ρυθμό απλών, ανιαρών μηνυμάτων, αποκαλύπτονται «κρυμμένα» περιστατικά ουσίας και παρουσιάζονται σε μία ανθρωπίνως κατανοητή, κατηγοριοποιημένη και συνοπτική μορφή. Ασφαλώς, τεχνικές σαν αυτή δεν αποτελούν πανάκεια: Λάθη σχεδιασμού μπορεί να υποβόσκουν. Άλλωστε τα πιο ύπουλα σφάλματα στο λογισμικό είναι αυτά που βρίσκονται σε φαινομενικά καλές και λειτουργικές υλοποιήσεις, περνώντας απαρατήρητα για ένα αόριστο χρονικό διάστημα. Όπως υποστηρίχθηκε στο προηγούμενο κεφάλαιο, θα πρέπει να υπάρχει

κατανόηση των διαδικασιών που διεξάγονται και όπως κάθε πρόγραμμα, θα πρέπει να συντηρείται και εσαεί να βελτιώνεται.
Σε ένα διαφορετικό επίπεδο ανάλυσης, η επιτυχής αναγνώριση ενός γεγονότος αποτελεί μόνο την αρχή μίας εξωσυστημικής, ως επί το πλείστον, διαδικασίας προς αντιμετώπιση της αντίστοιχης κατάστασης. Μικρή αξία έχει η πρόβλεψη μίας επικείμενης σύγκρουσης εάν δεν ειδοποιηθούν χωρίς καθυστέρηση οι γέφυρες στα συγκλίνοντα σκάφη. Παρομοίως, η ανταλλαγή ενός ύποπτου δέματος θα πρέπει να θέτει σε κίνηση την επιβεβλημένη απόπειρα εξακρίβωσης της φύσης του από τις αρμόδιες Αρχές, ύστερα από την πρόσδεση του παραλήπτη στον προορισμό του.

2. Το σύστημα μηνυμάτων AIS έχει εγγενείς δυνάμεις και αδυναμίες ως βάση αναγνώρισης καταστάσεων ενδιαφέροντος, δηλαδή ως τύπος LLE. Στις δυνάμεις του συγκαταλέγεται η πληθώρα των πληροφοριών που μπορεί ακόμη και ένα μόνο μήνυμα να μεταδώσει, η ψηφιακή ενθυλάκωση και δικτυακή μεταφορά του και φυσικά η καθολική αποδοχή του από τον παγκόσμιο στόλο. Οι αδυναμίες του περιορίζονται στην ευκολία με την οποία μπορεί να απενεργοποιηθεί ή ακόμη και να χειραγωγηθεί, μεταδίδοντας αναληθείς τιμές οι οποίες μπορεί να εκτείνονται στην ταυτότητα και θέση ενός πλοίου.
Χάρη στις «αντιθέσεις» αυτές, το σύστημα αναδεικνύεται σε εξαιρετικό εργαλείο αποφυγής ατυχημάτων, όπου δεν υπάρχει πρόθεση βλάβης ή παράβασης από ανθρώπους, αλλά ταυτόχρονα σε εξαιρετικά επισφαλές και αναξιόπιστο μέσο εντοπισμού κακόβουλων, εκούσιων και προμελετημένων ενεργειών.
Προς αντιμετώπιση της αδυναμίας αυτής, και όπως έχει ήδη υποστηριχθεί στην αντίστοιχη ενότητα, μπορούν να χρησιμοποιηθούν παράλληλα εναλλακτικές μέθοδοι εντοπισμού, αλλά μόνο μερικώς μπορούν αυτές να εγγυηθούν την ακρίβεια των αποτελεσμάτων.
Η σύσταση της ανάλυσης είναι η συμπληρωματική χρήση πολλαπλών πηγών γεγονότων χαμηλού επιπέδου διαφοροποιημένης προέλευσης και υπόστασης, ακόμη και η σύνδεση με ανεξάρτητα συστήματα επεξεργασίας γεγονότων.

Εν τέλει,

Οι υδάτινοι διάδρομοι του πλανήτη είναι αδιαμφισβήτητα ένα μέσο επικοινωνίας που έκρινε την πορεία και επιβίωση των σημαντικότερων πολιτισμών που εμφανίστηκαν επί αυτού. Ιστορικά, όλα τα μεγάλα έθνη είχαν εξασφαλίσει διαύλους επικοινωνίας με το πολιτικό και οικονομικό περιβάλλον της εποχής τους, αποκτώντας ξεχωριστή δυναμική στην εμπορική, πολιτιστική αλλά ακόμη και αποικιοκρατική επέκτασή τους. Ακόμη και σήμερα, και παρά την άνθηση των αερομεταφορών, οι πιο εύρωστες πόλεις του κόσμου είναι παράκτιες ή έχουν ανεμπόδιστη πρόσβαση σε κάποιο λιμάνι.

Δεν υπάρχει αμφιβολία πως η ναυτιλία είναι οδός επικοινωνίας που εάν εκμεταλλευτεί καταλλήλως εξελίσσεται σε μοχλό πολύπλευρης ανάπτυξης. Η εκμετάλλευση των θαλασσών όμως δεν είναι ένας καρπός που απλώς περιμένει να κοπεί από το δένδρο του: Είναι δρόμος γεμάτος προκλήσεις και κινδύνους. Η ενοχλητική τάση της ιστορίας να επαναλαμβάνεται δεν έχει εξαιρεθεί από τη ναυτιλία, όπου τραγωδίες και ναυάγια, ακόμη και εν καιρό ειρήνης, αποκαλύπτουν πως η τεχνολογική πρόοδος είναι αναγκαία αλλά όχι επαρκής συνθήκη εξασφάλισης απροβλημάτιστης δραστηριότητας στην ανοικτή θάλασσα.

Οι ναυτικές καταστροφές δεν είναι αποτέλεσμα της οργής κάποιου θεού, όπως πιστεύαμε κάποτε. Σήμερα γνωρίζουμε πως σχεδόν πάντοτε η ευθύνη για αυτές βαραίνει τον άνθρωπο, εκούσια ή όχι. Η διαπίστωση αυτή όμως αντανακλά και την δυνατότητα που του δίνεται να προσπαθεί και να υπερνικά, τελικά, τις προκλήσεις. Σε πνεύμα όμοιο με τους μηχανισμούς που περιγράφηκαν στις σελίδες αυτές, ο άνθρωπος έχει χρέος επαγρύπνησης και διαρκούς προόδου.

Το παρόν φιλοδοξεί να αποτελέσει ένα μικρό βήμα προς αυτή την κατεύθυνση.

> “But man is not made for defeat.,” he said.
> “A man can be destroyed but not defeated.”
>
> — *Ernest Hemingway, The Old Man and the Sea*

Βιβλιογραφία

Run-time Composite Event Recognition (2012)
Alexander Artikis, Marek Sergot, George Paliouras

Chronicle recognition improvement using temporal focusing and hierarchization (2012)
Christophe Dousson, Pierre Le Maigat

The Event Calculus Explained (1999)
Murray Shanahan

A Logic-based Calculus of Events (1986)
Robert Kowalski, Marek Sergot

TRIO: A Logic Language for Executable Specifications (1990)
Ghezzi C., Mandrioli D., Morzenti A.

Markov Logic Networks (2006)
Matthew Richardson, Pedro Domingos

TESLA: A Formally Defined Event Specification Language (2010)
Gianpaolo Cugola, Alessandro Margara

Complex Event Processing with T-REX (2010)
Gianpaolo Cugola, Alessandro Margara

Cayuga: A High-Performance Event Processing Engine (2007)
Lars Brenna, Alan Demers, Johannes Gehrke, Mingsheng Hong, Joel Ossher, Biswanath Panda, Mirek Riedewald, Mohit Thatte, Walker White

DejaVu: A Complex Event Processing System for Pattern Matching over Live and Historical Data Streams (2011)
Nihal Dindar, Peter M. Fischer, Nesime Tatbul

Maritime Situation Analysis Framework (2015)
Hamed Yaghoubi Shahir, Uwe Glasser, Amir Yaghoubi Shahir, Hans When

Complex event processing applied to early maritime threat detection (2012)
Juan Boubeta-Puig, Inmaculada Medina-Bulo, Guadalupe Ortiz, German Fuentes-Landi

A complex event processing approach to detect abnormal behaviours in the marine environment (2015)
Fernando Terroso-Saenz, Mercedes Valdes-Vela, Antonio F. Skarmeta-Gomez

Event Recognition for Maritime Surveillance (2015)
Kostas Patroumpas , Alexander Artikis, Nikos Katzouris, Marios Vodas, Yannis Theodoridis, Nikos Pelekis

How not to drown in a sea of information: An event recognition approach (2015)
Elias Alevizos, Alexander Artikis, Kostas Patroumpas, Marios Vodas, Yannis Theodoridis, Nikos Pelekis

Anomaly detection for sea surveillance (2008)
Rikard Laxhammar

Vessel pattern knowledge discovery from AIS data: a framework for anomaly detection and route prediction (2014)
Giuliana Pallota, Michele Vespe, Karna Bryan

On the Representation and Exploitation of Context Knowledge in a Harbor Surveillance Scenario (2011)
J. Garcia, J. Gomez-Romero, M. A. Patricio, J. M. Molina, Gala Rogova

An AIS data Visualization Model for Assessing Maritime Traffic Situation and its Applications (2012)
Pan Jiacai, Jiang Qingshan, Hu Jinxing, Shao Zheping

Open data for anomaly detection in maritime surveillance (2013)
Samira Kazemi, Shahrooz Abghari, Niklas Lavesson, Henric Johnson, Peter Ryman

Temporal abstraction and inductive logic programming for arrhythmia recognition from electrocardiograms (2003)
G. Carrault, M.-O. Cordier, R. Quiniou, F. Wang

Intelligent Highway Traffic Surveillance With Self-Diagnosis Abilities (2011)
Hsu-Yung Cheng, Shih-Han Hsu

Behaviour Recognition from video content: A logic programming approach (2010)
Alexander Artikis, Anastasios Skarlatidis, George Paliouras

Intelligent M2M: Complex event processing for machine-to-machine communication (2015)
Ralf Bruns, Jurgen Dunkel, Henrik Masbruch, Sebastian Stipkovic

Boat traffic characteristics in three passages of the Aegean Sea: evidence, risk of maritime accidents, strategy for protection (2010)
Anastasia Miliou, Monica Demetriou, Charlene Caprio, Thodoris Tsimpidis, Stephanie Barnicoat

On the Generation of Spatiotemporal Datasets (1999)
Y. Theodoridis, J.R.O. Silva, M.A. Nascimento

A Framework for Generating Network-Based Moving Objects (2002)
Thomas Brinkhoff

Anomaly Detection and Knowledge Discovery using Vessel Tracking Data (2016)
A. Alessandrini, M. Alvarez, H. Greidanus, V. Gammieri, V. Fernandez Arguedas, F. Mazzarella, C. Santamaria, M. Stasolla, D. Tarchi, M. Vespe

Methodology for Real-Time Detection of AIS Falsification (2016)
C. Ray, C. Iphar, A. Napoli

Detection of malicious AIS position spoofing by exploiting radar information (2013)
Fotios Katsileris, Paolo Braca, Stefano Coraluppi

Object-Oriented Sensor Data Fusion for Wide Maritime Surveillance (2010)
Fischer Yvonne, Bauer Alexander

Παράρτημα Α: Τα μέλη των τάξεων της εφαρμογής

Σε επέκταση της επεξήγησης της λειτουργικότητας του προγράμματος, όπου παρουσιάστηκε το περιεχόμενο των υποσυστημάτων και οι τύποι δεδομένων σε αυτά και η λειτουργικότητά τους, το παρόν παράρτημα προχωρά σε μία ειδικότερη και βαθύτερη ανάλυση, εστιασμένη στο εσωτερικό των τύπων. Για την ακρίβεια, παρατίθενται ιστοσελίδες τεκμηρίωσης ως προϊόντα του εργαλείου Javadoc.

Το Javadoc είναι ένα πρόγραμμα παραγωγής τεκμηρίωσης Java API, δηλαδή περιγραφής του τρόπου χρήσης αρχείων .class και των μελών κάθε τύπου που αυτά αντιπροσωπεύουν. Καλείται από τη γραμμή εντολών[115] και δέχεται ως όρισμα αρχεία πηγαίου κώδικα Java, τα οποία προαιρετικά περιέχουν μεταδεδομένα[116] και λοιπές παρατηρήσεις, με δομημένη μορφή εντός ειδικών μπλοκ σχολιασμού. Τα μεταδεδομένα αυτά αποτελούν μέσο πληρέστερης περιγραφής τάξεων και μελών αυτών. Ύστερα από ανάγνωση του κώδικα, το πρόγραμμα παράγει τα σχετικά αρχεία τεκμηρίωσης των αρχείων .class που αντιστοιχούν στο αποτέλεσμα μεταγλώττισης των δοθέντων αρχείων .java. Τα παραγόμενα έχουν μορφή .html, είναι δηλαδή σελίδες ιστού που μπορούν να προσπελαστούν από κοινούς φυλλομετρητές. Το πλεονέκτημα δημοσίευσης σε αυτή τη μορφή είναι η δυνατότητα διασύνδεσης μεταξύ εγγράφων τύπων κληρονομικής ή αναφορικής συνάφειας. Κατ' ελάχιστο, το πρόγραμμα εκδίδει για κάθε τάξη μία λίστα των μελών δημοσίου πρόσβασης ως signatures[117], ενώ με κατάλληλα options κατά την κλήση του, η τεκμηρίωση εμπλουτίζεται ποικιλοτρόπως. Τα options καθορίζουν παραμέτρους όπως το εύρος των μελών που θα συμπεριληφθούν στην τεκμηρίωση. Επιπλέον, τα μεταδεδομένα εφ' όσον υπάρχουν στα αρχεία εισαγωγής, αποτυπώνονται στις ιστοσελίδες και τοποθετούνται σε αυτές αναλόγως είδους της πληροφορίας που μεταφέρουν.

Στα αρχεία του παραρτήματος περιέχονται όλα τα μέλη, ακόμη και αυτά με ιδιωτική πρόσβαση. Έχει γίνει κλήση του Javadoc μία φορά ανά υποσύστημα και τα παραγόμενα αρχεία έχουν τοποθετηθεί σε δομή που αντίστοιχη με αυτή των υποφακέλων του sailAway\application:

```
AppendixA
├───dataTypes
│   ├───ais
│   ├───configuration
│   ├───exceptions
│   ├───gui
│   ├───lowLevelTypes
│   ├───parsers
│   └───userInput
├───listeners
└───main
```

Για μία συνολική εικόνα κάποιου υποσυστήματος, προτείνεται η φόρτωση της σελίδας index.html του φακέλου του και στη συνέχεια, χρήση των υπερσυνδέσεων σε αυτή για προβολή επιμέρους τύπων ή μεθόδων.

Σχετικά με τη λειτουργία της εφαρμογής σε χαμηλότερο επίπεδο και μία λεπτομερή ανάλυση των αλγορίθμων της, ο αναγνώστης παροτρύνεται να μελετήσει επιλεκτικά τα αρχεία πηγαίου κώδικα, στα οποία υπάρχει περιφραστικός και αναλυτικός σχολιασμός[118] σε σημεία που έχουν αυξημένο βαθμό

[115] Πρόγραμμα javadoc.exe, παρέχεται από το Java SDK στον ίδιο κατάλογο με τα java/javac.

[116] Ως key – value ζεύγη. Τα keys στην περίπτωση του Javadoc ονομάζονται απλώς tags. Περισσότερα: https://docs.oracle.com/javase/1.5.0/docs/tooldocs/solaris/javadoc.html

[117] Signature μίας συνάρτησης είναι το πλήρες όνομά της, δηλαδή αυτό που πέρα από το αναγνωριστικό της περιλαμβάνει όλα τα modifiers που έχει δηλώσει κατά τον ορισμό της (πρόσβαση, μέλος τάξης / αντικειμένου, επεκτασιμότητα και άλλοι), τη λίστα των παραμέτρων με τους τύπους τους, αλλά και τον τύπο που επιστρέφει. Κατά αναλογία με τις μεθόδους, τα πεδία μίας τάξης στο Javadoc documentation εμφανίζονται με το όνομά τους και τους modifiers τους.

[118] Τα σχόλια είναι γραμμένα στα αγγλικά, καθώς υπήρξαν compiler errors λόγω ελληνικών χαρακτήρων, παρόλο που αυτοί βρίσκονται σε μπλοκ (/* … */) ή inline (// …)σχόλια. Για αισθητικούς λόγους δεν προτιμήθηκε η συγγραφή ελληνικής σε λατινικό αλφάβητο (greeklish).

πολυπλοκότητας ή επιτελούν λειτουργίες πέραν των τετριμμένων για προγράμματα Java ή αντικειμενοστραφές λογισμικό γενικότερα.

Παράρτημα Β: Η βοηθητική εφαρμογή ScaleAndOffset

Στα πλαίσια ανάπτυξης της εφαρμογής που παρουσιάστηκε, δημιουργήθηκε το πρόγραμμα που αναλύεται στο παράρτημα αυτό, με σκοπό να διευκολύνει τη χρήση της sailAway σε ότι αφορά την παροχή δεδομένων εισόδου για αυτήν, προετοιμάζοντας καταλλήλως ορισμένους τύπους αρχείων.

Η βοηθητική εφαρμογή είναι χρήσιμη σε περιπτώσεις όπου ο χρήστης επιθυμεί να προσθέσει στο σενάριο εξομοίωσης περισσότερα νησιά, που όμως βρίσκονται σε διαφορετική κλίμακα από τα υφιστάμενα, δεν έχουν το ίδιο σημείο αναφοράς ως αρχή των αξόνων στο χάρτη, ή πάσχουν από ένα συνδυασμό αυτών των καταστάσεων. Παρομοίως, η εφαρμογή χρησιμοποιείται για να καταστήσει δυνατή τη χρήση δρομολογίων πλοίων (δηλαδή σειράς waypoints), που παρουσιάζουν τις ίδιες ασυμβατότητες σε σχέση με τις στεριές ανάμεσα από τις οποίες κινούνται, ή με άλλα υπάρχοντα δρομολόγια.

Η ScaleAndOffset λειτουργεί με τα αρχεία εισόδου .csv εκείνα που έκαστο περιέχουν ένα διάνυσμα διατεταγμένων ζευγών, αποτελούμενων από τιμές γεωγραφικού μήκους και γεωγραφικού πλάτους, δηλαδή συντεταγμένων. Πρόκειται λοιπόν για τα αρχεία πολλαπλών γραμμών (καταχωρήσεων) και δύο «στηλών» τα οποία είναι τα waypoints.csv και .csv νησιών. Η εφαρμογή δύναται να επεξεργαστεί επίσης και τα broadcasts.csv, με κάποιους περιορισμούς. Συγκεκριμένα για αυτά τα αρχεία γίνεται ειδική αναφορά στο τέλος του παραρτήματος.

Πριν την εκτέλεση του προγράμματος, εντός του φακέλου \input\ τοποθετούνται τα αρχεία που ο χρήστης επιθυμεί να μετατρέψει. Όμως τα περιεχόμενα του φακέλου αυτού δε μεταβάλλονται με κανένα τρόπο. Αντίθετα, στον φάκελο \output\ δημιουργούνται κατ' αντιστοιχία ομώνυμα αρχεία που έχουν υποστεί την επιθυμητή μετατροπή.

Η εφαρμογή μπορεί να ξεκινήσει με την κατάλληλη κλήση από τη γραμμή εντολών, java ScaleAndOffset από το directory όπου περιέχεται και το μοναδικό .class αρχείο της, δηλαδή από το \sailAway\helperApps\ScaleAndOffset\. Σε αντίθεση με την κυρίως εφαρμογή sailAway, η παρούσα δεν συνοδεύεται από βοηθητικό batch script εκκίνησης, καθώς δεν εξαρτάται από εξωγενείς τύπους δεδομένων οι οποίοι θα έπρεπε υπό άλλες συνθήκες να δηλωθούν στο -classpath (ή –cp) option.

Η εφαρμογή αρχικά εμφανίζει τον τίτλο και την περιγραφή της στην κονσόλα, ενώ δίνει την επιλογή στο χρήστη να εμφανίσει περισσότερες πληροφορίες για τη χρήση, τη λειτουργία και τη χρησιμότητά της. Στη συνέχεια, περνά στο στάδιο όπου συγκεντρώνει τις απαραίτητες παραμέτρους μετατροπής των συντεταγμένων εισόδου. Για αυτό το σκοπό ακολουθούν διάλογοι από τους οποίους συγκεντρώνονται με την παρακάτω σειρά οι μεταβλητές:

1. xScale
   Δεκαδικός διπλής ακριβείας, θετικών τιμών αποκλειστικά. Αντιπροσωπεύει τον συντελεστή με τον οποίο τα γεωγραφικά μήκη (longitudes, x-values) πολλαπλασιάζονται. Τιμές μικρότερες του 1 προκαλούν συρρίκνωση, ενώ σε αντίθετη περίπτωση, μεγέθυνση.
2. yScale
   Δεκαδικός διπλής ακριβείας, θετικών τιμών αποκλειστικά. Αντιπροσωπεύει τον συντελεστή με τον οποίο τα γεωγραφικά πλάτη (latitudes, y-values) πολλαπλασιάζονται. Τιμές μικρότερες του 1 προκαλούν συρρίκνωση, ενώ σε αντίθετη περίπτωση, μεγέθυνση. Κατά την εισαγωγή αυτής της παραμέτρου, η εφαρμογή έμμεσα παρακινεί το χρήστη να εισάγει την ίδια τιμή που εισήγαγε και για τα γεωγραφικά μήκη, υπενθυμίζοντάς την. Σε περίπτωση που ο χρήστης εισάγει κάτι διαφορετικό, τότε ο λόγος πλευρών του περιγεγραμμένου ορθογωνίου των νησιών (ή διαδρομών πλοίων), αλλάζει και παραμορφώνονται, αφού «τεντώνονται» άνισα ως προς κάποιο άξονα. Από την ύπαρξη της μεταβλητής αυτής φαίνεται πως η δυνατότητα αυτής της παραμόρφωσης δόθηκε εσκεμμένα στο χρήστη.
3. xOffset
   Δεκαδικός διπλής ακριβείας που προστίθεται στα γεωγραφικά μήκη, μετατοπίζοντας κατά μήκος του ισημερινού τις δοθείσες συντεταγμένες. Αρνητικές τιμές μετακινούν τα σημεία δυτικά, θετικές τιμές ανατολικά.
4. yOffset
   Δεκαδικός διπλής ακριβείας που προστίθεται στα γεωγραφικά πλάτη, μετατοπίζοντας κατά

μήκος των μεσημβρινών τις δοθείσες συντεταγμένες. Αρνητικές τιμές μετακινούν τα σημεία νότια, θετικές τιμές βόρεια.

5. omitFirstLine
   Μπουλιανή μεταβλητή (flag) που όταν δοθεί ως αληθής, υποδεικνύει στην εφαρμογή πως η πρώτη γραμμή σε κάθε αρχείο εισόδου αποτελεί γραμμή επικεφαλίδων των στηλών, και ως εκ τούτου θα πρέπει να αγνοηθεί. Σε αντίθετη περίπτωση, η επεξεργασία ξεκινά αμέσως από την πρώτη γραμμή.

Με την ολοκλήρωση της εισαγωγής των μεταβλητών αυτών, η εφαρμογή πολλαπλασιάζει και προσθέτει αναλόγως τις μεταβλητές αυτές στις κατάλληλες συνιστώσες των συντεταγμένων όλων των top-level αρχείων του φακέλου εισόδου, γράφοντας γραμμή προς γραμμή το αποτέλεσμα σε αρχεία εξόδου.

Όπως αναφέρεται και στις οδηγίες που κατά το runtime διατίθενται, η (από)κλιμάκωση οδηγεί αναπόφευκτα σε ένα ψευδό-offset, καθώς άλλωστε επιτυγχάνεται με πολλαπλασιασμό. Αυτή η επίδραση εξαρτάται ασφαλώς από το μέγεθος του συντελεστή, αλλά και από την απόσταση του εκάστοτε προς μεταβολή σημείου από την αρχή των αξόνων. Η αρχή των αξόνων είναι το μόνο σημείο που παραμένει ακλόνητο σε κάθε scaling.

Η ScaleAndOffset είναι θωρακισμένη από λανθασμένες εισόδους του χρήστη αναφορικά με τις πληροφορίες που απαιτούνται για τη λειτουργία της ίδιας, και οι οποίες παρέχονται στο runtime από τους διαλόγους γραμμής εντολών που αναφέρθηκαν παραπάνω. Σε περιπτώσεις εσφαλμένης εισαγωγής, όπως για παράδειγμα όταν εισάγονται τιμές εκτός εύρους ή αλφαριθμητικά εκεί όπου απαιτούνται αριθμητικές τιμές, παρέχονται μηνύματα σφάλματος, ενώ με χρήση επαναληπτικών δομών στον πηγαίο κώδικα ο χρήστης καλείται να εισάγει πάλι τη ζητούμενη πληροφορία.

Ωστόσο, η βοηθητική αυτή εφαρμογή, σε αντίθεση με την κύρια, δεν έχει τη δυνατότητα διεξαγωγής των πολυεπίπεδων έλεγχων εισόδου χρήστη επί του ίδιου του περιεχομένου των αρχείων (δηλαδή γεωγραφικών συντεταγμένων) που καλείται να μετατρέψει. Για παράδειγμα, σε ένα αρχείο που περιγράφει ένα νησί, η εφαρμογή δεν δύναται να διακρίνει εάν αυτό περιγράφεται από μόλις δύο σημεία, απεικονίζεται από μία γραμμή που τέμνει τον εαυτό της, ή συμπίπτει με κάποια άλλη στεριά. Η ScaleAndOffset πρακτικά κάνει ότι ακριβώς περιγράφει και ο τίτλος της, με την ορθότητα των αρχείων να επαφίεται στην εφαρμογή sailAway και στο χρήστη. Παρόλα αυτά, ασφαλώς υπάρχει η απαίτηση για αριθμητικές τιμές στις συντεταγμένες, αφού άλλωστε επί αυτών εκτελούνται αριθμητικές πράξεις. Σε περίπτωση μη ικανοποίησης αυτού του περιορισμού, ο χρήστης έρχεται αντιμέτωπος με το αντίστοιχο exception handle στην έξοδο της γραμμής εντολών και η μετατροπή του συγκεκριμένου αρχείου αποτυγχάνει.

Όπως ισχύει για την sailAway, η ScaleAndOffset έχει κάποιο περιθώριο βελτίωσης και έχουν εντοπιστεί μερικά σημεία που θα μπορούσαν να ληφθούν υπ’ όψη σε μεταγενέστερη έκδοση:

- Οι συντελεστές κλιμάκωσης (xScale, yScale) θα μπορούσαν να δέχονται και αρνητικές τιμές επιτρέποντας στο χρήστη, μεταξύ άλλων, “mirroring” δηλαδή συμμετρική προβολή των γραμμών εισόδου ως προς τους άξονες (μεσημβρινούς για αρνητικές xScale, ισημερινούς για αρνητικές yScale). Με χρήση αρνητικών συντελεστών εκατέρωθεν, καθίσταται δυνατή η και παραγωγή συμμετρικού, ως προς σημείο, σχήματος.
- Δεν υπάρχουν επί του παρόντος command prompts περί ολοκλήρωσης των εργασιών της εφαρμογής, ή προειδοποιητικό μήνυμα έλλειψης αρχείων εισόδου.
- Για την αντιμετώπιση του ψευδο-offset κατά την αλλαγή κλίμακας που αναφέρθηκε νωρίτερα, θα μπορούσε κατά την συλλογή δεδομένων από το χρήστη να ζητείται και ακόμα μία boolean flag, η οποία όταν δοθεί ως αληθής, πριν το λογισμικό πολλαπλασιάσει με τους δοθέντες συντελεστές (xScale, yScale) τις συντεταγμένες, θα μεταφέρει κάθε νησί / διαδρομή με τέτοιο τρόπο ώστε το κέντρο του περιγεγραμμένου ορθογωνίου του να συμπίπτει με την αρχή των αξόνων. Κατόπιν πολλαπλασιασμού θα μεταφέρει κάθε σχήμα σε θέση ώστε το κέντρο του περιγεγραμμένου ορθογωνίου του να βρεθεί πάλι στις αρχικές του συντεταγμένες. Έτσι αντιμετωπίζεται το πρόβλημα της ανεπιθύμητης ενέργειας της «μετακίνησης», αν και αυξάνεται ο κίνδυνος αλληλοεπικάλυψης (“overlap”) των σχημάτων, τα οποία όμως θα περάσουν εκ νέου από τους σχετικούς ελέγχους της sailAway.

- Εισαγωγή αρχείων: το πρόγραμμα διαβάζει μόνο τα αρχεία εντός του γονικού φακέλου \input\ και αγνοεί ότι τοποθετείται σε υποφακέλους του.
- Το πρόγραμμα αγνοεί, για κάθε γραμμή των αρχείων εισαγωγής, χαρακτήρες μετά το δεύτερο κόμμα («,»), δηλαδή τις στήλες πέραν της δεύτερης, χωρίς φυσικά να επεξεργάζεται με κάποιο τρόπο τα δεδομένα αυτά, αλλά δυστυχώς και χωρίς να τα μεταφέρει ως έχουν μεταφέρει στην έξοδο. Αυτός είναι και ό λόγος για τον οποίο η χρήση του με τα αρχεία broadcasts.csv είναι στην καλύτερη περίπτωση προβληματική, καθώς αυτά φέρουν απαραίτητες πληροφορίες και πέραν της $2^{ης}$ στήλης (στήλες heading και speed). Η επόμενη έκδοση θα πρέπει να αντιγράφει αυτούσιες τις περίσσειες στήλες στα αρχεία εξόδου που παράγει. Σε τέτοια περίπτωση εγείρεται βέβαια και το ερώτημα της σκοπιμότητας της προσαρμογής της στήλης της ταχύτητας, αναλόγως της κλιμάκωσης ενός τέτοιου δρομολογίου…

Παράρτημα Γ: Η άντληση και προσαρμογή δεδομένων στο dataset του στόλου

Κατά τα γνωστά και όπως έχει διατυπωθεί στο Κεφάλαιο 3, τα πλοία ως μέρη διαφόρων σεναρίων εξομοίωσης που συνοδεύουν την εφαρμογή, αποτελούνται από πληροφορίες συγκεντρωμένες σε αρχεία εισόδου ονοματισμένα "vessel.csv". Σε περιπτώσεις σεναρίων όπου συμμετέχουν γνησίως εξομοιούμενα ταξίδια, τα αρχεία των ταξιδιών αυτών πέρα από το αρχείο διαδρομής και πλοίου, συνοδεύονται και από ένα αρχείο παραμέτρων ταξιδιού ("parameters.csv") στο οποίο η κυριότερη πληροφόρηση είναι τα συνήθη φορτία που συνθέτουν τη συνολική μάζα του σκάφους.

Αρχεία εισόδου σχετικά με πλοία (vessel.csv, parameters.csv) βρίσκονται στους φακέλους:

- \sailAway\inputFiles\trips\:
  Υποφάκελος του ενεργού σεναρίου εξομοίωσης, \sailAway\inputFiles\. Πρόκειται για το γονικό φάκελο αρχείων εισαγωγής που αφορούν αποκλειστικά ταξίδια και από όπου η εφαρμογή αναγιγνώσκει τις ανάλογες πληροφορίες.
- \sailAway\other\scenarios\:
  Πρόκειται για κατάλογο αποθήκευσης σεναρίων πέραν του τρέχοντος (ενεργού). Η δομή αρχείων υπό το φάκελο αυτό κατηγοριοποιεί τα σενάρια, ενώ στις περισσότερες περιπτώσεις αυτά είναι πλήρη, ως στιγμιότυπα λειτουργικού συνόλου αρχείων του αντίστοιχου σεναρίου, υπό τον \inputFiles\.
- \sailAway\other\datasets\fleet\:
  Πρόκειται απλώς για συλλογή όλων των διαθέσιμων στην εφαρμογή αρχείων περιγραφής πλοίων μαζί με τις τυπικές παραμέτρους τους, για χρήση σε νέα σενάρια εξομοίωσης. Αξίζει να σημειωθεί πως για τον ίδιο σκοπό, συλλογές συμπλεγμάτων νησιών βρίσκονται στον «αδελφό» φάκελο \sailAway\other\datasets\archipelagoes\.

Στις παραπάνω τοποθεσίες και ανεξαρτήτως του συνολικού αριθμού διαφορετικών σεναρίων εξομοίωσης / όγκου δεδομένων, η sailAway περιλαμβάνει 14 υπαρκτά, γνωστά και εν ενεργεία[119] πλοία βάσει πληροφοριών που έχουν αντληθεί από έγκυρες διαδικτυακές πηγές. Οι απαραίτητες διαδικασίες, από τη συλλογή μέχρι την αξιοποίηση των δεδομένων, έχουν επιμεριστεί στις καρτέλες του αρχείου vessels.xlsx του παραρτήματος ως εξής:

- καρτέλα "dataSurvey":
  Περιέχει μόνο τα διαθέσιμα στις πηγές από όπου αντλήθηκαν δεδομένα, και στη μορφή που βρίσκονται σε αυτές. Οι εν λόγω πηγές προέλευσης αναφέρονται στην καρτέλα, ανά σκάφος. Τα ζητούμενα μεγέθη για χρήση στα πλαίσια της sailAway δεν βρέθηκαν στο σύνολό τους για κάθε πλοίο, ενώ οι μονάδες μέτρησης δεν συμπίπτουν πάντοτε με αυτές που χρησιμοποιεί η εφαρμογή. Σε κάθε περίπτωση, αναγράφονται μόνο τα διαθέσιμα, πρωτογενή από τις πηγές, στοιχεία.
- καρτέλα "dataAdaptation":
  Διεξάγει τις αναγκαίες μετατροπές, οι οποίες περιγράφονται εκεί, συνοπτικά. Οι προσαρμογές περιλαμβάνουν, ενδεικτικά, αλλαγές ονομάτων μεταβλητών, μετατροπές στις μονάδες της sailAway, στρογγυλοποίηση, κ.α. Όμως το σημαντικότερο έργο των υπολογισμών τις καρτέλας είναι η κατά προσέγγιση εκτίμηση και παραγωγή των τιμών που δεν βρέθηκαν στις πηγές, αλλά απαιτούνται από την εφαρμογή. Κύριο εργαλείο αυτού του εγχειρήματος είναι ένα σύνολο υποθέσεων και παραδοχών οι οποίες αναγράφονται και επεξηγούνται συνοπτικά. Σε γενικές γραμμές μπορεί να υποστηριχθεί πως σε πλοία ομοίου τύπου υπάρχουν σχετικά σταθερες αναλογίες μεταξύ κάποιων μεγεθών. Σε άλλες περιπτώσεις, τα ευρήματα βάσει των παραδοχών προσαρμόζονται για πλοία διαφορετικού μεγέθους, κατόπιν εφαρμογής αναλογικών διορθώσεων.
- καρτέλα "sailAwayFleet":
  Περιέχει τη σύνοψη των τεχνικών χαρακτηριστικών των σκαφών και των τυπικών παραμέτρων των ταξιδιών τους κατόπιν μετατροπών, όπως χρησιμοποιούνται στην sailAway.

[119] Κατά τη στιγμή συγγραφής του παρόντος.