ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ & ΔΙΚΤΥΩΝ

Διπλωματική Εργασία

ΔΥΝΑΜΙΚΗ ΕΠΙΛΟΓΗ ΣΥΜΠΕΡΙΦΟΡΑΣ ΣΕ ΠΟΛΥΠΡΑΚΤΟΡΙΚΑ ΣΥΣΤΗΜΑΤΑ

Φοιτητής: ΔΗΜΟΣΘΕΝΗΣ ΜΠΕΚΑΣ
Επιβλέποντες: ΑΣΠΑΣΙΑ ΔΑΣΚΑΛΟΠΟΥΛΟΥ, ΣΠΥΡΟΣ ΛΑΛΗΣ

ΙΟΥΛΙΟΣ 2005, ΒΟΛΟΣ

ΠΡΟΛΟΓΟΣ

Η εργασία βασίστηκε στην πλατφόρμα ανάπτυξης πρακτόρων JADE ενώ πραγματεύεται την:

- σχεδίαση και ανάπτυξη μιας κατάλληλης πολυπρακτορικής πλατφόρμας παροχής ετερογενών υπηρεσιών με χρήση οντολογιών για αναπαράσταση του ανοικτού περιβάλλοντος (μη-προσιτό, μη-ντετερμινιστικό, δυναμικό και συνεχές) και μηχανισμών δυναμικής επιλογής και εκτέλεσης συμπεριφορών για τον καθορισμό της πολυκριτηριακής διαπραγμάτευσης μεταξύ των πρακτόρων η οποία διεξάγεται λαμβάνοντας υπόψιν την τρέχουσα κατάσταση του περιβάλλοντος και τις προτιμήσεις των χρηστών
- ανάπτυξη, με βάση αυτή την πλατφόρμα, μιας (ενδεικτικής) εφαρμογής εύρεσης ταξί όπου οι τακτικές και τα πρωτόκολλα διαπραγμάτευσης επιλέγονται και αλλάζουν δυναμικά

Η σπουδαιότητα της υλοποίησης έγκειται στην επίλυση των προβλημάτων των υπαρχόντων πολυπρακτορικών συστημάτων όπως η παροχή μιας μόνο υπηρεσίας, η διαπραγμάτευση απλού ζητήματος, η υποβάθμιση της δυναμικής επιλογής συμπεριφοράς σε απλή επιλογή του πρωτοκόλλου διαπραγμάτευσης, η προκαθορισμένη λίστα πρωτοκόλλων, η μη δυνατότητα ανάπτυξης νέων συμπεριφορών από ανεξάρτητους προγραμματιστές και η μη δυνατότητα δυναμικής αναβάθμισης λογισμικού.

Στην πρώτη ενότητα γίνεται μια συνοπτική εισαγωγή στους πράκτορες, τα πολυπρακτορικά συστήματα και στην πλατφόρμα ανάπτυξης πρακτόρων που χρησιμοποιήθηκε. Στην δεύτερη ενότητα δίνεται η μοντελοποίηση της πλατφόρμας και το αφηρημένο αρχιτεκτονικό σχέδιο του συστήματος που περιγράφει τις οντότητες, το περιβάλλον και τις λειτουργίες του σε γενικό επίπεδο ανεξαρτήτως εφαρμογής. Στην τρίτη ενότητα δίνεται το εξειδικευμένο σχέδιο του συστήματος σχετικά με την υλοποίηση της υπηρεσίας εύρεσης ταξί και περιγράφονται, σε επίπεδο εφαρμογής, οι οντότητες, το περιβάλλον και οι λειτουργίες της. Στην τέταρτη ενότητα αναφέρονται τα συμπεράσματα που αποκομίσθηκαν από την εκπόνηση της συγκεκριμένης διπλωματικής εργασίας ενώ στην πέμπτη ενότητα αναφέρονται οι προοπτικές του συστήματος για μελλοντικές επεκτάσεις.

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ ΣΕ ΠΡΑΚΤΟΡΕΣ

Ο τομέας των έξυπνων πρακτόρων είναι από τους πιο δραστήριους και ενδιαφέροντες κλάδους έρευνας και ανάπτυξης στην Τεχνολογία της Πληροφορίας. Ανήκουν στην τρίτη γενιά ανάπτυξης των συστημάτων λογισμικού [Parunak, 1999] και αποτελούν υπολογιστικά, αλληλεπιδραστικά συστήματα ικανά για ευέλικτες, αυτόνομες ενέργειες σε ανοικτά και πολυπρακτορικά περιβάλλοντα.

1.1 Ορισμοί και Χαρακτηριστικά Πρακτόρων

Υπάρχουν πολλές κατηγοριοποιήσεις και ορισμοί για έξυπνους πράκτορες [Franklin & Graesser, 1997]. Στα πλαίσια αυτής της εργασίας αρκούμαστε στην κατηγορία πρακτόρων λογισμικού, και στον ευρέως αποδεκτό ορισμό του Wooldridge [Wooldridge & Jennings, 1995], σύμφωνα με τον οποίο: *έξυπνος πράκτορας είναι ένα υπολογιστικό σύστημα που βρίσκεται σε ένα περιβάλλον και είναι ικανό για αυτόνομη δράση μέσα σε αυτό προκειμένου να ικανοποιήσει τους σχεδιαστικούς του σκοπούς.*

Τα κύρια χαρακτηριστικά ενός έξυπνου πράκτορα αναλύονται με τον παρακάτω τρόπο στο [Wooldridge & Jennings, 1995]:

- *Αυτονομία*: Ένας έξυπνος πράκτορας έχει τους δικούς του στόχους και μπορεί να λειτουργεί χωρίς την παρέμβαση των χρηστών ή άλλων πρακτόρων.
- *Προνοητικότητα*: Ένας έξυπνος πράκτορας δεν αντιδρά απλώς στο περιβάλλον αλλά είναι ικανός να λαμβάνει πρωτοβουλίες για την επίτευξη των στόχων του, ανάλογα με τις συνθήκες οι οποίες εμφανίζονται στο περιβάλλον του.
- *Αντιδραστικότητα*: Ένας έξυπνος πράκτορας αντιλαμβάνεται το περιβάλλον του και αντιδράει μέσα σε λογικά και συγκεκριμένα χρονικά πλαίσια στις αλλαγές που επέρχονται σε αυτό.
- *Κοινωνικότητα*: Ένας έξυπνος πράκτορας είναι σε θέση να αλληλεπιδρά με άλλους πράκτορες μέσω μιας κοινά κατανοητής γλώσσας ώστε να μπορεί να συνεργαστεί για να επιτύχει τους στόχους του.

Να σημειωθεί ότι στους πράκτορες αποδίδονται και άλλα δευτερεύοντα χαρακτηριστικά τα οποία όμως δεν εμφανίζονται σε όλες τις κατηγορίες πρακτόρων. Κάποια από αυτά είναι η προσαρμοστικότητα, η ελαστικότητα και η κινητικότητα.

1.2 Περιβάλλοντα Πρακτόρων

Τα περιβάλλοντα πρακτόρων κατηγοριοποιούνται με τον παρακάτω τρόπο στο [Russell & Norvig, 1995]:

- *Προσιτά / Μη-Προσιτά* (Accessible / Inaccessible) ανάλογα με το εάν υπάρχει διαθέσιμη πλήρης, ακριβή και ανανεωμένη πληροφορία σχετικά με την κατάσταση του περιβάλλοντος. Τα περισσότερα περιβάλλοντα, όπως αυτά του πραγματικού κόσμου και του Διαδικτύου, είναι μη-προσιτά.
- *Ντετερμινιστικά / Μη-Ντετερμινιστικά* (Deterministic / Non-Deterministic) αναφορικά με το εάν μια συγκεκριμένη ενέργεια έχει ένα μοναδικό εγγυημένο αποτέλεσμα και συνεπώς δεν υπάρχει αβεβαιότητα σχετικά με την κατάσταση στην οποία θα εισέλθει το σύστημα από την εκτέλεση μιας ενέργειας. Όλα τα πραγματικά περιβάλλοντα πρέπει να θεωρούνται μη-ντετερμινιστικά.
- *Στατικά / Δυναμικά* (Static / Dynamic) σε σχέση με το εάν ένα περιβάλλον μεταβάλλεται μόνο από τις ενέργειες των πρακτόρων που δρουν μέσα σε αυτό ή μεταβάλλεται και από άλλες διαδικασίες και συστήματα του περιβάλλοντος. Ο πραγματικός κόσμος και το Διαδίκτυο αποτελούν υψηλά δυναμικά περιβάλλοντα.
- *Διακριτά / Συνεχή* (Discrete / Continuous) ως προς το εάν υπάρχει ή όχι ένας διακριτός, πεπερασμένος αριθμός καταστάσεων του περιβάλλοντος. Ένα διακριτό περιβάλλον είναι αυτό που μπορεί να εγγυηθεί την μετάβασή του σε ένα πεπερασμένο σύνολο δυνατών καταστάσεων ενώ ένα συνεχές μπορεί να βρεθεί σε μη μετρήσιμες καταστάσεις.

Η συγκεκριμένη διπλωματική μοντελοποιεί ανοικτά περιβάλλοντα, δηλαδή μη-προσιτά, μη-ντετερμινιστικά, δυναμικά και συνεχή.

1.3 Αρχιτεκτονικές Πρακτόρων

Οι αρχιτεκτονικές πρακτόρων είναι οι βασικές μηχανές που μοντελοποιούν τον τρόπο λειτουργίας των πρακτόρων στα πραγματικά και ανοικτά περιβάλλοντα καθώς και τον τρόπο της μεταξύ τους αλληλεπίδρασης.

Στην αφηρημένη αρχιτεκτονική πρακτόρων, το περιβάλλον βρίσκεται αρχικά σε μια δεδομένη κατάσταση και ο πράκτορας επιλέγει μια ενέργεια, από το σύνολο των δυνατών ενεργειών, την οποία εκτελεί με αποτέλεσμα το περιβάλλον να μεταβεί σε

μια νέα, εφικτή κατάσταση. Να σημειωθεί ότι παρόλο που για κάθε τρέχουσα κατάσταση και αντίστοιχη ενέργεια μπορεί να υπάρχουν πολλές επόμενες εφικτές καταστάσεις, το περιβάλλον μπορεί να μεταβεί μόνο σε μια κατάσταση ενώ ο πράκτορας δεν μπορεί να γνωρίζει εκ των προτέρων ποια είναι αυτή.

Ωστόσο, η αφηρημένη αρχιτεκτονική δεν αρκεί για να κατασκευάσουμε πράκτορες διότι απαιτείται να γίνει περισσότερο αναλυτική (δομές δεδομένων, πράξεις εκτέλεσης επί αυτών, ροή ελέγχου ανάμεσα στις δομές). Για αυτό τον λόγο έχουν αναπτυχθεί και άλλες αρχιτεκτονικές, οι σημαντικότερες εκ των οποίων είναι οι πράκτορες με εσωτερική αναπαράσταση, οι πράκτορες με πεποιθήσεις – επιθυμίες – προθέσεις, οι αντιδραστικοί πράκτορες, οι υβριδικοί και οι κινητοί πράκτορες. Να τονισθεί ότι οι παραπάνω δεν αποτελούν ανεξάρτητες αρχιτεκτονικές αλλά διαφορετικές προσεγγίσεις και υλοποιήσεις της αφηρημένης αρχιτεκτονικής. Περισσότερες πληροφορίες για τις αρχιτεκτονικές πρακτόρων μπορείτε να βρείτε στο [Wooldridge, 2002].

1.4 Πολυπρακτορικά Συστήματα

Οι πιο πρόσφατες προσπάθειες στον τομέα των πρακτόρων έχουν οδηγήσει σε μοντέλα ομαδικής δραστηριότητας στα οποία οι πράκτορες συνεργάζονται προς την επίτευξη συγκεκριμένων στόχων. Ένα πολυπρακτορικό σύστημα (Multi-Agent System) είναι ένα σύστημα που σχεδιάστηκε και υλοποιήθηκε ως ένα δίκτυο πρακτόρων που αλληλεπιδρούν, δηλαδή συνεργάζονται, συνεννοούνται, επικοινωνούν και διαπραγματεύονται. Επίσης στοχεύει στην επίλυση πολύπλοκων ή κατανεμημένων προβλημάτων και στη διασύνδεση και λειτουργία ήδη υπαρχόντων συστημάτων ώστε να είναι εύκολη η εκμετάλλευσή τους. Στα συστήματα αυτά, οι πράκτορες είτε εργάζονται αυτόνομα ανταλλάσσοντας πληροφορίες και προσπαθώντας να επιτύχουν τους δικούς τους ανεξάρτητους στόχους εμφανίζοντας μια ανταποδοτική (reciprocal) σχέση εξάρτησης είτε συνεργάζονται επιλύνοντας υποπροβλήματα έτσι ώστε ο συνδυασμός των επιμέρους λύσεων που θα προκύψουν να αποτελέσει την τελική λύση εμφανίζοντας μια αμοιβαία (mutual) σχέση εξάρτησης. Κύριο χαρακτηριστικό των συνεργαζόμενων πρακτόρων είναι η δυνατότητα διαπραγμάτευσης μέσω κάποιας γλώσσας επικοινωνίας ώστε να φθάσουν οι πράκτορες σε κοινά αποδεκτές συμφωνίες επιλύνοντας ενδεχόμενες συγκρούσεις.

Στη σχεδίαση και υλοποίηση ενός πολυπρακτορικού συστήματος υπάρχουν μερικά κρίσιμα σημεία στα οποία θα πρέπει να δοθεί ιδιάζουσα σημασία ώστε να βρεθεί ο αποτελεσματικότερος τρόπος αντιμετώπισής τους. Τα προβλήματα αυτά αφορούν κυρίως την επικοινωνία των πρακτόρων, τον τρόπο συνεργασίας τους αλλά και την δυνατότητα μάθησης που θα κατέχουν ώστε να μπορούν να προσαρμόζονται στις δυναμικές αλλαγές του περιβάλλοντος.

Σύμφωνα με το [Luck et al., 2003], τα πολυπρακτορικά συστήματα της παρούσας χρονικής φάσης (2003-2005) εμφανίζουν τα παρακάτω χαρακτηριστικά:

- Σχεδιάζονται διασυνεταιρικά, ώστε οι συμμετέχοντες πράκτορες να έχουν λιγότερους κοινούς στόχους αν και οι αλληλεπιδράσεις τους αφορούν ένα κοινό τομέα εφαρμογής.
- Όλοι οι συμμετέχοντες πράκτορες σχεδιάζονται από την ίδια ομάδα και μοιράζονται την βάση γνώσης.
- Χρησιμοποιούνται όλο και περισσότερο οι προτυποποιημένες γλώσσες επικοινωνίας όπως η FIPA ACL αλλά τα πρωτόκολλα διαπραγμάτευσης παραμένουν μη-προτυποποιημένα.
- Είναι σε θέση να υποστηρίζουν μεγάλο αριθμό συμμετεχόντων πρακτόρων αλλά σε προκαθορισμένα περιβάλλοντα όπως αυτά του υπολογιστικού πλέγματος (grid computing)

Σύμφωνα πάλι με το [Luck et al., 2003], τα πολυπρακτορικά συστήματα του βραχυπρόθεσμου μέλλοντος (2006-2009) εμφανίζουν τα παρακάτω χαρακτηριστικά:

- Επιτρέπουν συμμετοχή ετερογενών πρακτόρων σχεδιασμένων από διαφορετικές ομάδες και σχεδιαστές. Κάθε πράκτορας θα μπορεί να συμμετέχει στο σύστημα δεδομένου ότι η συμπεριφορά του συμβαδίζει με τις απαιτήσεις και τις προϋποθέσεις του συστήματος.
- Οι γλώσσες και τα πρωτόκολλα που χρησιμοποιούνται στα συστήματα θα είναι προσυμφωνημένα και προτυποποιημένα και ίσως θα ανακτώνται από κοινές βιβλιοθήκες, οι οποίες θα διαφέρουν ανάλογα με τον τομέα εφαρμογής. Για παράδειγμα, οι οντολογίες θα απαιτούνται για την διευθέτηση της σημασιολογικής ετερογένειας.
- Είναι κλιμακωτά και επιτρέπουν την δυναμική διείσδυση ή έξοδο πρακτόρων από το σύστημα.

- Γίνεται γεφύρωση των πρακτόρων όπου θα είναι εφικτή η επικοινωνία μεταξύ πρακτόρων από διαφορετικούς επιστημονικούς τομείς.
- Η ανάπτυξη συστημάτων προχωράει καθορίζοντας τυποποιημένες σχεδιαστικές μεθοδολογίες συμπεριλαμβανομένου προτύπων για διαφορετικού είδους πρακτόρων και ειδών πρακτορικών συστημάτων. Ζητήματα σημασιολογίας, όπως ο συντονισμός ανάμεσα σε ετερογενής πράκτορες και ο έλεγχος πρόσβασης, είναι ιδιάζουσας σημασίας.

Αξίζει να σημειωθεί ότι η εργασία αφορά άμεσα την επίλυση των προβλημάτων της παρούσας φάσης ανάπτυξης πρακτόρων καθώς μέσα από την ανοιχτή αρχιτεκτονική της πλατφόρμας που αναπτύχθηκε υποστηρίζεται η σχεδίαση και υλοποίηση πρακτόρων από διαφορετικές ομάδες ανθρώπων, με διαφορετικά πρωτόκολλα αλληλεπίδρασης σε δυναμικά περιβάλλοντα και με δυναμικές συμπεριφορές.

1.5 Επικοινωνία και Διαπραγμάτευση Πρακτόρων

Στην βιβλιογραφία έχουν προταθεί διάφοροι μηχανισμοί επικοινωνίας για τη συνεργασία ανάμεσα σε πράκτορες. Οι κυριότεροι μηχανισμοί είναι:

- ανταλλαγή μηνυμάτων (message passing)
- επικοινωνία μέσω μαυροπίνακα (shared blackboard)

Στην ανταλλαγή μηνυμάτων οι πράκτορες επικοινωνούν μέσω «προσωπικών» μηνυμάτων τα οποία αποστέλλουν ο ένας απευθείας στον άλλο. Αντίθετα, στην επικοινωνία μέσω μαυροπίνακα υπάρχει ένας κοινός χώρος για όλους τους πράκτορες όπου αποθηκεύονται / δημοσιοποιούνται μηνύματα και τα οποία είναι ορατά και προσπελάσιμα από όλους τους πράκτορες του συστήματος.

Σε ψηλότερο επίπεδο, ανεξάρτητα από τον μηχανισμό επικοινωνίας, πρέπει να υπάρχει επιπλέον συμφωνία για το λεγόμενο πρωτόκολλο επικοινωνίας μέσω του οποίου καθορίζεται ο τύπος, η μορφή, η σημασία και το περιεχόμενο των μηνυμάτων που ανταλλάσσουν οι πράκτορες, καθώς και οι επιτρεπτές αλληλεπιδράσεις (ποιο μήνυμα μπορεί να σταλεί ως απάντηση σε κάποιο άλλο, τι μορφή και περιεχόμενο πρέπει να έχει). Για παράδειγμα ένα πρωτόκολλο επικοινωνίας είναι δυνατό να περιλαμβάνει διαφορετικούς τύπους μηνυμάτων για την ανταλλαγή πληροφορίας που αφορά προτάσεις, αποδοχές, απορρίψεις, αποσύρσεις, διαφωνίες και ενέργειες.

Παρότι διάφορα συστήματα πρακτόρων παρέχουν εγγενή υποστήριξη για διάφορα πρωτόκολλα επικοινωνίας, τις περισσότερες φορές η σημασιολογία της επικοινωνίας πρέπει να οριστεί τελικά σε επίπεδο εφαρμογής, π.χ. μέσα από οντολογίες που αναπτύσσει ο σχεδιαστής της.

Μια ειδική κατηγορία πρωτοκόλλων επικοινωνίας πρακτόρων είναι τα πρωτόκολλα διαπραγμάτευσης [Rosenschein & Zlotkin, 1994]. Ένα πρωτόκολλο διαπραγμάτευσης δίνει συνήθως τη δυνατότητα στους πράκτορες να έχουν συζητήσεις, δηλαδή ακολουθίες ανταλλαγής μηνυμάτων με απώτερο σκοπό τη συνεργασία των πρακτόρων. Για παράδειγμα ένα πρωτόκολλο αλληλεπίδρασης που χρησιμοποιείται σε καταστάσεις διαπραγματεύσεων μεταξύ πρακτόρων, θα μπορούσε να ορίζει ότι όταν ένας πράκτορας προτείνει μια σειρά ενεργειών σε έναν άλλο πράκτορα, ο δεύτερος αφού αξιολογήσει την πρόταση θα πρέπει να απαντήσει με ένα μήνυμα αποδοχής, απόρριψης ή διαφωνίας.

Διαχωρίζουμε τα σενάρια διαπραγμάτευσης ανάμεσα σε αυτά των πολλαπλών ζητημάτων (multiple issues) και του απλού ζητήματος (single-issue). Ένα παράδειγμα διαπραγμάτευσης απλού ζητήματος είναι όταν δύο πράκτορες διαπραγματεύονται για την τιμή ενός συγκεκριμένου προϊόντος. Σε αυτή τη περίπτωση οι προτιμήσεις των πρακτόρων είναι συμμετρικές (symmetric) με την έννοια ότι η συμφωνία που είναι περισσότερη αρεστή για έναν πράκτορα, σίγουρα είναι λιγότερο ευνοϊκή για τον άλλο και το αντίστροφο. Στα σενάρια διαπραγμάτευσης πολλαπλών ζητημάτων οι πράκτορες διαπραγματεύονται για τις τιμές πολλαπλών ιδιοτήτων που μπορεί κιόλας να είναι αλληλοσυσχετιζόμενες. Σε αυτή τη περίπτωση είναι συνήθως λιγότερο προφανές τι σημαίνει και πως ορίζεται μια αληθινή υποχώρηση, διότι δεν είναι απλό να καθοριστούν (α) οι παράμετροι που πρέπει να αυξηθούν ή να ελαττωθούν και κατά πόσο, (β) ποιες είναι οι συσχετίσεις ανάμεσα στις παραμέτρους αυτές και (γ) ποια η βαρύτητα και η σπουδαιότητα της κάθε ιδιότητας. Επιπλέον, οι πολλαπλές ιδιότητες που πρέπει να ληφθούν υπόψιν σε μια διαπραγμάτευση, συνήθως οδηγούν σε μια εκθετική αύξηση του χώρου των δυνατών συμφωνιών. Οι συμμετέχοντες στην διαπραγμάτευση μπορεί να δυσκολεύονται ακόμα και να συμφωνήσουν ποιες είναι οι ιδιότητες ή τα ζητήματα προς διαπραγμάτευση.

Υπάρχουν διάφορα πρωτόκολλα διαπραγμάτευσης που σε γενικές γραμμές προσπαθούν να εγγυηθούν τις εξής ιδιότητες: (α) την αποφυγή καταστάσεων χάους,

(β) την ικανοποίηση καθολικών περιορισμών, (γ) την εκμετάλλευση κατανεμημένης εμπειρογνωμοσύνης και (δ) την αύξηση της αποδοτικότητας. Τα σημαντικότερα και δημοφιλέστερα πρωτόκολλα διαπραγμάτευσης αναλύονται στο [Smith, 1977; Smith & Davis, 1980] και είναι τα (α) Contract Net Protocol (CNP), (β) Contract Net With Confirmation Protocol (CNCP) και (γ) Holonic Contract Net With Confirmation Protocol (HCNCP) όπου τα δύο τελευταία αποτελούν επέκταση του CNP.

1.6 Πλατφόρμα Ανάπτυξης Πρακτόρων JADE

Για την εργασία επιλέχθηκε η πλατφόρμα του JADE. Παρακάτω περιγράφονται τα κυριότερα χαρακτηριστικά της ενώ πληροφορίες για περισσότερες πλατφόρμες ανάπτυξης πρακτόρων υπάρχουν στο [Mangina, 2002].

Το JADE (Java Agent DEvelopment) είναι μια εφαρμογή ενδιάμεσου στρώματος (middleware), που αναπτύχθηκε πλήρως σε Java από την Telecom Italia Lab, για την ανάπτυξη κατανεμημένων πολυπρακτορικών εφαρμογών βασισμένων στην peer-to-peer αρχιτεκτονική επικοινωνίας. Το JADE απλοποιεί την υλοποίηση και εκτέλεση πολυπρακτορικών εφαρμογών ενώ παράλληλα:

- υποστηρίζει την δυναμική εξέλιξη του πρακτορικού περιβάλλοντος επιτρέποντας την δυναμική είσοδο και έξοδο των πρακτόρων από το σύστημα ανάλογα με τις ανάγκες και απαιτήσεις της εφαρμογής
- υποστηρίζει την επικοινωνία και αλληλεπίδραση των πρακτόρων σε ενσύρματα και ασύρματα περιβάλλοντα
- υποστηρίζει και διαχειρίζεται μέσω γραφικού περιβάλλοντος τον κύκλο ζωής του εκάστοτε πράκτορα
- υποστηρίζει υπηρεσίες καταλόγου (white, yellow pages) για την εύρεση πρακτόρων και εφαρμογών
- συμβαδίζει με τις προδιαγραφές της FIPA
- παρέχει ένα σύνολο γραφικών εργαλείων που υποστηρίζουν την φάση της εκσφαλμάτωσης, διαχείρισης και υλοποίησης της εφαρμογής
- υποστηρίζει διάφορα πρωτόκολλα επικοινωνίας (FIPA ACL) καθώς και πρωτόκολλα διαπραγμάτευσης (Contract Net)
- υποστηρίζει την δημιουργία και διαχείριση μηνυμάτων περιεχομένου συμπεριλαμβανομένου της XML και RDF

- υποστηρίζει την μετακίνηση των πρακτόρων (κώδικα, κατάσταση εκτέλεσης) σε διαφορετικές τοποθεσίες (containers, platforms)

Η πλατφόρμα αυτή μπορεί να είναι κατανεμημένη σε μηχανήματα με διαφορετικό λειτουργικό σύστημα, διαφορετική υποδομή δικτύου ή και διαφορετική έκδοση της Java (το JADE παρέχει το ίδιο API για τα J2EE, J2SE και J2ME περιβάλλοντα) ενώ η διαμόρφωση (configuration) ή η παραμετροποίηση μπορεί να γίνει μέσω ενός απομακρυσμένου GUI. Η διαμόρφωση μπορεί επίσης να λαμβάνει χώρα σε πραγματικό χρόνο μετακινώντας τους πράκτορες από ένα μηχάνημα σε άλλο, όταν και όποτε αυτό απαιτείται.

Στο [Chmiel et al., 2004] αναφέρονται τα πειράματα που έγιναν στο JADE για να δοκιμάσουν την αποτελεσματικότητα της πλατφόρμας. Το συμπέρασμα που προέκυψε είναι ότι το JADE αποτελεί ένα ισχυρό περιβάλλον του οποίου οι δυνατότητες περιορίζονται από τη Java ενώ το ίδιο δεν εισάγει ουσιαστικούς περιορισμούς. Επίσης είναι ικανό να υποστηρίξει την μετακίνηση χιλιάδων πρακτόρων ανάμεσα σε διάφορα μηχανήματα καθώς και την ανταλλαγή δεκάδων χιλιάδων ACL μηνυμάτων ανάμεσα σε πράκτορες. Τέλος, μια αύξηση στο πλήθος των πρακτόρων προκαλεί γραμμική αύξηση στο χρόνο επεξεργασίας.

Όσον αφορά την αρχιτεκτονική των πρακτόρων, πρέπει να σημειωθεί ότι οι πράκτορες της πλατφόρμας του συστήματος δεν βασίζονται σε καμία από τις γνωστές αρχιτεκτονικές πρακτόρων, δηλαδή δεν έχουν εσωτερική κατάσταση, δεν είναι αντιδραστικοί ούτε και υβριδικοί. Η επιλογή της αρχιτεκτονικής περιορίζεται από την πλατφόρμα του JADE η οποία συμβαδίζει με τις προδιαγραφές της FIPA Agent System αρχιτεκτονικής [Bellifemine et al., 1999].

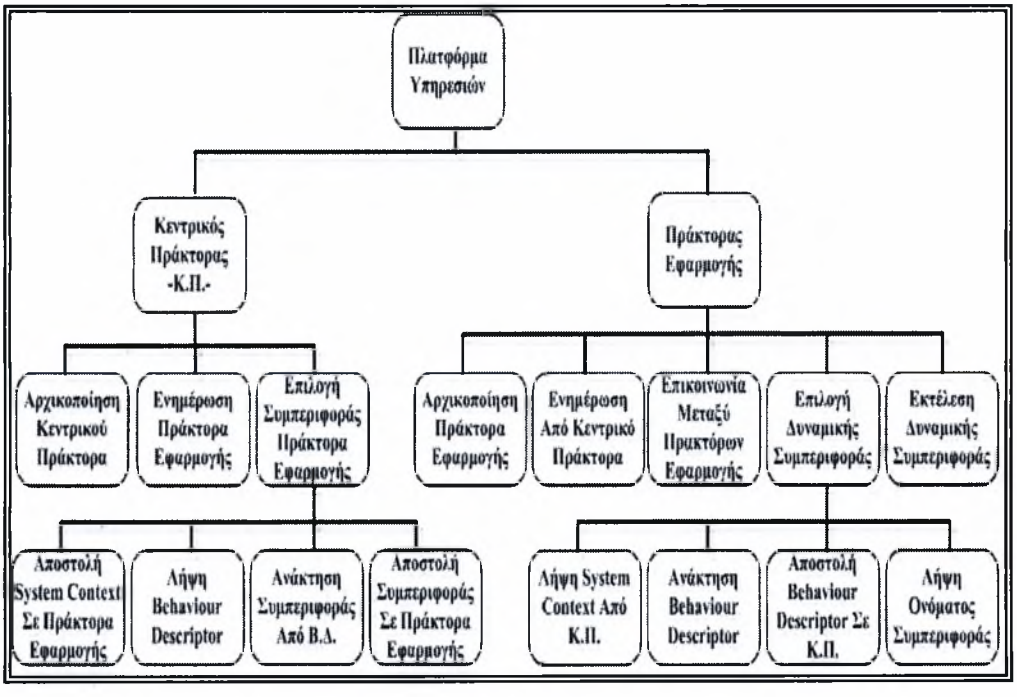
2. ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΟΣ ΔΥΝΑΜΙΚΗΣ ΕΠΙΛΟΓΗΣ ΣΥΜΠΕΡΙΦΟΡΩΝ

Υποθέτουμε ότι κάθε εφαρμογή αποτελείται από διάφορους πράκτορες που αλληλεπιδρούν μεταξύ τους για την παροχή της επιθυμητής λειτουργικότητας. Οι έννοιες καθώς και τα μηνύματα που αφορούν την εκάστοτε εφαρμογή ορίζονται μέσα από κατάλληλες οντολογίες που γνωρίζουν και χρησιμοποιούν οι αντίστοιχοι πράκτορες. Επίσης, η αλληλεπίδραση μεταξύ των πρακτόρων ρυθμίζεται μέσα από συμπεριφορές της εκάστοτε εφαρμογής που βρίσκονται αποθηκευμένες στο σύστημα και τις οποίες μπορούν να επιλέγουν και να ενεργοποιούν δυναμικά οι πράκτορες.

Με βάση αυτό το μοντέλο, η γενική πλατφόρμα (CAMP για Context-Aware Multi-agent Platform) καλείται να ενοποιήσει το σύνολο των ετερογενών εφαρμογών χωρίς να παρεμβαίνει στην (ιδιαίτερη) αλληλεπίδραση μεταξύ των πρακτόρων που τις υλοποιούν.

2.1 Διάγραμμα Αποσύνθεσης Λειτουργιών

Το διάγραμμα αποσύνθεσης λειτουργιών (functional decomposition diagram) περιγράφει τις λειτουργίες και τις οντότητες της πλατφόρμας.



Σχήμα 2.1: Διάγραμμα Αποσύνθεσης Λειτουργιών Συστήματος

Από το παραπάνω διάγραμμα παρατηρείται η αλληλεπίδραση μεταξύ δύο ειδών πρακτόρων εκ των οποίων ο ένας μοντελοποιεί τον πράκτορα συστήματος (κεντρικός πράκτορας) και ο άλλος τον πράκτορα εφαρμογής, ο οποίος εξειδικεύεται ανάλογα με την εφαρμογή. Η αρχιτεκτονική της πλατφόρμας εξηγείται παρακάτω.

2.2 Κεντρικός Πράκτορας

Ο κεντρικός πράκτορας δρα ως κεντρικός διαχειριστής για την αποθήκευση και ανάκληση των δεδομένων των πρακτόρων και των συμπεριφορών όλων των εφαρμογών της πλατφόρμας. Παράλληλα μοντελοποιεί τις παραμέτρους του περιβάλλοντος της εκάστοτε εφαρμογής τις οποίες μεταφέρει στους πράκτορες προκειμένου να έχουν την απαραίτητη γνώση για την κατάσταση του περιβάλλοντος, να εξάγουν συμπεράσματα και να επιλέξουν συμπεριφορές.

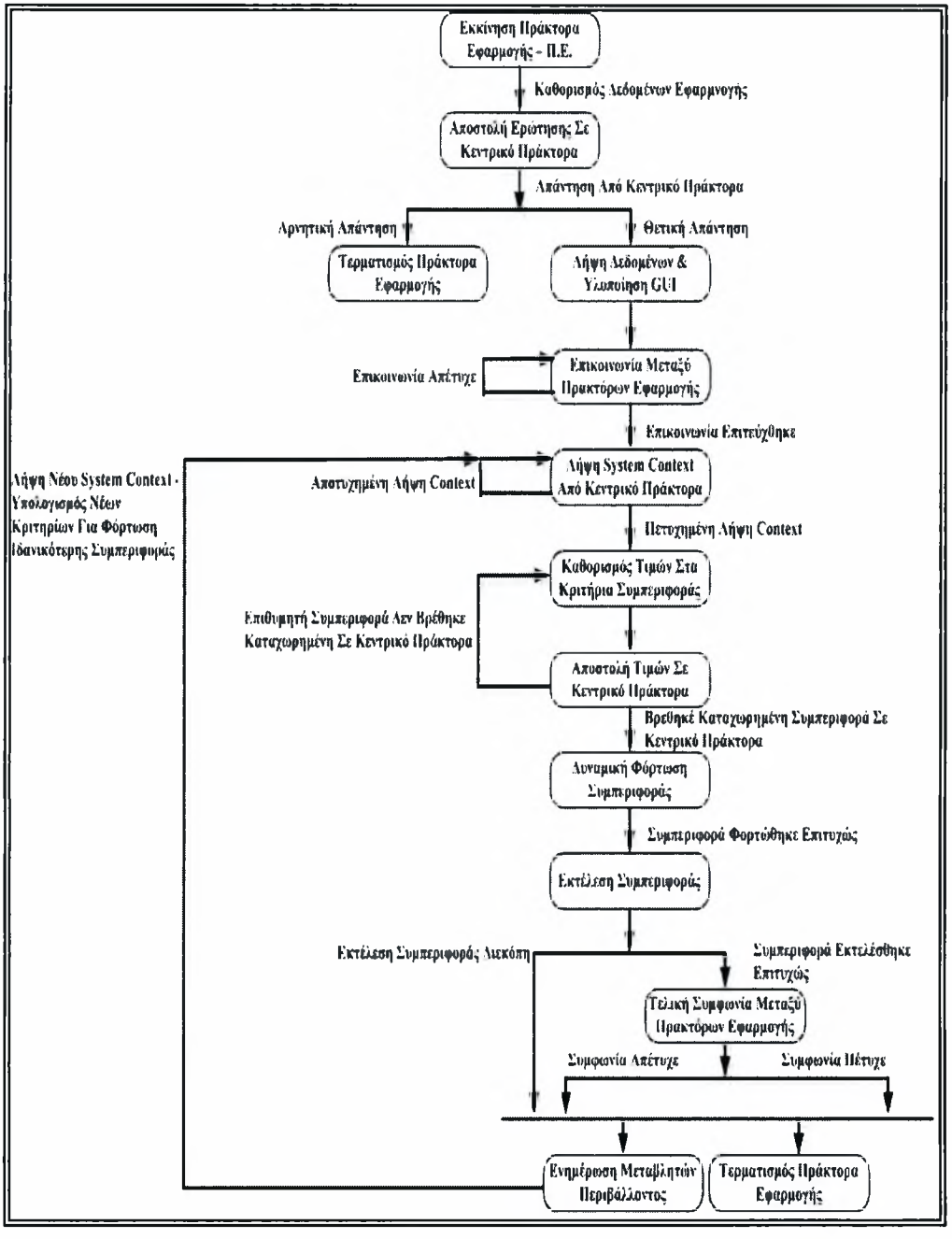
Συνεπώς, οι λειτουργίες του κεντρικού πράκτορα διακρίνονται στις:

- γενικές λειτουργίες (platform-specific) που λαμβάνουν χώρα ανεξαρτήτως της εφαρμογής και περιλαμβάνουν την αρχικοποίηση του κεντρικού πράκτορα (αρχικοποίηση πράκτορα, εγγραφή σε υπηρεσία καταλόγου-yellow page, δημιουργία γραφικού περιβάλλοντος)
- λειτουργίες της εφαρμογής (application-specific) οι οποίες περιλαμβάνουν (α) τις *διαχειριστικές* λειτουργίες, μέσω των οποίων ο διαχειριστής του συστήματος μπορεί να προσθέσει, τροποποιήσει, διαγράψει συμπεριφορές, οντολογίες, δεδομένα και παραμέτρους του περιβάλλοντος της εκάστοτε εφαρμογής και (β) τις *αλληλεπιδραστικές* λειτουργίες, μέσω των οποίων οι πράκτορες εφαρμογής της εκάστοτε υπηρεσίας ανακτούν (ιδιωτικά) δεδομένα και ενημερώνονται για την τρέχουσα κατάσταση του περιβάλλοντος της εφαρμογής προκειμένου να επιλέξουν συμπεριφορές. Οι αλληλεπιδραστικές λειτουργίες του κεντρικού πράκτορα αποτελούν τις βασικές λειτουργίες της πλατφόρμας δεδομένου ότι εμπλέκεται ο κεντρικός πράκτορας (συστήματος) ο οποίος αλληλεπιδρά με τους πράκτορες οποιασδήποτε εφαρμογής ώστε οι τελευταίοι να ολοκληρώσουν το έργο τους. Οι λειτουργίες αυτές θα εξεταστούν αναλυτικότερα στη συνέχεια.

Ο κεντρικός πράκτορας δεν εμφανίζει καμία δυναμικότητα. Δεν φορτώνει δυναμικά κώδικα από άλλες τοποθεσίες ούτε μπορεί ο κώδικάς του να επεκταθεί ή τροποποιηθεί από τους προγραμματιστές των εφαρμογών.

2.3 Πράκτορας Εφαρμογής

Ο πράκτορας εφαρμογής είναι υπεύθυνος για την παροχή μιας συγκεκριμένης λειτουργικότητας στα πλαίσια μιας εφαρμογής. Για παράδειγμα, μπορεί να είναι ο πράκτορας που διαπραγματεύεται για λογαριασμό ενός πελάτη ή αντίστοιχα ενός εξυπηρετητή. Παρακάτω παρατίθεται το διάγραμμα μεταβάσεων που αναλύει το διάγραμμα λειτουργιών (σχήμα 2.1) για τον πράκτορα εφαρμογής.



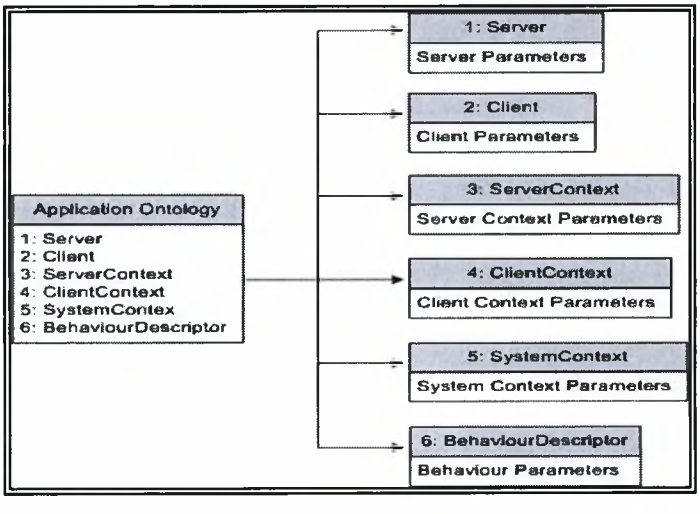
Σχήμα 2.2: Διάγραμμα Μεταβάσεων Πράκτορα Εφαρμογής

Ο πράκτορας εφαρμογής αποτελείται από ένα βασικό στατικό «πλαίσιο» κώδικα που υλοποιεί στοιχειώδεις λειτουργίες, όπως εκκίνηση, αρχικοποίηση, τερματισμό, επιλογή και εκτέλεση συμπεριφοράς. Συνεπώς, σε αυτό το αφηρημένο επίπεδο, δεν έχει εξαρχής συγκεκριμένες πεποιθήσεις, επιθυμίες ή στόχους, ούτε γνωρίζει τι επιμέρους ενέργειες πρέπει να εκτελέσει για να τις επιτύχει. Για τον λόγο αυτό, επικοινωνεί με τον κεντρικό πράκτορα ώστε να ενημερωθεί σχετικά με δεδομένα που αφορούν τον ίδιο ή την κατάσταση του περιβάλλοντος της εφαρμογής καθώς και να φορτώσει την συμπεριφορά που θεωρεί ότι είναι καταλληλότερη για τη τρέχουσα κατάσταση.

2.4 Περιβάλλον-Context

Το περιβάλλον ενός πολυπρακτορικού συστήματος αναπαρίσταται με τη χρήση οντολογιών, οι οποίες περιγράφουν τις έννοιες και τα δεδομένα που ορίζουν την κατάσταση του περιβάλλοντος και με βάση τα οποία αλληλεπιδρούν οι πράκτορες. Δεδομένου ότι η πλατφόρμα CAMP ενοποιεί ένα σύνολο ετερογενών εφαρμογών, είναι αυτονόητο ότι κάθε εφαρμογή εξαρτάται από το δικό της περιβάλλον και συνεπώς ορίζει την δική της οντολογία η οποία το αναπαριστά. Επομένως όλες οι οντολογίες του συστήματος ορίζονται σε επίπεδο εφαρμογής και όχι πλατφόρμας.

Παρόλο όμως που η πλατφόρμα δεν υλοποιεί καμία οντολογία σε γενικό επίπεδο παρά μόνο χρησιμοποιεί τις οντολογίες της εκάστοτε εφαρμογής, ωστόσο είναι σε θέση να μοντελοποιήσει τη βασική δομή που πρέπει να ακολουθούν οι οντολογίες της κάθε εφαρμογής και η οποία δίνεται παρακάτω.

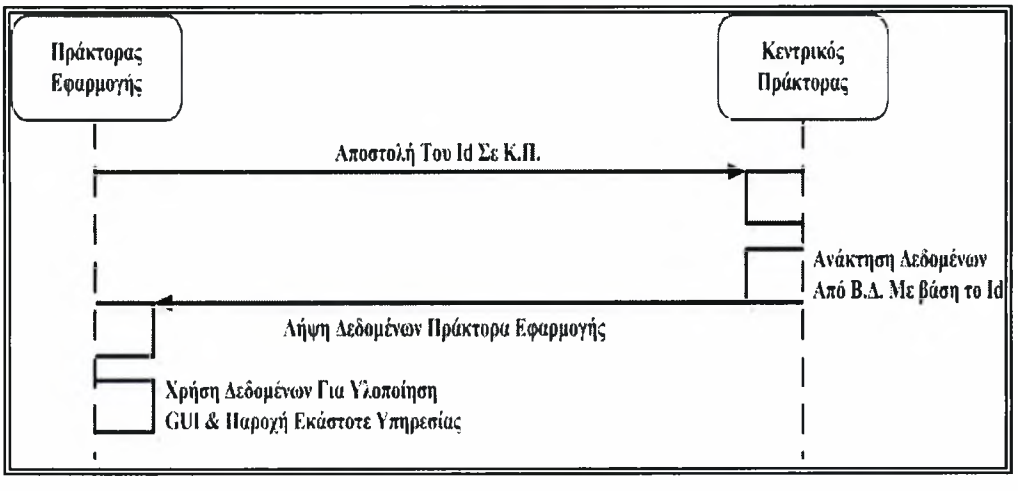


Σχήμα 2.3: Γενικό Διάγραμμα Οντολογίας Πλατφόρμας Υπηρεσιών

Με βάση το παραπάνω διάγραμμα, η οντολογία οποιασδήποτε εφαρμογής θα πρέπει να επεκτείνει την γενική αυτή οντολογία και να περιλαμβάνει (α) τα αντικείμενα Server και Client που μοντελοποιούν τα δεδομένα του πράκτορα εξυπηρετητή και πελάτη της εφαρμογής αντίστοιχα, (β) τα αντικείμενα ServerContext, ClientContext και SystemContext που μοντελοποιούν τις παραμέτρους του περιβάλλοντος του πράκτορα εξυπηρετητή, πελάτη και του συστήματος (μέσω του κεντρικού πράκτορα) της εφαρμογής αντίστοιχα και (γ) το αντικείμενο BehaviourDescriptor που περιλαμβάνει τα κριτήρια για την περιγραφή μιας συμπεριφοράς σε επίπεδο εφαρμογής.

2.5 Ενημέρωση Πράκτορα Εφαρμογής

Η ενημέρωση του πράκτορα εφαρμογής από τον κεντρικό πράκτορα ανήκει, όπως προαναφέρθηκε, στις διαχειριστικές λειτουργίες του κεντρικού πράκτορα και αποτελεί μία από τις λειτουργίες της πλατφόρμας διότι εκτελείται για οποιονδήποτε πράκτορα εφαρμογής ανεξάρτητα της υπηρεσίας που παρέχει. Η ενημέρωση αυτή αφορά την ανάκτηση, από τον κεντρικό πράκτορα, ιδιωτικών δεδομένων του πράκτορα εφαρμογής που πιθανώς ο πρώτος να έχει αποθηκευμένα στη Βάση Δεδομένων του. Στη συνέχεια παρατίθεται το αντίστοιχο διάγραμμα ακολουθίας.



Σχήμα 2.4: Διάγραμμα Ακολουθίας Για Την Περίπτωση Χρήσης "Ενημέρωση Πράκτορα Εφαρμογής"

2.6 Δυναμική Επιλογή Συμπεριφοράς

Ως συμπεριφορά ορίζεται το σύνολο των ενεργειών που εκτελεί ένας πράκτορας κατά την αλληλεπίδραση (π.χ. διαδικασία της διαπραγμάτευσης) του με τους υπόλοιπους πράκτορες της εφαρμογής. Σε επίπεδο λογισμικού, κάθε συμπεριφορά αποθηκεύεται σε ένα .class αρχείο το οποίο εμπεριέχει το σύνολο των μηνυμάτων που αποστέλλονται ανάμεσα στους πράκτορες εφαρμογής καθώς και το σύνολο των ενεργειών για την επίτευξη ενός στόχου σύμφωνα με αυτή τη συμπεριφορά.

Οι λόγοι υπέρ της υλοποίησης της δυναμικής επιλογής συμπεριφορών, έναντι της εκτέλεσης του στατικού και τοπικά αποθηκευμένου κώδικα, είναι ότι:

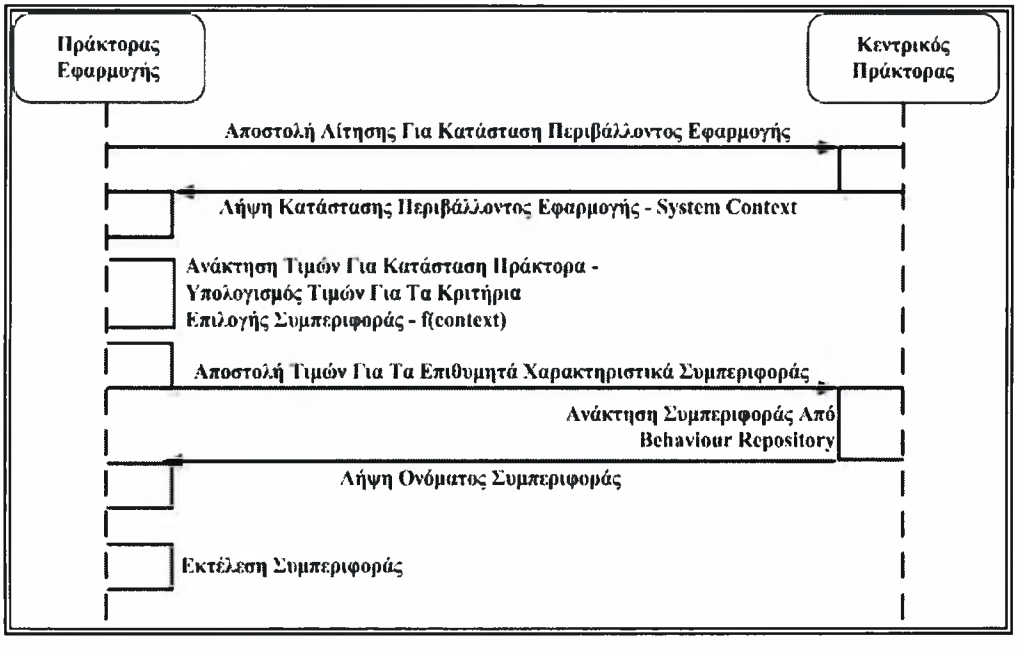
- ο πράκτορας μπορεί να αντιλαμβάνεται και να προσαρμόζεται στις καταστάσεις του περιβάλλοντός του φορτώνοντας δυναμικά την ανάλογη συμπεριφορά μέσω της οποίας υιοθετεί διαφορετικές αντιδράσεις στα πλαίσια μιας εφαρμογής (την μια στιγμή μπορεί να είναι επιφυλακτικός, λογικός, υπομονετικός, ενώ την άλλη να είναι βιαστικός, απερίσκεπτος, ριψοκίνδυνος)
- λύνεται το πρόβλημα της αναβάθμισης λογισμικού. Εφόσον ο πράκτορας φορτώνει κάθε φορά κώδικα από μια τοποθεσία, τότε μπορεί η τοποθεσία αυτή να ενημερώνεται με καινούργιες και βελτιωμένες εκδόσεις συμπεριφορών και πρωτοκόλλων διαπραγμάτευσης αφαιρώντας τις προβληματικές εκδόσεις

Η δυναμική επιλογή μιας συμπεριφοράς πραγματοποιείται ως εξής (σχήμα 2.5). Ο πράκτορας εφαρμογής ζητά από τον κεντρικό πράκτορα την τρέχουσα κατάσταση του περιβάλλοντος της εφαρμογής (την οποία ο ίδιος δεν είναι σε θέση να γνωρίζει). Αφού λάβει τα δεδομένα αυτά, τα εισάγει μαζί με τις τιμές που απεικονίζουν την κατάσταση του πράκτορα, μέσα σε μια συνάρτηση αντιστοίχισης των παραπάνω δεδομένων σε χαρακτηριστικά συμπεριφορών (περισσότερα για αυτό στην παρακάτω υποενότητα). Στη συνέχεια αποστέλλει αυτά τα χαρακτηριστικά στον κεντρικό πράκτορα ο οποίος αναζητά στην δεξαμενή των συμπεριφορών και επιστρέφει στον πελάτη την κατάλληλη συμπεριφορά την οποία ο τελευταίος φορτώνει από τον δικτυακό τόπο του κεντρικού πράκτορα και την εκτελεί.

Όπως φαίνεται και από το διάγραμμα αποσύνθεσης λειτουργιών του συστήματος (σχήμα 2.1), η δυναμική επιλογή συμπεριφοράς αποτελεί διαφορετική λειτουργία από την εκτέλεση της συμπεριφοράς. Η τελευταία λαμβάνει χώρα σε επίπεδο εφαρμογής

δεδομένου ότι ο κεντρικός πράκτορας και κατ' επέκταση το σύστημα δεν εμπλέκονται στην εκτέλεση μιας συμπεριφοράς. Για αυτόν το λόγο, τα ζητήματα που σχετίζονται με την εκτέλεση μιας συμπεριφοράς θα παρουσιαστούν στην επόμενη ενότητα ενώ και τα παρακάτω διαγράμματα εστιάζουν στην επιλογή της συμπεριφοράς και αγνοούν την εκτέλεσή της.

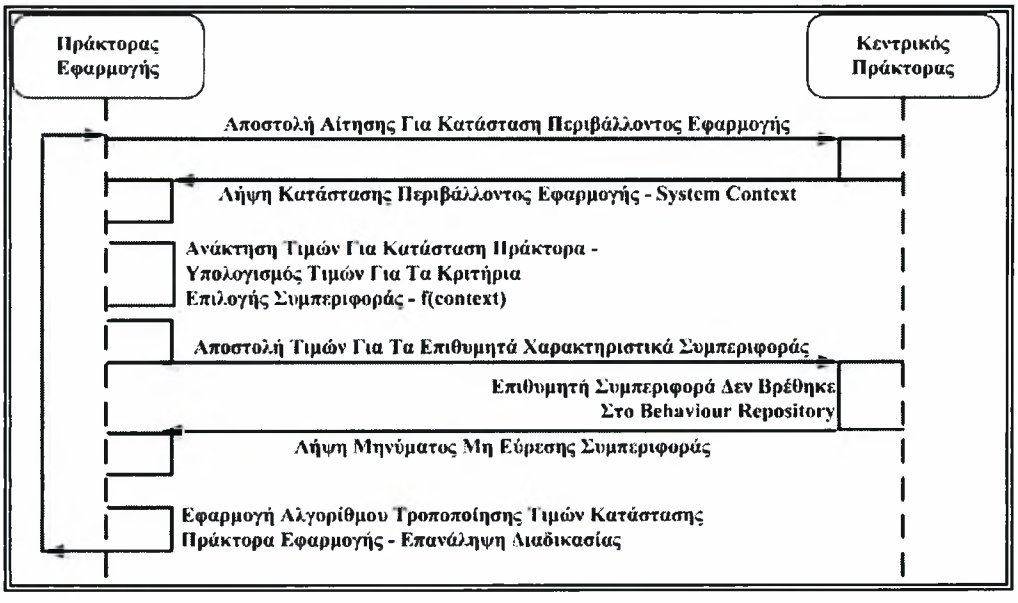
Η δεξαμενή συμπεριφορών (behaviour repository) μοντελοποιεί την Βάση Δεδομένων του κεντρικού πράκτορα στην οποία αποθηκεύονται όλες οι συμπεριφορές μιας συγκεκριμένης εφαρμογής και στην οποία γίνεται κάθε φορά η αναζήτηση για την ανάκτηση της κατάλληλης συμπεριφοράς.



Σχήμα 2.5: Διάγραμμα Ακολουθίας Για Την Κλασική Περίπτωση Χρήσης "Δυναμική Επιλογή Συμπεριφοράς"

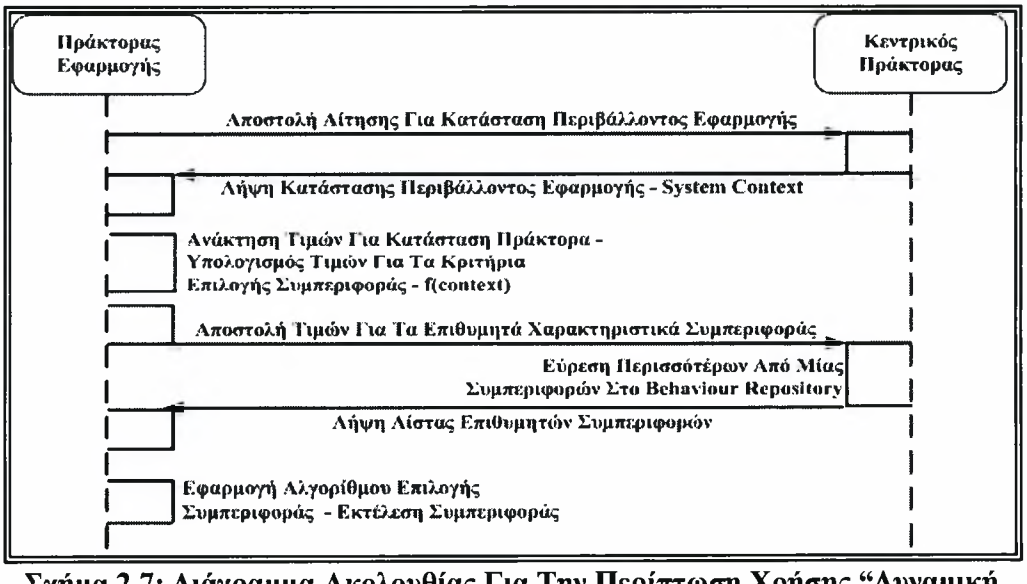
Να σημειωθεί ότι στα πλαίσια της παραπάνω διαδικασίας υπάρχουν και εξειδικευμένες περιπτώσεις κατά τις οποίες (α) ο κεντρικός πράκτορας μπορεί να μην βρει στην δεξαμενή συμπεριφορών κάποια κατάλληλη συμπεριφορά, (β) ο κεντρικός πράκτορας μπορεί να ανακτήσει από την δεξαμενή συμπεριφορών περισσότερες από μια δυνατές συμπεριφορές και (γ) ο πράκτορας εφαρμογής, ανεξαρτήτως των παραπάνω περιπτώσεων, επιλέγει και εκτελεί μια συμπεριφορά όπου στη συνέχεια η εκτέλεσή της διακόπτεται και επαναλαμβάνεται η παραπάνω διαδικασία επιλογής νέας συμπεριφοράς.

Στην πρώτη περίπτωση (Σχήμα 2.6), ο κεντρικός πράκτορας αποστέλλει στον πράκτορα εφαρμογής ένα μήνυμα μη εύρεσης κατάλληλης συμπεριφοράς. Στη συνέχεια ο τελευταίος τροποποιεί τις τιμές που απεικονίζουν την κατάστασή του και επαναλαμβάνει την παραπάνω διαδικασία.



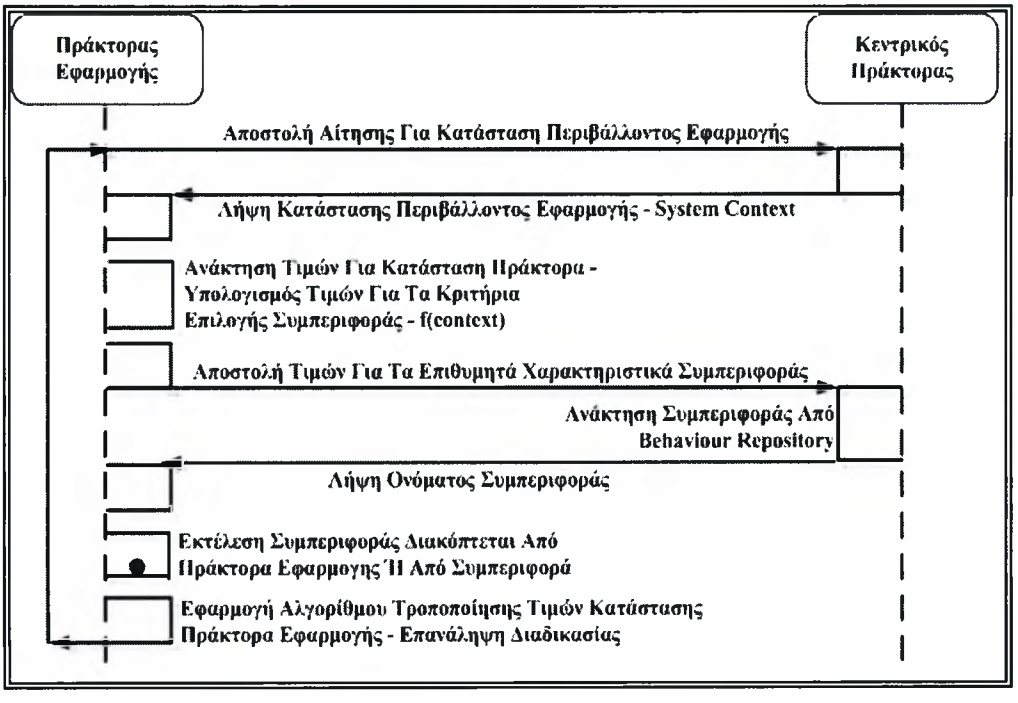
Σχήμα 2.6: Διάγραμμα Ακολουθίας Για Την Περίπτωση Χρήσης "Δυναμική Επιλογή Συμπεριφοράς - Δεν Βρέθηκε Κατάλληλη Συμπεριφορά"

Όσον αφορά την δεύτερη περίπτωση (Σχήμα 2.7), ο κεντρικός πράκτορας αποστέλλει στον πράκτορα εφαρμογής τη λίστα των συμπεριφορών που ανέκτησε και εξαρτάται πλέον από τον τελευταίο να επιλέξει μία από αυτές (π.χ. τυχαία).



Σχήμα 2.7: Διάγραμμα Ακολουθίας Για Την Περίπτωση Χρήσης "Δυναμική Επιλογή Συμπεριφοράς - Βρέθηκαν Περισσότερες Από Μία Συμπεριφορές"

Όσον αφορά την τρίτη περίπτωση (Σχήμα 2.8), η εκτέλεση της συμπεριφοράς διακόπτεται και επαναλαμβάνεται η παραπάνω διαδικασία επιλογής νέας προκειμένου να φορτωθεί μια ιδανικότερη συμπεριφορά για την τρέχουσα κατάσταση του συστήματος ή του πράκτορα. Την διακοπή της εκτέλεσης της συμπεριφοράς μπορεί να προκαλέσει (α) ο πράκτορας εφαρμογής ο οποίος αντιλαμβάνεται ότι, με βάση κάποια αλλαγή στην κατάσταση του περιβάλλοντος, η εκτελούμενη συμπεριφορά είναι ακατάλληλη, (β) η ίδια η συμπεριφορά η οποία αναγνωρίζει ότι τα χαρακτηριστικά που εμφανίζει δεν συμβαδίζουν με την τακτική του πράκτορα ή τα δεδομένα που λαμβάνει από άλλους πράκτορες.



Σχήμα 2.8: Διάγραμμα Ακολουθίας Για Την Περίπτωση Χρήσης "Δυναμική Επιλογή Νέας Συμπεριφοράς – Τερματισμός Εκτελούμενης Συμπεριφοράς"

Να σημειωθεί ότι ο αλγόριθμος τροποποίησης των τιμών της κατάστασης του πράκτορα εφαρμογής (περίπτωση α), ο αλγόριθμος επιλογής ανάμεσα σε «ισοδύναμες» συμπεριφορές (περίπτωση β) και η απόφαση για τον τερματισμό μιας συμπεριφοράς (περίπτωση γ) δεν αφορούν την γενική πλατφόρμα καθώς εξαρτώνται από την εκάστοτε εφαρμογή (application-specific) ή ακόμα και από τις ικανότητες του συγκεκριμένου πράκτορα εφαρμογής. Οι αλγόριθμοι αυτοί θα εξεταστούν στην επόμενη ενότητα για την συγκεκριμένη εφαρμογή εύρεσης ταξί.

2.6.1 Κριτήρια Επιλογής Συμπεριφορών και Μετασχηματισμοί

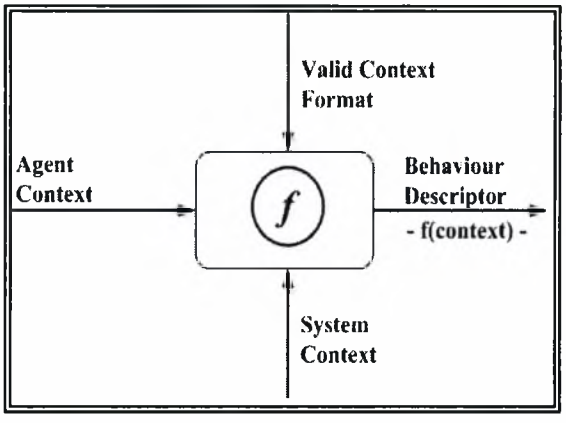
Ο καθορισμός των τιμών στα κριτήρια επιλογής συμπεριφορών μέσω της συνάρτησης αντιστοίχησης, που αναφέρθηκε παραπάνω, προϋποθέτει την γνώση δύο τύπων πληροφορίας (α) των τοπικών παραμέτρων του πράκτορα εφαρμογής που συνιστούν το περιβάλλον του πράκτορα (agent context) και (β) των γενικότερων παραμέτρων της εφαρμογής που συνιστούν το περιβάλλον του συστήματος (system context).

Τα (α) και (β) μαζί εισάγονται ως όρισμα η συνάρτηση αντιστοίχησης και παράγει τις τιμές για τα κριτήρια των υπό επιλογή συμπεριφορών (Σχήμα 2.9). Τα κριτήρια αυτά περιγράφουν (α) τις ιδιότητες της συμπεριφοράς που επιθυμεί να αποκτήσει ο πράκτορας και (β) το πρωτόκολλο διαπραγμάτευσης που επιθυμεί να χρησιμοποιήσει κατά την αλληλεπίδρασή του με άλλους πράκτορες.

Τα κριτήρια επιλογής συμπεριφορών, όπως και οι ίδιες οι συμπεριφορές:

- ορίζονται σε επίπεδο εφαρμογής (και όχι σε επίπεδο πλατφόρμας) από τον προγραμματιστή της εκάστοτε υπηρεσίας Ο λόγος είναι ότι κάθε εφαρμογή θέτει τα δικά της κριτήρια που περιγράφουν τις συμπεριφορές της και δεν υπάρχει ένα γενικό σύνολο κριτηρίων που να περιγράφουν τις συμπεριφορές όλων των εφαρμογών της πλατφόρμας
- καθορίζονται αυστηρά από την οντολογία της εκάστοτε εφαρμογής και αποτελούν αντικείμενα της τάξης BehaviourDescriptor (περιγράφηκε στην υποενότητα 2.4

Το μοναδικό κοινό κριτήριο επιλογής συμπεριφορών για οποιαδήποτε εφαρμογή είναι ο τύπος του πρωτοκόλλου διαπραγμάτευσης που χρησιμοποιεί μια συμπεριφορά για να επιτύχει τον σκοπό της. Το κριτήριο αυτό είναι αναγκαίο για να μπορούν οι αλληλεπιδρώντες πράκτορες της εφαρμογής να υιοθετήσουν κάποιο συμβατό πρωτόκολλο διαπραγμάτευσης προκειμένου να είναι εφικτή η μεταξύ τους αλληλεπίδραση. Στην αντίθετη περίπτωση, όπου οι αλληλεπιδρώντες πράκτορες φορτώσουν συμπεριφορές με διαφορετικούς τύπους πρωτοκόλλων, δεν είναι εφικτή η μεταξύ τους επικοινωνία.



Σχήμα 2.9: Διάγραμμα SADT Για Την Συνάρτηση Αντιστοίχησης Των Κριτηρίων Επιλογής Συμπεριφοράς

Η συνάρτηση f αντιστοιχεί τις τιμές της τρέχουσας κατάστασης του περιβάλλοντος στις τιμές των κριτηρίων της επιθυμητής συμπεριφοράς και ορίζεται σε ατομικό επίπεδο του πράκτορα εφαρμογής. Ο λόγος που η συνάρτηση δεν ορίζεται σε επίπεδο πλατφόρμας είναι γιατί κάθε συνάρτηση αντιστοίχησης έχει δικούς της συντελεστές για την μετατροπή της εισόδου σε έξοδο, οι οποίοι ορίζονται και αρχικοποιούνται από τον προγραμματιστή της συγκεκριμένης εφαρμογής. Στη συνέχεια βέβαια οι τιμές των συντελεστών μεταβάλλονται δυναμικά από τον εκάστοτε πράκτορα εφαρμογής ανάλογα με την τρέχουσα πορεία του και το ιστορικό των διαπραγματεύσεων στις οποίες συμμετείχε. Να σημειωθεί ότι ο αλγόριθμος τροποποίησης των τιμών του περιβάλλοντος του πράκτορα εφαρμογής, που αναφέρθηκε κατά την περίπτωση της αποτυχημένης επιλογής συμπεριφοράς (Σχήμα 2.6), εμπεριέχει και την μεταβολή των συντελεστών της f ώστε να μετασχηματίζουν την είσοδο σε διαφορετική έξοδο που να εγγυάται καλύτερα αποτελέσματα.

Τονίζουμε ότι η γενική πλατφόρμα CAMP δεν υλοποιεί κάποια συγκεκριμένη συμπεριφορά ή πρωτόκολλο διαπραγμάτευσης πρακτόρων, καθώς δεν μεσολαβεί καθόλου στην επικοινωνία μεταξύ των διαφόρων πρακτόρων της εφαρμογής. Αντίθετα, η αλληλεπίδραση μεταξύ των πρακτόρων εφαρμογής γίνεται στα πλαίσια παροχής της λειτουργικότητας της συγκεκριμένης εφαρμογής και κατ' επέκταση μέσω ενός κατάλληλα διαμορφωμένου πρωτοκόλλου.

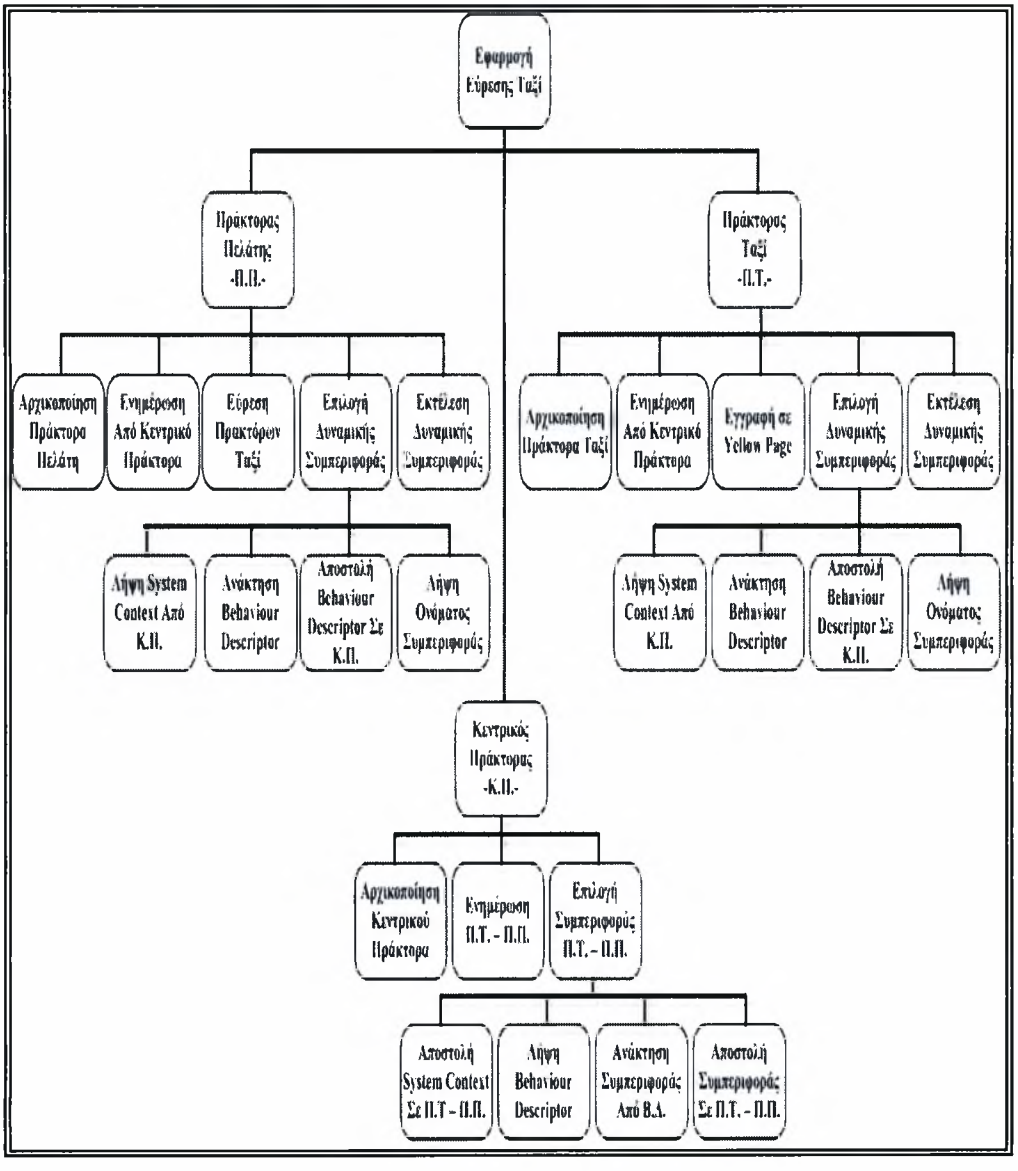
3. ΥΛΟΠΟΙΗΣΗ-ΕΦΑΡΜΟΓΗ

Παράλληλα με την σχεδίαση και ανάπτυξη των μηχανισμών της γενικής πλατφόρμας που περιγράφηκε στην προηγούμενη ενότητα, αναπτύχθηκε και μια δοκιμαστική εφαρμογή. Η λειτουργικότητα αυτής της εφαρμογής είναι η εύρεση ταξί σε μια πόλη.

Σκοπός της εφαρμογής δεν ήταν η υλοποίηση μιας όσο το δυνατόν ρεαλιστικότερης και πληρέστερης λειτουργικότητας αλλά ο έλεγχος για το αν τα χαρακτηριστικά και οι μηχανισμοί της ταυτίζονται με τη διαμόρφωση των απαιτήσεων που επιβάλλει η σχεδίαση της γενικής πλατφόρμας. Γενικότερα, αυτό που ενδιέφερε ήταν μια (οποιαδήποτε) εφαρμογή η οποία να εμπεριέχει τα στοιχεία της δυναμικότητας που καλείται να υποστηρίζει η συγκεκριμένη πλατφόρμα, πράγμα που ισχύει σε μεγάλο βαθμό για την προκειμένη εφαρμογή . Πιο συγκεκριμένα, τα ταξί συνεχώς κινούνται και αλλάζουν τοποθεσίες, με ή χωρίς πελατεία, και μπορεί να υιοθετούν διαφορετικές τακτικές χρέωσης, ενώ αντίστοιχα οι πελάτες εμφανίζουν μια πληθώρα διαφορετικών προτιμήσεων βάσει τοποθεσίας, προορισμού, κόστους, χρόνου παραλαβής και άφιξης στον προορισμό. Ένας άλλος λόγος για τον οποίο επιλέχθηκε αυτή η εφαρμογή είναι γιατί απαιτεί μόνο δύο τύπους πρακτόρων, τον πράκτορα-ταξί και τον πράκτορα-πελάτη, πράγμα που απλοποιεί την υλοποίηση, χωρίς όμως αυτό να απλοποιεί και τις απαιτήσεις σχετικά με την δυναμική επιλογή συμπεριφορών που πρέπει να υποστηρίζει η γενική πλατφόρμα.

3.1 Διάγραμμα Αποσύνθεσης Λειτουργιών

Ακολουθεί το διάγραμμα αποσύνθεσης λειτουργιών (functional decomposition diagram) της συγκεκριμένης εφαρμογής ενώ η λειτουργία και υλοποίηση της εφαρμογής αναλύεται παρακάτω.



Σχήμα 3.1: Διάγραμμα Αποσύνθεσης Λειτουργιών Της Εφαρμογής Εύρεσης Ταξί

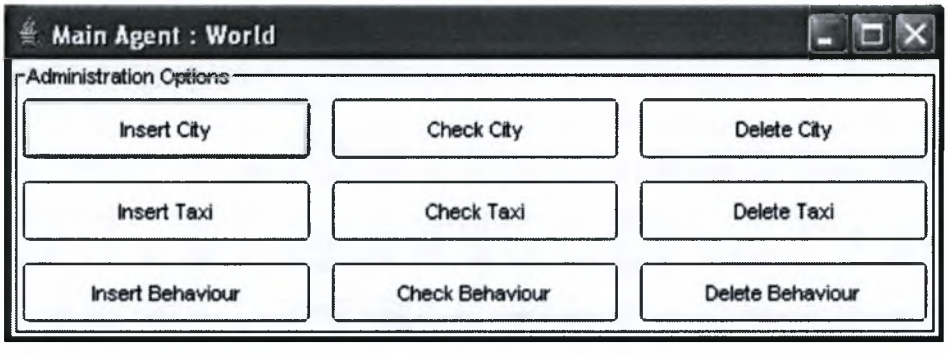
Από το παραπάνω διάγραμμα παρατηρούμε ότι ο πράκτορας της συγκεκριμένης εφαρμογής διακρίνεται σε δύο ειδών πράκτορες, τον πράκτορα πελάτη και τον πράκτορα εξυπηρετητή, που θα εξεταστούν σε παρακάτω υποενότητα.

Να σημειωθεί ότι η λειτουργία αρχικοποίησης του πράκτορα πελάτη και εξυπηρετητή περιλαμβάνει και την δημιουργία του γραφικού περιβάλλοντος ενώ η αρχικοποίηση του κεντρικού πράκτορα περιλαμβάνει την εγγραφή του στην υπηρεσία καταλόγου (yellow page) καθώς και τη δημιουργία του γραφικού του περιβάλλοντος.

3.2 Κεντρικός Πράκτορας

Ο κεντρικός πράκτορας της εφαρμογής είναι ο ίδιος από άποψη οντότητας με τον κεντρικό πράκτορα της γενικής πλατφόρμας. Η διαφορά είναι λειτουργική δεδομένου ότι ο κεντρικός πράκτορας της εφαρμογής εξειδικεύεται ώστε να παρέχει όλες τις βασικές υπηρεσίες (που αναλύθηκαν στην προηγούμενη ενότητα) για την συγκεκριμένη μόνο εφαρμογή.

Ο διαχειριστής της εφαρμογής αλληλεπιδρά με τον κεντρικό πράκτορα μέσα από ένα απλό γραφικό περιβάλλον που απεικονίζεται στο παρακάτω σχήμα.



Σχήμα 3.2: Γραφικό Περιβάλλον Κεντρικού Πράκτορα Πλατφόρμας Για Την Εφαρμογή Εύρεσης Ταξί

Από το παραπάνω γραφικό περιβάλλον παρατηρούμε ότι ο διαχειριστής της εφαρμογής μπορεί να προσθέσει, ελέγξει και αφαιρέσει (α) τα στοιχεία των πόλεων στις οποίες παρέχεται η υπηρεσία εύρεσης ταξί, (β) τα στοιχεία των ταξιτζήδων που προσφέρουν την υπηρεσία και (γ) τα χαρακτηριστικά των συμπεριφορών (τιμές για τα κριτήρια τους) που υποστηρίζουν την εφαρμογή. Αναλυτικότερα τα στοιχεία της κατηγορίας (β) και (γ) παρατίθενται στην οντολογία της εφαρμογής (Σχήμα 3.9).

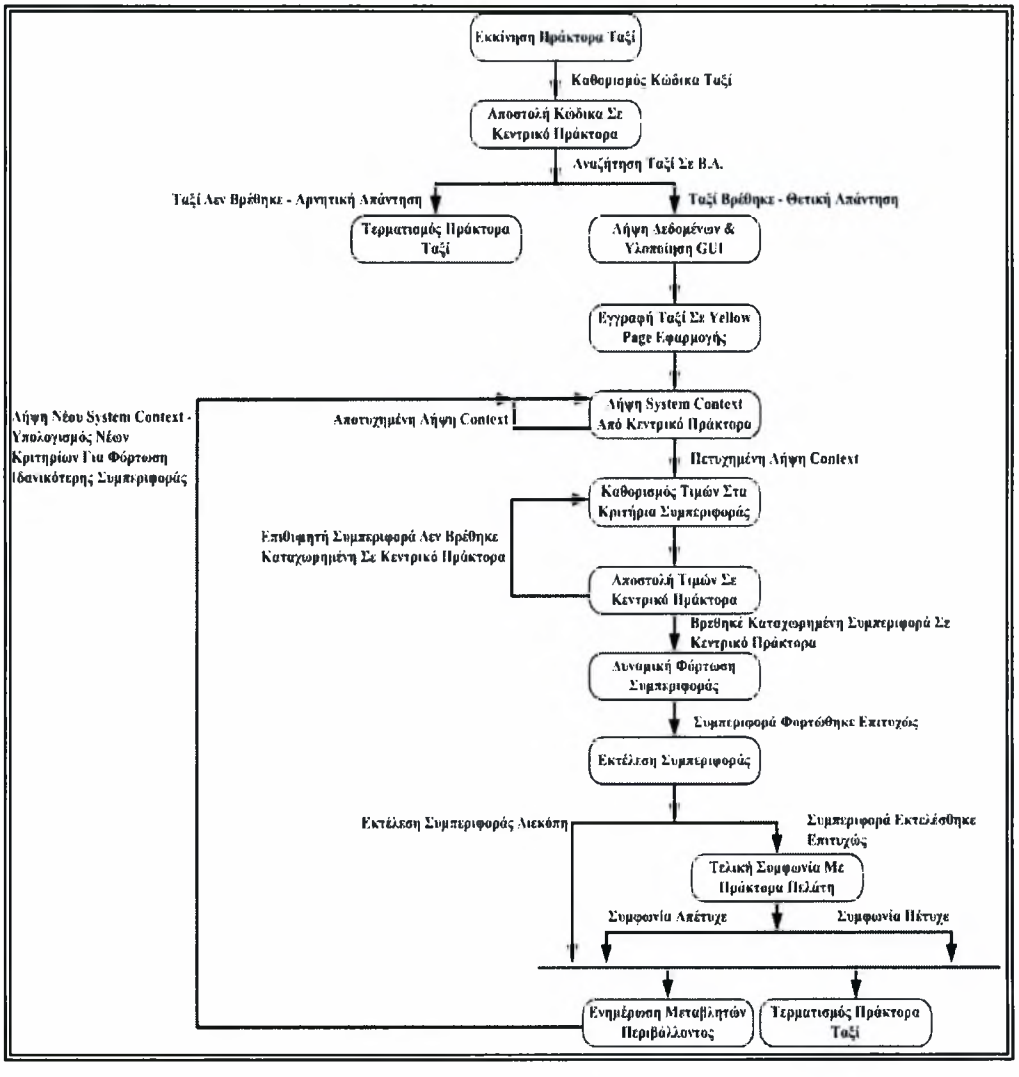
Να σημειωθεί ότι τα δεδομένα σχετικά με την πόλη και τα ταξί εντάσσονται στο ευρύτερο πλαίσιο της πρώτης λειτουργίας της πλατφόρμας που αφορά την ενημέρωση των πρακτόρων εφαρμογής (εξετάστηκε στην ενότητα 2.5), ενώ τα δεδομένα σχετικά με τις συμπεριφορές σχετίζονται με την δεύτερη λειτουργία της πλατφόρμας που αφορά την δυναμική επιλογή συμπεριφοράς (εξετάστηκε στην ενότητα 2.6).

3.3 Πράκτορας-Εφαρμογής

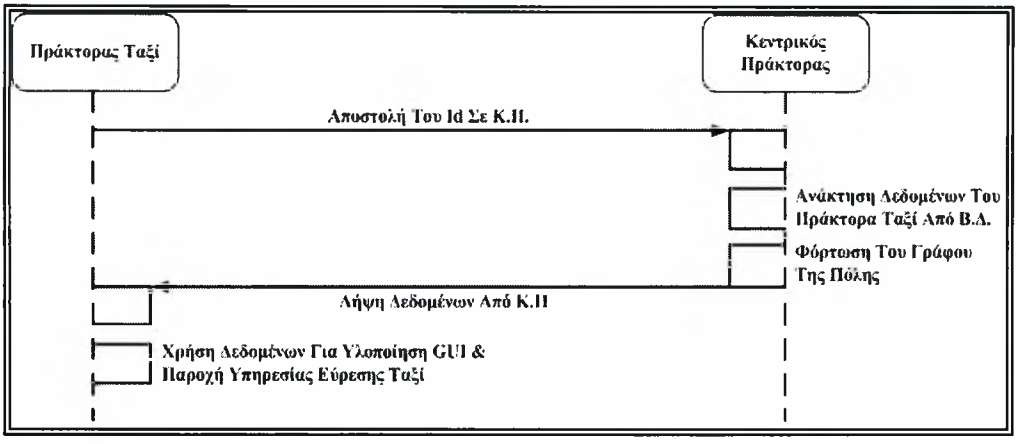
Ο πράκτορας εφαρμογής μοντελοποιήθηκε σε επίπεδο πλατφόρμας και είναι υπεύθυνος για την παροχή μιας συγκεκριμένης λειτουργικότητας στα πλαίσια μιας εφαρμογής. Στο επίπεδο όμως της συγκεκριμένης εφαρμογής δεν υφίσταται (ακέραια) τέτοια οντότητα αλλά διακρίνεται σε δύο πιο εξειδικευμένους πράκτορες οι οποίοι διαπραγματεύονται για λογαριασμό ενός ταξιτζή και ενός πελάτη.

3.3.1 Πράκτορας-Ταξί

Ο πράκτορας-ταξί δημιουργείται από τον διαχειριστή της συγκεκριμένης εφαρμογής και είναι υπεύθυνος για την παροχή της υπηρεσίας ενός ταξί, επεκτείνοντας κατάλληλα την λειτουργικότητα του πράκτορα εφαρμογής της γενικής πλατφόρμας.

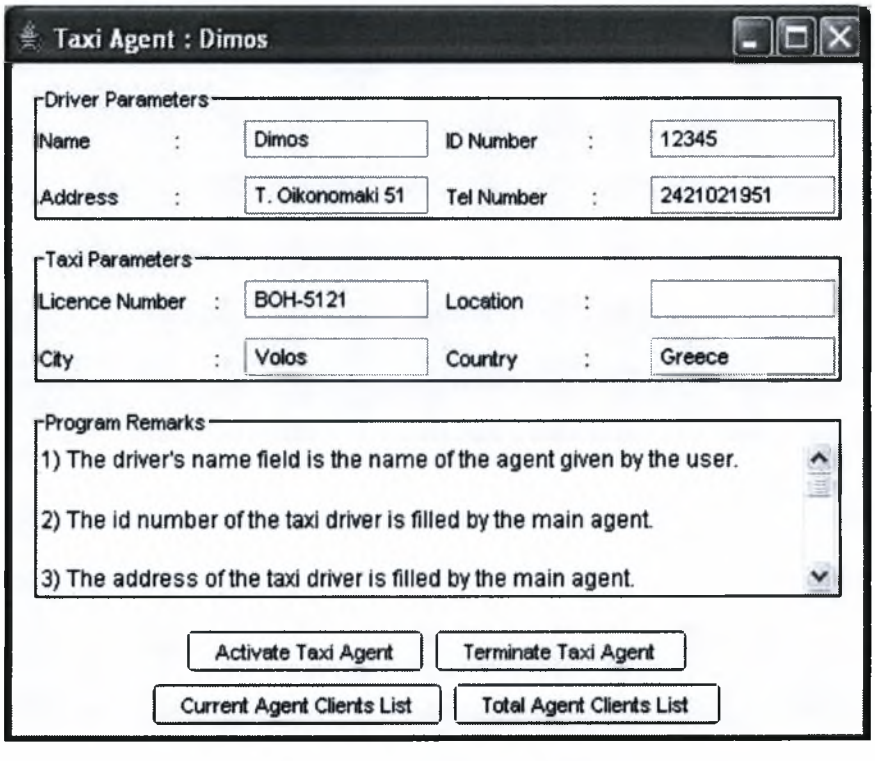


Σχήμα 3.3: Διάγραμμα Μεταβάσεων Πράκτορα-Ταξί



Σχήμα 3.4: Διάγραμμα Ακολουθίας Για Την Περίπτωση Χρήσης "Ενημέρωση Πράκτορα-Ταξί"

Τα παραπάνω διαγράμματα μεταβάσεων και αλληλεπίδρασης του πράκτορα-ταξί με τον κεντρικό πράκτορα αποτελούν επέκταση-εξειδίκευση των αντίστοιχων διαγραμμάτων της προηγούμενης ενότητας (Σχήματα 2.2, 2.4). Ακολουθεί το γραφικό περιβάλλον του πράκτορα-ταξί.

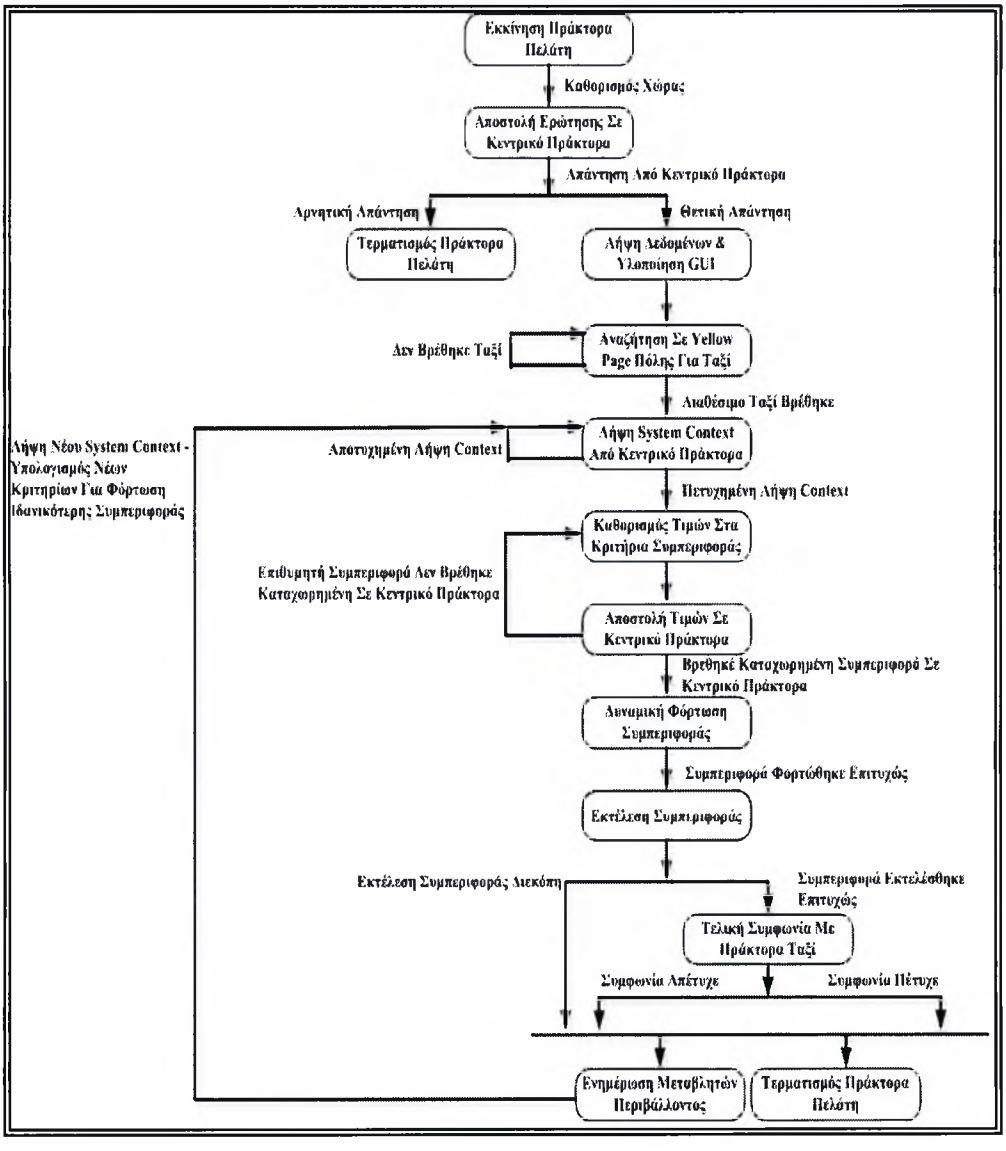


Σχήμα 3.5: Γραφικό Περιβάλλον Πράκτορα-Ταξί

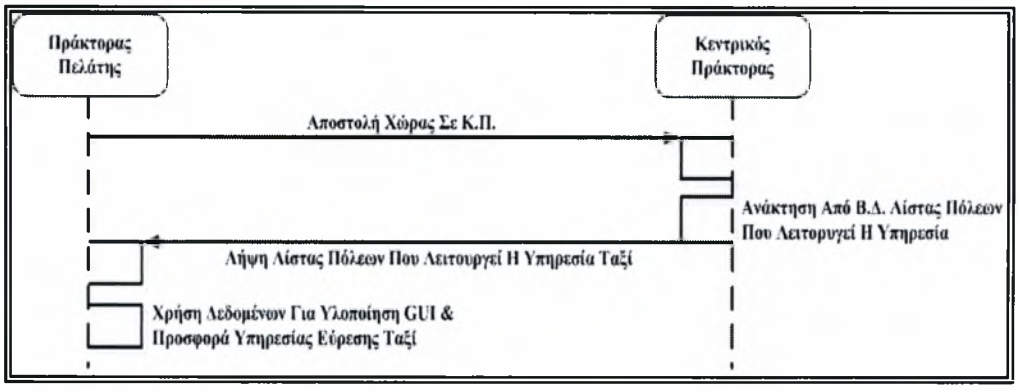
3.3.2 Πράκτορας-Πελάτης

Ο πράκτορας-πελάτης επιφορτίζεται με την ευθύνη να διαπραγματευτεί με διάφορους πράκτορες-ταξί και να επιλέξει την βέλτιστη προσφορά με βάση τις προτιμήσεις του χρήστη που εκπροσωπεί. Ο πράκτορας-πελάτης επεκτείνει τον πράκτορα εφαρμογής της γενικής πλατφόρμας εξειδικεύοντάς τον για την παροχή της συγκεκριμένης υπηρεσίας και δημιουργείται στον υπολογιστή του χρήστη.

Τα παρακάτω διαγράμματα μεταβάσεων και αλληλεπίδρασης του πράκτορα-πελάτη με τον κεντρικό πράκτορα αποτελούν επέκταση-εξειδίκευση των αντίστοιχων διαγραμμάτων της προηγούμενης ενότητας (Σχήματα 2.2, 2.4).

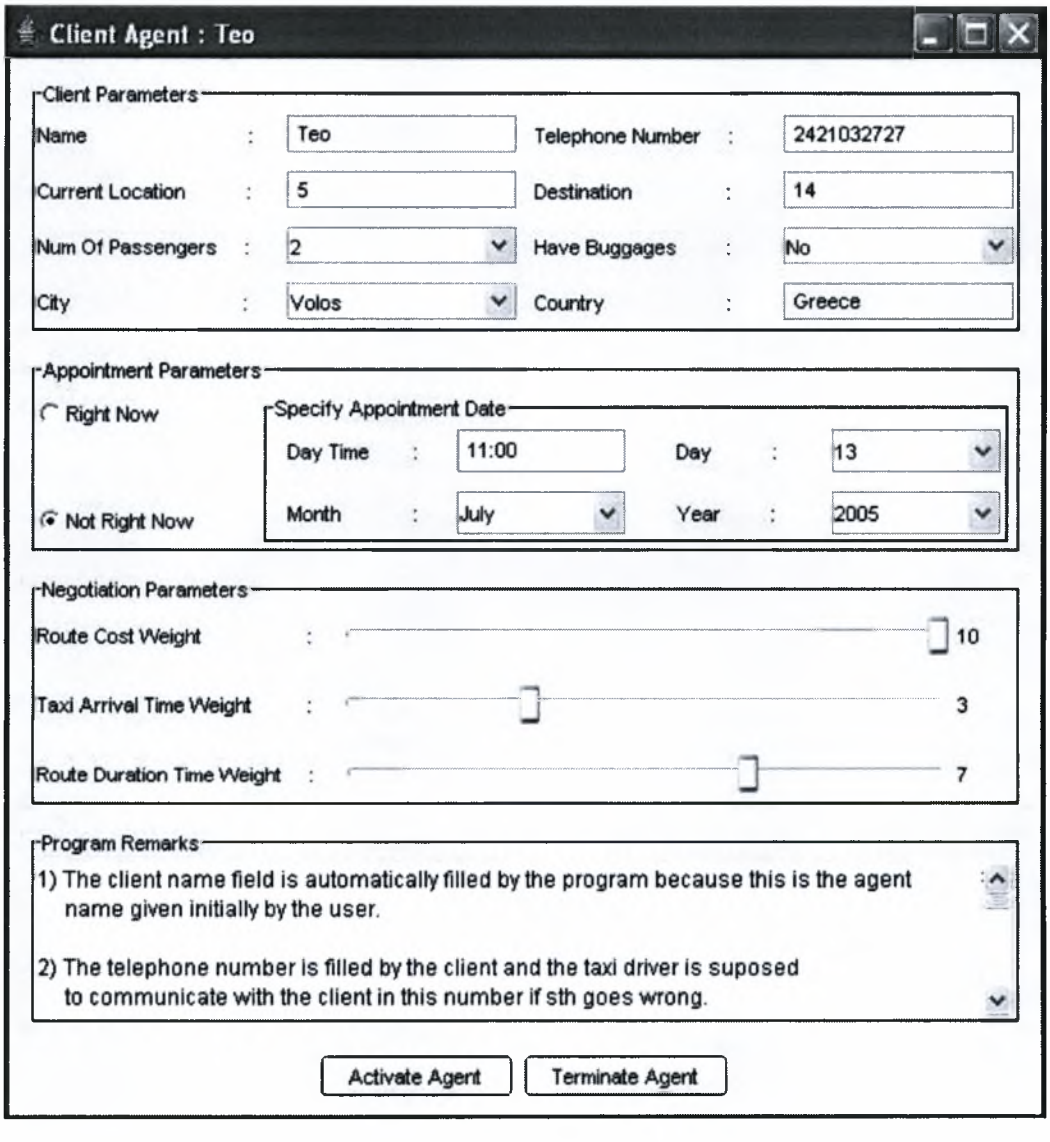


Σχήμα 3.6: Διάγραμμα Μεταβάσεων Πράκτορα-Πελάτη



Σχήμα 3.7: Διάγραμμα Ακολουθίας Για Την Περίπτωση Χρήσης "Ενημέρωση Πράκτορα-Πελάτη"

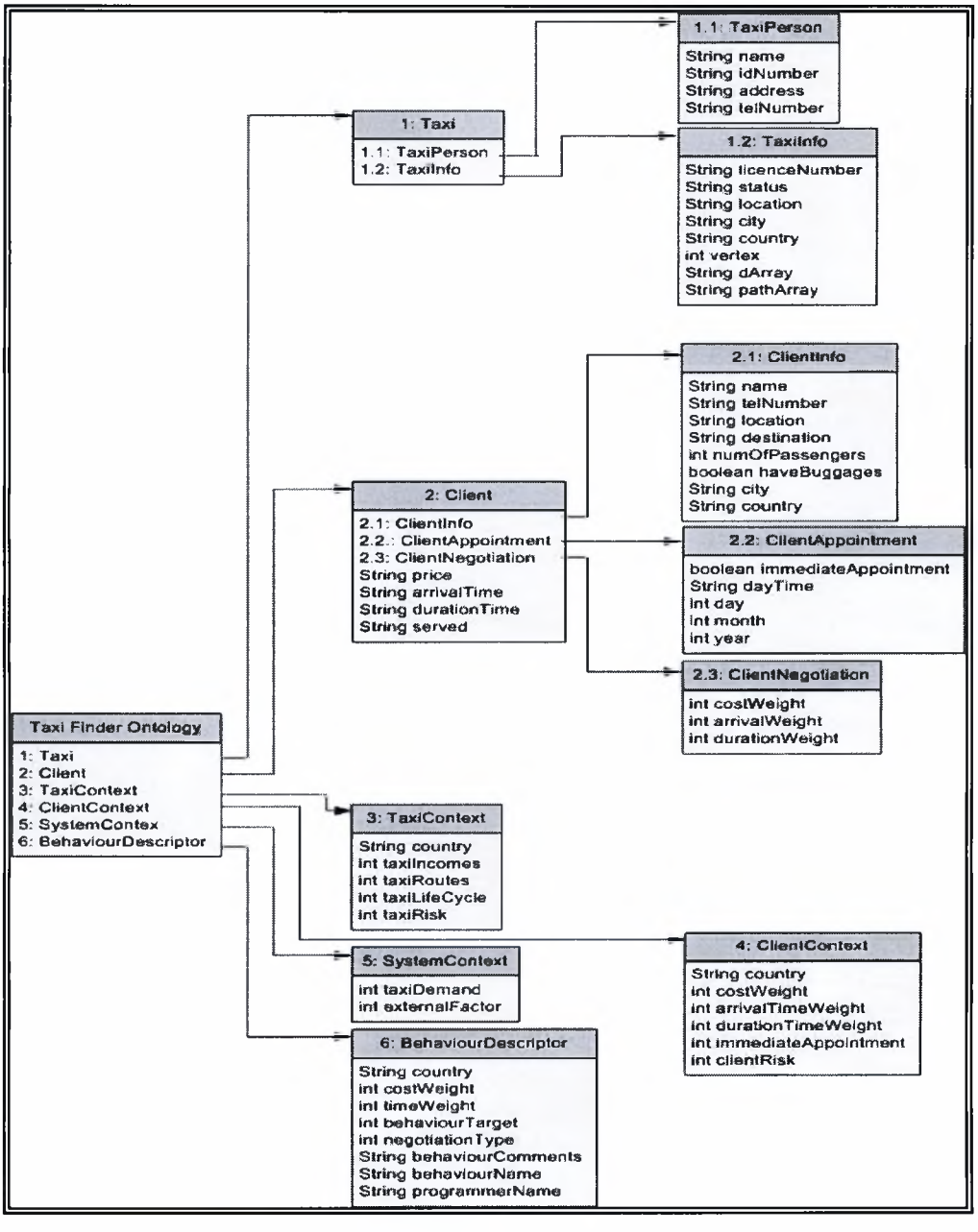
Ακολουθεί το γραφικό περιβάλλον του πράκτορα-πελάτη.



Σχήμα 3.8: Γραφικό Περιβάλλον Πράκτορα-Πελάτη

3.4 Περιβάλλον-Context

Όπως προαναφέρθηκε, κάθε εφαρμογή ορίζει την δική της οντολογία όπου περιγράφονται οι έννοιες που ορίζουν την κατάσταση του περιβάλλοντος και με βάση τις οποίες αλληλεπιδρούν οι πράκτορες της εφαρμογής. Η οντολογία της εφαρμογής εύρεσης ταξί δίνεται παρακάτω και αποτελεί στιγμιότυπο της οντολογίας του σχήματος 2.3.



Σχήμα 3.9: Διάγραμμα Οντολογίας Υπηρεσίας Εύρεσης Ταξί

Εξηγώντας την παραπάνω δομή:

- Το αντικείμενο Taxi περιέχει τα δεδομένα του πράκτορα-ταξί, και περιλαμβάνει στοιχεία για τον οδηγό και το όχημα. Τα ονόματα των μεταβλητών περιγράφουν επαρκώς και την σημασία τους. Να αναφερθεί ότι οι μεταβλητές vertex, darray και patharray περιέχουν τα δεδομένα του γράφου της πόλης τα οποία ο πράκτορας λαμβάνει από τον κεντρικό πράκτορα προκειμένου να υπολογίζει τις αποστάσεις ανάμεσα στο ταξί και τον πελάτη.
- Το αντικείμενο Client περιέχει τα δεδομένα του πράκτορα-πελάτη. Περιλαμβάνει στοιχεία που εισάγει ο πελάτης, όπως την τοποθεσία παραλαβής και προορισμού, την επιθυμητή ώρα άφιξης και δεδομένα σχετικά με τις προτιμήσεις του πελάτη ως προς την βαρύτητα διαφόρων κριτηρίων.
- Το αντικείμενο TaxiContext μοντελοποιεί το περιβάλλον που γνωρίζει ο πράκτορας-ταξί και οι μεταβλητές του καθορίζονται από αυτόν.
- Το αντικείμενο ClientContext μοντελοποιεί το περιβάλλον που γνωρίζει ο πράκτορας-πελάτης και οι μεταβλητές του καθορίζονται από αυτόν.
- Το αντικείμενο SystemContext μοντελοποιεί το περιβάλλον που γνωρίζει ο κεντρικός πράκτορας. Όπως φαίνεται στο αντίστοιχο διάγραμμα ακολουθίας για την περίπτωση δυναμικής επιλογής συμπεριφοράς (Σχήμα 2.5), ο κεντρικός πράκτορας αποστέλλει το αντικείμενο SystemContext στους πράκτορες-πελάτη και ταξί ώστε να ληφθεί υπόψιν για την επιλογή συμπεριφοράς.
- Το αντικείμενο BehaviourDescriptor περιλαμβάνει τα κριτήρια για την περιγραφή μιας συμπεριφοράς με βάση την αντίστοιχη οντολογία σε επίπεδο εφαρμογής. Όπως προαναφέρθηκε, το αντικείμενο BehaviourDescriptor παράγεται σαν αποτέλεσμα της συνάρτησης αντιστοίχισης f και στέλνεται στον κεντρικό πράκτορα για να επιλεγεί η κατάλληλη συμπεριφορά της επιθυμητής εφαρμογής.

3.5 Διασύνδεση Συμπεριφορών

Για να επιτευχθεί η απρόσκοπτη μεταφορά κατάστασης ανάμεσα σε έναν πράκτορα εφαρμογής και στις συμπεριφορές που επιλέγει και εκτελεί δυναμικά, πρέπει να υπάρχει η δυνατότητα (α) ο πράκτορας να διαβιβάσει δεδομένα της εφαρμογής στην συμπεριφορά και (β) η συμπεριφορά να διαβιβάσει τυχόν αλλαγές ή επιπλέον δεδομένα πίσω στον πράκτορα. Να σημειωθεί ότι αυτά τα δεδομένα πρέπει να

βρίσκονται στην κυριότητα και έλεγχο του πράκτορα και όχι της εκάστοτε συμπεριφοράς που φορτώνεται και ξεφορτώνεται δυναμικά.

Αυτή η διαβίβαση δεδομένων επιτυγχάνεται μέσα από κατάλληλη διασύνδεση (interface) που πρέπει υποχρεωτικά να υλοποιεί κάθε πράκτορας εφαρμογής που συμμετέχει στο σύστημα, και η οποία είναι γνωστή εκ των προτέρων έτσι ώστε η υλοποίηση από πράκτορες και συμπεριφορές να μπορεί να γίνει από ανεξάρτητες ομάδες προγραμματιστών χωρίς κάποιο συγχρονισμό μεταξύ τους. Παρακάτω δίνονται σχηματικά οι διεπαφές.

```
public static final int AUTHENTICATE_TAXI_EVENT = 201;
public static final int CREATE_TAXI_GUI_EVENT = 202;
public static final int REGISTER_TAXI_EVENT = 203;
public static final int DEFINE_LOADED_BEHAVIOUR_EVENT = 204;
public static final int EXECUTE_LOADED_BEHAVIOUR_EVENT = 205;
public static final int TERMINATE_TAXI_EVENT = 206;

public void setup();                                    public void TerminateTaxiAgent();
public void OnGuiEvent(GuiEvent ev);                    public Agent GetAgentInstance();
public Taxi GetTaxiInstance();                          public Client GetClientInstance();
public void SetClientInstance(Client c);                public TaxiAgentGui GetTaxiAgentGui();
public AID GetMainAgent();                              public void SetMainAgent(AID main);
public int GetTaxiIncomes();                            public int GetTaxiRoutes();
public int GetTaxiLifeCycle();                          public int GetTaxiRisk();
public void SetTaxiRisk(int risk);
public BehaviourDescriptor GetProperBehaviour(TaxiContext taxi, SystemContext system);
public Codec GetCodec();                                public Ontology GetOntology();
public void SetVertex(int v);                           public int GetVertex();
public void SetDArray(String d_array);                  public int [][] GetDArray();
public void SetPathArray(String path_array);            public int [][] GetPathArray();
public int GetArrivalRouteCost();                       public void SetArrivalRouteCost(int cost);
public String GetArrivalRoutePath();                    public void SetArrivalRoutePath(String path_str);
public int GetDestinationRouteCost();                   public void SetDestinationRouteCost(int cost);
public String GetDestinationRoutePath();                public void SetDestinationRoutePath(String path_str);
public AID GetAID();                                    public void SetAID(AID a);
public int GetFlag();                                   public void SetFlag(int f);
public boolean GetAgentActivated();                     public void SetAgentActivated(boolean activated);
```

Σχήμα 3.10: Διεπαφή Για Πράκτορα-Ταξί

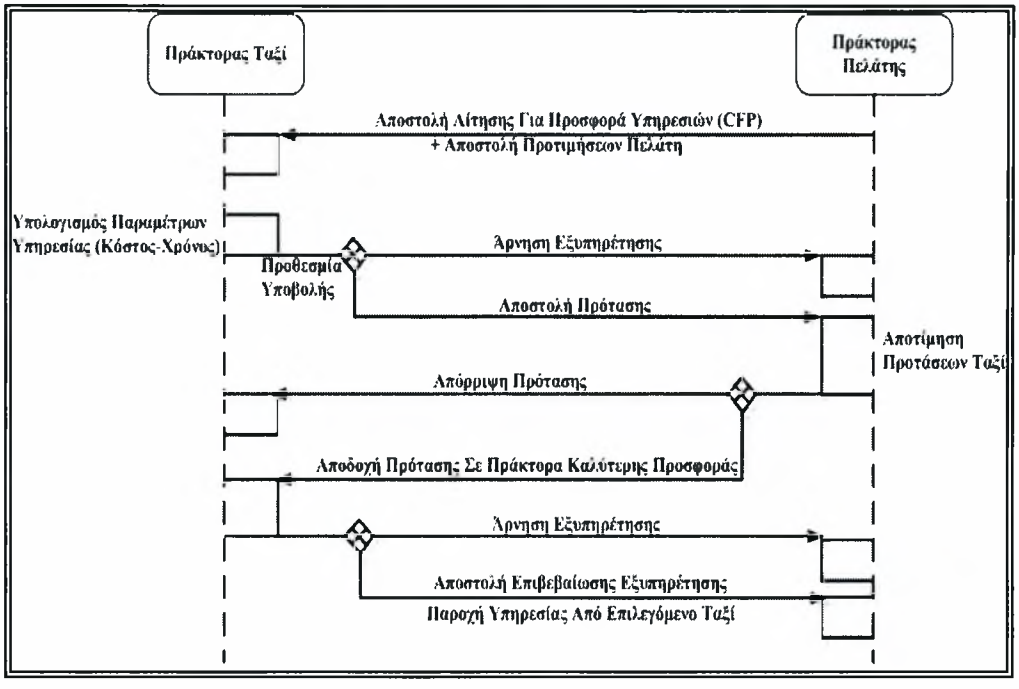
```
public static final int AUTHENTICATE_CLIENT_EVENT = 301;
public static final int CREATE_CITY_GUI = 302;
public static final int ACTIVATE_CLIENT_EVENT = 303;
public static final int DEFINE_LOADED_BEHAVIOUR_EVENT = 304;
public static final int EXECUTE_LOADED_BEHAVIOUR_EVENT = 305;
public static final int SEARCH_TAXI_EVENT = 306;
public static final int TERMINATE_CLIENT_EVENT = 307;

public void setup();                                  public void OnGuiEvent(GuiEvent ev);
public void TerminateClientAgent();                   public Vector RetrieveCityData(String city);
public Agent GetAgentInstance();                      public Client GetClientInstance();
public void SetClientInstance(Client c);              public ClientAgentGui GetClientAgentGui();
public AID GetMainAgent();                            public void SetMainAgent(AID main);
public AID[] GetTaxiAgentList();                      public void SetTaxiAgentList(DFAgentDescription[] list);
public int GetCostWeight();                           public int GetArrivalTimeWeight();
public int GetDurationTimeWeight();                   public int GetClientRisk();
public void SetTaxiRisk(int risk);
public BehaviourDescriptor GetProperBehaviour(ClientContext client, SystemContext system);
public Codec GetCodec();                              public Ontology GetOntology();
public int GetFlag();                                 public void SetFlag(int f);
public String GetCountry();                           public void SetCountry(String c);
```

Σχήμα 3.11: Διεπαφή Για Πράκτορα-Πελάτη

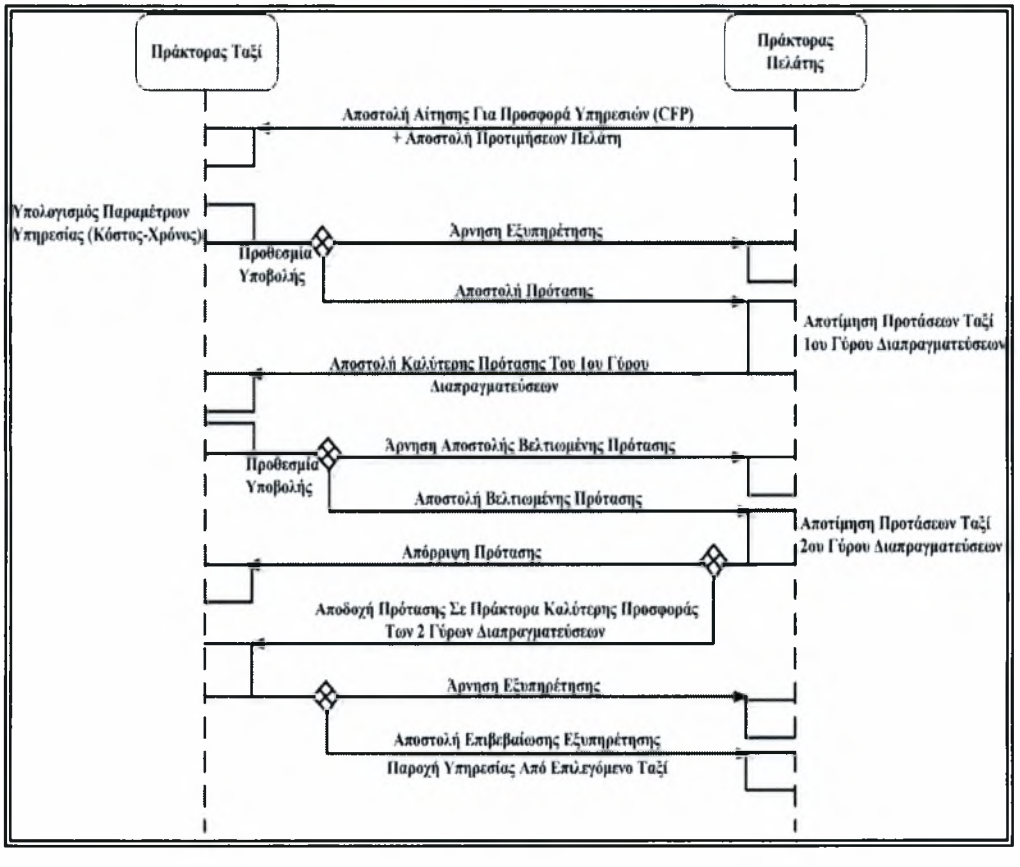
3.6 Συμπεριφορές και Πρωτόκολλα Διαπραγμάτευσης

Για τους σκοπούς της εφαρμογής, και με βάση τις παραπάνω διασυνδέσεις, αναπτύχθηκαν διάφορες συμπεριφορές σύμφωνα με διαφορετικές τακτικές και πρωτόκολλα διαπραγμάτευσης. Στην υποενότητα αυτή θα εξεταστεί η εκτέλεση των συμπεριφορών ανάμεσα στον πράκτορα-πελάτη και πράκτορα-ταξί, η οποία ορίζεται σε επίπεδο εφαρμογής. Αντίθετα δεν πρόκειται να σχολιαστεί η δυναμική επιλογή των συμπεριφορών αυτών, η οποία ορίζεται σε επίπεδο πλατφόρμας και σχολιάστηκε στην ενότητα 2.6.

Παρακάτω παρατίθενται τα αντίστοιχα διαγράμματα για κάθε έναν τύπο πρωτοκόλλου διαπραγμάτευσης που υλοποιούν οι συμπεριφορές της εφαρμογής. Δεδομένου ότι στα διαγράμματα φαίνεται μόνο η ακολουθία των μηνυμάτων μεταξύ των πρακτόρων, μπορεί εύκολα να δοθεί το πρωτόκολλο διαπραγμάτευσης που χρησιμοποιεί η συμπεριφορά ενώ δεν είναι εφικτό να δοθεί η τακτική που ακολουθεί η εκάστοτε συμπεριφορά διότι η τακτική δεν παίζει κάποιο ρόλο στην ακολουθία των μηνυμάτων αλλά στον καθορισμό των περιεχομένων τους. Παρακάτω δίνονται δύο διαφορετικές συμπεριφορές της εφαρμογής με βάση (μόνο) τον τύπο του πρωτοκόλλου διαπραγμάτευσης που υλοποιούν.



Σχήμα 3.12: Διάγραμμα Ακολουθίας Για Την Περίπτωση Χρήσης "Εκτέλεση Συμπεριφοράς Με Υλοποίηση Contact Net Protocol (CNP)"



Σχήμα 3.13: Διάγραμμα Ακολουθίας Για Την Περίπτωση Χρήσης "Εκτέλεση Συμπεριφοράς Με Υλοποίηση Τροποποιημένου Contact Net Protocol Με 2 Γύρους Διαπραγματεύσεων (2RCNP)"

Το πρώτο πρωτόκολλο (CNP) περιλαμβάνει έναν γύρο διαπραγμάτευσης ενώ το δεύτερο αποτελεί επέκταση του πρώτου ώστε να περιλαμβάνει και ένα δεύτερο γύρο διαπραγματεύσεων. Στο δεύτερο γύρο διαπραγμάτευσης ο πράκτορας-πελάτης στέλνει αίτηση, μόνο στους πράκτορες-ταξί που συμμετείχαν στον πρώτο γύρο, για να κάνουν καλύτερη προσφορά στέλνοντας τους παράλληλα και την καλύτερη προσφορά που έλαβε από τον πρώτο γύρο διαπραγμάτευσης. Να σημειωθεί ότι θεωρούμε πως οι συμπεριφορές του πελάτη και του ταξί στο δεύτερο τύπο πρωτοκόλλου είναι ειλικρινείς και καλοπροαίρετες και άρα ο πράκτορας-πελάτης δεν μπορεί να πει ψέματα στους πράκτορες-ταξί σχετικά με την καλύτερη προσφορά που έχει λάβει. Τελικά, οι πράκτορες-ταξί αποφασίζουν αν μπορούν να «χτυπήσουν» την καλύτερη προσφορά του πρώτου γύρου ή όχι. Ο πράκτορας-πελάτης αποτιμά τις νέες προσφορές που λαμβάνει, τις συγκρίνει με τις παλιές και επιλέγει την καλύτερη από τους δύο γύρους διαπραγμάτευσης στέλνοντας στον αντίστοιχο πράκτορα-ταξί αποδοχή της πρότασης. Να αναφερθεί ότι θα μπορούσαν να υλοποιηθούν και άλλα πρωτόκολλα διαπραγμάτευσης απλού γύρου όπως (CNCP ή HCNCP) καθώς και πρωτόκολλα πολλαπλών γύρων διαπραγμάτευσης.

Ωστόσο, ο τύπος ενός πρωτοκόλλου αποτελεί ένα μόνο από τα χαρακτηριστικά της συμπεριφοράς (BehaviourDescriptor) ενώ τα υπόλοιπα χαρακτηριστικά (κόστος, χρόνος) καθορίζονται από την τακτική της. Συνεπώς μπορεί να υπάρχουν περισσότερες από μία συμπεριφορές με τον ίδιο τύπο πρωτοκόλλου διαπραγμάτευσης (έστω CNP) αλλά με διαφορετική τακτική. Για παράδειγμα μία συμπεριφορά μπορεί (α) να δίνει βαρύτητα στον χρόνο παροχής της υπηρεσίας επομένως η προθεσμία υποβολής προτάσεων (deadline) να είναι πολύ σύντομη ώστε βασικά να δέχεται την πρώτη προσφορά που λαμβάνει ή (β) να επιθυμεί να μεγιστοποιήσει το κέρδος και άρα η προθεσμία υποβολής προτάσεων από τους πράκτορες-ταξί να είναι υπερβολικά μεγάλη προκειμένου να λάβει περισσότερες προσφορές για να έχει τυχόν καλύτερες «οικονομικά» προτάσεις.

Στην ενότητα 2.6 αναφέρθηκαν οι αλγόριθμοι που συμμετέχουν στη δυναμική επιλογή συμπεριφοράς και που υλοποιούνται σε επίπεδο εφαρμογής. Για την συγκεκριμένη λοιπόν εφαρμογή εύρεσης ταξί δίνονται οι αλγόριθμοι αυτοί.

3.6.1 Προσομοίωση Παραμέτρων Διαπραγμάτευσης

Ο πράκτορας-ταξί πρέπει να είναι σε θέση να προτείνει μόνος του στον πράκτορα-πελάτη τις παραμέτρους με βάση τις οποίες διαπραγματεύονται, δηλαδή το κόστος της κούρσας, το χρόνο άφιξης του ταξί στον πελάτη και το χρόνο διαδρομής. Συνεπώς ο πράκτορας-ταξί πρέπει να γνωρίζει που βρίσκεται το ταξί, που βρίσκεται ο πελάτης και ποιο είναι το συντομότερο μονοπάτι (α) ανάμεσα στον ίδιο και τον πελάτη και (β) ανάμεσα στην τοποθεσία του πελάτη και στον προορισμό του.

Η ανεύρεση του βέλτιστου μονοπατιού διευκολύνεται από τον κεντρικό πράκτορα, ο οποίος, με βάση τον πίνακα γειτνιάσεως (adjacency matrix) κάθε πόλης, που διαβάζει από ένα ASCII αρχείο, χρησιμοποιεί τον αλγόριθμο του Floyd για να υπολογίσει τα συντομότερα μονοπάτια όλων των ζευγών (ΣΜΟΖ), δηλαδή τα συντομότερα μονοπάτια για κάθε ζεύγος τοποθεσιών. Αυτός ο πίνακας στέλνεται στη συνέχεια σε κάθε πράκτορα-ταξί που δραστηριοποιείται στην συγκεκριμένη πόλη.

Όταν ο πράκτορας-ταξί λάβει μια αίτηση ενός πελάτη, χρησιμοποιεί αυτό τον πίνακα για να βρει το συντομότερο μονοπάτι, ώστε να υπολογίσει το κόστος καθώς και την εκτιμώμενη διάρκεια της διαδρομής χρησιμοποιώντας τις παρακάτω συναρτήσεις μετατροπής. Ο χρόνος άφιξης του ταξί στον πελάτη και ο χρόνος διαδρομής παράγονται από μία σταθερή συνάρτηση, βάσει της απόστασης που υπάρχει ανάμεσα στον πελάτη και το ταξί, ενώ το κόστος της διαδρομής δεν δίνεται από μία προκαθορισμένη και σταθερή συνάρτηση αλλά διαφέρει ανάλογα με την τακτική-συμπεριφορά που ακολουθεί το ταξί και το ρίσκο που είναι διατεθειμένο να πάρει (εν προκειμένω, οι συμπεριφορές με βάση το κόστος διακρίνονται σε δύο περιπτώσεις όπου το ταξί: (α) ζητάει πολλά χρήματα και (β) ζητάει λίγα χρήματα).

- *Χρόνος άφιξης ταξί στον πελάτη* = (2/3)*(απόσταση μεταξύ ταξί και αφετηρίας)
- *Χρόνος διαδρομής* = (2/3)*(απόσταση μεταξύ αφετηρίας και προορισμού)
- *Συνολικό μήκος διαδρομής* = (απόσταση μεταξύ ταξί και αφετηρίας) + (απόσταση μεταξύ αφετηρίας και προορισμού)
- *Κόστος* (για συμπεριφορά πράκτορα που ζητάει λίγα χρήματα) = 1 Ευρώ (ταρίφα) + Συνολικό μήκος διαδρομής * 0.2 Ευρώ
- *Κόστος* (για συμπεριφορά πράκτορα που ζητάει πολλά χρήματα) = 2 Ευρώ (ταρίφα) + Συνολικό μήκος διαδρομής * 0.3 Ευρώ

Να σημειωθεί ότι οι παραπάνω συναρτήσεις είναι αυστηρά υποκειμενικές ενώ οι τιμές των συντελεστών είναι αυθαίρετες και επιλέχθηκαν διαισθητικά.

3.6.2 Συνάρτηση Χρησιμότητας Πράκτορα Πελάτη

Η συνάρτηση χρησιμότητας του πράκτορα-πελάτη λαμβάνει σαν όρισμα την πρόταση του πράκτορα-ταξί (τις τρεις παραμέτρους διαπραγμάτευσης που εξετάστηκαν στην ενότητα 3.6.1) και παράγει ως έξοδο μία τιμή που καθορίζει το πόσο καλή είναι η πρόταση για τον πελάτη που συγκρινόμενη με άλλες καθορίζει τη βέλτιστη πρόταση. Στην υλοποίηση της συγκεκριμένης εφαρμογής, κάθε συμπεριφορά έχει ξεχωριστή συνάρτηση χρησιμότητας, η οποία καθορίζεται από τον προγραμματιστή της. Ο λόγος είναι ότι υπάρχουν συμπεριφορές με σκοπό την μεγιστοποίηση του κέρδους ή τη βελτιστοποίηση του χρόνου παροχής της υπηρεσίας ή την επίτευξη και των δυο παραμέτρων. Συνεπώς, η κάθε συμπεριφορά πρέπει να αποτιμά διαφορετικά την πρόταση του πράκτορα-ταξί όσον αφορά το κόστος της υπηρεσίας, το χρόνο άφιξης του ταξί στον πελάτη και τον χρόνο της διαδρομής πολλαπλασιάζοντας αυτές τις παραμέτρους με διαφορετικούς συντελεστές.

Στην συγκεκριμένη υλοποίηση, για κάθε τύπο πρωτοκόλλου διαπραγμάτευσης ορίζονται στην καλύτερη των περιπτώσεων 4 συμπεριφορές, (α) το κόστος έχει μεγάλη βαρύτητα και ο χρόνος μικρή, (β) το κόστος και ο χρόνος έχουν μεγάλη βαρύτητα, (γ) το κόστος και ο χρόνος έχουν μικρή βαρύτητα και (δ) το κόστος έχει μικρή βαρύτητα και ο χρόνος μεγάλη. Οι 4 λοιπόν συναρτήσεις χρησιμότητας με αντίστοιχη σειρά είναι:

1) *utility_value*= price;
2) *utility_value*=(c_weight*price+a_weight*a_time+d_weight*d_time)/(c_weight+a_weight+d_weight);
3) *utility_value*=(c_weight*price+a_weight*a_time+d_weight*d_time)/(c_weight+a_weight+d_weight);
4) *utility_value*=(a_weight*a_time+d_weight*d_time)/(a_weight+d_weight);

όπου ισχύει ότι c_weight a_weight, d_weight είναι τα βάρη για την παράμετρο του κόστους, του χρόνου άφιξης και του χρόνους διαδρομής αντίστοιχα, που έδωσε ο πελάτης μέσω του γραφικού περιβάλλοντος ενώ price, a_time, d_time είναι οι παράμετροι διαπραγμάτευσης της πρότασης του πράκτορα-ταξί.

3.6.3 Αλγόριθμος Καθορισμού Κριτηρίων Συμπεριφοράς

Όπως αναφέρθηκε, οι πράκτορες πελάτη και ταξί χρησιμοποιούν συναρτήσεις αντιστοίχησης της κατάστασης του περιβάλλοντος (TaxiContext+SystemContext και ClientContext+SystemContext) σε χαρακτηριστικά των κριτηρίων συμπεριφορών της εκάστοτε εφαρμογής (BehaviourDescriptor). Και στις δύο περιπτώσεις, η συνάρτηση απεικόνισης καθορίζει τη βαρύτητα της επιθυμητής συμπεριφοράς ως προς το κόστος, το συνολικό χρόνο παροχής της συγκεκριμένης υπηρεσίας εύρεσης ταξί και τον τύπο του πρωτοκόλλου διαπραγμάτευσης που θα υλοποιεί η συμπεριφορά. Οι συναρτήσεις αντιστοίχησης για τον πράκτορα ταξί και πελάτη είναι δύσκολο να δοθούν (λόγω του μεγάλου μεγέθους τους) ωστόσο θα παρουσιαστούν οι παράμετροι του TaxiContext, ClientContext και SystemContext που καθορίζουν τις παραμέτρους του κόστους, του χρόνου και του πρωτοκόλλου διαπραγμάτευσης.

Για τον πράκτορα-ταξί:

- *cost_weight*: taxiIncomes, taxiDemand, taxiRisk
- *time_weight*: taxiLifeCycle, externalFactor
- *negotiation_type*: time_weight, random

Για τον πράκτορα-πελάτη:

- *cost_weight*: costWeight, taxiDemand, clientRisk
- *time_weight*: immediateAppointment, arrivalTimeWeight, durationTimeWeight, externalFactor
- *negotiation_type*: time_weight, random

3.6.4 Αλγόριθμος Επιλογής Σε Ισοδύναμες Συμπεριφορές

Στην ενότητα 2.6 αναφέρθηκε η περίπτωση στην οποία ο κεντρικός πράκτορας ανακτά από τη Β.Δ. περισσότερες από μία κατάλληλες συμπεριφορές για τον πράκτορα εφαρμογής. Ο τελευταίος λαμβάνει τη λίστα και επιλέγει μία από αυτές. Ο αλγόριθμος με τον οποίο ο πράκτορας πελάτης ή ταξί επιλέγει μία συμπεριφορά από τη λίστα συμπεριφορών ορίζεται σε επίπεδο εφαρμογής.

Στη εφαρμογή εύρεσης ταξί, ο πράκτορας εξετάζει στη Β.Δ. που διαθέτει για το ποια συμπεριφορά από αυτές που ανήκουν στη λίστα χρησιμοποιήθηκε περισσότερες φορές. Αν υπάρχουν περισσότερες από μία συμπεριφορές που χρησιμοποιήθηκαν τον

ίδιο αριθμό φορών, τότε επιλέγει την πιο «καινούργια» συμπεριφορά με την προοπτική ότι εφόσον υλοποιήθηκε αργότερα θα είναι και περισσότερο βελτιωμένη. Σαν τελική εναλλακτική επιλέγει μια συμπεριφορά τυχαία. Να σημειωθεί ότι ο αλγόριθμος θα μπορούσε να ελέγχει ποια συμπεριφορά φορτώθηκε περισσότερες φορές για την συγκεκριμένη χρονική στιγμή της ημέρας ή ποια επέφερε περισσότερα χρήματα ή ποια διακόπηκε λιγότερες φορές.

3.6.5 Αλγόριθμος Τροποποίησης Τιμών Κατάστασης Πράκτορα

Στην ενότητα 2.6 αναφέρθηκε η περίπτωση κατά την οποία η δυναμική επιλογή συμπεριφοράς αποτυγχάνει γιατί (α) ο κεντρικός πράκτορας δεν βρήκε στη δεξαμενή την κατάλληλη συμπεριφορά και (β) η συμπεριφορά διακόπτεται κατά την διάρκεια της εκτέλεσής της. Και στις δύο περιπτώσεις παρατηρήθηκε ότι ο πράκτορας της εκάστοτε εφαρμογής εφαρμόζει κάποιον αλγόριθμο ενημέρωσης των τιμών της κατάστασής του.

Για την συγκεκριμένη εφαρμογή εύρεσης ταξί, ο αλγόριθμος μεταβάλλει ανάλογα την τιμή μιας μεταβλητής που έχει ο πράκτορας-ταξί (taxiRisk) και ο πράκτορας-πελάτης (clientRisk) και ορίζει το ρίσκο που είναι διατεθειμένοι να πάρουν για την επιλογή μιας συμπεριφοράς. Κάθε φορά που η δυναμική επιλογή συμπεριφοράς αποτύχει (για τους λόγους που αναφέρθηκαν παραπάνω) ο πράκτορας μειώνει την μεταβλητή του ρίσκου και μεταβάλλει την κατάστασή του ώστε να είναι πιο διαλλακτικός και μετριόφρων σχετικά με το κόστος της υπηρεσίας. Στην περίπτωση που η συμπεριφορά επιλεχθεί και εκτελεσθεί με επιτυχία, τότε

- ο πράκτορας-ταξί (αν κρίνει σκόπιμο) μπορεί να αυξήσει τη μεταβλητή με αποτέλεσμα να εμφανίσει μεγαλύτερο ρίσκο στην επόμενη διαπραγμάτευση φορτώνοντας μια πιο «απαιτητική» συμπεριφορά και ζητώντας περισσότερα χρήματα
- ο πράκτορας-πελάτης δεν χρειάζεται να αυξήσει τη μεταβλητή εφόσον τερματίζεται μετά την παροχή της υπηρεσίας. Δεδομένου ότι η τιμή της μεταβλητής μειώνεται μετά από κάθε αποτυχία, κατά την εκκίνηση του πράκτορα η τιμή αρχικοποιείται σε έναν μεγάλο αριθμό ώστε αρχικά να εμφανίζει διάθεση ρίσκου διότι αν δεν γίνει κάτι τέτοιο ο πελάτης θα διαπραγματεύεται εξαρχής με διαλλακτική και υποχωρητική στάση που μπορεί να αποφέρει ζημιά για τον ίδιο

4. ΣΥΜΠΕΡΑΣΜΑΤΑ

Η ενασχόληση, στα πλαίσια της διπλωματικής, με τον σχεδιασμό και την υλοποίηση μιας πολυπρακτορικής πλατφόρμας αλλά και την ανάπτυξη της εφαρμογής εύρεσης ταξί, οδήγησε στην εξαγωγή των ακόλουθων συμπερασμάτων.

Η πλατφόρμα CAMP ικανοποιεί τα χαρακτηριστικά των [Luck et al., 2003] για τα πολυπρακτορικά συστήματα της παρούσας φάσης (2003-2005) ενώ κατά το σχεδιασμό λήφθηκαν υπόψιν τα χαρακτηριστικά που απαιτούν τα πολυπρακτορικά συστήματα της επόμενης φάσης (2006-2008). Προκειμένου βέβαια να επιβεβαιωθεί ότι το σύστημα ικανοποιεί τα χαρακτηριστικά της επόμενης φάσης απαιτείται περαιτέρω πειραματισμός με την πλατφόρμα και ανάπτυξη και άλλων εφαρμογών (εκτός αυτής του ταξί). Φυσικά μια αρχική θεωρητική μοντελοποίηση για κάποιες άλλες εφαρμογές, όπως η εύρεση πακέτου διακοπών, δείχνει ότι η πλατφόρμα είναι σε θέση να υποστηρίξει την παροχή και άλλων εφαρμογών αρκεί αυτές να στηρίζονται στο γενικό αρχιτεκτονικό μοντέλο που περιγράφεται στη δεύτερη ενότητα.

Δεν αξιοποιήθηκαν πλήρως οι δυνατότητες του JADE όσον αφορά λειτουργικά εργαλεία και ειδικότερα πακέτα σχετικά με πρωτόκολλα διαπραγμάτευσης. Αυτή θα ήταν μια αξιόλογη κατεύθυνση για μελλοντική δουλειά όμως σκοπός της διπλωματικής δεν ήταν η πλήρης εκμάθηση του εργαλείου αυτού αλλά η χρησιμοποίησή του για την ενασχόληση με θέματα δυναμικής επιλογής και εκτέλεσης συμπεριφοράς.

Οι τεχνικές μάθησης ορίζονται σε επίπεδο εφαρμογής και όχι πλατφόρμας. Όσον αφορά τις τεχνικές για την εφαρμογή εύρεσης ταξί αυτές υλοποιήθηκαν μέσα από τους αλγορίθμους επιλογής ισοδύναμων συμπεριφορών (ενότητα 3.6.4) και τροποποίησης των τιμών της κατάστασης του πράκτορα (ενότητα 3.6.5). Με βάση τους απλούς αυτούς μηχανισμούς, οι πράκτορες της εφαρμογής αποκτούν κάποιου είδους «εμπειρία» μέσω των προηγούμενων διαπραγματεύσεων που στη συνέχεια εκμεταλλεύονται στις επόμενες επιλογές τους. Να σημειωθεί ότι η μάθηση σε ένα σύστημα πρακτόρων δεν σχετίζεται πάντα με πολύπλοκες μεθόδους ή απαιτητικές τεχνικές όπως τα νευρωνικά δίκτυα. Μάθηση, σύμφωνα με τον Wooldridge [Wooldridge, 2002], επιδεικνύει οποιοδήποτε πρακτορικό σύστημα λαμβάνει υπόψιν

παλιότερες καταστάσεις του περιβάλλοντος ή προηγούμενες επιλογές του πράκτορα για την λήψη επόμενων αποφάσεων οι οποίες χάριν της γνώσης των δεδομένων αυτών αναμένεται να είναι καλύτερες. Συμπεραίνει κανείς ότι η υλοποίηση ενός έξυπνου πράκτορα δεν συνεπάγεται την εγγενή ευφυΐα του συστήματος διότι αυτή χαρακτηρίζεται από τον τρόπο που οι πράκτορες λειτουργούν μέσα στην εφαρμογή και επιτυγχάνεται μόνο μέσα από τεχνικές μάθησης και εξαγωγής συμπερασμάτων.

Η ανάπτυξη πρακτορικών συστημάτων πρέπει να βασίζεται σε κάποια πλατφόρμα ανάπτυξης πρακτόρων όπως το JADE. Διαφορετικά είναι υπερβολικά δύσκολη η ανάπτυξη μιας εφαρμογής και παράλληλα η ανάπτυξη των εργαλείων για την υλοποίησή της. Επιπλέον κατά την επιλογή ενός πρακτορικού εργαλείου θα πρέπει να λαμβάνονται υπόψιν οι δυνατότητές του σε σχέση με την υπό ανάπτυξη εφαρμογή διότι το εργαλείο αυτό υπάρχει για να διευκολύνει σε κάποια πράγματα και να κατευθύνει σε κάποια άλλα. Για τη συγκεκριμένη διπλωματική, το JADE έλυσε το πρόβλημα της επικοινωνίας και του συγχρονισμού των πρακτόρων ενώ έδωσε κατευθύνσεις για την ανάπτυξη των συμπεριφορών του πράκτορα αλλά και της οντολογίας για την αναπαράσταση του περιβάλλοντος.

Τέλος να σημειωθεί ότι τα χαρακτηριστικά (α) των σεναρίων διαπραγμάτευσης πολλαπλών ζητημάτων, (β) της δυναμικής επιλογής συμπεριφορών και πρωτοκόλλων με βάση το περιβάλλον και (γ) της μάθησης αναδεικνύουν την λειτουργικότητα και σπουδαιότητα της διπλωματικής σε σχέση με παρόμοιες εργασίες [Angryk et al., 2002], [Gordon & Paprzycki, 2005], [Paprzycki et al., 2004], [Pîrvănescu et al., 2005]. Τα παραπάνω συστήματα υποβαθμίζουν την δυναμική επιλογή της συμπεριφοράς σε απλή επιλογή του πρωτοκόλλου διαπραγμάτευσης χωρίς να λαμβάνουν υπόψιν την κατάσταση των πρακτόρων. Επιπλέον εμφανίζουν κάποια αρνητικά χαρακτηριστικά όπως (α) παρέχουν μία συγκεκριμένη εφαρμογή και δεν μπορούν να επεκταθούν για την παροχή περισσοτέρων, (β) τα σενάρια διαπραγμάτευσης είναι απλού ζητήματος, (γ) η δυναμική επιλογή πρωτοκόλλου διαπραγμάτευσης ισχύει μόνο για τον πράκτορα εξυπηρετητή ενώ ο πελάτης φορτώνει το πρωτόκολλο που έχει η πλειοψηφία των εξυπηρετητών μη λαμβάνοντας υπόψιν το δικό του περιβάλλον, (δ) η λίστα των πρωτοκόλλων είναι εξαρχής καθορισμένη και περιγράφει συγκεκριμένα μόνο πρωτόκολλα και (ε) δεν παρέχεται η δυνατότητα ανάπτυξης νέων συμπεριφορών από ανεξάρτητους προγραμματιστές.

5. ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Έχοντας παρουσιάσει τη γενική σχεδίαση της πλατφόρμας του συστήματος αλλά και την υλοποίηση της συγκεκριμένης εφαρμογής, αξίζει να αναφερθούν κάποιες επεκτάσεις ή τροποποιήσεις που μπορούν να γίνουν σε επίπεδο πλατφόρμας ή εφαρμογής και θα εμφάνιζαν ιδιαίτερο ερευνητικό ενδιαφέρον.

5.1 Υλοποίηση Γενικού Πράκτορα

Οι πράκτορες που απαρτίζουν την πλατφόρμα μπορεί να μην έχουν στατικό κώδικα υπό την έννοια της προκαθορισμένης συμπεριφοράς ωστόσο έχουν στατικό κώδικα για την υλοποίηση του γραφικού περιβάλλοντος καθώς και για την οντολογία που αναπαριστά το περιβάλλον της κάθε υπηρεσίας. Βάσει αυτών συνεπάγεται ότι ο εκάστοτε πράκτορας εφαρμογής (πελάτης ή εξυπηρετητής) εξειδικεύεται στην παροχή μίας μόνο υπηρεσίας και συνεπώς υιοθετεί έναν συγκεκριμένο ρόλο.

Η προτεινόμενη επέκταση αφορά την σχεδιαστική τροποποίηση των πρακτόρων ώστε να παρέχουν οποιαδήποτε υπηρεσία και να μην εξειδικεύονται μόνο σε μία. Η λύση αφορά την ανάπτυξη γενικών (generic) οντοτήτων πρακτόρων οι οποίες μπορούν να εκτελέσουν οποιονδήποτε ρόλο, να έχουν οποιαδήποτε υπόσταση (όντας πελάτες ή εξυπηρετητές) και να παρέχουν οποιαδήποτε υπηρεσία. Η προσέγγιση αυτή επιφέρει σοβαρές τροποποιήσεις στο μοντέλο της πλατφόρμας CAMP δεδομένου ότι ο generic πράκτορας θα πρέπει να είναι σε θέση να φορτώνει όχι μόνο δυναμικές συμπεριφορές για την αλληλεπίδρασή του με άλλους πράκτορες αλλά και τον ρόλο που επιθυμεί να έχει (πελάτης ή εξυπηρετητής), την οντολογία της εκάστοτε υπηρεσίας και το γραφικό περιβάλλον του πράκτορα του οποίου τον ρόλο υποδύεται.

Οι λόγοι που μπορεί μια τέτοια υλοποίηση να προτιμηθεί είναι γιατί (α) παρουσιάζει καινοτομικά χαρακτηριστικά, (β) ο χρήστης έχει στη διάθεσή του μεγάλο bandwidth και δεν προβληματίζεται για την μεταφορά όλων αυτών των δεδομένων και (γ) η συσκευή του χρήστη στην οποία τρέχει η εφαρμογή δεν έχει πολύ αποθηκευτικό χώρο για να διατηρεί τοπικά όλα αυτά τα αρχεία για κάθε πράκτορα που παρέχει μια υπηρεσία. Χαρακτηριστικό παράδειγμα η περίπτωση να τρέχει η εφαρμογή σε κινητό τηλέφωνο.

Οι δυσκολίες μιας τέτοιας υλοποίησης αφορούν (α) το πρόβλημα στον συγχρονισμό των ρόλων του μοναδικού πράκτορα που διαθέτει ο χρήστης όταν θέλει να του προσφερθούν ταυτόχρονα δύο υπηρεσίες και (β) την περίπτωση που ο χρήστης έχει μειωμένο bandwidth ή έλλειψη διαρκής σύνδεσης στο Διαδίκτυο για την μεταφορά όλων των δεδομένων με αποτέλεσμα να προτιμά να υπάρχουν αποθηκευμένα τοπικά.

5.2 Υλοποίηση Κινητών Πρακτόρων

Κυρίαρχο γνώρισμα των κινητών πρακτόρων, οι οποίοι υποστηρίζονται από την πλατφόρμα JADE, είναι η δυνατότητά τους να «μετακινούνται» μέσα σε ένα δικτυακό περιβάλλον για να επιτύχουν τους στόχους τους. Πρόκειται δηλαδή για διεργασίες (software processes), οι οποίες κατά την διάρκεια της εκτέλεσής τους μεταφέρονται στους υπολογιστές που συμμετέχουν στο δίκτυο - περιβάλλον.

Μια δυνατή επέκταση της πλατφόρμας CAMP είναι η υλοποίηση κινητών πρακτόρων που δεν θα εκτελούνται τοπικά στη συσκευή του χρήστη αλλά θα μεταφέρονται σε άλλες τοποθεσίες (κεντρικό πράκτορα) για να διαπραγματευτούν για μια υπηρεσία.

Οι λόγοι για τους οποίους μια τέτοια υλοποίηση μπορεί να προτιμηθεί είναι γιατί (α) οι συναλλαγές μεταξύ του κεντρικού πράκτορα και του πράκτορα πελάτη ή εξυπηρετητή γίνονται σε τοπικό επίπεδο μειώνοντας ουσιαστικά τον όγκο δεδομένων που θα έπρεπε να μεταφερθεί μέσω του δικτύου και (β) παρουσιάζει μεγαλύτερη αξιοπιστία καθώς η λειτουργία του συνολικού συστήματος ανεξαρτητοποιείται εν μέρει από τη διαθεσιμότητα του δικτύου.

Οι δυσκολίες μιας τέτοιας υλοποίησης αφορούν (α) το πρόβλημα της ασφάλειας του πράκτορα αλλά και του συστήματος που θα τον φιλοξενήσει, (β) το πιθανό κόστος της μεταφοράς του πράκτορα από μία τοποθεσία σε άλλη που μπορεί να είναι συγκρίσιμο με αυτό της αλληλεπίδρασης από απόσταση και (γ) τη δυσκολία του απομακρυσμένου ελέγχου των κινητών πρακτόρων καθώς δεν μπορεί να καθοριστεί με ακρίβεια η θέση ή η τρέχουσα κατάστασή του.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[Angryk et al., 2002]: R. Angryk, V. Galant, M. Gordon, M. Paprzycki, (2002), Travel Support System - an Agent-Based Framework. In Proceedings of the International Conference on Internet Computing (IC'02), Las Vegas, NV

[Bellifemine et al., 1999]: F. Bellifemine, A. Poggi, G. Rimassa (1999), JADE – A FIPA-compliant agent framework, In Proceedings of the Practical Applications of Intelligent Agents

[Chmiel et al., 2004]: K. Chmiel, D. Tomiak, M. Gawinecki, P. Kaczmarek, M. Szymczak, M. Paprzycki (2004), Testing the Efficiency of JADE Agent Platform, In Proceedings of the ISPDC *2004* Conference, IEEE Computer Society Press, Los Alamitos, CA, pp. 49-57

[Franklin & Graesser, 1997]: S. Franklin, A. Graesser (1997), Is it an agent, or just a program? In Intelligent Agents III, LNAI Volume 1193, pp. 21-36, Springer, Berlin

[Gordon & Paprzycki, 2005]: M. Gordon, M. Paprzycki (2005), Designing Agent Based Travel Support System, In Proceedings of the ISPDC 2005 Conference

[Griffel et al., 1998]: F. Griffel, W. Lamersdorf, M. Merz, (1998), A plug-in architecture for providing dynamic negotiation capabilities for mobile agents

[Lashkari et al., 1994]: Y. Lashkari, M. Metral, and P. Maes (1994), Collaborative Interface Agents, In Proceedings of the Twelfth National Conference on Artificial Intelligence, Vol. 1, AAAI Press, Seattle, WA

[Luck et al., 2003]: M. Luck, P. McBurney, C. Preist (2003), Agent Technology: Enabling Next Generation Computing, A Roadmap for Agent Based Computing, European Network of Excellence for Agent-based Computing, pp. 41-46

[Mangina, 2002]: E. Mangina (2002), Review of software products for multi-agent systems, European Network of Excellence for Agent-based Computing, pp. 41-46

[Nwana et al., 1996]: H. Nwana, L. Lee, N. Jennings (1996), Coordination in software agent systems, BT Laboratories internal report

[Paprzycki et al., 2004]: M. Paprzycki, A. Abraham, A. Pîrvănescu, C. Bădică, (2004), Implementing Agents Capable of Dynamic Negotiations. In Proceedings SYNASC04: Symbolic and Numeric Algorithms for Scientific Computing, Romania, Mirton Press

[Parunak, 1999]: H.V.D. Parunak (1999), Industrial & Practical Applications of DAI In Multi-Agent Systems, pp. 377-421, MIT Press, Cambridge, MA

[Pîrvănescu et al., 2005]: A. Pîrvănescu, C. Bădică, M. Paprzycki, (2005), Developing a JADE-based Multi-Agent E-Commerce Environment. In Proceedings IADIS AC'05, International Conference on Applied Computing, Portugal, IADIS Press, Lisbon

[Rosenschein & Zlotkin, 1994]: J. S. Rosenchein, G. Zlotkin (1994), Rules Of Encounter Designing Conventions for Automated Negotiation among Computers, MIT Press, Cambridge, MA

[Russell & Norvig, 1995]: S. Russell, P. Norvig (1995), Artificial Intelligence: A Modern Approach, Prentice-Hall, Englewood Cliffs, NJ

[Russell & Subramanian, 1995]: S. Russell, D. Subramanian (1995), Provably bounded-optimal agents, Journal of AI Research, pp.575-609

[Sichman et al., 1994]: J. S. Sichman et al. (1994), A social reasoning mechanism based on dependence networks , In Proceedings of the 11th European Conference on Artificial Intelligence (ECAI-94), Amsterdam, pp. 188-192

[Sichman & Demazeau, 1995]: J. Sichman, Y. Demazeau (1995), Exploiting social reasoning to deal with agency level inconsistency, In Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, CA, pp. 352-359

[Sierra et al., 1997]: C. Sierra, P. Faratin, N. R. Jennings (1997), A Service-Oriented Negotiation Model between Autonomous Agents, In Proceedings of 8th European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW-97), Ronneby, Sweden

[Smith, 1977]: R. G. Smith (1977), The CONTRACT NET: a formalism for the control of distributed problem solving, In Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI-77), Cambridge, MA

[Smith & Davis, 1980]: R. G. Smith, R. Davis (1980), Frameworks for cooperation in distributed problem solving, IEEE Transactions on Systems, Man and Cybernetics, 11(1)

[Weiss, 1999]: G. Weiss (1999), Multi-Agent Systems, MIT Press, Cambridge, MA

[Wooldridge & Jennings, 1995]: M. Wooldridge, N. R. Jennings (1995), Intelligent Agents: Theory & Practice. The Knowledge Engineering Review, 10(2), pp. 115-152

[Wooldridge & Jennings, 1999]: M. Wooldridge, N. R. Jennings (1999), The cooperative problem solving process, Journal of Logic and Computation, 9(4), pp. 563-592

[Wooldridge, 2000]: M. Wooldridge (2000), The computational complexity of agent design problem, In Proceedings of the 4th International Conference on Multi-Agent Systems (ICMAS-2000), Boston, MA, pp.341-348

[Wooldridge, 2002]: M Wooldridge (2002), An introduction to multi-agent systems, John Wiley & Sons