ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Πολυπρακτορικός Σχεδιασμός Δράσης

Διπλωματική εργασία

του

Θεόδωρου Στ. Τσότσου

Επιβλέποντες Καθηγητές: Δασκαλοπούλου Ασπασία
Επίκουρος Καθηγήτρια

Αντωνόπουλος Χρήστος
Επίκουρος Καθηγητής

Βόλος, Σεπτέμβριος 2012

Στη μνήμη της γιαγιάς μου
και στην οικογένειά μου

Ευχαριστίες

Με την περάτωση της παρούσας διπλωματικής εργασίας θα ήθελα να ευχαριστήσω θερμά την κύρια επιβλέπουσα καθηγήτρια της εργασίας αυτής, κα. Ασπασία Δασκαλοπούλου, για την πολύτιμη βοήθειά της και την καθοδήγησή της σε όλη την πορεία της εργασίας. Επίσης θα ήθελα να ευχαριστήσω και τον δεύτερο επιβλέποντα καθηγητή κ. Χρήστο Αντωνόπουλο για τις χρήσιμες παρατηρήσεις του, στην προσπάθεια ολοκλήρωσης της εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω όλους μου τους φίλους, σε Πτολεμαΐδα και Βόλο, για την υποστήριξή τους, κυρίως ηθική και ψυχολογική, στα εύκολα και τα δύσκολα, και για όλες τις ωραίες στιγμές που περάσαμε μαζί όλα αυτά τα χρόνια.

Τέλος, ένα μεγάλο ευχαριστώ, από βάθους καρδίας, ανήκει στους γονείς μου και την οικογένεια, χωρίς τη στήριξη και την καθοδήγηση των οποίων σε όλη μου τη ζωή, δε θα μπορούσε να επιτευχθεί τίποτα.

Θεόδωρος Τσότσος

Βόλος, 2012

Περιεχόμενα

Κατάλογος Πινάκων iv

Κατάλογος Εικόνων v

Περίληψη vi

1 Εισαγωγή 1

1.1 Αντικείμενο και σκοπός της διπλωματικής 1

1.2 Διάθρωση της διπλωματικής εργασίας 2

2 Πολυπρακτορικός Σχεδιασμός 3

2.1 Βασικές έννοιες και ορισμοί 3

2.2 Κοινή διαχείριση εργασιών 6

2.3 Κοινή διαχείριση αποτελεσμάτων 8

2.4 Κοινός σχεδιασμός δράσης 9

2.4.1 Συγκεντρωτικός σχεδιασμός για κατανεμημένα πλάνα 9

2.4.2 Κατανεμημένος σχεδιασμός 11

2.4.3 Κατανεμημένος σχεδιασμός για κατανεμημένα πλάνα 11

2.5 Συγχώνευση πλάνων 12

2.6 Γλώσσες επικοινωνίας 14

2.6.1 KQML 14

2.6.2 FIPA ACL 17

3 Σχεδιασμός της Υλοποίησης 19

3.1 Περιγραφή του προβλήματος 19

3.2 Κατανομή ρεπερτορίου ενεργειών στους πράκτορες 20

Περιεχόμενα

3.2.1 Πράκτορες με διαφορετικό ρεπερτόριο ενεργειών και κοινή γνώση χάρτη 20

3.2.2 Πράκτορες με κοινό ρεπερτόριο ενεργειών, κοινό πρόβλημα, μη κοινή γνώση χάρτη 21

3.3 Αναπαράσταση Βάσης Γνώσης 22

3.3.1 Έμμεση δήλωση Βάσης Γνώσης 22

3.3.2 Άμεση δήλωση Βάσης Γνώσης μέσω κλάσεων 23

3.3.3 Άμεση δήλωση Βάσης Γνώσης μέσω Βάσης Δεδομένων 23

3.3.4 Άμεση δήλωση Βάσης Γνώσης μέσω αρχείου 23

3.4 Εισαγωγή στην πλατφόρμα JADE 24

3.4.1 Ειδικοί πράκτορες 27

3.4.2 Εργαλεία γραφικού περιβάλλοντος 29

4 Υλοποίηση της εφαρμογής 31

4.1 Περιγραφή της υλοποίησης 31

4.2 Δομή της εφαρμογής 33

4.2.1 Αρχεία της εφαρμογής 33

4.2.2 UML διαγράμματα της εφαρμογής 34

4.3 Περιγραφή ροής εκτέλεσης 36

5 Επίλογος 39

5.1 Συμπεράσματα 39

5.2 Μελλοντικές επεκτάσεις 40

Βιβλιογραφία 41

Κατάλογος Πινάκων

2.1 Τελεστικά της KQML 16
2.2 Τελεστικά ανά κατηγορία στη FIPA ACL 18

Κατάλογος Εικόνων

2.1 Ένας πράκτορας στο περιβάλλον του. 3
2.2 Τυπική δομή ενός πολυπρακτορικού συστήματος. 4
2.3 Παράδειγμα για το πρόβλημα των Πύργων του Ανόι 7
2.4 Επίπεδα πρωτοκόλλου KQML 15
2.5 Παράδειγμα μηνύματος KQML 15
2.6 Παράδειγμα μηνύματος FIPA ACL 17
3.1 Παράδειγμα έμμεσης δήλωσης ΒΓ 22
3.2 Κατανομή πρακτόρων στο δίκτυο 25
3.3 Επικοινωνία πρακτόρων στο JADE 25
3.4 Διάγραμμα εκτέλεσης νήματος ενός πράκτορα 26
3.5 Στιγμιότυπο RMA GUI 27
3.6 Στιγμιότυπο DF GUI 28
3.7 Στιγμιότυπο Dummy Agent 29
3.8 Στιγμιότυπο Sniffer Agent 30
3.9 Στιγμιότυπο Introspector Agent 30
4.1 Διάγραμμα κλάσεων του πράκτορα συντονιστή 35
4.2 Διάγραμμα κλάσεων των πρακτόρων χειριστών 35
4.3 Διάγραμμα κλάσεων όλης της εφαρμογής 36
4.4 Παράθυρο αλληλεπίδρασης με το χρήστη 37
4.5 Αναζήτηση πρακτόρων από το συντονιστή 37
4.6 Ανταλλαγή μηνυμάτων ανάμεσα στους πράκτορες 38
4.7 Εμφάνιση προτεινόμενης διαδρομής 38

Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται ένα από τα σημαντικότερα ζητήματα των Πολυπρακτορικών Συστημάτων και της Τεχνητής Νοημοσύνης, το ζήτημα του Πολυπρακτορικού Σχεδιασμού Δράσης. Το συγκεκριμένο ζήτημα συνίσταται στον σχεδιασμό των δραστηριοτήτων μιας ομάδας πρακτόρων, οι οποίοι δοθέντος ενός συγκεκριμένου προβλήματος προσπαθούν από κοινού να καταλήξουν στο στόχο που είναι η επίλυση του προβλήματος.

Βασικά θέματα που εξετάζονται στον πολυπρακτορικό σχεδιασμό είναι ο καταμερισμός των εργασιών ανάμεσα στους πράκτορες που συμμετέχουν, η εξαγωγή των επιμέρους πλάνων, η επικοινωνία των πρακτόρων και ο συντονισμός των ενεργειών τους, καθώς και η σύνθεση των επιμέρους πλάνων με σκοπό την εξαγωγή του τελικού πλάνου για την επίτευξη του στόχου του προβλήματος. Τα παραπάνω αντικείμενα μελετώνται στην συγκεκριμένη εργασία τόσο θεωρητικά, όσο και μέσω μιας εφαρμογής που υλοποιήθηκε στα πλαίσια της εργασίας.

Η εφαρμογή μοντελοποιεί τα ανωτέρω ζητήματα και πραγματεύεται το πρόβλημα εύρεσης τυχαίας διαδρομής ανάμεσα σε δυο πόλεις μέσω συγκεκριμένων μέσων μεταφοράς. Είναι υλοποιημένη σε JAVA, για την πλατφόρμα υλοποίησης και επικοινωνίας πολυπρακτορικών συστημάτων JADE και ακολουθεί όλες τις προδιαγραφές FIPA.

Λέξεις κλειδιά:

πράκτορας, πολυπρακτορικά συστήματα, πολυπρακτορικός σχεδιασμός, επικοινωνία, JADE

Κεφάλαιο 1

Εισαγωγή

1.1 Αντικείμενο και σκοπός της διπλωματικής

Μέχρι πρόσφατα τα ευφυή υπολογιστικά συστήματα χρησιμοποιούνταν μόνο σε απλούς τομείς των βιομηχανικών και εμπορικών εφαρμογών, συνήθως αντικαθιστώντας απλές μηχανικές ή αναλογικές συσκευές. Η σταδιακή, όμως, αποδοχή των τεχνικών της Τεχνητής Νοημοσύνης επέκτεινε το πεδίο εφαρμογής τους. Οι εταιρίες πλέον, προσπαθούν να δημιουργήσουν όσο το δυνατόν περισσότερο εξελιγμένα συστήματα βασισμένα στο λογισμικό, τα οποία χρησιμοποιούνται για την εποπτεία και τον έλεγχο της γραμμής παραγωγής, στα δίκτυα ενέργειας και επικοινωνίας, στον έλεγχο εναέριας κυκλοφορίας, στην αναζήτηση πληροφοριών σε κατανεμημένες βάσεις δεδομένων, στα παιχνίδια και σε πολλούς άλλους τομείς..

Τέτοιου είδους συστήματα είναι τα πολυπρακτορικά συστήματα, στα οποία η αλληλεπίδραση είναι δυναμική και όχι στατικά καθορισμένη, δίνοντας τη δυνατότητα στο σύστημα να συμπεριφερθεί με σωστό τρόπο, ακόμη και σε περιπτώσεις που δεν είχαν αρχικά προβλεφθεί. Η δυναμική φύση της αλληλεπίδρασης των πρακτόρων, καθώς και το γεγονός ότι οι πράκτορες αποτελούν αυτόνομες οντότητες, εγείρει μια σειρά αρκετά δύσκολων θεμάτων που αφορούν στην υλοποίηση πολυπρακτορικών συστημάτων.

Ένα από τα σημαντικότερα ζητήματα στη μελέτη των πολυπρακτορικών συστημάτων είναι αυτό του πολυπρακτορικού σχεδιασμού. Το συγκεκριμένο ζήτημα μελετά τις τεχνικές και τις διαδικασίες μέσω των οποίων οι πράκτορες αλληλεπιδρούν, συντονίζονται, συνάπτουν συμφωνίες για να σχεδιάσουν τα πλάνα των ενεργειών τους κατά την επίλυση του προβλήματος που ανατίθεται στο σύστημα.

Βασικό ζητούμενο του πολυπρακτορικού σχεδιασμού είναι ο συντονισμός και η επικοινωνία των πρακτόρων, ώστε τα δημιουργηθέντα πλάνα ενεργειών να είναι αποδοτικότερα και ακριβέστερα στην επίλυση του προβλήματος.

Στην παρούσα διπλωματική εργασία γίνεται προσπάθεια να μοντελοποιηθούν κάποιες από τις μελετώμενες τεχνικές, μέσω της δημιουργίας μιας πολυπρακτορικής εφαρμογής. Η εφαρμογή, δεδομένης μιας βάσης γνώσης (ενός χάρτη), αναζητεί και κατασκευάζει μια διαδρομή ανάμεσα σε δύο πόλεις, χρησιμοποιώντας διαφορετικά μέσα μεταφοράς. Παρατηρούμε έτσι τον τρόπο επικοινωνίας και συνεργασίας των πρακτόρων και μελετάμε τη συμπεριφορά τους.

1.2 Διάθρωση της διπλωματικής εργασίας

Στο Κεφάλαιο 2 παρατίθεται μια εισαγωγή σε βασικές έννοιες πολυπρακτορικών συστημάτων, καθώς και η θεωρητική μελέτη των βασικότερων τεχνικών που χρησιμοποιούνται στον πολυπρακτορικό σχεδιασμό.

Στο Κεφάλαιο 3 παρουσιάζεται η διαδικασία σχεδιασμού της υλοποίησης της εφαρμογής, τα χαρακτηριστικά της πλατφόρμας υλοποίησης και οι τεχνικές ανάμεσα στις οποίες έγινε η επιλογή για την εφαρμογή.

Στο Κεφάλαιο 4 παρατίθεται η υλοποίηση της εφαρμογής μέσω UML διαγραμμάτων και στιγμιότυπων από την εκτέλεσή της.

Τέλος, στο Κεφάλαιο 5 παρουσιάζονται τα συμπεράσματα της διπλωματικής εργασίας και η συζήτηση για μελλοντική επέκταση της εργασίας.

Κεφάλαιο 2

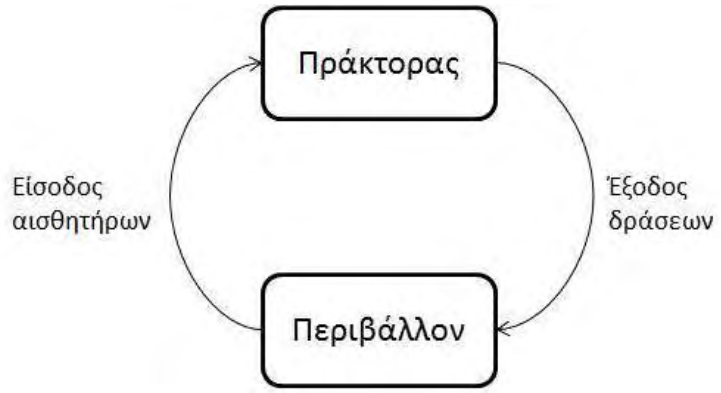
Πολυπρακτορικός Σχεδιασμός

Στο συγκεκριμένο κεφάλαιο γίνεται μια εισαγωγή στον τομέα των Πολυπρακτορικών συστημάτων και μελετώνται βασικές αρχές του Πολυπρακτορικού Σχεδιασμού.

2.1 Βασικές έννοιες και ορισμοί

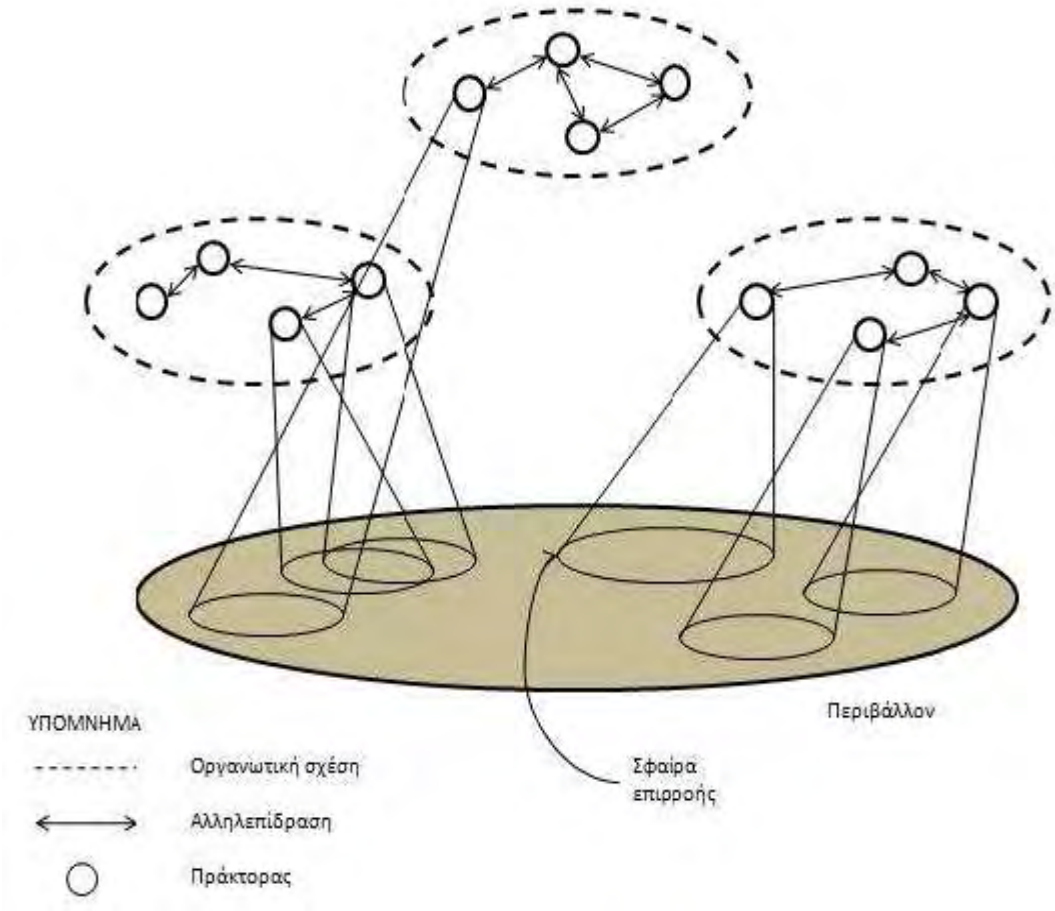
Αρχικά θα πρέπει να δοθεί ο ορισμός του όρου πράκτορας, για την εισαγωγή σε ευρύτερες έννοιες και τεχνικές των πολυπρακτορικών συστημάτων. Επειδή δεν υπάρχει καθολικά αποδεκτός ορισμός, θα παρουσιαστεί ο ορισμός των Wooldrigde και Jennings (1995) όπως αυτός παρέχεται από το βιβλίο «Εισαγωγή στα Πολυπρακτορικά Συστήματα» [12].

Ορισμός: *Ο πράκτορας είναι ένα υπολογιστικό σύστημα που βρίσκεται σε κάποιο περιβάλλον, και το οποίο είναι ικανό για αυτόνομη δράση μέσα σε αυτό το περιβάλλον προκειμένου να ικανοποιήσει τους σχεδιαστικούς του στόχους.*



Εικόνα 2.1 Ένας πράκτορας στο περιβάλλον του

Στον πραγματικό κόσμο όμως σπάνια, έως καθόλου, συναντάμε σύστημα με έναν μόνο πράκτορα. Η πλειονότητα των συστημάτων περιέχει δύο ή και περισσότερους πράκτορες, οι οποίοι αλληλεπιδρούν και συνεπώς η πολυπλοκότητα στην προσπάθεια επίλυσης ενός προβλήματος αυξάνει. Εισάγεται πλέον στο σύστημα η έννοια της επικοινωνίας, καθώς οι πράκτορες μπορεί να έχουν επικαλυπτόμενη επιρροή στο περιβάλλον και σχέσεις εξάρτησης μεταξύ τους.



Εικόνα 2.2 Τυπική δομή ενός πολυπρακτορικού συστήματος

Στο σημείο αυτό είναι σημαντικό να εισάγουμε στη μελέτη την έννοια του κατανεμημένου, καθώς η επίλυση του προβλήματος απαιτεί πλέον συλλογική προσπάθεια. Η ανάγκη αυτή βασίζεται στην εγγενή κατανομή των πόρων, στο σύστημα και στους πράκτορες, όπως η βάση γνώσης, η ικανότητα, η πληροφορία και η εξειδίκευση. Η ενασχόληση με το ζήτημα ξεκίνησε με τη Συνεργατική κατανεμημένη επίλυση προβλήματος (ΣΚΕΠ).

Ορισμός: *Η ΣΚΕΠ μελετά τον τρόπο με τον οποίο ένα χαλαρά συνδεδεμένο δίκτυο επιλυτών προβλημάτων μπορεί να συνεργαστεί για να επιλύσει προβλήματα που είναι πέρα από τις ατομικές ικανότητες των επιλυτών αυτών. Κάθε κόμβος επίλυσης προβλήματος στο δίκτυο είναι ικανός για εξελιγμένη επίλυση και μπορεί να εργαστεί ανεξάρτητα, αλλά τα προβλήματα που αντιμετωπίζουν οι κόμβοι δεν μπορούν να ολοκληρωθούν χωρίς συνεργασία. Η συνεργασία είναι απαραίτητη επειδή κανένας μεμονωμένος κόμβος δεν έχει αρκετή εξειδικευμένη γνώση, πόρους και πληροφορίες για να επιλύσει ένα πρόβλημα και οι διάφοροι κόμβοι μπορεί να διαθέτουν εξειδικευμένη γνώση για την επίλυση διαφορετικών τμημάτων του προβλήματος.*[5]

Η επίλυση αυτού του είδους των προβλημάτων απαιτεί, αφενός συνοχή στην ομάδα των πρακτόρων (να επιθυμούν να συνεργαστούν καλά) και αφετέρου συντονισμός αρμοδιοτήτων (να γνωρίζουν πως θα συνεργαστούν καλά).

- Συνοχή (coherence): Η συνοχή μπορεί να μετρηθεί από την άποψη της ποιότητας της λύσης, της αποδοτικότητας της χρήσης πόρων, της εννοιολογικής σαφήνειας της λειτουργίας, ή της ομαλότητας στη μείωση της απόδοσης του συστήματος, όταν υπάρχει αβεβαιότητα ή αστοχία.

- Συντονισμός αρμοδιοτήτων (competence coordination): Σε ένα τέλεια συντονισμένο σύστημα, οι πράκτορες δεν θα ασχολούνται κατά λάθος με τους υποστόχους των άλλων πρακτόρων, στην προσπάθειά τους για την επίτευξη ενός κοινού στόχου. Η παρουσία διενέξεων μεταξύ των πρακτόρων, με την έννοια της καταστροφικής επέμβασης του ενός στον άλλο, είναι ένδειξη κακού συντονισμού.

Ένα βασικό ζήτημα στην επίλυση πολυπρακτορικών προβλημάτων είναι αυτό του σχεδιασμού των ενεργειών των πρακτόρων. Σε αντίθεση με τα μονοπρακτορικά συστήματα το πρόβλημα έγκειται στον κοινό σχεδιασμό δράσης και στην παραγωγή ενός συνολικού πλάνου για το σύστημα, καθώς πλέον οι πράκτορες δεν λειτουργούν ατομικά αλλά ως ομάδα για την επίλυση του προβλήματος. Πολλές φορές οι ενέργειες ενός πράκτορα έρχονται σε σύγκρουση με τις ενέργειες ενός άλλου πράκτορα ή οι προϋποθέσεις μιας ενέργειας a_i ενός πράκτορα Α, δύναται να ισχύουν

εφόσον έχει εκτελεστεί μια ενέργεια b_j ενός πράκτορα Β και συνεπώς η ενέργεια b_j πρέπει να προηγείται. Παρατηρούμε συνεπώς πως οι πράκτορες εκτός από το να ασχολούνται με την επίλυση του προβλήματος, πρέπει να ασχολούνται και με τον σχεδιασμό των ενεργειών τους. Η ανάγκη αυτή συνίσταται στο πως οι πράκτορες θα εργαστούν ομαδικά, να αποσυνθέσουν το πρόβλημα σε υποπροβλήματα, να αναθέσουν τα υποπροβλήματα, να ανταλλάξουν τις λύσεις των υποπροβλημάτων και να συνθέσουν τις επιμέρους λύσεις στην συνολική.

Τη λύση σε αυτά τα προβλήματα έρχεται να δώσει ο πολυπρακτορικός σχεδιασμός και η ύπαρξη συγχρονισμού στις ενέργειες των πρακτόρων.

2.2 Κοινή διαχείριση εργασιών

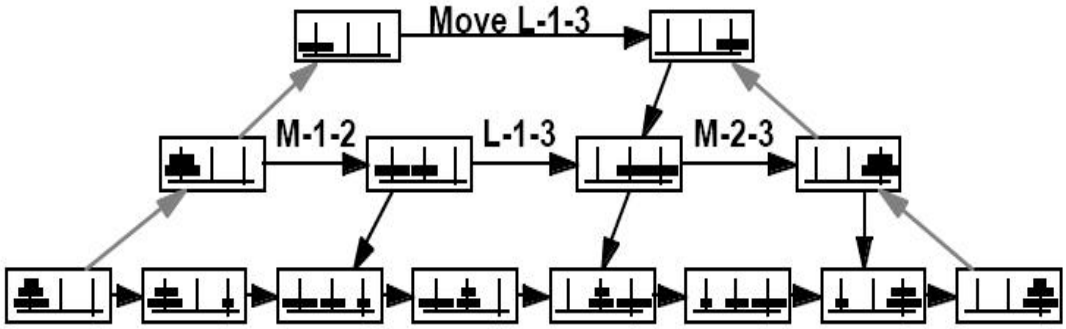
Η κοινή διαχείριση εργασιών πραγματοποιείται κατά την αποσύνθεση του προβλήματος και την ανάθεση των εργασιών των υποπροβλημάτων σε διαφορετικούς πράκτορες. Η ιδέα βασίζεται στο γεγονός ότι αν ένας πράκτορας έχει πολλές εργασίες να διεκπεραιώσει, πρέπει να βοηθηθεί από άλλους πράκτορες με λιγότερες ή καθόλου εργασίες. Τα βήματα για να επιτευχθεί η διαμοίραση των εργασιών είναι:

- **Αποσύνθεση του προβλήματος**: Το αρχικό μας πρόβλημα αποσυντίθεται ιεραρχικά σε μικρότερα υποπροβλήματα μέχρι να δημιουργηθούν τέτοια υποπροβλήματα σε μέγεθος που να επιδέχονται επίλυσης από έναν μεμονωμένο πράκτορα. Η διαδικασία της αποσύνθεσης μπορεί να γίνει από έναν πράκτορα, με την προϋπόθεση ότι αυτός κατέχει την απαραίτητη γνώση τόσο για τη δομή της εργασίας του προβλήματος, ώστε μπορεί να την διαμοιράσει σε μικρότερα μη επικαλυπτόμενα κομμάτια, όσο και για τους συμμετέχοντες στην επίλυση πράκτορες, για να μην υπάρξουν υποπροβλήματα που δε θα μπορούν να επιλυθούν από κάποιον πράκτορα. Αν την ανωτέρω γνώση την κατέχουν περισσότεροι του ενός πράκτορες, τότε η διαδικασία της αποσύνθεσης μπορεί να αντιμετωπιστεί ως συνεργατική διαδικασία, με ενδεχομένως καλύτερα αποτελέσματα, γεγονός όμως που απαιτεί επιπρόσθετο συντονισμό ενεργειών.

- **Ανάθεση υποπροβλημάτων**: Στο παρόν στάδιο τα υποπροβλήματα που δημιουργήθηκαν από την αποσύνθεση διαβιβάζονται στους κατάλληλους πράκτορες για επίλυση (η διαδικασία αυτή μπορεί να θεωρηθεί και ως μέρος της διαδικασίας αποσύνθεσης).

- **Επίλυση υποπροβλημάτων**: Τα ανωτέρω υποπροβλήματα επιλύονται, το καθένα ξεχωριστά, από τους πράκτορες που τους έχουν ανατεθεί. Κατά τη διαδικασία επίλυσης ενδεχομένως οι πράκτορες να χρειάζονται πληροφορίες από κάποιον άλλο πράκτορα για να συνεχίσουν την εργασία τους. Συνεπώς απαιτείται κοινή διαχείριση πληροφορίας και επικοινωνία μεταξύ των πρακτόρων.

- **Σύνθεση αποτελεσμάτων**: Οι επιμέρους λύσεις των υποπροβλημάτων ενοποιούνται αναδρομικά (με ιεραρχία ανάστροφη της αποσύνθεσης) στην τελική συνολική λύση του αρχικού προβλήματος. Ομοίως με τη διαδικασία της αποσύνθεσης, μπορεί να επιτευχθεί είτε από έναν εξειδικευμένο πράκτορα, είτε από περισσότερους μέσω επικοινωνίας και συντονισμού.



Εικόνα 2.3 Παράδειγμα για το πρόβλημα των Πύργων του Ανόι

Το γεγονός αυτό επιτυγχάνεται σχετικά απλά όταν όλοι οι πράκτορες είναι ομοιογενείς ως προς τις ικανότητές τους. Στις περιπτώσεις ανομοιογενών ή πλήρως αυτόνομων πρακτόρων, σε κάθε πράκτορα πρέπει να ανατεθεί μια εργασία ανάλογα με τις ικανότητές του, ενώ υπάρχει και η περίπτωση οι πράκτορες να μην διακατέχονται από την υπόθεση της αγαθής προαίρεσης, να είναι ανταγωνιστικοί και να απαιτείται σύναψη συμφωνιών.

2.3 Κοινή διαχείριση αποτελεσμάτων

Κατά την διάρκεια της διαδικασίας της κοινής διαχείρισης αποτελεσμάτων οι πράκτορες ανταλλάσουν πληροφορίες μεταξύ τους. Ένας πράκτορας αποστέλλει μια πληροφορία σε έναν άλλο είτε ενεργητικά, επειδή πιστεύει πως αυτή η πληροφορία θα του φανεί χρήσιμη στο μέλλον, είτε αντιδραστικά απατώντας στο αίτημα κάποιου άλλου πράκτορα για την συγκεκριμένη πληροφορία. Τα αποτελέσματα όμως που εξάγονται κατά την επίλυση μιας εργασίας από έναν πράκτορα μπορεί να διαφέρουν από τα αποτελέσματα που εξάγει ένας άλλος πράκτορας για την ίδια εργασία. Για παράδειγμα στους μαθητές μιας τάξης δίδεται η ίδια εργασία, όμως η επίλυσή της διαφέρει από μαθητή σε μαθητή. Για να βελτιωθεί η ομαδική απόδοση των πρακτόρων κατά την κοινή διαχείριση αποτελεσμάτων ο Durfee [4] προτείνει τους ακόλουθους τρόπους:

- **Εμπιστοσύνη (Confidence):** Οι ανεξάρτητα παραγόμενες, από τους πράκτορες, λύσεις επαληθεύονται συγκρινόμενες μεταξύ τους, αποδίδοντας ένα συλλογικό αποτέλεσμα που θα έχει μεγαλύτερο βαθμό εμπιστοσύνης στο να είναι σωστό. Στο παράδειγμα με τους μαθητές, μπορούν να συγκρίνουν τα αποτελέσματά τους για να αυξήσουν την εμπιστοσύνη τους στις απαντήσεις.

- **Πληρότητα (Completeness):** Κάθε πράκτορας διατυπώνει το αποτέλεσμά του για κάθε εργασία που μπορεί (ή του έχει ανατεθεί) να εκτελέσει (τοπική άποψη). Μοιραζόμενοι τα αποτελέσματα αυτά οι πράκτορες καλύπτουν ένα πληρέστερο τμήμα της συνολικής εργασίας (σφαιρική άποψη).

- **Ακρίβεια (Precision):** Για να βελτιώσει τη δική του λύση ένας πράκτορας χρειάζεται να γνωρίζει περισσότερα για τις λύσεις που έχουν διατυπώσει οι άλλοι πράκτορες. Για παράδειγμα σε μια μηχανική εφαρμογή κάθε πράκτορας θα μπορούσε να καταλήξει ξεχωριστά στις προδιαγραφές ενός εξαρτήματος. Γνωρίζοντας όμως τις προδιαγραφές των εξαρτημάτων και των άλλων πρακτόρων θα μπορούσε να καταλήξει σε προδιαγραφές που ταιριάζουν μεταξύ τους με μεγαλύτερη ακρίβεια.

- **Επικαιρότητα (Timeliness):** Ακόμα και αν ένας πράκτορας μπορούσε να επιλύσει ένα μεγάλο μέρος του προβλήματος μόνος του, με την κοινή διαχείριση της λύσης θα μπορούσε να παραχθεί το αποτέλεσμα γρηγορότερα.

Παρατηρώντας τα πλεονεκτήματα της κοινής διαχείρισης αποτελεσμάτων καταλήγουμε στο συμπέρασμα πως οι πράκτορες πρέπει να ανταλλάσουν αποτελέσματα. Η διαδικασία όμως αυτή, δεν είναι τόσο εύκολη καθώς οι πράκτορες θα πρέπει να γνωρίζουν πώς να αφομοιώσουν τα κοινόχρηστα αποτελέσματα στα δικά τους. Επιπλέον, δεδομένου ότι η αφομοίωση μπορεί να μην είναι τετριμμένη, η ανταλλαγή μεγάλου όγκου αποτελεσμάτων μπορεί να είναι δαπανηρή, συνεπώς οι πράκτορες θα πρέπει να είναι όσο το δυνατόν πιο επιλεκτικοί στα αποτελέσματα που ανταλλάσουν.

2.4 Κοινός σχεδιασμός δράσης

Η διαδικασία σχεδιασμού των ενεργειών των πρακτόρων και η ενσωμάτωσή τους σε υποπλάνα που θα κληθούν να εξυπηρετήσουν οι πράκτορες, αποτελεί μια σύνθετη διαδικασία στην πολυπρακτορική επίλυση προβλημάτων. Το γεγονός οφείλεται πρωτίστως στην ασάφεια του όρου «κατανεμημένο», που εισάγαμε παραπάνω, στο σύστημα και στην διαδικασία. Το υπό εξέταση ζήτημα θα μπορούσε να εστιαστεί στο σημείο που θα εισάγουμε τον όρο στη διαδικασία σχεδιασμού, ούτως ώστε να είναι λειτουργικότερη. Μια θεώρηση είναι τα παραγόμενα υποπλάνα από τη διαδικασία σχεδιασμού να κατανέμονται ανάμεσα στους πράκτορες, που βρισκονται εντός ενός ή περισσοτέρων συστημάτων. Εναλλακτική θεώρηση είναι η διαδικασία σχεδιασμού και παραγωγής υποπλάνων να είναι κατανεμημένη ανάμεσα στους πράκτορες. Είτε να συμβαίνουν και οι δυο θεωρήσεις.

2.4.1 Συγκεντρωτικός σχεδιασμός για κατανεμημένα πλάνα (centralized planning for distributed plans)

Στόχος του συγκεκριμένου τρόπου σχεδιασμού ενεργειών είναι η δημιουργία ξεχωριστών πλάνων για κάθε πράκτορα. Συνεπώς τα πλάνα που πρέπει να

εκτελεστούν κατανεμημένα μπορούν να δημιουργηθούν με συγκεντρωτικό τρόπο, από ένα σύστημα σχεδιασμού, από το οποίο ορίζεται ο καταμερισμός και η διάταξη εργασιών. Δεν χρειάζεται να υπάρχει αυστηρή σειρά ανάμεσα στις ενέργειες καθώς αυτές μπορούν να εκτελούνται παράλληλα. Έτσι έχουμε έναν συγκεντρωτικό συντονιστή (πράκτορας «αφέντης») ο οποίος αναλαμβάνει να διαμοιράσει το πλάνο στους πράκτορες που θα το εκτελέσουν (πράκτορες «σκλάβοι»), καθένας από τους οποίους εκτελεί το μέρος του πλάνου που του αναλογεί. Σκοπός είναι λειτουργώντας οι πράκτορες παράλληλα να επιτύχουν αυτήν την κατάσταση του κόσμου που θα περιέχει τους στόχους του πλάνου, σύμφωνα με τις υποθέσεις της ορθότητας της γνώσης και του προβλέψιμου το κόσμου.

Ο αλγόριθμος που δίνει την διαδικασία αυτήν είναι ο εξής:

1. Δεδομένης μιας περιγραφής του στόχου, του συνόλου των λειτουργιών και μιας περιγραφής της αρχικής κατάστασης δημιουργείται το μερικώς διατεταγμένο πλάνο. Όπου είναι πιθανό, δια μέσω έρευνας, βρίσκεται ένα σχέδιο όπου τα βήματα έχουν λίγες διατεταγμένες δεσμεύσεις μεταξύ τους.
2. Αποσύνθεση του πλάνου σε υποπλάνα όπου οι διατεταγμένες σχέσεις ανάμεσα στα βήματα τείνουν να συγκεντρώνονται εντός αυτών και να ελαχιστοποιούνται ανάμεσα σε αυτά.
3. Εισαγωγή συγχρονισμού στις ενέργειες των υποπλάνων.
4. Ανάθεση των υποπλάνων στους πράκτορες «σκλάβους». Σε περίπτωση αποτυχίας επιστροφή σε προηγούμενα βήματα (διαφορετική αποσύνθεση ή διαφορετικό πλάνο).
5. Εκκίνηση της εκτέλεσης των πλάνων.

Παρατηρούμε πως ο ανωτέρω αλγόριθμος αποτελεί μια εξειδίκευση της διαδικασίας κοινής διαχείρισης εργασιών που περιγράφηκε παραπάνω. Ουσιαστικά, ο αντικειμενικός στόχος είναι να βρεθεί, μέσω της αποσύνθεσης και της ανάθεσης, από όλα τα πιθανά πλάνα που επιτυγχάνουν τον στόχο, το πλάνο που μπορεί να αποσυντεθεί και να κατανεμηθεί πιο αποτελεσματικά.

Η επικοινωνιακή υποδομή έχει μεγάλο αντίκτυπο στον βαθμό αποσύνθεσης και κατανομής και συνεπώς το χρηματικό και χρονικό κόστος πρέπει να λαμβάνεται υπ’ όψιν. Σε χαλαρά συνδεδεμένα δίκτυα οδηγεί το σύστημα με λίγους πράκτορες να

επιλύει μεγάλες εργασίες ενώ σε ισχυρά συνδεδεμένα ο βαθμός της αποσύνθεσης και η παράλληλη εκτέλεση αυξάνονται.

2.4.2 Κατανεμημένος σχεδιασμός (distributed planning)

Ο σχεδιασμός της παραγωγής ενός σύνθετου συγκεντρωτικού πλάνου συχνά απαιτεί τη συνεργασία ανάμεσα σε διαφορετικούς πράκτορες, εξειδικευμένους στο σχεδιασμό. Καθένας από αυτούς συνεισφέρει σε ένα μέρος του πλάνο, ανάλογα με το κομμάτι στο οποίο ειδικεύεται, χωρίς απαραίτητα να είναι και ένας από αυτούς που θα το εκτελέσει.

Σε αυτόν τον τρόπο σχεδιασμού, χρησιμοποιούνται συχνά οι τεχνικές της κοινής διαχείρισης εργασιών και αποτελεσμάτων. Το αρχικό πρόβλημα αποσυντίθεται και κατανέμεται στους/από τους «ειδικούς» πράκτορες, μέχρι να συνθέσουν το συγκεντρωτικό πλάνο. Στις αλληλεπιδράσεις αυτές ανάμεσα στους πράκτορες σχεδιαστές η πληροφορία που ανταλλάσσεται είναι το μερικώς εξειδικευμένο πλάνο.

Οι πιο ασύγχρονες ενέργειες στο κομμάτι του σχεδιασμού επιλύονται μέσω της κοινής διαχείρισης αποτελεσμάτων. Αντί οι πράκτορες να περιφέρουν ένα μεμονωμένο πλάνο, το οποίο θα επεξεργάζονται σειριακά, και ενδεχομένως να καταλήξει σε αδιέξοδο και να πρέπει να επιστραφεί, οι πράκτορες σχεδιάζουν παράλληλα μερικά πλάνα τα οποία συνθέτουν στο συγκεντρωτικό πλάνο.

Ο συγκεκριμένος τρόπος σχεδιασμού βρίσκει, κυρίως, εφαρμογή στον τομέα των logistics, στον κατασκευαστικό τομέα και στον τομέα των δικτύων επικοινωνίας.

2.4.3 Κατανεμημένος σχεδιασμός για κατανεμημένα πλάνα (distributed planning for distributed plans)

Αποτελεί τον πιο σύνθετο και δύσκολο τρόπο σχεδιασμού ενεργειών καθώς εδώ η έννοια του κατανεμημένου βρίσκεται τόσο στην διαδικασία σχεδίασης του πλάνου, όσο και στο αποτέλεσμα αυτής που είναι η εκτέλεση των παραγομένων πλάνων. Μια ομάδα πρακτόρων συνεργάζονται μεταξύ τους για την κατάστρωση διαφορετικών ατομικών πλάνων, συντονίζοντας δυναμικά τις ενέργειές τους.

Συνεπώς είναι σχεδόν άχρηστο να υπάρχει ένα συγκεντρωτικό πολυπρακτορικό πλάνο οπουδήποτε στο σύστημα, αλλά κομμάτια αυτού κατανεμημένα στους πράκτορες που θα τα εκτελέσουν.

Επειδή οι πράκτορες συνεργάζονται τόσο κατά την σχεδίαση, όσο και κατά την εκτέλεση των πλάνων, και ενδέχεται να είναι ιδιοτελείς, πρέπει να υπάρχει καλή επικοινωνία στο σύστημα για να επιτυγχάνονται οι διαπραγματεύσεις ανάμεσα τους. Επιπρόσθετα απαιτείται συμβατότητα ανάμεσα στα κατανεμημένα κομμάτια του πλάνου, για την αποφυγή διενέξεων, ενώ είναι επιθυμητό να αλληλοβοηθούνται έτσι ώστε να επιτευχθούν οι στόχοι τους.

2.5 Συγχώνευση πλάνων

Υπάρχουν αρκετές τεχνικές για να επιτευχθεί αυτή η συνεργασία, ανάμεσα σε αυτές είναι και ένας αλγόριθμος που πρότεινε ο Georgeff [6] για την συγχώνευση πλάνων (plan merging) με στόχο την οργάνωση των ενεργειών των πρακτόρων, ούτως ώστε αυτοί να μπορούν να συνεργάζονται και να αποφεύγουν τις συγκρούσεις.

Οι ενέργειες περιγράφονται ως μια ακολουθία συνόλου καταστάσεων $S_1,S_2,S_3,\ldots,S_n$ που περιγράφουν διαισθητικά τις καταστάσεις από τις οποίες διέρχεται μια ενέργεια, με S_1 το πεδίο ορισμού των αρχικών καταστάσεων και S_n το εύρος των τελικών καταστάσεων μιας ενέργειας. Η περιγραφή αυτή μας επιτρέπει να λάβουμε υπ' όψιν τόσο τις συνεργατικές όσο και τις επιβλαβείς, για το σύστημα, αλληλεπιδράσεις μεταξύ των πρακτόρων.

Οι ενέργειες στα πλάνα προσδιορίζονται με τη χρήση γενικευμένης σημειογραφίας STRIPS όπου εκτός από τις γνωστές λίστες προϋποθέσεων (S_1) – προσθέσεων-διαγραφών (S_n), υπάρχει και μια λίστα διάρκειας για τις ενέργειες. Η λίστα αυτή περιέχει ένα σύνολο συνθηκών, οι οποίες θα πρέπει να είναι αληθείς κατά τη διάρκεια εκτέλεσης της ενέργειας. Υποθέτουμε ότι η διάρκεια εκτέλεσης μιας ενέργειας δεν είναι εξ αρχής ορισμένη και ότι το σύστημα δύναται να τη γνωρίζει μόνο όταν η ενέργεια ολοκληρωθεί και ερωτηθεί ο αντίστοιχος πράκτορας.

Υποθέτουμε ότι δεδομένου ενός προβλήματος σχεδιασμού έχουμε αποσυνθέσει τον αρχικό στόχο σε υποστόχους. Χωρίς να χαθεί η αρχή της

γενικότητας, υποθέτουμε ότι αποσυντίθεται σε δυο υποστόχους. Επίσης υποθέτουμε πως έχουμε καταστρώσει δυο ξεχωριστά πλάνα που επιλύουν αυτούς τους υποστόχους. Το πρόβλημα συνεπώς είναι να συνδυάσουμε αυτά τα δυο πλάνα σε ένα πολυπρακτορικό πλάνο που να αποφεύγει τις συγκρούσεις και να επιτρέπει όσο το δυνατόν περισσότερες ενέργειες να εκτελεστούν παράλληλα.

Το πρώτο πράγμα που πρέπει να εξεταστεί είναι ο τρόπος με τον οποίο κάποιες ατομικές ενέργειες αλληλεπιδρούν με κάποιες άλλες. Μετά χρειάζεται να προσδιορίσουμε ποιες από τις καταστάσεις, των ενεργειών αυτών, είναι μη ασφαλείς (unsafe), δηλαδή οδηγούν σε αδιέξοδο, και να προσθέσουμε μηχανισμούς συγχρονισμού στα δύο υποπλάνα.

Η διαδικασία αυτή αποτελείται από τρία στάδια:

1. **Ανάλυση αλληλεπιδράσεων (interaction analysis)**: Στο στάδιο αυτό παράγεται μια περιγραφή των τρόπων με τους οποίους αλληλεπιδρούν τα πλάνα. Για να περιγραφεί η σχέση των αλληλεπιδράσεων των στόχων χρησιμοποιούνται οι έννοιες της ικανοποιησιμότητας (satisfiability), της αντιμεταθετικότητας (commutativity), και της προτεραιότητας (precedence). Δυο ενέργειες ονομάζονται ικανοποιήσιμες αν υπάρχει κάποια σειρά εκτέλεσης χωρίς να ακυρώνονται οι προϋποθέσεις της μιας ή και των δυο. Υποπερίπτωση της ικανοποιησιμότητας είναι η αντιμεταθετικότητα, όπου οι δυο ενέργειες μπορούν να εκτελεστούν παράλληλα. Αν δυο ενέργειες είναι αντιμεταθετικές είτε δεν αλληλεπιδρούν ή οι αλληλεπιδράσεις είναι ακίνδυνες. Κατά τη διάρκεια της ανάλυσης αλληλεπιδράσεων γίνεται αναζήτηση στα πλάνα των όλων πρακτόρων, για την ανίχνευση όλων των αλληλεπιδράσεων που υπάρχουν.

2. **Ανάλυση ασφάλειας (safety analysis)**: Αφού προσδιοριστούν οι πιθανές αλληλεπιδράσεις ανάμεσα στα πλάνα πρέπει να εξεταστούν ως προς την ασφάλεια τους. Η ασφάλεια ορίζεται ως το ζεύγος ενεργειών σε σχέση με την προτεραιότητα και την αντιμεταθετικότητα του ζεύγους. Πρώτον, όλες οι ενέργειες που είναι ασφαλείς (δεν υπάρχει αλληλεπίδραση ή οι ενέργειες είναι αντιμεταθετικές) αφαιρούνται από το πλάνο (απλοποίηση). Στη συνέχεια παράγεται το σύνολο όλων των μη ασφαλών αλληλεπιδράσεων.

3. **Διευθέτηση αλληλεπιδράσεων (interaction resolution)**: Για την επίλυση των διενέξεων οι μη ασφαλείς αλληλεπιδράσεις ορίζονται ως κρίσιμες περιοχές (critical sections) και απαιτείται αμοιβαίος αποκλεισμός για τον χειρισμό αυτών (π.χ. σημαφόροι).

Πρόκειται για μια απλή και αποδοτική τεχνική δημιουργίας ευέλικτων πολυπρακτορικών πλάνων από τη συγχώνευση απλών μονοπρακτορικών πλάνων. Τα βασικότερα χαρακτηριστικά της είναι, αφενός, ο τρόπος αναπαράστασης των ενεργειών, που επιτρέπει την έκφραση πιο σύνθετων μορφών αλληλεπίδρασης, αφετέρου η χρήση γενικευμένης σημειογραφίας STRIPS και οι εφαρμογές της αντιμεταθετικότητας, συνεισφέρουν στην κατηγορηματική αποφυγή δημιουργίας πιθανών διενέξεων μεταξύ των ενεργειών που περιλαμβάνονται στο πλάνο.

2.6 Γλώσσες επικοινωνίας

Έχει γίνει πολλές φορές αναφορά στην ανάγκη του πολυπρακτορικού σχεδιασμού για επικοινωνία ανάμεσα στους πράκτορες, ώστε να επιτευχθεί συγχρονισμός ενεργειών και χειρισμός επιβλαβών αλληλεπιδράσεων. Για να υλοποιηθεί η απαιτούμενη επικοινωνία είναι απαραίτητη μια γλώσσα επικοινωνίας, ώστε οι συμβαλλόμενοι πράκτορες να μπορούν να ανταλλάσουν μηνύματα, τα οποία θα είναι κατανοητά και από τους δύο. Αν και έχουν προταθεί πλήθος γλωσσών επικοινωνίας πρακτόρων, δύο είναι σήμερα οι επικρατέστερες η KQML και η FIPA ACL.

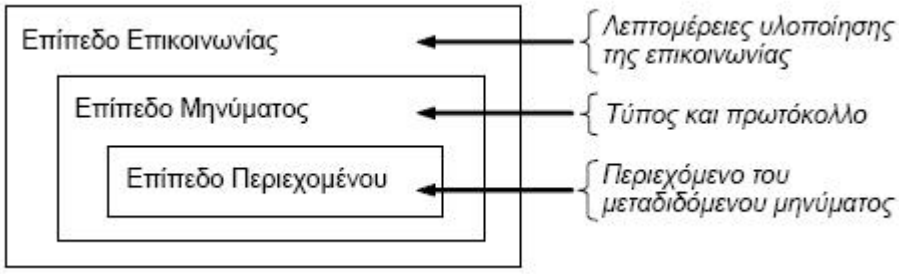
2.6.1 KQML

Η KQML (Knowledge and Query Manipulation Language), αποτελεί ταυτόχρονα ένα πρωτόκολλο και μια γλώσσα επικοινωνίας πρακτόρων βασισμένη σε μηνύματα, τα οποία έχουν μια κοινή μορφή. Στην KQML ορίζονται τρία διαφορετικά επίπεδα:

- **Επίπεδο περιεχομένου (content layer):** Το οποίο αφορά το περιεχόμενο του μεταδιδόμενου μηνύματος. Αυτό καθορίζεται από την αντίστοιχη εφαρμογή και μπορεί να είναι οτιδήποτε.

- **Επίπεδο μηνύματος (message layer):** Το οποίο είναι ο πυρήνας της γλώσσας. Στο επίπεδο αυτό καθορίζεται ο τύπος του μηνύματος, με κάποια από τις διαθέσιμες δηλώσεις (performative) από το σύνολο που έχει καθοριστεί για τη γλώσσα (tell, ask, reply, inform, advertise, κτλ) καθώς και το πρωτόκολλο (protocol), δηλαδή ο τρόπος μετάδοσης και λήψης των μηνυμάτων, όπως για παράδειγμα σύγχρονος, ασύγχρονος, κτλ.

- **Επίπεδο επικοινωνίας (communication layer):** Στο οποίο καθορίζονται οι χαμηλού επιπέδου λεπτομέρειες της υλοποίησης της επικοινωνίας.



Εικόνα 2.4 Επίπεδα πρωτοκόλλου KQML

Τα KQML μηνύματα μπορούν να θεωρηθούν ως αντικείμενα, σύμφωνα με την ευρεία έννοια του αντικειμενοστραφούς προγραμματισμού, καθένα από τα οποία περιέχει: ένα τελεστικό (κλάση) και έναν αριθμό παραμέτρων (μεταβλητές).

```
(<performative>
    :sender     <ο αποστολέας>
    :receiver   <ο παραλήπτης>
    :language   <η γλώσσα στην οποία έχει κωδικοποιηθεί το περιεχόμενο>
    :ontology   <η οντολογία είναι το λεξιλόγιο το οποίο πρέπει
                    να χρησιμοποιηθεί για να γίνει κατανοητό το μήνυμα>
    :content    <το περιεχόμενο του μηνύματος>
)
```

Εικόνα 2.5 Παράδειγμα μηνύματος KQML

Στον πίνακα που ακολουθεί παρουσιάζονται τα συνολικά 41 τελεστικά της γλώσσας. Το S συμβολίζει το πεδίο του αποστολέα (sender), το R το πεδίο το παραλήπτη (receiver) και το C το περιεχόμενο του μηνύματος (content).

Τελεστικό	**Σημασία**
achieve	Ο S θέλει να κάνει ο R κάτι αληθές στο περιβάλλον του
advertise	Ο S δηλώνει κατάλληλος για την επεξεργασία ενός τελεστικού
ask-about	Ο S θέλει όλες τις συναφείς προτάσεις της εικονικής βάσης γνώσης (VKB) του R
ask-all	Ο S θέλει όλες τις απαντήσεις του R για μια ερώτηση C
ask-if	Ο S θέλει να ξέρει αν η απάντηση του R υπάρχει στο VKB του C
ask-one	Ο S θέλει μια από τις απαντήσεις του R για μια ερώτηση C
break	Ο S θέλει να διακόψει ο R μια διοχέτευση (pipe) που έχει εγκαθιδρύσει
broadcast	Ο S θέλει να στείλει ο R ένα τελεστικό σε όλες τις συνδέσεις
broker-all	Ο S θέλει να μαζέψει ο R όλες τις απαντήσεις σε ένα τελεστικό
broker-one	Ο S θέλει να πάρει ο R βοήθεια στην απάντηση σε ένα τελεστικό
deny	Το ενσωματωμένο τελεστικό δεν εφαρμόζεται πλέον στον S
delete-all	Ο S θέλει να αφαιρέσει ο R όλες τις προτάσεις που ταιριάζουν με τη C
delete-one	Ο S θέλει να αφαιρέσει ο R μια πρόταση που ταιριάζει με τη C
discard	Ο S δε θέλει τις εναπομένουσες απαντήσεις του R για ένα ερώτημα
eos	Τέλος μιας ροής απάντησης για ένα ερώτημα
error	Ο S θεωρεί ότι ένα προηγούμενο ερώτημα του R είναι κακοδιατυπωμένο
evaluate	Ο S θέλει να αποτιμήσει ο R το C
forward	Ο S θέλει να προωθήσει ο R ένα μήνυμα σε κάποιον άλλον πράκτορα
generator	Ίδιο με το standby για ένα stream-all
insert	Ο S ζητά από τον R να προσθέσει περιεχόμενο στη δική του VKB
monitor	Ο S θέλει ενημερώσεις για την απάντηση του R σε ένα stream-all
next	Ο S θέλει την επόμενη απάντηση του R σε ένα τελεστικό ροής
pipe	Ο S θέλει να δρομολογήσει ο R όλα τα υπόλοιπα τελεστικά σε άλλον πράκτορα
ready	Ο S είναι έτοιμος να απαντήσει σε ένα προηγούμενο τελεστικό του R
recommend-all	Ο S θέλει όλα τα ονόματα των πρακτόρων που μπορούν να απαντήσουν στο C
recommend-one	Ο S θέλει το όνομα ενός πράκτορα που μπορεί να απαντήσει στο C
recruit-all	Ο S θέλει να κάνει ο R όλους τους κατάλληλους πράκτορες να απαντήσουν στο C
recruit-one	Ο S θέλει να κάνει ο R έναν κατάλληλο πράκτορα να απαντήσει στο C
register	Ο S μπορεί να παραδώσει τελεστικά σε κάποιον κατονομαζόμενο πράκτορα
reply	Μεταδίδει μια αναμενόμενη απάντηση
rest	Ο S θέλει τις εναπομένουσες απαντήσεις του R για έναν προηγούμενο τελεστικό
sorry	Ο S δε μπορεί να δώσει μια πιο κατατοπιστική απάντηση

Τελεστικό	Σημασία
standby	Ο S θέλει να είναι έτοιμος ο R να απαντήσει σε ένα τελεστικό
stream-about	Η εκδοχή πολλών απαντήσεων του ask-about
stream-all	Η εκδοχή πολλών απαντήσεων του ask-all
subscribe	Ο S θέλει ενημερώσεις για την απάντηση του R σε ένα τελεστικό
tell	Ο S ισχυρίζεται στον R ότι το C υπάρχει στη VKB του
transport-address	Ο S συσχετίζει ένα συμβολικό όνομα με τη διεύθυνση μεταφοράς
unregister	Άρνηση του register
untell	Ο S ισχυρίζεται στον R ότι το C δεν υπάρχει στη VKB του

Πίνακας 2.1 Τελεστικά της KQML

2.6.2 FIPA ACL

Το Ίδρυμα για Ευφυείς Φυσικούς Πράκτορες (Foundation for Intelligent Physical Agents, FIPA), από το 1995, είχε ως στόχο την καθιέρωση προτύπων, ώστε να εξασφαλιστεί η διαλειτουργικότητα των συστημάτων πρακτόρων που θα αναπτυχθούν στη βιομηχανία. Αποτέλεσμα της προσπάθειας αυτής ήταν η ανάπτυξη της γλώσσας επικοινωνίας πρακτόρων FIPA ACL.

Συντακτικά το μήνυμα της FIPA ACL παρουσιάζει αρκετές ομοιότητες με αυτό της KQML. Περιέχει, ομοίως, ένα τελεστικό (κλάση) που ορίζει την επιδιωκόμενη ερμηνεία των μηνυμάτων και έναν αριθμό παραμέτρων (μεταβλητών). Η διαφορά έγκειται στις τιμές που μπορούν να λάβουν τα πεδία αυτά.

```
(<performative>
    :sender     <ο αποστολέας>
    :receiver   <ο παραλήπτης>
    :language   <η γλώσσα στην οποία έχει κωδικοποιηθεί το περιεχόμενο>
    :ontology   <η οντολογία είναι το λεξιλόγιο το οποίο πρέπει
                    να χρησιμοποιηθεί για να γίνει κατανοητό το μήνυμα>
    :content    <το περιεχόμενο του μηνύματος>
)
```

Εικόνα 2.6 Παράδειγμα μηνύματος FIPA ACL

Η πιο σημαντική διαφορά μεταξύ των δυο γλωσσών έγκειται στα σύνολα των τελεστικών που παρέχουν. Στην FIPA ACL παρέχονται 20 τελεστικά τα οποία κατηγοριοποιούνται ανάλογα με το είδος της επικοινωνίας.

Τελεστικό	**Μεταβίβαση πληροφορίας**	**Αίτηση πληροφορίας**	**Διαπραγμάτευση**	**Εκτέλεση ενεργειών**	**Χειρισμός σφαλμάτων**
accept-proposal			X		
agree				X	
cancel		X		X	
cfp			X		
confirm	X				
disconfirm	X				
failure					X
inform	X				
inform-if	X				
inform-ref	X				
not-understood					X
propagate				X	
propose			X		
proxy				X	
query-if		X			
query-ref		X			
refuse				X	
reject-proposal			X		
request				X	
request-when				X	
request-whenever				X	
subscribe		X			

Πίνακας 2.2 Τελεστικά ανά κατηγορία στη FIPA ACL

Κεφάλαιο 3

Σχεδιασμός της Υλοποίησης

Στο συγκεκριμένο κεφάλαιο γίνεται μια περιγραφή της φύσης του προβλήματος που επιλύει η εφαρμογή. Επιπλέον, παρουσιάζονται οι επιλογές και οι περιπτώσεις που προκύπτουν στη σχεδίαση, λόγω των διαφορετικών τεχνικών που μελετήσαμε. Παρουσιάζεται επίσης μια περιγραφή της πλατφόρμας υλοποίησης της εφαρμογής και των χαρακτηριστικών της.

3.1 Περιγραφή του προβλήματος

Η εφαρμογή που υλοποιήθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας επιχειρεί να μοντελοποιήσει κάποιες από τις τεχνικές σχεδίασης που αναπτύχθηκαν θεωρητικά, καθώς και να μας δώσει τη δυνατότητα να μελετήσουμε την επικοινωνία και τον τρόπο με τον οποίο αυτή επιτυγχάνεται μεταξύ των πρακτόρων, ώστε να υπάρχει συντονισμός ενεργειών.

Το πρόβλημα που πρέπει να επιλυθεί έγκειται στην κατασκευή ενός δρομολογίου. Αναλυτικότερα, στην εύρεση μιας τυχαίας διαδρομής ανάμεσα σε δύο πόλεις, αποτελούμενη ενδεχομένως από ενδιάμεσους σταθμούς, όπου η μετακίνηση από μια πόλη σε άλλη μπορεί να επιτευχθεί με τη χρήση διάφορων μεταφορικών μέσων. Στην περίπτωση μας έχουμε τρία μέσα μεταφοράς, αεροπλάνο, τρένο και λεωφορείο.

Ανάλογα με τα προτεινόμενα είδη σχεδιασμού έχουμε και διαφορετικές περιπτώσεις σεναρίων για το σύστημα, το ρεπερτόριο ενεργειών κάθε πράκτορα, την αναπαράσταση της βάσης γνώσης, ανάμεσα στα οποία πρέπει να επιλέξουμε ώστε να προχωρήσουμε στην υλοποίηση της εφαρμογής.

3.2 Κατανομή ρεπερτορίου ενεργειών στους πράκτορες

Ανάλογα με το είδος του κοινού σχεδιασμού δράσης προκύπτει η ανάγκη για δημιουργία διαφορετικών πρακτόρων, πέρα από τους επιλυτές του προβλήματος, οι οποίοι θα είναι επιφορτισμένοι με τον σχεδιασμό των πλάνων. Όσον αφορά τους επιλυτές του προβλήματος, εφόσον βρούμε τον τρόπο σχεδιασμού, πρέπει να βρούμε πόσοι χρειάζονται ανάλογα με το ρεπερτόριο ενεργειών τους για να επιλύεται το πρόβλημα. Παρατηρούμε πως ο αριθμός των πρακτόρων σε μια εφαρμογή, το ρεπερτόριο ενεργειών τους, καθώς και ο τρόπος του σχεδιασμού, είναι σχέσεις αμφίδρομες και αλληλοεξαρτώμενες.

3.2.1 Πράκτορες με διαφορετικό ρεπερτόριο ενεργειών και κοινή γνώση χάρτη

Στην περίπτωση αυτή έχουμε πράκτορες με διαφορετικό ρεπερτόριο ενεργειών, όπου ο κάθε πράκτορας μπορεί χα χειρίζεται ένα διαφορετικό μέσω μεταφοράς. Δίνεται η δυνατότητα να έχουμε είτε συγκεντρωτικό, είτε κατανεμημένο σχεδιασμό για κατανεμημένα πλάνα.

Στην πρώτη περίπτωση θα χρειαστούμε έναν επιπλέον πράκτορα στο σύστημά μας, που θα αναλάβει το ρόλο του συντονιστή των ενεργειών των υπολοίπων. Θα είναι ο «προσωπικός βοηθός» (PDA) του χρήστη καθώς θα είναι ο μόνος που θα αλληλεπιδρά άμεσα μαζί του. Είναι υπεύθυνος για το σχεδιασμό των πλάνων των πρακτόρων, την ανταλλαγή πληροφοριών και τη σύνθεση των μερικών λύσεων στην τελική.

Στη δεύτερη περίπτωση οι πράκτορες χειριστές των μέσων μεταφοράς, αναλαμβάνουν να συντονίσουν τις ενέργειές τους μέσω μερικού σχεδιασμού των υπό εκτέλεση πλάνων. Πρέπει μόνοι τους να συνάψουν συμφωνίες για την ανάθεση ενός έργου σε κάποιον που μπορεί να το επιλύσει, να αποτρέψουν διενέξεις και όπου υπάρξουν να τις λύσουν καθώς και να συνθέσουν τις μερικές λύσεις σε τελική. Παρατηρούμε επομένως ότι ο φόρτος εργασιών ενός πράκτορα αυξάνεται αισθητά.

Η κοινή γνώση αρχικής και τελικής κατάστασης διευκολύνει τους πράκτορες στο συγχρονισμό των ενεργειών στην αναζήτηση του δρομολογίου, καθώς όλοι γνωρίζουν πως και για πόσο πρέπει να συνεχίζουν την αναζήτηση αυτή. Η κοινή γνώση του χάρτη αυξάνει την προσβασιμότητα των πρακτόρων σε γνώση, άρα οι

πράκτορες εκτελούν την εργασία τους παράλληλα χωρίς να περιμένουν πληροφορία από άλλον πράκτορα, αυξάνει όμως και την ανάγκη του συστήματος για χρήση πόρων. Το πρόβλημα τώρα έγκειται στον τρόπο με τον οποίο θα αναπαρασταθεί η γνώση αυτή.

3.2.2 Πράκτορες με κοινό ρεπερτόριο ενεργειών, κοινό πρόβλημα, μη κοινή γνώση χάρτη

Τώρα οι πράκτορες εκτελούν όλοι τις ίδιες ενέργειες, κοινό ρεπερτόριο, δηλαδή μπορεί να χειρίζεται ο καθένας και τα τρία μέσα μεταφοράς. Επειδή όμως η βάση γνώσης δεν είναι κοινή, δεν μπορεί κανένας να δώσει λύση μόνος του αν η αρχική και η τελική κατάσταση δεν βρίσκεται στο δικό του πεδίο γνώσης. Συνεπώς μπορούμε να πούμε ότι κάθε πράκτορας γνωρίζει όλα τα δρομολόγια, όλων των μέσων μεταφοράς μόνο για ένα τμήμα του χάρτη, όπως ένα γεωγραφικό διαμέρισμα.

Στην περίπτωση αυτή έχουμε κατανεμημένο σχεδιασμό ενεργειών, όπου ο κάθε πράκτορας, ειδικός σε ένα μέρος του προβλήματος, βρίσκει τη λύση στο δικό του μέρος, οι οποίες στο τέλος συντίθενται από όλους. Όλοι οι πράκτορες εκτελούν το ίδιο πλάνο, το οποίο σχεδιάζεται από κοινού. Δίνεται οι δυνατότητα να έχουμε διαφορετικούς πράκτορες σχεδιαστές των πλάνων από τους επιλυτές ή να αναλάβουν το σχεδιασμό οι ήδη υπάρχοντες πράκτορες.

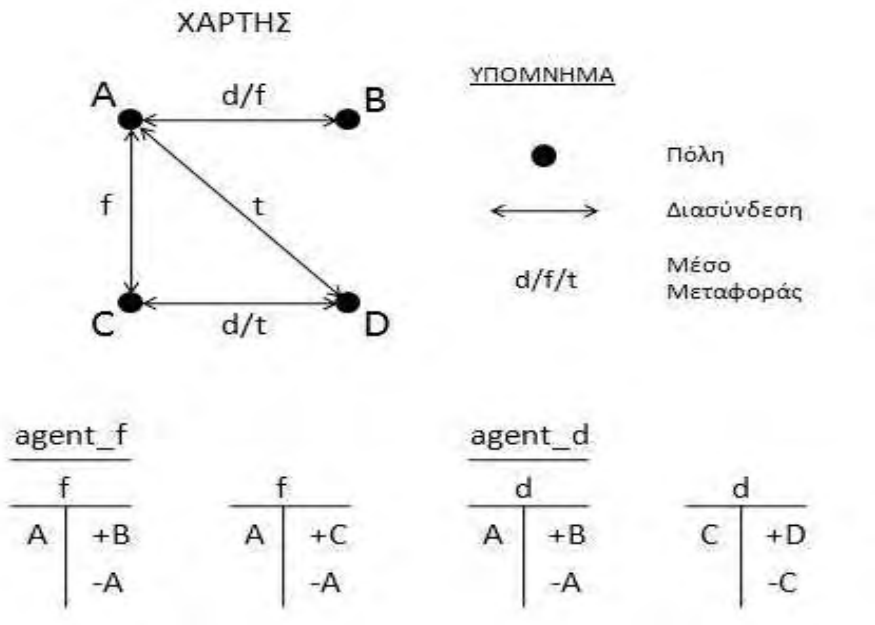
Στο ερώτημα που πρέπει να απαντήσουμε είναι αν γνωρίζει κάθε πράκτορας ποιοί άλλοι πράκτορες υπάρχουν και σε πιο κομμάτι του χάρτη έχει πρόσβαση ο καθένας. Αν δεν γνωρίζουν την ύπαρξη των άλλων πρακτόρων, για να επιτύχουμε την κοινή διαχείριση αποτελεσμάτων, μπορούμε να προσθέσουμε ένα πράκτορα, κι εδώ εν είδη PDA, που θα αναλάβει την επικοινωνία μεταξύ των πρακτόρων, μεταβιβάζοντας μόνο την πληροφορία ότι ο ένας τελείωσε την εργασία του και εισήλθε στο τάδε γεωγραφικό διαμέρισμα, ενημερώνοντας τον υπεύθυνο πράκτορα να συνεχίσει.

3.3 Αναπαράσταση Βάσης Γνώσης

Όπως έχει αναφερθεί και παραπάνω η βάση γνώσης (knowledge base) και η πρόσβαση σε αυτήν αποτελεί βασικό παράγοντα στα πολυπρακτορικά συστήματα καθώς η επίλυση ενός προβλήματος εξαρτάται από τη γνώση που έχεις πάνω σε αυτό. Συνεπώς σχεδιάζοντας μια εφαρμογή πρέπει να λάβουμε σοβαρά υπ' όψιν τον τρόπο με τον οποίο θα αναπαρασταθεί η γνώση αυτή, ώστε να είναι γρηγορότερη η πρόσβαση και λειτουργικότερη η εφαρμογή.

3.3.1 Έμμεση δήλωση Βάσης Γνώσης

Οι πράκτορες έχουν πρόσβαση σε αυτή, δηλαδή στις πληροφορίες που χρειάζονται για την εκτέλεση μιας εργασίας, σύμφωνα με το ρεπερτόριο ενεργειών τους. Αποτελεί την πιο απλή μορφή αναπαράστασης καθώς το ρεπερτόριο ενεργειών, οι προϋποθέσεις και οι προσθέσεις-διαγραφές τους είναι ήδη γνωστά σε ένα πράκτορα.



Εικόνα 3.1 Παράδειγμα έμμεσης δήλωσης ΒΓ

Σύμφωνα με το παράδειγμα, παρατηρούμε πως ο πράκτορας agent_d γνωρίζει από τις διαθέσιμες για αυτόν ενέργειες αν μπορεί να μεταβεί από την πόλη C στην

πόλη D. Παρά την απλότητά της η συγκεκριμένη αναπαράσταση υστερεί στο γεγονός ότι δεν μπορούμε να πετύχουμε κοινή βάση γνώσης για όλους τους πράκτορες.

3.3.2 Άμεση δήλωση Βάσης Γνώσης μέσω κλάσεων

Ένας άλλος τρόπος δήλωσης του χάρτη είναι μέσω κλάσεων, από τη στιγμή που η εφαρμογή θα υλοποιηθεί σε JAVA. Έτσι το σύστημα μας είναι ενιαίο και δε χρειάζεται τροποποιήσεις ή προσθήκες συμβατότητας. Η πρόσβαση στις πληροφορίες είναι απλή και γίνεται με απλή προσπέλαση των τιμών των μεταβλητών των αντικειμένων των κλάσεων.

Το πρόβλημα είναι να βρούμε τον αποδοτικότερο για τις σχεδιαστικές μας ανάγκες τρόπο δήλωσης των κλάσεων, έτσι ώστε να αναπαριστά ευκρινώς το χάρτη, να διευκολύνει το χειρισμό τον δεδομένων από τους πράκτορες και να τροποποιείται ανεξάρτητα από την υπόλοιπη εφαρμογή.

3.3.3 Άμεση δήλωση Βάσης Γνώσης μέσω Βάσης Δεδομένων

Ένας εύκολος τρόπος δημιουργίας της βάσης γνώσης είναι μέσω μιας βάσης δεδομένων, όπως για παράδειγμα η SQL. Η απεικόνιση είναι εύκολη, καθώς μια πόλη συσχετίζεται με μια άλλη με κάποιο μεταφορικό μέσο. Αποτελεί τον πληρέστερο τρόπο απεικόνισης ενός χάρτη και είναι αρκετά εύχρηστος σε περιπτώσεις μεγάλου όγκου δεδομένων. Έχει το πλεονέκτημα ότι είναι ανεξάρτητος της εφαρμογής και συνεπώς εύκολα τροποποιήσιμος, χωρίς αλλαγές στην εφαρμογή.

Αυτό που πρέπει να χειριστούμε είναι βασικά ερωτήματα SQL, κάτι όμως που απαιτεί γνώση SQL, αλλά και γνώση χειρισμού της μέσα από JAVA καθώς ο υπόλοιπος κώδικας θα είναι γραμμένος σε αυτήν.

3.3.4 Άμεση δήλωση Βάσης Γνώσης μέσω αρχείου

Ο τελευταίος τρόπος αναπαράστασης του χάρτη είναι μέσω ενός αρχείου .txt από το οποίο οι πράκτορες θα διαβάζουν τις πληροφορίες που επιθυμούν. Και εδώ η βάση γνώσης είναι ανεξάρτητη της εφαρμογής και ευκόλως τροποποιήσιμη. Παρόλο

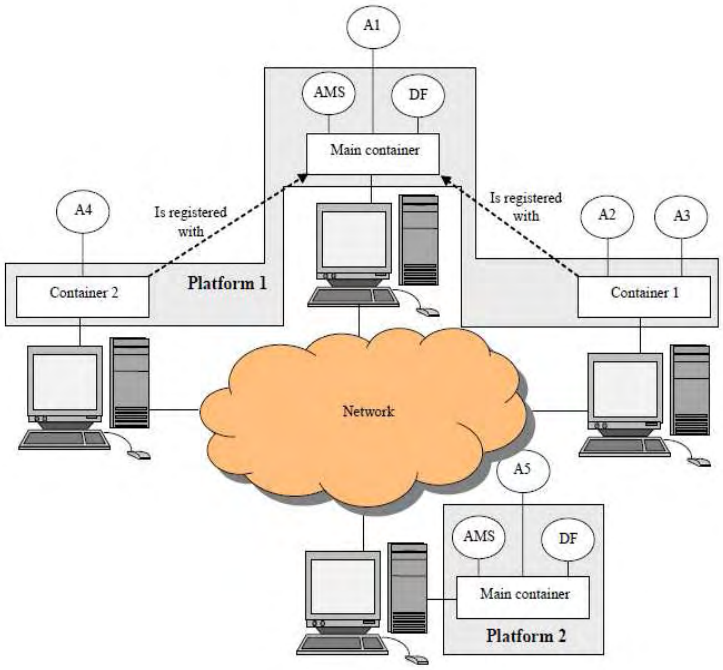
που ο χειρισμός αρχείων μέσα από τη JAVA είναι κάτι εύκολο, το γεγονός των συσχετίσεων που πρέπει να προσδώσουμε καθώς και ο πιθανός μεγάλος όγκος δεδομένων αυξάνουν την πολυπλοκότητα χειρισμού.

3.4 Εισαγωγή στην πλατφόρμα JADE

Η κατασκευή και η ανάπτυξη πολυπρακτορικών συστημάτων είναι μια δύσκολη και επίπονη διαδικασία για τον προγραμματιστή όταν αυτή γίνεται από το μηδέν. Μεγάλο μέρος της δυσκολίας έγκειται στην επικοινωνία των πρακτόρων και στο πως αυτοί θα ανταλλάσουν μηνύματα για το σχεδιασμό πλάνων για την επίτευξη ενός στόχου ή την επίλυση ενός προβλήματος. Στα πλαίσια ερευνών και κατασκευής κάποιων μεγάλων έργων αναπτύχθηκαν κάποια εργαλεία που δίνουν λύσεις στα προβλήματα αυτά. Εδώ γίνεται μια εισαγωγή στην πλατφόρμα υλοποίησης και εκτέλεσης πολυπρακτορικών συστημάτων JADE.

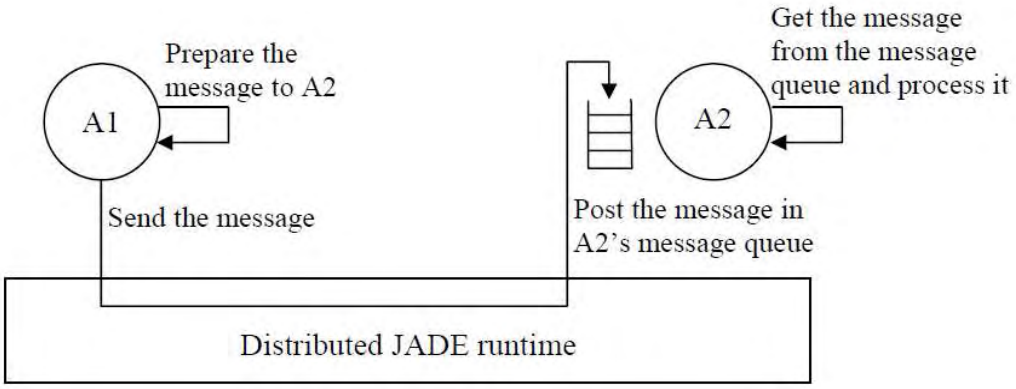
Το JADE (Java Agent Development fragement) είναι ένα πλαίσιο λογισμικού για την ανάπτυξη εφαρμογών βασισμένων στους πράκτορες, υλοποιημένο εξ ολοκλήρου σε JAVA. Απλοποιεί την υλοποίηση πολυπρακτορικών συστημάτων μέσω ενός ενδιάμεσου λογισμικού (middleware), που ακολουθεί όλες τις προδιαγραφές FIPA. Για το σκοπό αυτό παρέχει μια βιβλιοθήκη κλάσεων, τις οποίες μπορεί να χρησιμοποιήσει ο προγραμματιστής στην ανάπτυξη των πρακτόρων και ένα σύνολο από εργαλεία γραφικού περιβάλλοντος. Ασχολείται, επίσης, και με όλες τις πτυχές ενός συστήματος που δεν έχουν να κάνουν με τα εσωτερικά του πράκτορα και είναι ανεξάρτητα της υλοποίησης, όπως η μεταφορά μηνυμάτων, η κωδικοποίηση και η ανάλυση σφαλμάτων και ο κύκλος ζωής ενός πράκτορα.

Το αναπτυσσόμενο σύστημα μπορεί να κατανεμηθεί, ακόμα και κατά την διάρκεια της εκτέλεσής του, πάνω από δίκτυο. Οι πράκτορες μπορούν να εκτελούνται ταυτόχρονα είτε στο ίδιο, είτε σε διαφορετικά τερματικά (τα οποία δεν είναι απαραίτητο να χρησιμοποιούν το ίδιο λειτουργικό σύστημα), μέσω της Java Virtual Machine (JVM).



Εικόνα 3.2 Κατανομή πρακτόρων στο δίκτυο

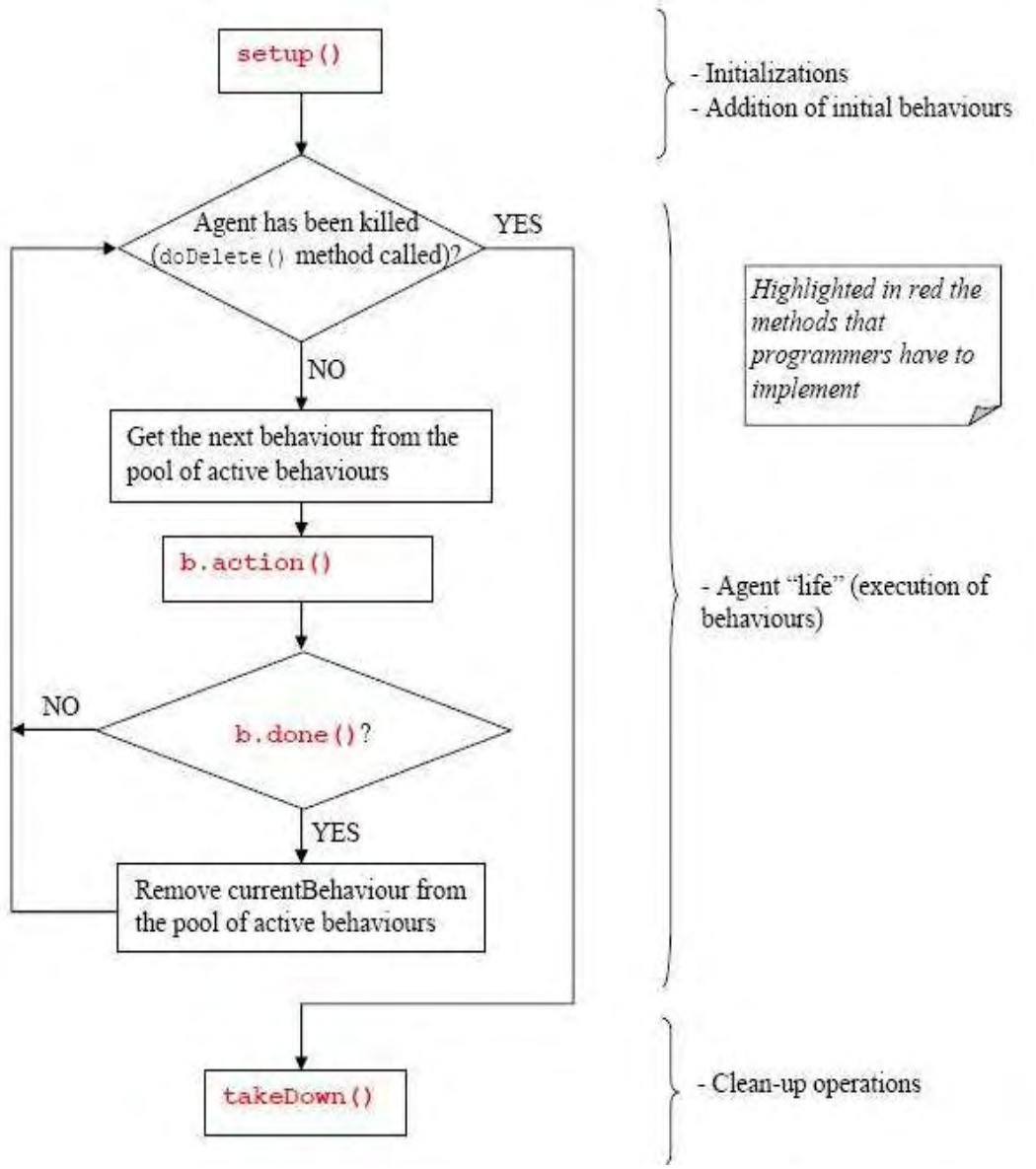
Η αρχιτεκτονική της επικοινωνίας προσφέρει έναν ευέλικτο και αποδοτικό τρόπο ανταλλαγής μηνυμάτων. Το JADE δημιουργεί και χειρίζεται μια ουρά εισερχομένων μηνυμάτων τύπου FIPA ACL, ιδιωτικά σε κάθε πράκτορα. Οι πράκτορες μπορούν να έχουν πρόσβαση στην ουρά μέσω ενός συνδυασμού από διάφορες λειτουργίες. Έχει υλοποιηθεί το πλήρες μοντέλο επικοινωνίας FIPA και τα συστατικά του έχουν διακριθεί σαφώς και ενσωματωθεί πλήρως στην πλατφόρμα.



Εικόνα 3.3 Επικοινωνία πρακτόρων στο JADE

Οι πράκτορες είναι υλοποιημένοι ως ένα νήμα για κάθε πράκτορα, ενώ ο επιπρόσθετος πολυνηματικός χειρισμός και συγχρονισμός, προσφέρεται για χρήση

απ' ευθείας από τη JAVA. Το JADE υποστηρίζει, επίσης, τον προγραμματισμό συνεργατικών μεθόδων «συμπεριφορών» (behaviours) για κάθε πράκτορα, μέσω των οποίων εκτελούν τις ενέργειές τους. Το runtime περιλαμβάνει επίσης κάποιες έτοιμες προς χρήση συμπεριφορές για τις πιο συνηθισμένες εργασίες στον προγραμματισμό πρακτόρων, όπως πρωτόκολλα αλληλεπίδρασης FIPA, αφύπνιση υπό ορισμένες προϋποθέσεις, καθώς και τη δόμηση πολύπλοκων εργασιών από ομαδοποιήσεις απλούστερων.

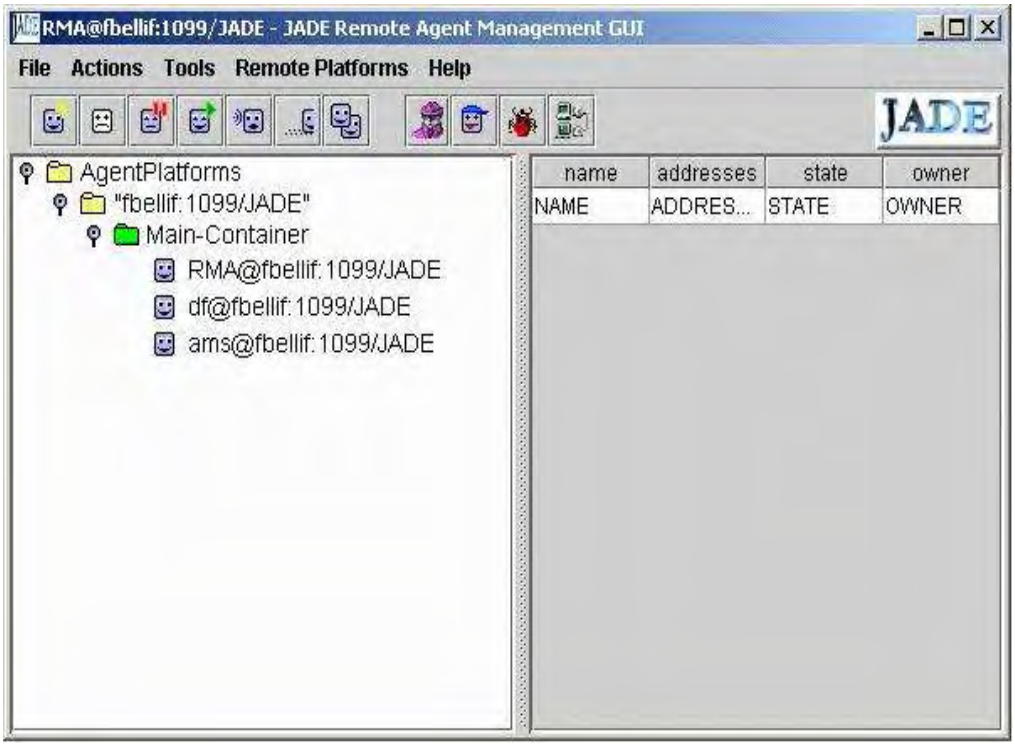


Εικόνα 3.4 Διάγραμμα εκτέλεσης νήματος ενός πράκτορα

3.4.1 Ειδικοί πράκτορες

Κατά την εκκίνηση της πλατφόρμας JADE παρέχονται κάποιοι ειδικοί πράκτορες (special agents) για τη διευκόλυνση του χρήστη στο χειρισμό και την ανάπτυξη πολυπρακτορικών συστημάτων.

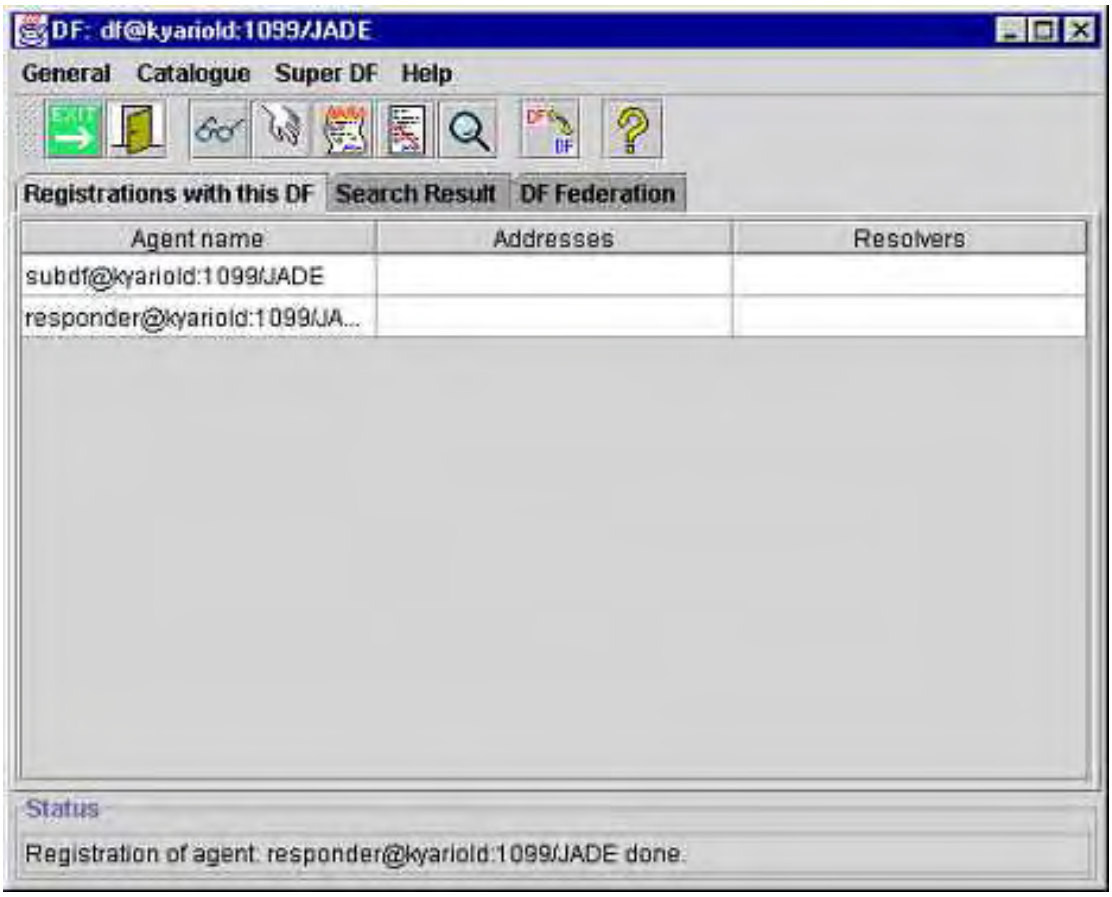
- **Remote Management Agent (RMA):** Λειτουργεί ως ένα γραφικό περιβάλλον διεπαφής για τον χρήστη (GUI) μέσω του οποίου γίνεται απομακρυσμένη διαχείριση, παρακολούθηση και έλεγχος της κατάστασης των πρακτόρων. Το πρώτο στιγμιότυπο του RMA μπορεί να ενεργοποιηθεί με την επιλογή στην γραμμή εντολών «-gui», και μετά μέσω αυτού μπορούν να ενεργοποιηθούν περισσότερα. Η συνοχή ανάμεσα σε πολλαπλά RMAs διατηρείται με κοινοποίηση (multicasting) των γεγονότων σε καθένα από αυτά. Επίσης, μέσω της κονσόλας RMA μπορούν να ξεκινήσουν όλα τα παρεχόμενα εργαλεία και να χειρισθούν άλλες απομακρυσμένες πλατφόρμες πρακτόρων, συμβατές με τις προδιαγραφές FIPA.



Εικόνα 3.5 Στιγμιότυπο RMA GUI

- **Agent Management System (AMS):** Είναι ο πράκτορας που ασκεί εποπτικό έλεγχο σχετικά με την πρόσβαση και χρήση της πλατφόρμας. Μόνο ένας AMS μπορεί να υπάρχει σε κάθε πλατφόρμα. Επίσης ο AMS παρέχει υπηρεσίες όπως μια λευκή σελίδα (white page) και τον κύκλο ζωής των πρακτόρων, διατηρώντας έναν κατάλογο των αναγνωριστικών (AID) και της κατάστασης κάθε πράκτορα. Όλοι οι πράκτορες πρέπει να εγγραφούν σε έναν AMS, προκειμένου να πάρουν μια έγκυρη τιμή AID. Έχει ακόμα τη δυνατότητα να δημιουργήσει ή να «σκοτώσει» πράκτορες.

- **Directory Facilitator (DF):** Είναι ο πράκτορας που παρέχει την υπηρεσία κίτρινων σελίδων (yellow pages) στο σύστημα, δηλαδή την υπηρεσία καταλόγου. Κάθε πράκτορας μπορεί να εγγραφεί σε αυτήν δηλώνοντας το είδος των υπηρεσιών που προσφέρει, έτσι ώστε να μπορεί να βρεθεί από άλλους πράκτορες που αναζητούν τη σχετική πληροφορία, προκειμένου να επιτύχουν τους στόχους τους.

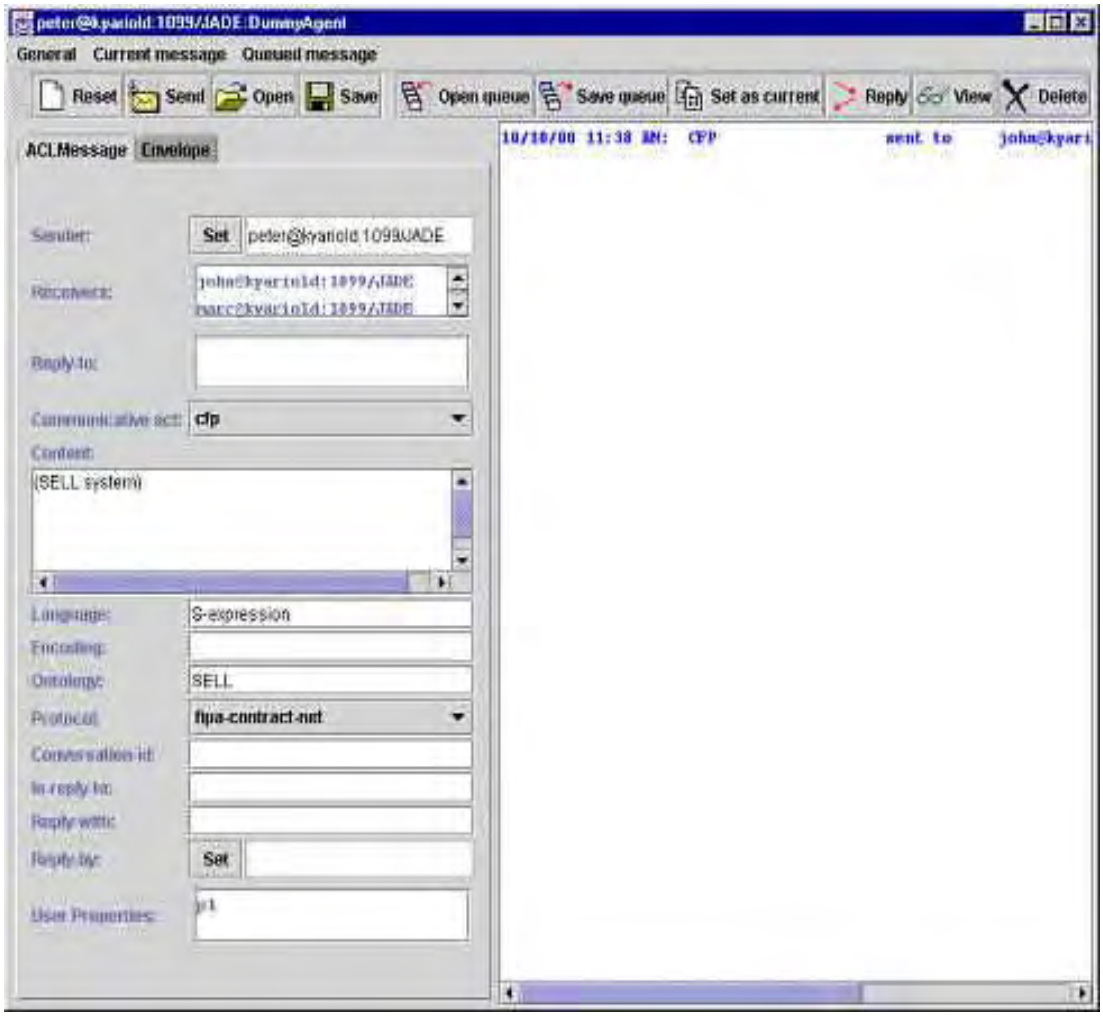


Εικόνα 3.6 Στιγμιότυπο DF GUI

3.4.2 Εργαλεία γραφικού περιβάλλοντος

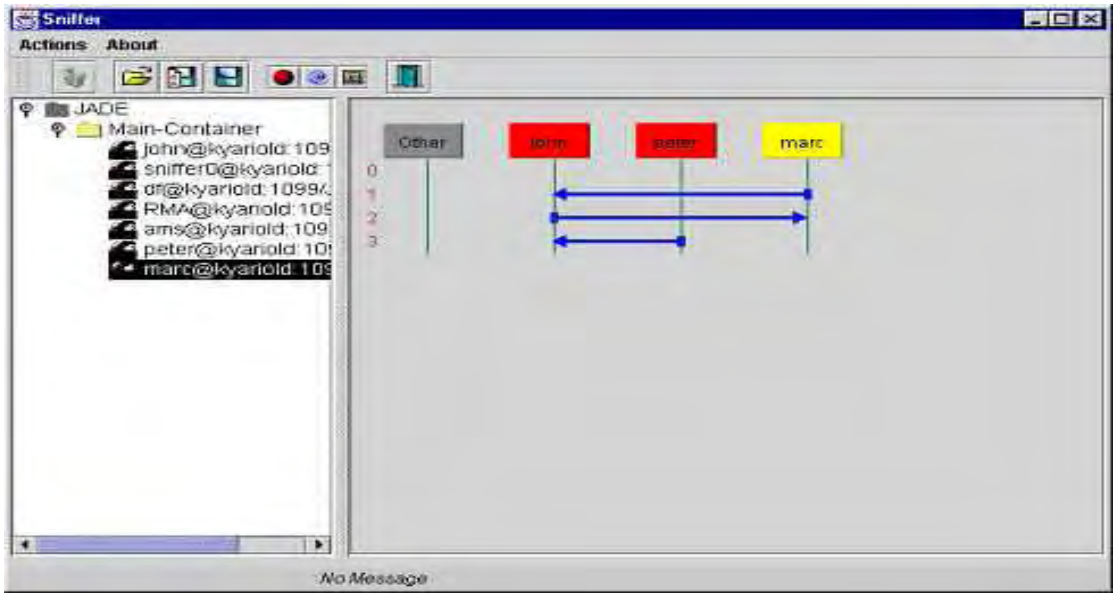
Το JADE παρέχει ένα σύνολο από εργαλεία γραφικού περιβάλλοντος, τα οποία είναι υλοποιημένα μέσω αντίστοιχων πρακτόρων.

- **Dummy Agent:** Είναι ένα εργαλείο για τον έλεγχο και τον εντοπισμό σφαλμάτων στο σύστημα. Παρέχει τη δυνατότητα σύνθεσης και αποστολής μηνυμάτων σε άλλους πράκτορες, καθώς και εμφάνισης της λίστας εισερχομένων και απεσταλμένων μηνυμάτων συμπληρωμένων μια χρονοσφραγίδα (timestamp), προκειμένου να γίνει δοκιμή και καταγραφή της επικοινωνίας και της αντίδρασης κάποιου πράκτορα.



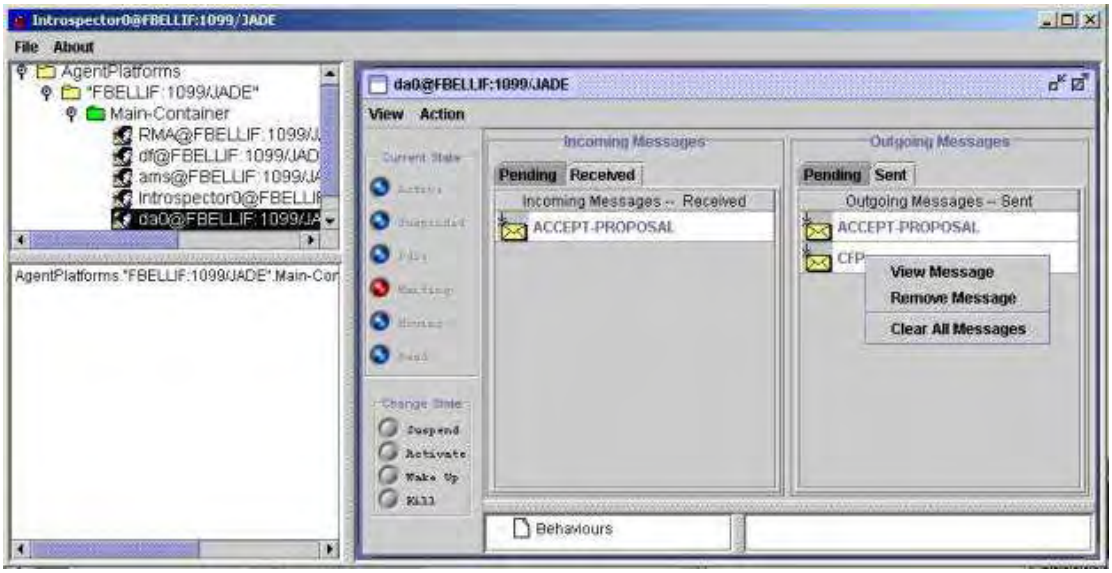
Εικόνα 3.7 Στιγμιότυπο Dummy Agent

- **Sniffer Agent:** Είναι ένας πράκτορας που μπορεί να «υποκλέψει» μηνύματα, κατά τη διάρκεια ανταλλαγής τους από τους πράκτορες, και να τα εμφανίσει γραφικά, χρησιμοποιώντας ένα συμβολισμό παρόμοιο με αυτό των UML διαγραμμάτων ροής. Είναι χρήσιμο για τον εντοπισμό σφαλμάτων στις κοινωνίες πρακτόρων, παρατηρώντας πώς αντιδρούν στα ανταλλασσόμενα μηνύματα.



Εικόνα 3.8 Στιγμιότυπο Sniffer Agent

- **Introspector Agent:** Είναι ένας πράκτορας που επιτρέπει τον έλεγχο του κύκλου ζωής ενός πράκτορα, των μηνυμάτων που έχει ανταλλάξει και των συμπεριφορών που εκτελεί.



Εικόνα 3.9 Στιγμιότυπο Introspector Agent

Κεφάλαιο 4

Υλοποίηση της εφαρμογής

Στο κεφάλαιο αυτό παρουσιάζεται η περιγραφή της υλοποίησης της εφαρμογής. Ως συνέχεια του κεφαλαίου 3, παρατίθεται ο τρόπος με τον οποίο οι επιλογές αυτές έγιναν πράξη.

4.1 Περιγραφή της υλοποίησης

Ξεκινώντας την περιγραφή της πορείας υλοποίησης της εφαρμογής πρέπει πρώτα να αναφέρουμε τις επιλογές που έγιναν, ανάμεσα σε αυτές που παρουσιάστηκαν κατά το σχεδιασμό της. Δεδομένου του προβλήματος που πρέπει να επιλυθεί, αποφασίστηκε η εφαρμογή να υλοποιήσει το σενάριο κατανομής ενεργειών για τους πράκτορες που περιγράφηκε στην ενότητα 3.2.1, και η περίπτωση όπου έχουμε συγκεντρωτικό σχεδιασμό ενεργειών για κατανεμημένα πλάνα. Η συγκεκριμένη επιλογή έγινε για λόγους απλούστευσης της προσπάθειας του συστήματος για συντονισμό ενεργειών και επικοινωνία. Απαιτείται συνεπώς η κατασκευή τεσσάρων (4) πρακτόρων: ένας πράκτορας συντονιστής, ένας πράκτορας χειριστής δρομολογίων αεροπλάνων, ένας πράκτορας χειριστής δρομολογίων λεωφορείων κι ένας πράκτορας χειριστής δρομολογίων τρένων.

Η άλλη επιλογή που έπρεπε να γίνει είναι του τρόπου αναπαράστασης του χάρτη, δηλαδή της βάσης γνώσης για το σύστημα. Επιλέχθηκε η αναπαράσταση να γίνει μέσω κλάσεων και ο χειρισμός μέσω των αντικειμένων αυτών. Η επιλογή αυτή έγινε αφενός, για λόγους ευκολίας στο χειρισμό των αναζητήσεων πληροφοριών και αφετέρου, για λόγους ομοιομορφίας και συνοχής του συστήματος. Επίσης για να μην αυξηθεί η πολυπλοκότητα προγραμματισμού της εφαρμογής, παρόλο που η λύση με

την καλύτερη απεικόνιση και αναζήτηση ήταν μέσω βάσης δεδομένων, όμως γινόταν πολυπλοκότερος ο τρόπος προγραμματισμού, γεγονός που ήταν πέρα από το σκοπό αυτής της διπλωματικής.

Η υλοποίηση βασίζεται στον ακόλουθο γενικό αλγόριθμο επίλυσης του προβλήματος:

1. Εισαγωγή από το χρήστη τιμών αφετηρίας και προορισμού στον συντονιστή
2. Έλεγχος του συντονιστή στη ΒΓ για ορθότητα τιμών
    a. Μη αποδεκτές τιμές, πήγαινε στο 1
    b. Αποδεκτές τιμές, κάνε την αφετηρία επόμενη πόλη
3. Αποστολή από το συντονιστή, στους πράκτορες χειριστές δρομολογίων, της τιμής της επόμενης πόλης
4. Αναζήτηση από τους χειριστές δρομολογίων στη ΒΓ, σύμφωνα με το μέσο τους, της πόλης που συνορεύει με την επόμενη πόλη
    a. Αν υπάρχει, αποστολή στο συντονιστή
    b. Αν όχι, αποστολή σφάλματος στο αίτημα
5. Λήψη από το συντονιστή των 3 απαντήσεων
    a. Αν και οι 3 ή 2 ή 1 έγκυρες, κράτα την πρώτη (κάνε την ληφθείσα πόλη επόμενη πόλη)
    b. Αν και οι 3 σφάλμα, πήγαινε στο 7
6. Έλεγχος του συντονιστή αν έφτασε στον προορισμό
    a. Αν η επόμενη πόλη είναι ο προορισμός, πήγαινε στο 7
    b. Αν όχι, πήγαινε στο 3
7. Εμφάνιση της διαδρομής
8. Τερματισμός

Επειδή ο πράκτορας συντονιστής πρέπει να γνωρίζει τους πράκτορας χειριστές δρομολογίων, με τους οποίους θα συνεργαστεί, έτσι ώστε να μπορεί να ανταλλάσει μαζί τους μηνύματα, κάθε πράκτορας κατά τη εκκίνησή του εγγράφεται στην υπηρεσία καταλόγου (yellow pages) του DF Agent. Όταν εκκινείτε ο συντονιστής αναζητά στον κατάλογο τρεις χειριστές δρομολογίων, έναν για κάθε μεταφορικό μέσο, και μέχρι να τους βρει συνεχίζει την αναζήτηση.

4.2 Δομή της εφαρμογής

Σύμφωνα με την περιγραφή της υλοποίησης και τον γενικό αλγόριθμο επίλυσης του προβλήματος έχει υλοποιηθεί η εφαρμογή, η δομή της οποίας παρουσιάζεται παρακάτω.

4.2.1 Αρχεία της εφαρμογής

Εδώ παρουσιάζονται τα εφτά αρχεία του απαρτίζουν την εφαρμογή, καθώς και η περιγραφή του περιεχομένου τους.

- **CoordinatorAgent.java:** Πρόκειται για το αρχείο που περιέχει την υλοποίηση του πράκτορα συντονιστή. Βασικές συμπεριφορές που εκτελεί είναι η findTransportAgents(), που αναζητά τους πράκτορες χειριστές με τους οποίους θα συνεργαστεί και η findRoute(), που υλοποιεί την ανταλλαγή μηνυμάτων με τους άλλους πράκτορες και την εύρεση του δρομολογίου.

- **CoordinatorGui.java:** Είναι το αρχείο όπου υλοποιείται το γραφικό περιβάλλον αλληλεπίδρασης του χρήστη με τον πράκτορα συντονιστή. Δημιουργεί ένα παράθυρο στο οποίο ο χρήστης δίνει τις τιμές της αφετηρίας και του προορισμού του δρομολογίου που επιθυμεί.

- **AirplaneHandlerAgent.java:** Περιέχει τον κώδικα του πράκτορα χειριστή δρομολογίων με αεροπλάνο. Οι λειτουργίες που εκτελεί είναι εγγραφή στην υπηρεσία καταλόγου και η συμπεριφορά findNextCityAirplane(), όπου αναζητά στη ΒΓ, δεδομένης μιας πόλης, μια πόλη που συνδέεται με αυτή και επιτρέπει τη μετακίνηση με αεροπλάνο.

- **BusHandlerAgent.java:** Περιέχει τον κώδικα του πράκτορα χειριστή δρομολογίων με λεωφορείο. Οι λειτουργίες που εκτελεί είναι εγγραφή στην υπηρεσία καταλόγου και η συμπεριφορά findNextCityBus(), όπου αναζητά στη ΒΓ, δεδομένης μιας πόλης, μια πόλη που συνδέεται με αυτή και επιτρέπει τη μετακίνηση με λεωφορείο.

- **TrainHandlerAgent.java:** Περιέχει τον κώδικα του πράκτορα χειριστή δρομολογίων με τρένο. Οι λειτουργίες που εκτελεί είναι εγγραφή στην υπηρεσία καταλόγου και η συμπεριφορά findNextCityTrain(), όπου αναζητά στη ΒΓ, δεδομένης μιας πόλης, μια πόλη που συνδέεται με αυτή και επιτρέπει τη μετακίνηση με τρένο.

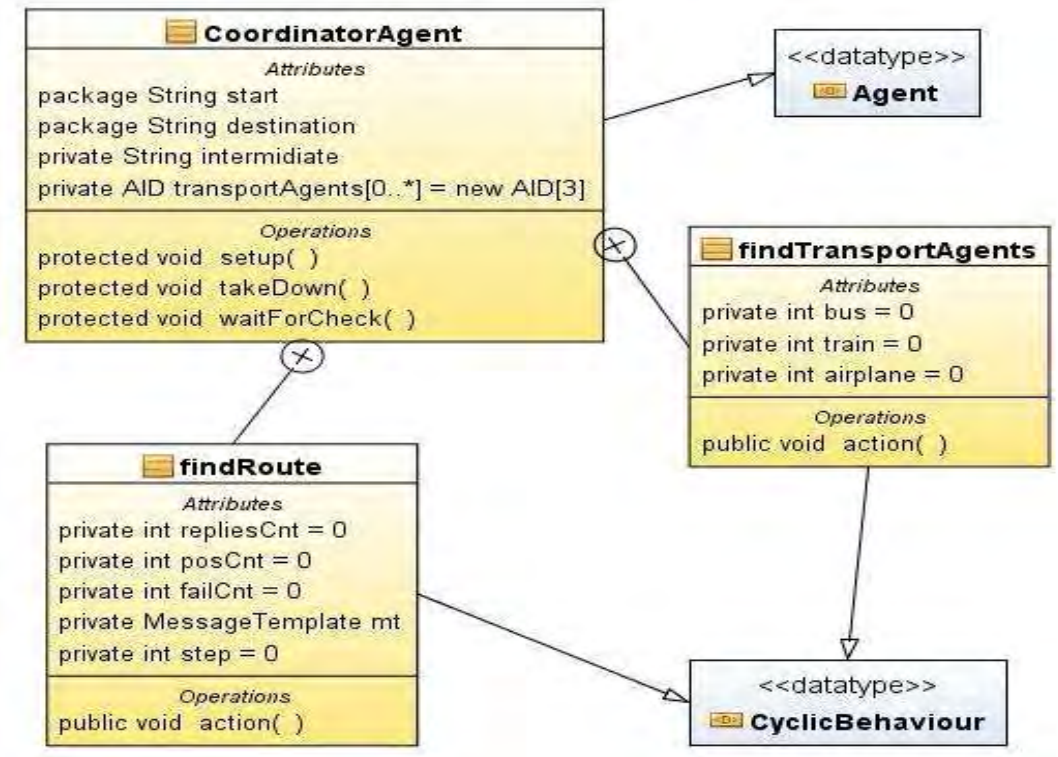
- **City.java:** Αποτελεί, μαζί με το αρχείο map.java, τη ΒΓ για το σύστημα. Περιέχει την κλάση City από την οποία δημιουργούνται τα αντικείμενα των πόλεων στο χάρτη.

- **Map.java:** Πρόκειται για το αρχείο που περιέχει τον κώδικα που δημιουργεί τον χάρτη στον οποίο γίνονται οι αναζητήσεις για την κατασκευή του ζητούμενου δρομολογίου.

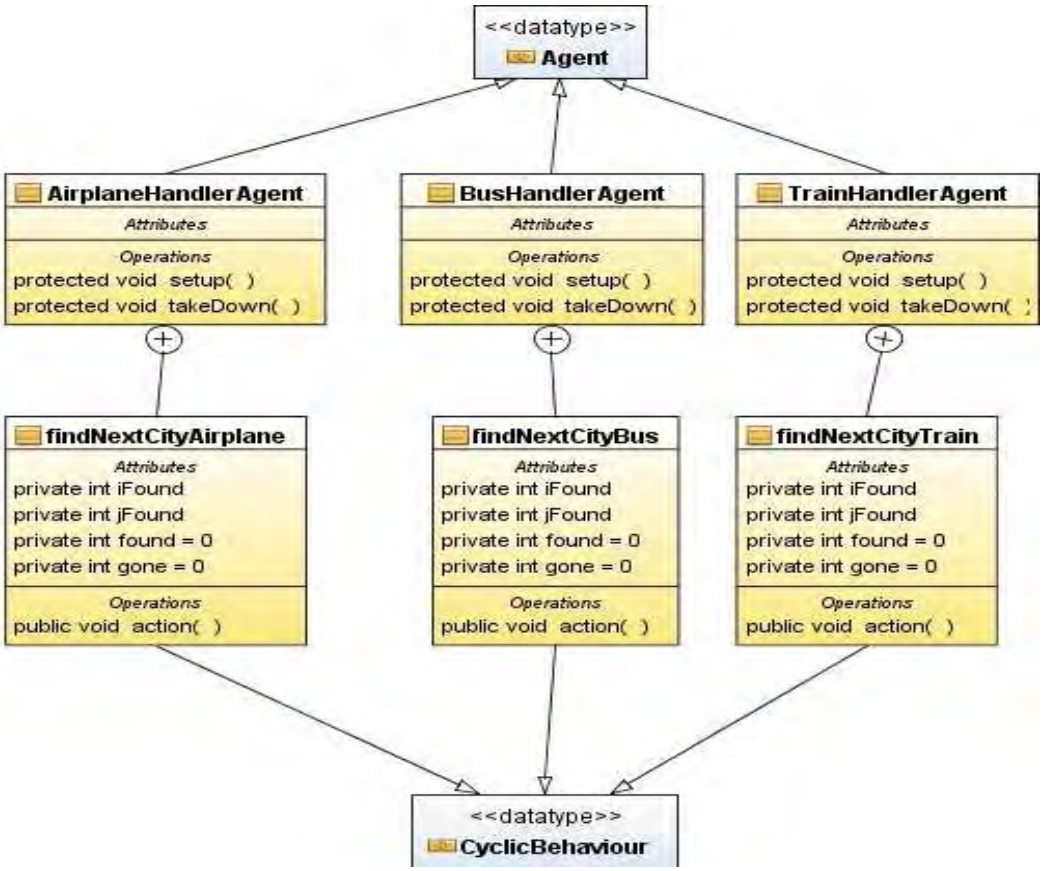
4.2.2 UML διαγράμματα της εφαρμογής

Είναι ιδιαιτέρως χρήσιμο για τον αναγνώστη, στην προσπάθεια του για την κατανόηση της δομής και της διαδικασίας υλοποίησης της εφαρμογής, η οπτικοποίηση της με τη χρήση της Ενοποιημένης Γλώσσας Μοντελοποίησης (Unified Model Language – UML).

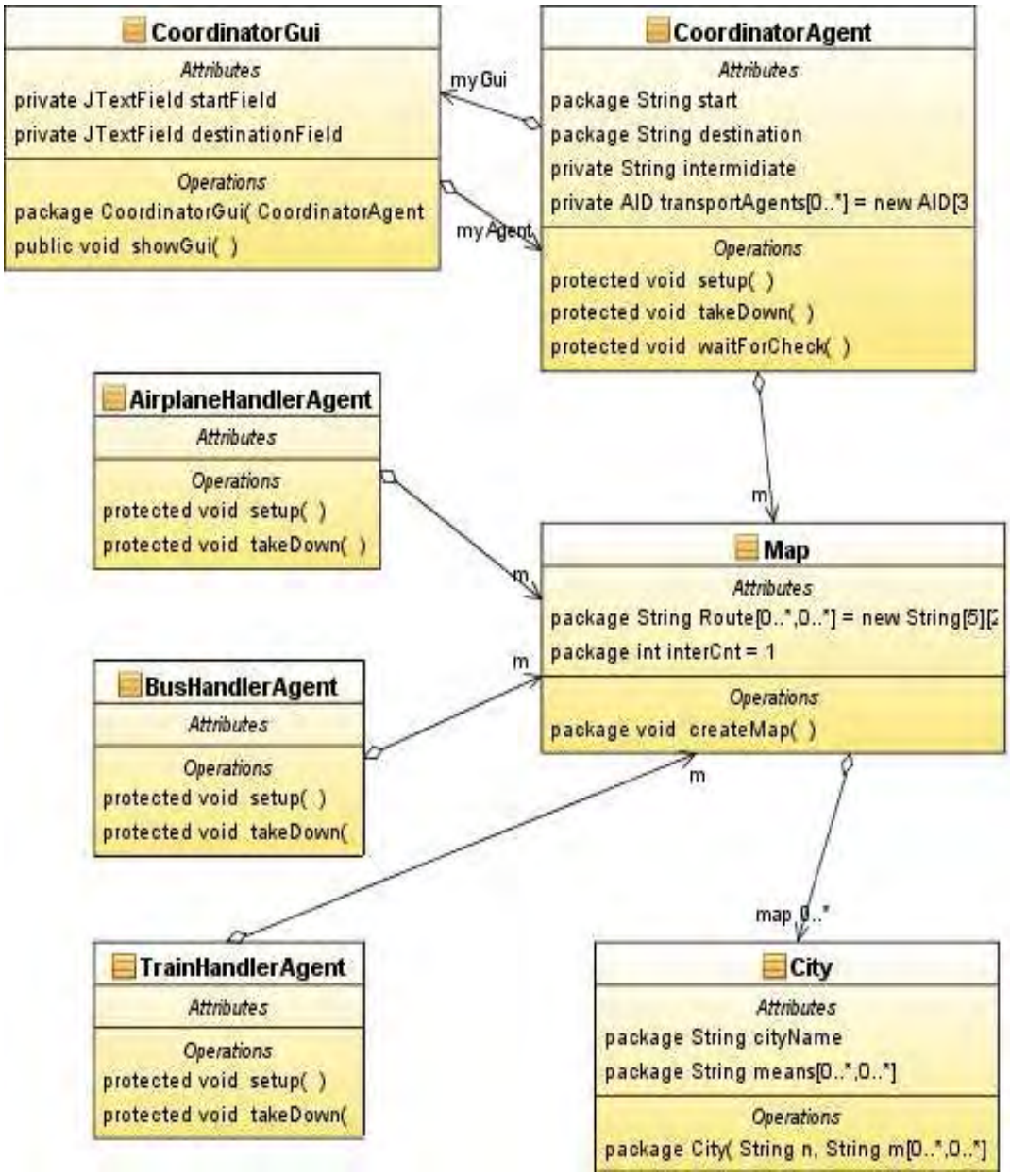
Στην ενότητα αυτή παρουσιάζονται τα UML διαγράμματα κλάσεων των αρχείων της εφαρμογής, όπως αυτά προέκυψαν από την ανάλυσή τους από το εργαλείο UML designer plugin της πλατφόρμας ανάπτυξης προγραμματιστικών εφαρμογών NetBeans IDE 7.2.



Εικόνα 4.1 Διάγραμμα κλάσεων του πράκτορα συντονιστή



Εικόνα 4.2 Διάγραμμα κλάσεων των πρακτόρων χειριστών



Εικόνα 4.3 Διάγραμμα κλάσεων όλης της εφαρμογής

4.3 Περιγραφή ροής εκτέλεσης

Για την πλήρη κατανόηση της εφαρμογής, και έχοντας ήδη περιγράψει το περιεχόμενο και τη δομή της, παρουσιάζεται στην παρούσα ενότητα όλη η ροή εκτέλεσής της. Η εκκίνηση της πλατφόρμας JADE γίνεται μέσω της γραμμής εντολών, από το φάκελο JADE, με την εντολή:

java -cp <jade lib\jade.jar;classes jade.Boot –gui

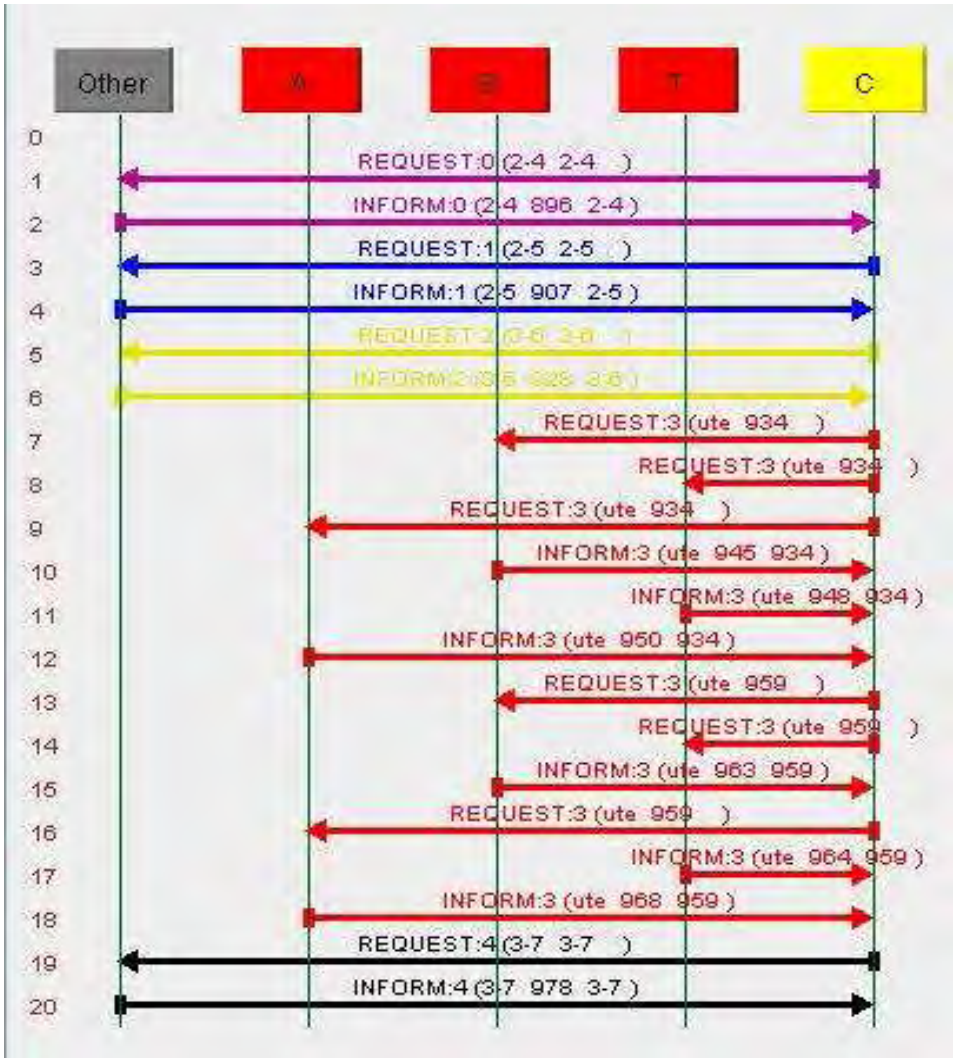
Στη συνέχεια γίνεται η εισαγωγή πρακτόρων στην πλατφόρμα, και αυτοί με τη σειρά τους εγγράφονται στην υπηρεσία καταλόγου του DF. Μόλις εκκινήσει ο πράκτορας συντονιστής, ανοίγει το παράθυρο αλληλεπίδρασης με το χρήστη

Εικόνα 4.4 Παράθυρο αλληλεπίδρασης με το χρήστη

Εφόσον εισαχθούν οι σωστές τιμές, ο συντονιστής αναζητά στην υπηρεσία καταλόγων τους πράκτορες που χρειάζεται για την επίλυση του προβλήματος.

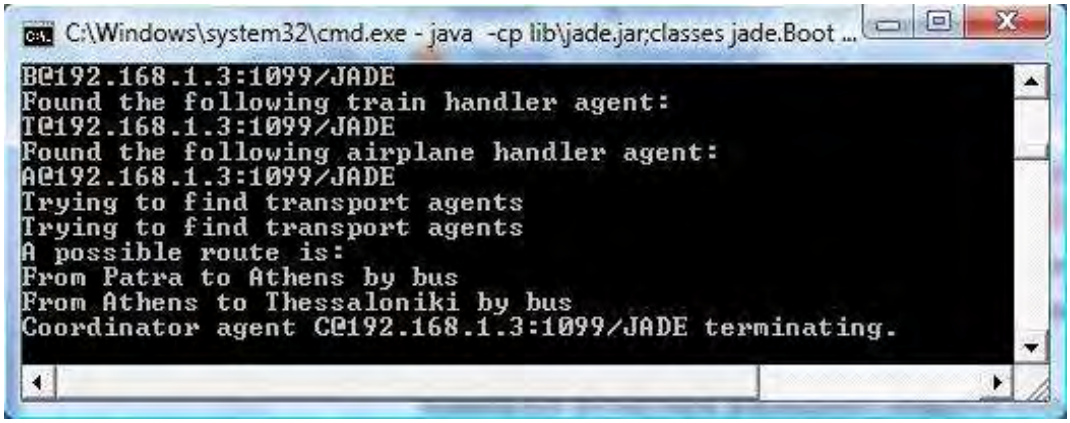
Εικόνα 4.5 Αναζήτηση πρακτόρων από το συντονιστή

Από τη στιγμή που σχηματίζεται η ομάδα επίλυσης, οι πράκτορες αρχίζουν την ανταλλαγή μηνυμάτων για την εύρεση και κατασκευή του δρομολογίου. Τα ανταλλασσόμενα μηνύματα μπορούμε να τα παρακολουθήσουμε μέσω του Sniffer Agent, όπως φαίνεται στην εικόνα 4.6.



Εικόνα 4.6 Ανταλλαγή μηνυμάτων ανάμεσα στους πράκτορες

Τέλος, παρουσιάζεται στο χρήστη η προτεινόμενη από το σύστημα διαδρομή ανάμεσα στις πόλεις που δόθηκαν και τερματίζεται ο πράκτορας συντονιστής.



Εικόνα 4.7 Εμφάνιση προτεινόμενης διαδρομής

Κεφάλαιο 5

Επίλογος

5.1 Συμπεράσματα

Μέσα από μια ενδελεχή καταγραφή και παρουσίαση, του θεωρητικού υπόβαθρου και των τεχνικών υλοποίησης του πολυπρακτορικού σχεδιασμού, καταλήξαμε στην μοντελοποίηση μέρους των τεχνικών αυτών μέσω της υλοποίησης της ανωτέρω εφαρμογής. Παρατηρούμε πως η εφαρμογή μοντελοποιεί αρκετά ικανοποιητικά της επιλεγόμενες τεχνικές, καθώς παρουσιάζει τον τρόπο με τον οποίο επιτυγχάνεται ο σχεδιασμός των ενεργειών των πρακτόρων, η επικοινωνία και ο συντονισμός τους, μέχρι την επίτευξη του στόχου τους.

Επιπλέον, οι επιλογές που έγιναν κατά τη σχεδίαση της εφαρμογής της δίνουν τα πλεονεκτήματα, αφενός, της συμβατότητας με διάφορα συστήματα λόγω της πλατφόρμας εκτέλεσης και υλοποίησης JADE και της JVM και αφετέρου, της επεκτασιμότητας της βάσης γνώσης.

Στα μειονεκτήματα της εφαρμογής μπορεί να καταγραφεί το γεγονός ότι λόγω της τυχαιότητας της αναζήτησης πληροφορίας στη ΒΓ και της αποδοχής των απαντήσεων των πρακτόρων χειριστών δρομολογίων, από τον πράκτορα συντονιστή, ενδέχεται να έχουμε επισκεφθεί όλες τις ενδιάμεσες πόλεις και να μην έχουμε φθάσει στον προορισμό. Το γεγονός αυτό οφείλεται, επίσης, στη μικρή για το παράδειγμά μας ΒΓ και στη μη ρεαλιστική της προσέγγιση. Είναι όμως ανεξάρτητο από την ανάγκη υλοποίησης της εφαρμογής για την εργασία αυτή, καθώς αυτή ήταν ο συντονισμός και η επικοινωνία των πρακτόρων, κάτι που και σε αυτήν την περίπτωση επιτυγχάνεται.

5.2 Μελλοντικές επεκτάσεις

Καθώς η συγκεκριμένη εφαρμογή υλοποιήθηκε για ερευνητικούς λόγους στα πλαίσια της παρούσας διπλωματικής εργασίας, προκύπτει η ανάγκη για μελλοντική επέκτασή της, έτσι ώστε να καλύπτει περισσότερο ρεαλιστικούς στόχους. Έτσι από μια ερευνητική εφαρμογή μπορεί εύκολα να μετατραπεί σε μια εφαρμογή για εμπορικούς λόγους, που θα καλύπτει την ανάγκη του επιβατικού κοινού για μετακινήσεις. Μερικοί από τους στόχους αυτούς είναι οι ακόλουθοι:

- Εξέταση της κλιμάκωσης της εφαρμογής, μέσω της κλιμάκωσης της ΒΓ
- Επέκταση της εφαρμογής για χρήση περισσοτέρων μεταφορικών μέσων
- Ρεαλιστική απεικόνιση της ΒΓ. Ο χάρτης μπορεί να περιλαμβάνει ένα πλήθος από πόλεις και τα πραγματικά δρομολόγια μεταφορικών μέσων που τις συνδέουν.
- Γραφική απεικόνιση των χαρτών της ΒΓ και πιθανή σύνδεσή τους με το Παγκόσμιο Σύστημα Εντοπισμού Θέσης (Global Positioning System – GPS)
- Αναζήτηση στις πληροφορίες της ΒΓ, για εύρεση επόμενης πόλης, βάση κριτηρίων όπως:
    - Ταχύτητα χρόνου μετάβασης
    - Κόστος μετάβασης
    - Συντομότερη χιλιομετρικά διαδρομή
    - Προτίμηση σε μεταφορικό μέσο

  Ο συνδυασμός όλων των παραπάνω, δηλαδή μια πολυκριτηριακή αναζήτηση, απαιτεί ανάθεση προτεραιοτήτων στα κριτήρια, γεγονός που, εντέλει, ανάγεται σε επιμέρους μονοκριτηριακές αναζητήσεις.

Βιβλιογραφία

[1] http://eprints.agentlink.org

[2] http://jade.tilab.com

[3] Durfee E. H., Planning in distributed artificial intelligence. In Foundation of Distributed Artificial Intelligence, 1996, pp 231-245

[4] Durfee E. H., Distributed problem solving and planning. In Multiagent Systems, MIT Press, Cambridge, 1999, pp 121-164

[5] Durfee E. H., Lesser V. R., Corkill D. D., Trends in cooperative distributed problem solving. In IEEE Transactions on Knowledge and Data Engineering, 1989, pp 63-83

[6] Georgeff M. P., Communication and interaction in multi-agent planning. In proceedings of the 3rd National Conference on Artificial Intelligence (AAAI-83), Washington DC, 1983, pp 125-129

[7] Jennings N. R., Controlling cooperative problem solving in industrial multi-agent systems using joint intensions. In Artificial Intelligence, 1995, pp 195-240

[8] Russell S., Norvig P., Artificial Intelligence A Modern Aproach, Kleidarithmos Publications, 2nd Edition, Athens, 2004, pp 64-169

[9] Vlahavas I., Kefalas P., Bassiliades N., Kokkoras F., Sakellariou I., Artificial Inteligence, V.Giurdas Publications, 3rd Edition, Athens, 2006, pp 609-630

[10] Weerdt de M., Clement B., Introduction to Planning in Multiagent Systems. In Multiagent and Grid Systems, IOS Press,2009, pp 345-355

[11] Wooldridge M., Jennings N. R., The cooperative problem solving process. In Journal of Logic and Computation, 1999, pp 563-592

[12] Wooldridge M, An Introduction to Multiagent Systems. In Kleidarithmos Publications, Athens, 2008