ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

EASYAGENT: ΠΕΡΙΒΆΛΛΟΝ ΓΙΑ ΚΙΝΟΎΜΕΝΕΣ ΟΝΤΌΤΗΤΕΣ ΤΎΠΟΥ JAVA, ΣΥΜΒΑΤΌ ΜΕ ΤΟ ΠΡΌΤΥΠΟ MASIF

Αντώνης Χατζησταματίου

ΜΕΤΑΠΤΥΧΙΑΚΉ ΕΡΓΑΣΙΑ

Ηράκλειο, Φεβρουάριος 1999

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

EASYAGENT: ΠΕΡΙΒΆΛΛΟΝ ΓΙΑ ΚΙΝΟΎΜΕΝΕΣ ΟΝΤΌΤΗΤΕΣ ΤΎΠΟΥ JAVA, ΣΥΜΒΑΤΌ ΜΕ ΤΟ ΠΡΌΤΥΠΟ MASIF

Εργασία που υποβλήθηκε από τον
Αντώνη Χατζησταματίου
ως μερική εκπλήρωση των απαιτήσεων για την
απόκτηση

Μεταπτυχιακού Διπλώματος Ειδίκευσης στην
Επιστήμη Υπολογιστών

Συγγραφέας: _____
Αντώνης Χατζησταματίου
Τμήμα Επιστήμης Υπολογιστών

Εισηγητική Επιτροπή: _____
Χρήστος Νικολάου, Καθηγητής, Επόπτης

_____
Απόστολος Τραγανίτης, Αναπληρωτής Καθηγητής, Μέλος

_____
Αικατερίνη Χούστη, Αναπληρώτρια Καθηγήτρια, Μέλος

Δεκτή:

_____
Πάνος Κωνσταντόπουλος, Καθηγητής
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Ηράκλειο, Φεβρουάριος 1999

EASYAGENT: A MASIF COMPLIANT ENVIRONMENT FOR MOBILE JAVA OBJECTS

Antonis Hatzistamatiou

Master of Science Thesis

Department of Computer Science
University of Crete

ABSTRACT

The World Wide Web is a very popular and still rapidly growing information system, strictly limited to client-server interaction. The explosion of the W3 brings with it a number of related challenges for automation. “Mobile code” approaches like Java or the “interacting distributed objects” model of CORBA try to solve some of the disadvantages of the existing Web, but they either lack a more general interaction and communication model (Java) or the possibilities to transfer code and the right distribution granularity (CORBA). An approach that provides both mobile code and interacting objects (or object clusters) is the Mobile Agent model.

Many developments of mobile agent systems are under way in both academic and industrial environments. The differences among mobile agent systems prevent interoperability and rapid proliferation of agent technology, and has probably impeded the growth of the industry. The OMG MASIF standard has been initiated in order to achieve interoperability between agent platforms of different manufacturers.

EasyAgent is a MASIF compiant mobile agent platform that is built on top of a distributed processing environment. EasyAgent’s runtime environment consists of an agent and a finder system, for hosting and locating mobile objects respectively. Except from the functionality addressed by MASIF, EasyAgent agent system provides a transparent way for communication among agents. An API is also provided for programmers in order to develop agent based applications. Monitoring and management of agents is supported through a graphical user interface. We discuss the design and implementation of the EasyAgent environment and provide scenarios of use.

Supervisor: Christos Nikolaou
Professor of Computer Science
University of Crete

EASYAGENT: ΠΕΡΙΒΆΛΛΟΝ ΓΙΑ ΚΙΝΟΎΜΕΝΕΣ ΟΝΤΟΤΗΤΕΣ ΤΎΠΟΥ JAVA, ΣΥΜΒΑΤΟ ΜΕ ΤΟ ΠΡΟΤΥΠΟ MASIF.

Αντώνης Χατζησταματίου

Μεταπτυχιακή Εργασία

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

ΠΕΡΊΛΗΨΗ

Ο Παγκόσμιος Ιστός είναι ένα ευρέως διαδεδομένο και συνεχώς αναπτυσσόμενο σύστημα πληροφοριών. Η εξάπλωσή του συνοδεύεται από τη μεγάλη ανάγκη για αυτοματοποίηση ορισμένων λειτουργιών. Όμως, το μοντέλο του εξυπηρετούμενου/εξυπηρέτη, που υποστηρίζει, δε μπορεί να αποτελέσει τη βάση για μία υποδομή που θα προσφέρει αυτές τις υπηρεσίες, γιατί αναφέρεται σε αλληλεπίδραση στατικών αντικειμένων των οποίων η διεπαφή επικοινωνίας είναι από πριν καθορισμένη. Μία λύση στο πρόβλημα έρχεται να δώσει η τεχνολογία των κινούμενων πρακτόρων. Συνδυάζοντας τη γλώσσα προγραμματισμού Java και το περιβάλλον αλληλεπίδρασης κατανεμημένων οντοτήτων, όπως ορίζεται από την αρχιτεκτονική CORBA, παρέχει τη δυνατότητα μετακίνησης των οντοτήτων και την υποδομή για την υποστήριξη της διαφανούς επικοινωνίας και αλληλεπίδρασης των αντικειμένων αντίστοιχα.

Έρευνα σε θέματα που αφορούν τους κινούμενους πράκτορες διεξάγεται σε εργαστήρια, πανεπιστήμια και άλλα ιδρύματα. Οι διαφορές που συναντώνται στο σύνολο των συστημάτων πρακτόρων εμποδίζουν τόσο την διαλειτουργικότητα (interoperability), όσο και τους ρυθμούς ανάπτυξης της συγκεκριμένης τεχνολογίας. Το πρόβλημα έρχεται να λύσει το πρότυπο OMG MASIF. Τυποποιώντας θέματα της τεχνολογίας πρακτόρων, επιτυγχάνει τη διασφάλιση τρόπων για την αλληλεπίδραση των συστημάτων.

Το περιβάλλον κινούμενων πρακτόρων EasyAgent, που αναπτύχθηκε στα πλαίσια της εργασίας αυτής, είναι συμβατό με τους ορισμούς του προτύπου. Δομικά του στοιχεία είναι το σύστημα πρακτόρων για τη δραστηριοποίηση των κινούμενων οντοτήτων και το σύστημα ευρετηρίου για τον εντοπισμό τους. Εκτός των λειτουργιών που καθορίζονται από το πρότυπο, παρέχει τη δυνατότητα της επίκλησης μεθόδου ενός πράκτορα με διαφανή τρόπο. Επιπλέον, παρέχει διεπαφή προγραμματισμού (API) για την δημιουργία νέων πρακτόρων αλλά και την απόδοση συγκεκριμένων χαρακτηριστικών στις οντότητες του περιβάλλοντος. Τέλος, επιτρέπει την παρακολούθηση και διαχείριση των πρακτόρων μέσω μίας διεπαφής χρήσης (user interface). Συζητάμε το σχεδιασμό και την υλοποίηση του περιβάλλοντος EasyAgent και παρέχουμε σενάρια χρήσης του.

Επόπτης: Χρίστος Νικολάου
Καθηγητής Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης

ΠΕΡΙΕΧΌΜΕΝΑ

ΠΙΝΑΚΕΣ

ΣΧΗΜΑΤΑ

ΕΙΚΟΝΕΣ

1 Εισαγωγή

Ο Παγκόσμιος Ιστός είναι ένα ευρέως διαδεδομένο και συνεχώς αναπτυσσόμενο σύστημα πληροφοριών. Η εξάπλωσή του συνοδεύεται από τη μεγάλη ανάγκη για αυτοματοποίηση ορισμένων λειτουργιών. Κατά πρώτο λόγο, χρήστες χωρίς τεχνικό υπόβαθρο θα πρέπει να έχουν τη δυνατότητα να επωφεληθούν από την πληροφορία που είναι διαθέσιμη στον Παγκόσμιο Ιστό, χωρίς να κατακλύζονται από τεχνικές λεπτομέρειες. Επίσης, οι χρήστες θα πρέπει να απελευθερωθούν από τη βαρετή επαναληπτική διαδικασία ξεφυλλίσματος (browsing) των ιστοσελίδων. Τέλος, η πληροφορία που παρέχεται από τον Ιστό πρέπει να είναι διαθέσιμη σε μορφή που να εξυπηρετεί τις ανάγκες του χρήστη, ανεξάρτητα από τον τρόπο που είναι καταχωρημένη.

Οι διαδικασίες που λαμβάνουν χώρα πάνω στο διαδίκτυο είναι τόσο ανομοιογενείς όσο και η πληροφορία που είναι αποθηκευμένη σε αυτό, αλλά και οι άνθρωποι που τη χρησιμοποιούν. Λόγω της ανομοιογένειας αυτής, η αυτοματοποίηση των λειτουργιών που αφορούν τον Παγκόσμιο Ιστό είναι αρκετά δύσκολο έργο. Ως παράδειγμα μπορούν να αναφερθούν οι εξής δημοφιλείς λειτουργίες:

- αναζήτηση στις ιστοσελίδες εφημερίδων ή και περιοδικών για την εύρεση άρθρων σχετικών με την επιχείρηση Άλφα,
- παρακολούθηση τιμής και συμπεριφοράς μιας μετοχής, μαζεύοντας ταυτόχρονα αναλύσεις από ειδικούς αλλά και άρθρα, που αφορούν την εν λόγω εταιρία, από τις ιστοσελίδες των εφημερίδων
- αναζήτηση στις ιστοσελίδες εφημερίδων και περιοδικών για την εύρεση άρθρων που αφορούν πελάτες και ανταγωνιστές της εταιρίας Άλφα, χρησιμοποιώντας τα πιο πρόσφατα δεδομένα που υπάρχουν στους κόμβους του εσωτερικού δικτύου της.

Αν και αναφέρονται μόνο τρία παραδείγματα, είναι εύκολο να διαφανεί ότι η ανάγκη για αυτοματοποίηση είναι όχι μόνο επιτακτική αλλά και πολυμορφική. Όσον αφορά το πρώτο παράδειγμα, η λεξικογραφική αναζήτηση είναι μία πολύ εύκολη διαδικασία και παρέχεται ως υπηρεσία από πολλές εταιρίες. Πάραυτα, η διαδικασία αυτή δε μπορεί να διεκπεραιωθεί αν ο χρήστης δεν συμπληρώσει κάποιες φόρμες αναζήτησης και δε συγκεντρώσει τα αποτελέσματα. Επιπρόσθετα, οι τελικοί αποδέκτες τις αναζήτησης μπορεί να θέλουν τα αποτελέσματα σε κάποια συγκεκριμένη μορφή που θα εξυπηρετεί τις ανάγκες τους. Στο δεύτερο παράδειγμα, η συλλογή πληροφοριών για συγκεκριμένη μετοχή, φανερώνει την ανάγκη για αυτοματοποίηση διεργασιών που παρουσιάζουν κοινά χαρακτηριστικά στο πλάνο δράσης. Στη συγκεκριμένη περίπτωση, το μόνο που αλλάζει είναι το όνομα της μετοχής. Επίσης, διαφαίνεται η ανάγκη για συγκέντρωση ανομοιογενών πληροφοριών όπως γράφοι, δεδομένα και άρθρα σε μία αναφορά. Στο τελευταίο παράδειγμα, η συλλογή πληροφοριών για τις συγκεκριμένες εταιρίες απαιτεί τη χρήση δεδομένων από τις προσωπικές πηγές τις εταιρίας Άλφα.

Στα παραδείγματα αυτά διαφαίνεται η ανάγκη για αυτοματοποίηση διαδικασιών που αφορούν την αλληλεπίδραση του χρήστη με τον Παγκόσμιο Ιστό. Αναλυτικά είναι:

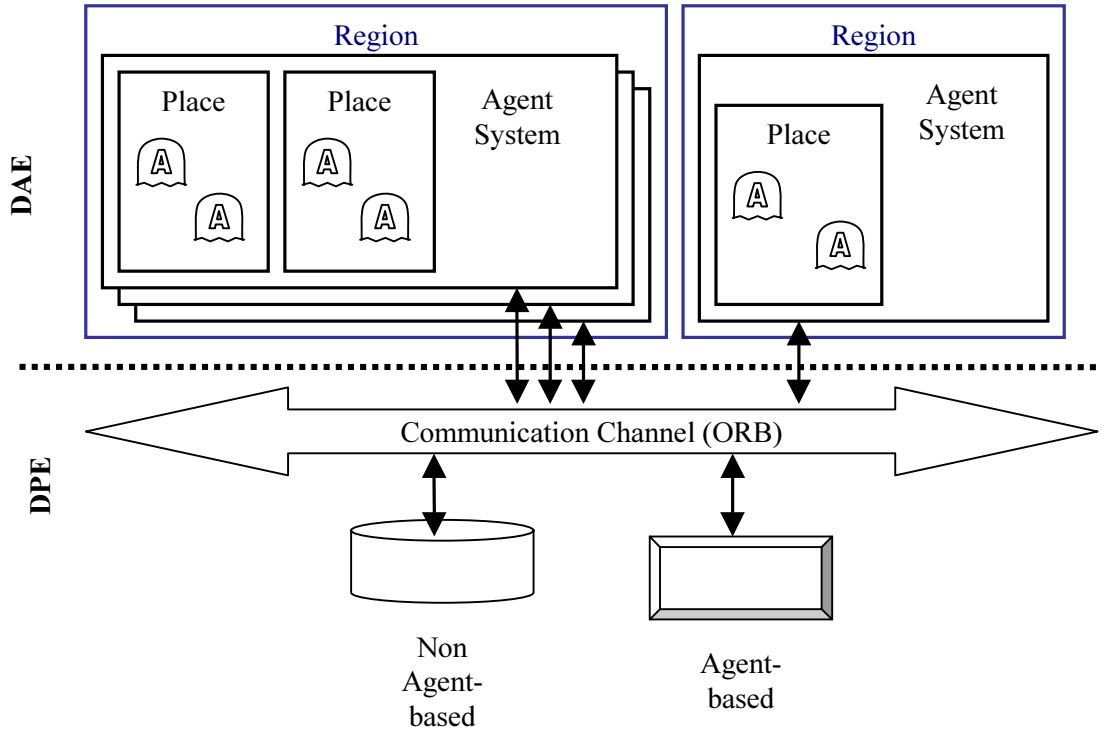
- αυτοματοποίηση των δραστηριοτήτων του χρήστη, περιλαμβάνει και τη συμπλήρωση στοιχείων για αναζήτηση
- συγκέντρωση πληροφοριών από διαφορετικές διευθύνσεις του διαδικτύου σε μία αναφορά
- δυνατότητα να ενεργείς όχι μόνο μετά από εντολή αλλά και βάση προγράμματος (scheduled basis)
- ικανότητα συγκομιδής πληροφοριών από πηγές δημόσιας και ιδιωτικής χρήσης
- και άμεση υποστήριξη συνεργασίας με τις υπάρχοντες εφαρμογές λογισμικού.

Όμως, το μοντέλο του εξυπηρετούμενου/εξυπηρέτη δε μπορεί να αποτελέσει τη βάση για μία υποδομή που θα προσφέρει αυτές τις υπηρεσίες. Αναφέρεται σε αλληλεπίδραση στατικών αντικειμένων των οποίων η διεπαφή επικοινωνίας είναι από πριν καθορισμένη. Για την ικανοποίηση των αναγκών χρειάζονται εφαρμογές οπλισμένες με τη δυνατότητα της επικοινωνίας και μεταφοράς, δηλαδή για εφαρμογές κινούμενων πρακτόρων[13][15][20]. Οι εφαρμογές αυτές θα χρησιμοποιούν το διαδίκτυο για να εντοπίσουν και να αλληλεπιδράσουν με πρόσωπα και πληροφορία εκ μέρους του καταναλωτή. Για τη δημιουργία του περιβάλλοντος υποστήριξης των εφαρμογών απαιτείται τόσο η ικανότητα για μεταφορά κώδικα και κατάστασης των αντικειμένων, όσο και η ύπαρξη ενός ευρύτερου πλαισίου επικοινωνίας και εντοπισμού των οντοτήτων που το συνθέτουν. Το περιβάλλον ανάπτυξης και εκτέλεσης των εφαρμογών υλοποιημένων σε γλώσσα προγραμματισμού Java καλύπτει μερικώς τις ανάγκες της υποδομής. Παρέχει όχι μόνο την, απαραίτητη για το περιβάλλον, δυνατότητα μετακίνησης αντικειμένων, αλλά και ένα σύνολο εργαλείων απαραίτητων για την πλειοψηφία των εφαρμογών (διεπαφές χρήσης, γραφικές αναπαραστάσεις, επικοινωνία με βάσης δεδομένων). Βέβαια, στερείται ενός ευρύτερου πλαισίου κατανομής και αλληλεπίδρασης των οντοτήτων. Το μειονέκτημα αυτό έρχεται να καλύψει το περιβάλλον για κατανεμημένες οντότητες όπως ορίζεται από την αρχιτεκτονική CORBA[2].

Η έκρηξη των ερευνητικών δραστηριοτήτων που σχετίζονται με την τεχνολογία των κινούμενων πρακτόρων άρχισε στις αρχές της δεκαετίας του 90. Ένας μεγάλος αριθμός ερευνητών και κατασκευαστών έχουν ασχοληθεί με την ανάπτυξη περιβαλλόντων πρακτόρων, που απευθύνονται σε διαφορετικά λειτουργικά συστήματα, χρησιμοποιώντας ένα ευρύ σύνολο από γλώσσες προγραμματισμού και τεχνικές υλοποίησης. Νέες γλώσσες αναπτύχθηκαν με μοναδικό σκοπό την υποστήριξη των κινούμενων πρακτόρων. Η διαφοροποίηση αυτή των προσπαθειών αλλά και των κατευθύνσεων έχει αρχίσει να γεφυρώνεται τα τελευταία χρόνια. Από τη μία, υπάρχει μία τάση για χρησιμοποίηση interpreter-based γλωσσών προγραμματισμού, όπως η Java, η οποία αποτελεί και τη γλώσσα υλοποίησης των περισσότερων περιβαλλόντων πρακτόρων σήμερα. Από την άλλη, υπάρχουν αρκετές προσπάθειες για τη σύζευξη των κινούμενων πρακτόρων με τις κατανεμημένες οντότητες που ορίζονται από την αρχιτεκτονική CORBA, όπως εξηγείται στην παρακάτω παράγραφο. Η ανομοιογένεια που παρουσιάζεται στο σύνολο των περιβαλλόντων πρακτόρων δυσχεραίνει την αλληλεπίδραση των οντοτήτων. Μία από τις βασικές ιδιότητες, που πρέπει να έχει μία εφαρμογή για την ικανοποίηση των αναγκών για αυτοματοποίηση, είναι η συνεργασία με εφαρμογές διαφορετικών κατασκευαστών. Με σκοπό λοιπόν τη δημιουργία και υποστήριξη μίας κοινής βάσης που θα διασφαλίζει τη διαλειτουργικότητα των εφαρμογών κινούμενων πρακτόρων διαφορετικών κατασκευαστών, η ομάδα Object Management Group (OMG) καθόρισε το σύνολο των βασικών λειτουργιών αλλά και δυνατοτήτων των ετερογενών συστημάτων πρακτόρων, μέσω του προτύπου Mobile Agent System Interoperability Facility (MASIF)[1]. Το πρότυπο τυποποιώντας θέματα που αφορούν τη μετακίνηση και τη διαχείριση των κινούμενων πρακτόρων καταφερνεί να ενισχύει τα περιβάλλοντα κινούμενων πρακτόρων με μία κοινή διεπαφή επικοινωνίας. Έτσι, επιτυγχάνεται η συνεργασία των συστημάτων πρακτόρων κατά το χρόνο εκτέλεσης αλλά και η διαφοροποίηση των εν λόγω συστημάτων όσον αφορά την υλοποίησή τους.

Το Σχήμα 1-1 παρουσιάζει μία αφαιρετική άποψη ενός συστήματος κινούμενων πρακτόρων που στηρίζει την εκτέλεσή του στο περιβάλλον που ορίζει η αρχιτεκτονική CORBA. Οι οντότητες που σχετίζονται με τους κινούμενους πράκτορες συνθέτουν το περιβάλλον κατανεμημένων πρακτόρων (Distributed Agent Environment) που στηρίζεται στο περιβάλλον κατανεμημένης επεξεργασίας (Distributed Processing Environment) της αρχιτεκτονικής CORBA. Τη φιλοξενία και εκτέλεση των πρακτόρων υποστηρίζουν τα συστήματα πρακτόρων που στο σύνολό τους απαρτίζουν το περιβάλλον κατανεμημένων πρακτόρων. Τη μετακίνηση των πρακτόρων όπως επίσης και την αλληλεπίδραση μεταξύ των

συστημάτων πρακτόρων αλλά και των τελευταίων με ξένες, ως προς τους κινούμενους πράκτορες, οντότητες και εφαρμογές, αναλαμβάνει να διεκπεραιώσει η υποδομή που προσφέρει το περιβάλλον CORBA. Με το τρόπο αυτό είναι δυνατή ή χρήση των υπηρεσιών που έχουν οριστεί από την αρχιτεκτονική, όπως υπηρεσία ονοματολογίας (naming service), υπηρεσία διαπραγμάτευσης (trading service), για την ενδυνάμωση της λειτουργικότητας των συστημάτων πρακτόρων. Ο εξοπλισμός των συστημάτων πρακτόρων με διεπαφές (intefaces) τύπου CORBA, όπως αυτές καθορίζονται από το πρότυπο MASIF, επιτρέπει τη διαφανή αλληλεπίδραση με οποιοδήποτε σύστημα πρακτόρων[9].



Σχήμα 1-1: Η δομή ενός περιβάλλοντος κινούμενων πρακτόρων ενταγμένο στο γενικότερο πλαίσιο που ορίζει η αρχιτεκτονική CORBA.

Σκοπός της εργασίας αυτής είναι η υλοποίηση ενός περιβάλλοντος κινούμενων πρακτόρων συμβατό με το πρότυπο MASIF. Το περιβάλλον ονομάζεται EasyAgent και επιγραμματικά παρέχει:

- σύστημα πρακτόρων για τη φιλοξενία και εκτέλεση πρακτόρων και σύστημα ευρετηρίου για τον εντοπισμό των οντοτήτων. Και τα δύο συστήματα υλοποιούν τη διεπαφή επικοινωνίας που ορίζεται από το πρότυπο.
- επαυξημένο σύνολο λειτουργιών λόγω της ικανότητας απομακρυσμένης επίκλησης μεθόδου ενός κινούμενου πράκτορα μέσω της διεπαφής του συστήματος πρακτόρων. Για την υποστήριξη της νέας λειτουργίας χρειάστηκε να γίνει μία προσθήκη στον ορισμό διεπαφής του συστήματος πρακτόρων.
- διεπαφή προγραμματισμού κινούμενων πρακτόρων (Application Programming Interface)
- διεπαφή χρήσης (user interface) για την παρακολούθηση και διαχείριση των κινούμενων πρακτόρων.

Αντιπροσωπευτικά παραδείγματα από τα υπάρχοντα συστήματα πρακτόρων παρουσιάζονται στο επόμενο κεφάλαιο. Στο τρίτο κεφάλαιο αναλύεται το πρότυπο MASIF. Μια και το κεφάλαιο αυτό αποτελεί την αναλυτική εισαγωγή σε θέματα συναφή με την τεχνολογία των κινούμενων πρακτόρων, παρέχονται αρκετοί ορισμοί και αναλύσεις. Τα τρία κεφάλαια που ακολουθούν ασχολούνται με το περιβάλλον EasyAgent. Συγκεκριμένα, το τέταρτο κεφάλαιο

αναφέρεται στην προσθήκη που έγινε στον ορισμό της διεπαφής, καθώς και σε ειδικά θέματα που παρουσιάστηκαν κατά τη διαδικασία της σχεδίασης. Στο δεύτερο μισό του κεφαλαίου περιγράφεται η υλοποίηση του συστήματος πρακτόρων και ευρετηρίου. Στο πέμπτο κεφάλαιο παρουσιάζονται οι διεπαφές χρήσης, ενώ στο έκτο η διεπαφή προγραμματισμού, που ολοκληρώνει το περιβάλλον. Ένα σενάριο χρήσης του συστήματος EasyAgent παρουσιάζεται στο έβδομο κεφάλαιο. Τα συμπεράσματα και οι προεκτάσεις παραθέτονται στο τελευταίο κεφάλαιο.

2 Επισκόπηση συναφών περιβαλλόντων

Πολλές εταιρίες χρησιμοποιώντας την ταχύτατη αποδοχή της γλώσσας προγραμματισμού Java επιχειρούν να εισάγουν έννοιες σχετικές με την τεχνολογία των κινούμενων πρακτόρων[25]. Κάθε μία ισχυρίζεται ότι η τεχνολογία των κινούμενων πρακτόρων θα αλλάξει τον τρόπο που ζούμε και εργαζόμαστε και προσπαθεί τόσο να καθιερώσει την άποψή της για την αρχιτεκτονική των συστημάτων της νέας αυτής τεχνολογίας όσο και να προσφέρει συστήματα έτοιμα για χρήση. Οι δυνατότητες των συστημάτων αυτών αλλά και οι ιδιαιτερότητές τους θα αναλυθούν στο κεφάλαιο αυτό με μία ανάλυση των σημαντικότερων αντιπροσώπων τους.

2.1 Εισαγωγή

Η έκρηξη του ενδιαφέροντος σε συστήματα κινούμενων πρακτόρων προξενήθηκε λόγω της μεγάλης απήχησης που είχε και έχει η γλώσσα προγραμματισμού Java[3]. Πριν από τέσσερα χρόνια μόνο μερικά συστήματα πρακτόρων ήταν υπό κατασκευή και αυτά βασίζονταν σε πειραματικές γλώσσες προγραμματισμού όπως Tcl[32], Sheme[33], Oblique, και Rosette. Μόνο ένα σύστημα υπήρχε σαν εμπορικό προϊόν και αυτό ήταν το σύστημα Telescript από την εταιρία General Magic.

Η γλώσσα προγραμματισμού Java ήρθε να αλλάξει δραματικά την επικρατούσα κατάσταση. Κατά την διάρκεια των δύο τελευταίων χρόνων πάνω από δέκα συστήματα πρακτόρων, διαφορετικών κατασκευαστών, υλοποιημένα σε Java έχουν εμφανιστεί είτε σαν εμπορικά προϊόντα είτε σαν ερευνητικά προγράμματα. Η ιδεατή μηχανή που χρησιμοποιείται σαν περιβάλλον εκτέλεσης των εφαρμογών που είναι υλοποιημένες σε Java και ο μηχανισμός για το φόρτωμα κλάσεων που χρησιμοποιεί μαζί με μερικές ικανότητες που έχει η γλώσσα, όπως σειριοποίηση, απομακρυσμένη επίκληση μεθόδου (remote method invocation) και χρήση ινών στην εκτέλεση, καταστούν την κατασκευή συστημάτων πρακτόρων μία σχετικά εύκολη εργασία.

Η εξέταση των υπαρχόντων στο εμπόριο συστημάτων πρακτόρων έδειξε ότι όλα μοιράζονται ορισμένα κοινά χαρακτηριστικά που προκύπτουν από την χρήση της γλώσσας προγραμματισμού Java.

- *Όλα παρέχουν κάποιου είδους εξυπηρετητή για πράκτορες.* Ο εξυπηρετητής πρακτόρων είναι το σημείο σύνδεσης με τον δεδομένο υπολογιστή. Η πλατφόρμα πάνω στην οποία θα μετακινηθεί και θα δράσει ο πράκτορας.
- *Οι πράκτορες μπορούν να μετακομίζουν από εξυπηρετητή σε εξυπηρετητή με κάποιο καθορισμένο τρόπο μεταφέροντας ταυτόχρονα και την κατάστασή τους.* Μερικά συστήματα μεταφέρουν αυτόματα τους πράκτορες βασισμένα σε ένα δρομολόγιο που ορίζεται κατά την κατασκευή του πράκτορα, ενώ άλλα αφήνουν τους πράκτορες να μετακινούνται με δική τους πρωτοβουλία.
- *Οι πράκτορες μπορούν να φορτώνουν τον κώδικά τους από πολλές πηγές.* Το δικαίωμα αυτό τους παρέχεται αυτόματα από τη στιγμή που όλα τα συστήματα χρησιμοποιούν δική τους έκδοση του φορτωτή κλάσεων (class loader). Έτσι είναι δυνατή η φόρτωση κλάσεων από το τοπικό σύστημα αρχείων και από το διαδίκτυο.
- *Τα συστήματα είναι γνήσιες εφαρμογές σε Java και χρησιμοποιούν όλες τις δυνατότητες που προσφέρει το περιβάλλον προγραμματισμού της.* Αυτό σημαίνει ότι μπορούν να εκτελεστούν σε οποιονδήποτε υπολογιστή που προσφέρει περιβάλλον εκτέλεσης ικανό να υποστηρίξει εφαρμογές σε Java.

Τα παραπάνω χαρακτηριστικά είναι κοινά για όλα τα συστήματα πρακτόρων και για το λόγο αυτό δεν αναλύονται περισσότερο στις περιγραφές των συστημάτων που απασχολούν το

κεφάλαιο αυτό. Αντιθέτως, το κεφάλαιο αυτό στέκεται στις ιδιαιτερότητες των συστημάτων. Μία ακόμα παρατήρηση που ισχύει για τα υπάρχοντα συστήματα πρακτόρων είναι η έλλειψη ενδεικτικών, για τις δυνατότητες των συστημάτων, παραδειγμάτων με αποτέλεσμα να μην είναι φανερή με μία πρώτη ματιά η λειτουργικότητα που παρέχουν. Επιπλέον, τα βοηθήματα που παρέχονται για τη δημιουργία νέων πρακτόρων, αλλά και για τη χρήση του συστήματος είναι ατελή. Στα περισσότερα συστήματα το μόνο βοήθημα που δίνεται είναι το βοήθημα για το περιβάλλον προγραμματισμού (API) που παρέχουν.

Πριν όμως αρχίσουμε τη συζήτηση για τα συγκεκριμένα συστήματα πρακτόρων θα ήταν προτιμότερο να σταθούμε λίγο στον όρο πράκτορας και στον τρόπο που αυτός χρησιμοποιείται. Ο όρος πράκτορας χρησιμοποιείται ευρέως για αναφορές σε συστήματα ρομποτικής έως και σε φίλτρα για το ηλεκτρονικό ταχυδρομείο. Ακόμη χρησιμοποιείται για περιγραφή εφαρμογών που βοηθούν στη χρήση παραθυρικών περιβαλλόντων επικοινωνίας με το χρήστη αλλά και για την περιγραφή κινούμενου κώδικα. Η σημασιολογία του όρου πράκτορας είναι μόνο η αρχή του προβλήματος. Άλλες διαστάσεις του είναι η χρήση των εννοιών κινούμενος και στατικός (mobile and stationary) πράκτορας, η διαφορά μεταξύ έξυπνου και αυτόνομου (intelligent and autonomous) πράκτορα, και η έννοια της συνεργασίας των πρακτόρων. Για να μη χαθούμε στο γενικότερο πλαίσιο της τεχνολογίας των πρακτόρων είναι προτιμότερο να σκεφτούμε τον πράκτορα λογισμικού σαν ένα διαμάντι με μία ξεχωριστή ιδιότητα ενσωματωμένη σε καθεμία τέσσερις γωνίες του. Πιο συγκεκριμένα το κάθε ζευγάρι των αντιδιαμετρικών γωνιών συμβολίζει τα ζευγάρια των αντικρουόμενων ιδιοτήτων που είναι η κινητικότητα με την στασιμότητα και η συνεργασία με την απομόνωση αντίστοιχα.

2.2 Σύγχρονα συστήματα πρακτόρων

Από τα συστήματα πρακτόρων που εμφανίστηκαν τα δύο τελευταία χρόνια επιλέξαμε τέσσερα συστήματα κινούμενων πρακτόρων που είναι σχετικά ώριμα όσον αφορά θέματα έρευνας και υλοποίησης. Τα συστήματα μπορούν να αποκτηθούν από το διαδίκτυο και είναι: το σύστημα Odyssey[35][36] της εταιρίας General Magic, το σύστημα Aglets[23] της εταιρίας IBM, το Voyager[34] της ObjectSpace και το Concordia[21] της Mitsubishi.

2.2.1 Odyssey της General Magic

Το Odyssey κυκλοφορεί για λειτουργικά συστήματα τύπου Unix και Windows. Χρησιμοποιεί ως περιβάλλον εκτέλεσης, την ιδεατή μηχανή του JDK1.1 μια και στηρίζει την εκτέλεσή του στην δυνατότητα της απομακρυσμένης κλήσης μεθόδου (JavaRMI) που παρέχει το περιβάλλον αυτό. Παρέχεται επιπλέον από την εταιρία λογισμικό για την υποστήριξη του πρωτοκόλλου επικοινωνίας τύπου IIOP[7] (Internet Inter-ORB Protocol) που ορίζεται από την αρχιτεκτονική CORBA και για υποστήριξη του πρωτοκόλλου DCOM[8][10]. Η δυνατότητα αυτή κάνει το σύστημα Odyssey να είναι το μοναδικό που υποστηρίζει πολλαπλούς μηχανισμούς για μεταφορά πρακτόρων. Η αρχικοποίηση του περιβάλλοντος για τη χρήση του συστήματος Odyssey περιλαμβάνει την αλλαγή σε συγκεκριμένες μεταβλητές περιβάλλοντος και την εκκίνηση της διεργασίας rmiregistry που παρέχεται μαζί με το περιβάλλον τύπου JDK1.1.

Μία δυνατότητα που παρέχει το σύστημα Odyssey είναι ο μηχανισμός για τον έλεγχο δράσης των οντοτήτων. Οι οντότητες μπορούν να κατευθύνουν την έξοδό τους αλλά και οποιαδήποτε άλλη πληροφορία επιθυμούν σε αρχεία του τοπικού συστήματος, σε κάποιο παράθυρο επικοινωνίας μιας εφαρμογής που τρέχει στον ίδιο ξενιστή, και στην προκαθορισμένη έξοδο (standard output) της εφαρμογής. Η δυνατότητα αυτή αποτελεί μεγάλο πλεονέκτημα στα χέρια των προγραμματιστών των πρακτόρων αλλά και των διαχειριστών του συστήματος. Όσον αφορά τους προγραμματιστές, τους δίνεται η

δυνατότητα ελέγχου της δράσης των πρακτόρων, όταν αυτοί είναι ακόμα στο στάδιο της υλοποίησης. Πολλές φορές η δράση των πρακτόρων αλλά και η αλληλεπίδρασή τους με άλλους πράκτορες δεν είναι απλή διαδικασία. Από την άλλη πλευρά, όσον αφορά τους διαχειριστές τους δίνει τη δυνατότητα της παρακολούθησης των πρακτόρων καθ' όλη τη διάρκεια της φιλοξενίας τους από το σύστημα. Όμως, η τρέχουσα υλοποίηση αντιμετωπίζει το εξής πρόβλημα με το μηχανισμό ελέγχου δράσης. Όταν ένας πράκτορας αρχικοποιεί τον μηχανισμό για την ανακατεύθυνση της εξόδου στην προκαθορισμένη έξοδο τότε τα μηνύματα εμφανίζονται στην προκαθορισμένη έξοδο του κάθε ξενιστή ο οποίος φιλοξενεί τον πράκτορα τη συγκεκριμένη στιγμή. Όταν όμως, η ανακατεύθυνση γίνεται σε κάποιο παράθυρο εφαρμογής κανένα μήνυμα δεν εμφανίζεται στο παράθυρο όταν ο πράκτορας δεν βρίσκεται στον ίδιο ξενιστή στον οποίο βρίσκεται η παραθυρική εφαρμογή.

Το σύστημα Odyssey παρέχει επίσης και ένα μηχανισμό για συνεργασία πρακτόρων δίνοντας τη δυνατότητα στους τελευταίους να καθορίζουν σημεία συνάντησης σε συγκεκριμένες μηχανές στο δίκτυο. Όμως η έλλειψη υποστήριξης μέσω εγγράφων καθώς και παραδειγμάτων δεν δίνει τη δυνατότητα στους προγραμματιστές να έχουν μια πλήρη εικόνα του τρόπου λειτουργίας του μηχανισμού αυτού. Ένας ακόμη μηχανισμός που παρέχει το σύστημα είναι αυτός της ανταλλαγής αναφορών (references) των δημοσιευμένων, κατά την ορολογία που εισάγει η κατασκευάστρια εταιρία, αντικειμένων. Ένας πράκτορας μπορεί να δημοσιεύσει οποιοδήποτε αριθμό αντικειμένων, δημιουργώντας ολικές αναφορές για αυτά, καταστώντας τα έτσι διαθέσιμα για χρήση από τους υπόλοιπους πράκτορες του ιδίου συστήματος πρακτόρων. Ο μηχανισμός της δημοσίευσης των αντικειμένων φαίνεται να είναι χρήσιμη ιδιότητα του συστήματος Odyssey αλλά η έλλειψη τεκμηρίωσης μέσω εγγράφων καθιστά την χρήση του δύσκολη από τους προγραμματιστές.

Γενικά, το σύστημα Odyssey παρέχει τη λειτουργικότητα που απαιτείτε για την υποστήριξη κινούμενων πρακτόρων επαυξημένη με τους μηχανισμούς που αναφέραμε στην παραπάνω παράγραφο. Όμως, η έλλειψη επαρκούς τεκμηρίωσης μέσω εγγράφων αλλά και παραδειγμάτων χρήσης των ιδιαίτερων μηχανισμών που παρέχει καθιστούν την εκμάθηση και τη χρήση του δύσκολη. Το σύστημα διανέμεται δωρεάν για χρήση από την ερευνητική κοινότητα.

2.2.2 Aglets της IBM

Από τα υπάρχοντα συστήματα πρακτόρων, το σύστημα Aglets είναι το περισσότερο διαδεδομένο. Σίγουρα υπάρχουν αρκετοί αναγνωρισμένοι ανταγωνιστές στο χώρο των κινούμενων πρακτόρων αλλά το σύστημα Aglets είναι περισσότερο δημοφιλές για τρεις λόγους:

- την ευκολία στην εγκατάσταση
- τα παραδείγματα των εφαρμογών των κινούμενων πρακτόρων είναι εντυπωσιακά (έχουν παραθυρικό περιβάλλον επικοινωνίας με τον χρήστη), και
- εκτός του γεγονότος ότι η κατασκευάστρια εταιρία είναι η IBM, το σύστημα Aglets είναι μία τεχνολογία κινούμενων πρακτόρων που φαίνεται σαν φυσική προέκταση της γλώσσας προγραμματισμού Java.

Ίσως το κυριότερο γνώρισμα της αρχιτεκτονικής του συστήματος να είναι ότι επιτρέπει στον αρχάριο χρήστη της γλώσσας προγραμματισμού Java να ασχοληθεί σχετικά εύκολα με την τεχνολογία των κινούμενων πρακτόρων αλλά και με την ανάπτυξη εφαρμογών κινούμενων πρακτόρων. Κάθε κομμάτι του συστήματος έχει εύχρηστο παραθυρικό περιβάλλον συνεργασίας με τον χρήστη. Το σύστημα Aglets μπορεί να εκτελεστεί σε μηχανές με λειτουργικό τύπου Unix και Windows, μέσα από την ιδεατή μηχανή τύπου Java. Το στάδιο της αρχικοποίησης περιλαμβάνει την αλλαγή μερικών μεταβλητών του περιβάλλοντος αλλά

και την έναρξη εκτέλεσης του εξυπηρετητή των κινούμενων πρακτόρων, που ονομάζονται aglets (το όνομα προκύπτει από συγχώνευση του όρου applet με τον όρο agent).

Ο εξυπηρετητής των κινούμενων πρακτόρων ονομάζεται Tahiti. Την πρώτη φορά που εκτελείται εμφανίζεται ένα παράθυρο που προτρέπει το χρήστη να γνωστοποιήσει τον εαυτό του στο σύστημα. Η πληροφορία αυτή ενσωματώνεται στους κινούμενους πράκτορες που δημιουργούνται από το σύστημα, έτσι ώστε να είναι γνωστή η ταυτότητα του δημιουργού τους από τα συστήματα που επισκέπτονται. Μετά την εγγραφή, εμφανίζεται το κυρίως παράθυρο του συστήματος που επιτρέπει την παρακολούθηση των πρακτόρων που φιλοξενούνται στο σύστημα σε πραγματικό χρόνο. Επίσης, παρέχει ένα σύνολο από μεθόδους που εξυπηρετούν τις λειτουργίες διαχείρισης. Επιπλέον, παρέχεται η δυνατότητα παρακολούθησης της χρήσης μνήμης, της κατάστασης εκτέλεσης των ινών, και των μηνυμάτων του συστήματος. Παρέχει ακόμη, μεθόδους για να θέτει ο χρήστης της προτιμήσεις του όσον αφορά τον τρόπο επικοινωνίας του συστήματος, αλλά και σε θέματα ασφάλειας. Έτσι, μπορεί να ορίσει ποιους πράκτορες επιθυμεί να φιλοξενήσει το σύστημά του, αλλά και τι προνόμια μπορούν να έχουν οι πράκτορες που φιλοξενούνται. Εν γένει, το παραθυρικό περιβάλλον επικοινωνίας με το χρήστη είναι λειτουργικότατο για την παρακολούθηση και τη διαχείριση μικρού αριθμού πρακτόρων. Σε αυτό το γεγονός δεν πρέπει να παραληφθεί η πλήρης τεκμηρίωση μέσω εγγράφων, αλλά και το σύνολο των παραδειγμάτων.

Σχετικά με τον σχεδιασμό του συστήματος πρέπει να τονιστεί η απλότητά του αλλά και η ομοιότητά του με παρόμοια κομμάτια που παρέχονται από το περιβάλλον προγραμματισμού τις Java. Για παράδειγμα, οι πράκτορες έχουν πολλές ομοιότητες στην δομή με τις οντότητες τύπου applets, όπως αυτές ορίζονται από τη γλώσσα Java. Το σύστημα παρέχει επίσης μηχανισμό για μεταβίβαση μηνυμάτων μεταξύ των πρακτόρων. Η αποστολή των μηνυμάτων μπορεί να είναι σύγχρονη και ασύγχρονη. Επιτρέπει επιπλέον, την ανάκτηση της πληροφορίας του δημιουργού των πρακτόρων.

Στον αντίποδα πρέπει να αναφερθεί η έλλειψη επαρκούς τεκμηρίωσης για την συγγραφή καινούργιου εξυπηρετητή πρακτόρων καθώς και το γεγονός ότι ο κύριος αρχιτέκτονας του συστήματος μετακόμισε στην εταιρία General Magic.

2.2.3 Voyager της ObjectSpace

Το σύστημα Voyager κυκλοφορεί για λειτουργικά συστήματα τύπου Unix και Windows. Χρησιμοποιεί την ιδεατή μηχανή της Java για να εκτελεστεί μια και στηρίζει την εκτέλεσή του στους μηχανισμούς της σειριοποίησης και του reflection. Στον προγραμματιστή παρέχεται επίσης και μεγάλος αριθμός εγγράφων για την τεκμηρίωση του συστήματος, που πρέπει να ειπωθεί ότι είναι η πληρέστερη των υπαρχόντων συστημάτων κινούμενων πρακτόρων. Υπάρχουν επιπλέον και αρκετά παραδείγματα τα περισσότερα από τα οποία είναι πολύ απλά, αλλά επεξηγούν ακριβώς συγκεκριμένους μηχανισμούς του συστήματος. Το εγχειρίδιο χρήσης του συστήματος οδηγεί το χρήστη μέσα στο σύστημα, επιτρέποντάς του να γνωρίσει όλους τους μηχανισμούς μέσα από απλά παραδείγματα κώδικα. Ο λόγος της αναλυτικής αυτής επεξήγησης του συστήματος είναι να μπορέσει ο τελικός χρήστης να κατανοήσει τις καινούργιες έννοιες και μηχανισμούς που εισάγονται από την κατασκευάστρια εταιρία, όπως ιδεατά αντικείμενα, υποδομή για επικοινωνία των πρακτόρων και τρόποι ελέγχου της δράσης των πρακτόρων.

Το ιδεατό αντικείμενο παίζει το ρόλο του μεσάζοντα, του ενδιάμεσου αντικειμένου (proxy) μιας απομακρυσμένης οντότητας ή και πράκτορα. Συστήματα που στηρίζουν την επικοινωνία μεταξύ των οντοτήτων που ορίζουν, σε μηχανισμούς παρόμοιους με τον μηχανισμό της απομακρυσμένης κλήσης διαδικασίας (Remote Procedure Call), όπως είναι ο μηχανισμός της

απομακρυσμένης επίκλησης μεθόδου (Remote Method Invocation) που προσφέρει η γλώσσα προγραμματισμού της Java, απαιτούν από τον προγραμματιστή να ορίσει πρώτα την διεπαφή επικοινωνίας και μετά την υλοποίηση οποιουδήποτε αντικειμένου. Σε αντίθεση, το σύστημα Voyager δημιουργεί, με ένα εργαλείο που έχει αναπτυχθεί και αποστέλλεται μαζί με αυτό, ένα ιδεατό αντικείμενο που είναι καθρέπτης του πραγματικού αντικειμένου. Έτσι, αποφεύγονται από το σύστημα αυτό τα στάδια που επιφέρει η χρησιμοποίηση του μηχανισμού της απομακρυσμένης επίκλησης μεθόδου. Για παράδειγμα, ας υποθέσουμε ότι έχουμε την κλάση που ορίζεται από το αρχείο A.class. Με το εργαλείο που παρέχεται, δημιουργείται η κλάση που περιγράφεται στο αρχείο VA.class και είναι το ιδεατό αντικείμενο της αρχικής κλάσης. Χρησιμοποιώντας το ιδεατό αυτό αντικείμενο μπορεί κανείς να δημιουργήσει νέα στιγμιότυπα αντικειμένων που περιγράφονται από την κλάση A.class, να επικοινωνήσει με αυτά αλλά και να τα μετακινήσει πάνω από το δίκτυο. Δηλαδή, το αντικείμενο που προκύπτει από την επεξεργασία εμπλουτίζεται με ιδιότητες που φέρουν οι πράκτορες. Βέβαια η διαφορά με τους τελευταίους είναι ότι το ιδεατό αντικείμενο δεν έχει την δική του ίνα εκτέλεσης. Είναι ένα παθητικό αντικείμενο που μεταφέρεται από σύστημα σε σύστημα. Μπορεί όμως να σχεδιαστεί για να παρέχει την ιδιότητα αυτή.

Ο μηχανισμός επικοινωνίας που προσφέρει το σύστημα είναι πολύ ευέλικτος. Έχει την δυνατότητα για σύγχρονη και ασύγχρονη επικοινωνία αλλά και επίκληση απομακρυσμένων μεθόδων. Η επίκληση μεθόδων οντοτήτων που φιλοξενούνται από διαφορετικά μηχανήματα στο δίκτυο γίνεται με την ίδια σημασιολογία που γίνεται και η επίκληση μεθόδων σε τοπικά οντότητες. Η ενσωμάτωση των αντικειμένων μέσα στα ιδεατά αντικείμενα επιτρέπει την πρόσβαση σε τοπικά και απομακρυσμένα αντικείμενα να γίνεται με την ίδια ακριβώς σύνταξη. Δηλαδή, επιτρέπει την χρήση των αντικειμένων σαν κατανεμημένα αντικείμενα. Από την άλλη πλευρά, ο μηχανισμός που ευθύνεται για την μετακίνηση των αντικειμένων είναι πρωτοποριακός. Παρέχεται ένας εξυπηρετητής κινούμενων πρακτόρων για την φιλοξενία των αντικειμένων αλλά σε αντίθεση με τα άλλα συστήματα πρακτόρων δεν είναι απαραίτητη η ύπαρξή του σε όλους τους ξενιστές. Και αυτό γιατί τα ιδεατά αντικείμενα μπορούν να μεταφέρονται όχι μόνο μεταξύ των εξυπηρετητών αλλά και μεταξύ των ιδεατών μηχανών που χρησιμοποιούν οι εφαρμογές σε Java σαν περιβάλλον εκτέλεσης.

Βέβαια, όπως όλα τα συστήματα πρακτόρων, το σύστημα Voyager έχει και τα μειονεκτήματά του. Η κύρια πηγή των μειονεκτημάτων είναι η μετατροπή των αντικειμένων σε ιδεατά αντικείμενα. Κατά πρώτον, η χρήση των ιδεατών αντικειμένων αλλάζει τις τεχνικές υλοποίησης που χρησιμοποιούν οι περισσότεροι προγραμματιστές. Η μετατροπή του αντικειμένου σε ιδεατό αντικείμενο απαιτεί τη χρησιμοποίηση συγκεκριμένου εργαλείου αλλά και την ορθότητα στη σύνταξη του αρχείου που περιγράφει το αντικείμενο αυτό. Επιπλέον, όταν το αντικείμενο περιέχει αναφορές σε άλλα αντικείμενα απαιτείται η μετατροπή και των αντικειμένων αυτών σε ιδεατά αντικείμενα. Λόγω του γεγονότος ότι οι αναφορές μπορούν να έχουν απεριόριστο βάθος ο γράφος των εξαρτήσεων για κάποιο αντικείμενο μπορεί να είναι πολύπλοκος.

2.2.4 Concordia της Mitsubishi

Το σύστημα Concordia έχει υλοποιηθεί με γλώσσα προγραμματισμού Java και η εκτέλεσή του εξαρτάται αποκλειστικά στο μηχανισμό της απομακρυσμένης επίκλησης μεθόδου (JavaRMI) σαν πρωτόκολλο τόσο για την μεταφορά των οντοτήτων όσο και για την επικοινωνία μεταξύ τους. Όπως και όλα τα παραπάνω συστήματα χρησιμοποιεί το περιβάλλον προγραμματισμού JDK1.1. Μία ειδοποιός διαφορά με τα υπόλοιπα συστήματα είναι ο τμηματικός σχεδιασμός του. Το σύστημα Concordia αποτελείται από πολλές οντότητες κάθε μία από τις οποίες αναλαμβάνει την εξυπηρέτηση συγκεκριμένης εργασίας.

Το σημαντικότερο κομμάτι του συστήματος, ονομάζεται Conduit Server, και είναι η οντότητα που παρέχει την υποδομή για την εξυπηρέτηση των αναγκών της επικοινωνίας αλλά και το περιβάλλον εκτέλεσης των πρακτόρων. Οι εξυπηρετητές αυτοί, ένας ή και περισσότεροι εκτελούνται σε κάθε κόμβο του δικτύου που επιθυμεί να φιλοξενήσει πράκτορες.

Υπάρχει επίσης η οντότητα διαχείρισης του συστήματος, που παρέχοντας μία διεπαφή χρήσης επιτρέπει την διαχείρισης των υπόλοιπων οντοτήτων του συστήματος. Μόνο μία τέτοια οντότητα είναι απαραίτητη για κάθε δίκτυο από εξυπηρετητές (Conduit Servers) που παρέχει το σύστημα Concordia. Η οντότητα αυτή παρέχει κεντρικοποιημένο (centralized) μηχανισμό ελέγχου και διαχείρισης τόσο των συστημάτων όσο και των πρακτόρων που δρουν στο δίκτυο των εξυπηρετητών. Η ύπαρξη της οντότητας διαχείρισης με τα χαρακτηριστικά αυτά δείχνει καθαρά ότι το σύστημα Concordia απευθύνεται για χρήση κυρίως από επιχειρήσεις όπου ο κεντρικοποιημένος μηχανισμός ελέγχου είναι επιθυμητός και μερικές φορές απαραίτητος.

Μία ακόμη λειτουργία του συστήματος, που είναι ιδιαίτερα σημαντική για το κοινό στο οποίο απευθύνεται το σύστημα, είναι η υποστήριξη της σταθερότητας των πρακτόρων αλλά και η ανοχή του σε σφάλματα. Δύο είναι οι οντότητες που συνεργάζονται για την αξιοπιστία του συστήματος. Η μία είναι υπεύθυνη για τη διατήρηση της πληροφορίας που αφορά την κατάσταση των πρακτόρων και η άλλη είναι υπεύθυνη για θέματα που σχετίζονται με τη μεταφορά των πρακτόρων. Και οι δύο οντότητες συνεργάζονται με τον εξυπηρετητή του συστήματος προκειμένου να μπορούν να παρέχουν την επιθυμητή αξιοπιστία και ανοχή σε λάθη. Ο βαθμός αξιοπιστίας του συστήματος μπορεί να καθοριστεί μέσω της οντότητας διαχείρισης.

Το σύστημα Concordia παρέχει επίσης υποστήριξη για συνεργασία πρακτόρων. Η οντότητα για τη διαχείριση των γεγονότων, με την ονομασία Event Manager, υλοποιεί ένα ασύγχρονο κατανεμημένο μοντέλο για γεγονότα, που επιτρέπει στους πράκτορες την αποστολή γεγονότων (post events) αλλά και την εγγραφή για λήψη γεγονότων (listen for events) που στάλθηκαν από άλλους πράκτορες. Παρέχεται επιπλέον και τρόπος για την ομαδοποίηση των πρακτόρων και τη μεταξύ τους συνεργασία, που είναι ανάλογος με τον μηχανισμό καθορισμού σημείου συνάντησης πρακτόρων του συστήματος Odyssey. Επιπρόσθετα, δίνεται η δυνατότητα στους πράκτορες για απευθείας επικοινωνία μεταξύ τους, χωρίς τη χρήση της οντότητας διαχείρισης γεγονότων. Οι μηχανισμοί που παρέχει το σύστημα Concordia για τη συνεργασία των πρακτόρων του δίνουν την πρωτοπορία στο θέμα αυτό.

Τελειώνοντας, πρέπει να αναφερθεί και η προσοχή των κατασκευαστών σε θέματα ασφάλειας. Είναι από τα λίγα συστήματα που ενσωμάτωνουν θέματα ασφάλειας στο σχεδιασμό τους. Το σύστημα υποστηρίζει την προστασία των πρακτόρων κατά τη διάρκεια της φιλοξενίας τους, ενώ το θέμα της προστασίας κατά τη διάρκεια της μετακίνησης των τελευταίων αφήνεται στη χρήση του secure socket layer. Το σύστημα επιπρόσθετα σχετίζει με κάθε πράκτορα μία ταυτότητα που υποδηλώνει τον κατασκευαστή του. Έτσι, γίνεται άμεση η απόδοση των δικαιωμάτων στους πράκτορες. Παρέχει επίσης και υποστήριξη για μηχανισμούς πιστοποίησης αυθεντικότητας.

2.2.5 Συζήτηση

Μία πρώτη παρατήρηση που ισχύει όχι μόνο για τα παραπάνω περιγραφόμενα συστήματα είναι ότι στο σύνολό τους υποστηρίζουν με κάποιο διαφορετικό μηχανισμό την μετακίνηση και την διαχείριση των οντοτήτων που αυτά ορίζουν. Μία δεύτερη παρατήρηση, που είναι ακόμη πιο σημαντική σχετίζεται με τους μηχανισμούς αλληλεπίδρασης των πρακτόρων που προσφέρουν. Όλα τα συστήματα προσπαθούν να παρέχουν μηχανισμούς συνεργασίας για τις

οντότητες που ορίζουν, χωρίς βέβαια να το επιτυγχάνουν όλα το ίδιο καλά. Έτσι το κάθε σύστημα καταφέρνει να παρέχει τις βασικές λειτουργίες που χρειάζονται οι πράκτορες για να δραστηριοποιηθούν και να μετακινηθούν. Από την άλλη πλευρά, η τεχνολογία των κινούμενων πρακτόρων, που έρχεται, όπως προσδοκούν οι ερευνητές που ασχολούνται με την αυτήν, να λύσει σημαντικό μέρος των προβλημάτων που παρουσιάζονται στις συναλλαγές κάθε είδους (π.χ. πληροφορίας, προϊόντων) στον Παγκόσμιο Ιστό απαιτεί την εκτενή αλληλεπίδραση πρακτόρων διαφορετικών κατασκευαστών. Για το σκοπό αυτό είναι απαραίτητη η τυποποίηση μερικών θεμάτων της τεχνολογίας, έτσι ώστε και να διατηρηθεί η ιδιαιτερότητα των συστημάτων άλλα και να προαχθεί η διαλειτουργικότητά τους.

2.3 Τυποποίηση

Το πρώτο πρότυπο σε θέματα που αφορούν την τεχνολογία των κινούμενων πρακτόρων ονομάζεται Mobile Agent System Interoperability Facility (MASIF) και προέρχεται από την ομάδα Object Management Group (OMG). Το πρότυπο υιοθετήθηκε τον Φεβρουάριο του 1998. Ασχολείται με αρκετά σημαντικά θέματα όπως διαχείριση, κινητικότητα, ονομασία και εντοπισμό των πρακτόρων. Έτσι επιτυγχάνεται η διασφάλιση της ελάχιστης διαλειτουργικότητας που πρέπει να χαρακτηρίζει τα συστήματα κινούμενων πρακτόρων. Πιθανότατα, η ίδια ομάδα να ξεκινήσει τη διαδικασία για τυποποίηση και άλλων θεμάτων που σχετίζονται με τους κινούμενους πράκτορες. Το σύνολο των προτύπων που θα προκύψει θα συντελεί στη δημιουργία ενός ευρέως αποδεκτού περιβάλλοντος για ανάπτυξη και χρησιμοποίηση συστημάτων κινούμενων πρακτόρων.

2.4 Συστήματα πρακτόρων βάση προτύπου

2.4.1 Περιβάλλον Grasshopper

Το πρώτο σύστημα κινούμενων πρακτόρων, που έχει αναπτυχθεί όπως το πρότυπο ορίζει, ονομάζεται Grasshopper και προέρχεται από την ομάδα IMA-CC του ινστιτούτου GMD. Η υλοποίηση αυτή παρέχει συστήματα πρακτόρων που υποστηρίζουν όλες τις λειτουργίες που ορίζει το πρότυπο. Επιπλέον, ενισχύει τα συστήματα με

- δυνατότητα υποστήριξης πολλαπλών τρόπων επικοινωνίας, που είναι το RMI της Java, το IIOP που ορίζεται από την αρχιτεκτονική CORBA, επικοινωνία μέσω sockets η οποία μπορεί να χρησιμοποιεί ως μηχανισμό προστασίας το Secure Socket Layer (SSL).
- δυνατότητα σύγχρονης (synchronous) και ασύγχρονης (asynchronous) επικοινωνίας των πρακτόρων με τα συστήματα που τους φιλοξενούν, αλλά και δυνατότητα ταυτόχρονης αποστολής πολλαπλών μηνυμάτων (multicast) σε ομάδες συστημάτων.
- υποστήριξη μηχανισμών για την εξωτερική ασφάλεια των συστημάτων, όπως μηχανισμός παροχής κωδικών πιστοποίησης γνησιότητας των μηνυμάτων (Message Authentication Codes).
- μηχανισμό σταθερότητας για την ορθή λειτουργίας του συστήματος, ώστε να μπορεί να αντιμετωπίζει τυχών σφάλματα στην εκτέλεσή του.

Το σύστημα διανέμεται με πλήρη λειτουργικότητα ως εμπορικό προϊόν αλλά και με κάποια βασική λειτουργικότητα ως προϊόν χωρίς επιβάρυνση.

2.4.2 Περιβάλλον EasyAgent

Το σύστημα που περιγράφεται στην εργασία αυτή είναι το δεύτερο σύστημα κινούμενων πρακτόρων που είναι συμβατό με το πρότυπο MASIF. Ονομάζεται EasyAgent και διασφαλίζει τη διαλειτουργικότητα των συστημάτων πρακτόρων όπως αυτή ορίζεται από το πρότυπο. Επιπλέον, παρέχει ένα προκαθορισμένο τρόπο επικοινωνίας των πρακτόρων Προσφέρει επιπρόσθετα και περιβάλλον εκτέλεσης πρακτόρων πάνω από πλοηγητές διαδικτύου. Η αναλυτική περιγραφή του συστήματος περιέχεται σε επόμενο κεφάλαιο. Πριν όμως προχωρήσουμε σε θέματα υλοποίησης παραθέτουμε ένα κεφάλαιο που επεξηγεί αναλυτικότερα το πρότυπο.

3 Το πρότυπο

Η έρευνα σε θέματα που αφορούν την τεχνολογία των κινούμενων πρακτόρων άρχισε σχετικά πρόσφατα, με συνέπεια η ανάπτυξη συστημάτων και εφαρμογών της ίδιας τεχνολογίας να είναι σε ακόμη πιο πρώιμα στάδια. Για το λόγο αυτό τα υπάρχοντα συστήματα κινούμενων πρακτόρων παρουσιάζουν σημαντικές διαφορές σε θέματα σχετικά με την αρχιτεκτονική του συστήματος αλλά και την υλοποίησή του. Οι διαφορές αυτές απέτρεψαν τη διαλειτουργικότητα (interoperability) των συστημάτων αλλά και την εξάπλωση της τεχνολογίας των πρακτόρων. Για να προαχθεί τόσο η διαλειτουργικότητα όσο και η διαφοροποίηση μεταξύ των συστημάτων κινούμενων πρακτόρων, μερικά θέματα που αφορούν την τεχνολογική τους υποδομή πρέπει να τυποποιηθούν.

Στο κεφάλαιο αυτό υπάρχει μια ανάλυση θεμάτων που έχουν τυποποιηθεί μετά από μία συλλογική υποβολή πρότασης με τίτλο MASIF (Mobile Agent System Interoperability Facilities Specification) από την ομάδα IMA-CC (Intelligent Mobile Agent Center of Competence) του GMD (GMD National Research Center for Information Technology) σε συνεργασία με τις επιχειρήσεις Crystaliz, General Magic, IBM και την ομάδα The Open Group. Η πρόταση αυτή κατατέθηκε ως απάντηση σε μία πρόσκληση για εκδήλωση πρότασης, από την ομάδα OMG MAF(Open Managment Group Mobile Agent Facility), με αντικείμενο τον τρόπο που μπορούν να ορισθούν πράκτορες και συστήματα πρακτόρων, που να έχουν ως βάση της αρχιτεκτονικής τους την αρχιτεκτονική CORBA και η τελική της αποδοχή έγινε το φθινόπωρο του 1997. Το κεφάλαιο ξεκινάει με μια ανάλυση του όρου διαλειτουργικότητα όσον αφορά τα συστήματα πρακτόρων και συνεχίζει αναλύοντας την ορολογία που εισάγει η τεχνολογία των κινούμενων πρακτόρων. Ακολουθεί μια παρουσίαση της συλλογής των ορισμών και περιγραφών που ορίζουν το πρότυπο.

3.1 Διαλειτουργικότητα

Ο σημαντικότερος σκοπός της τεχνολογίας των κινούμενων πρακτόρων είναι η διαφάνεια, η συμβατότητα, ή ακόμη γενικότερα η διαλειτουργικότητα μεταξύ των συστημάτων πρακτόρων που παρέχονται από διαφορετικούς κατασκευαστές. Ο σκοπός αυτός είναι δυνατό να επιτευχθεί εάν λειτουργίες, όπως μεταφορά πρακτόρων και κλάσεων και διαχείριση πρακτόρων, είναι τυποποιημένες. Αυτές ακριβώς οι λειτουργίες τυποποιήθηκαν στο πρότυπο που περιγράφετε στο κεφάλαιο αυτό.

Το πρότυπο MASIF διασφαλίζει τη διαλειτουργικότητα μεταξύ συστημάτων υλοποιημένων χρησιμοποιώντας την ίδια γλώσσα προγραμματισμού αλλά τυχόν από διαφορετικούς κατασκευαστές. Δεν ασχολείται με το θέμα της διαφορετική γλώσσα προγραμματισμού και της τυποποίησης λειτουργιών που αφορούν την αλληλεπίδραση των πρακτόρων με το σύστημα όπως μετάφραση (interpretation), σειριοποίηση (serialization), εκτέλεση (execution) ή αποσειριοποίηση (deserialization) των πρακτόρων.

3.1.1 Διασφάλιση διαλειτουργικότητας

Οι περιοχές της τεχνολογίας των κινούμενων πρακτόρων που είναι απαραίτητο να τυποποιηθούν για τη διασφάλιση της διαλειτουργικότητας των συστημάτων είναι:

- διαχείριση των πρακτόρων
- μεταφορά των πρακτόρων
- ονομασία πρακτόρων και συστημάτων
- ο τύπος του συστήματος πρακτόρων
- η σύνταξη του ονόματος της τοποθεσίας

Στην ενότητα αυτή θα αναλυθούν οι λόγοι που συνιστούν στην τυποποίηση των συγκεκριμένων περιοχών.

Διαχείριση πρακτόρων
Ο διαχειριστής συστήματος (άνθρωπος ή και πρόγραμμα) θα πρέπει να έχει τη δυνατότητα να χειρίζεται τα συστήματα πρακτόρων μέσα από ένα σύνολο τυποποιημένων λειτουργιών. Θα πρέπει να έχει τη δυνατότητα να δημιουργεί έναν πράκτορα δίνοντας το όνομα της κλάσης του, να διακόπτει προσωρινά (suspend), να ξαναρχίζει (resume) ή και να τερματίζει την εκτέλεση ενός πράκτορα. Η τυποποίηση των λειτουργιών διαχείρισης προσφέρει στο χειριστή ένα κοινό τρόπο αλληλεπίδρασης με όλα τα συστήματα.

Μεταφορά πρακτόρων
Είναι χρήσιμο να μπορούν δύο πράκτορες να επικοινωνούν βρισκόμενοι στην ίδια τοποθεσία παρά διαμέσου του δικτύου για δύο λόγους. Ο πρώτος είναι ότι περιορίζεται σημαντικά ο αριθμός των δικτυακών δοσοληψιών μια και το μόνο δικτυακό κόστος είναι η μεταφορά του πράκτορα. Περιορίζεται, δηλαδή, στο ελάχιστο δυνατόν το πέρασμα μηνυμάτων πάνω από το δίκτυο που συνεπάγεται σε μικρότερη καθυστέρηση στην επικοινωνία. Ο δεύτερος λόγος είναι ότι η τοπικότητα στην επικοινωνία ευνοεί λειτουργίες που σχετίζονται με παρακολούθηση δεδομένων (π.χ. το να περιμένεις για κάποια μετοχή να φτάσει κάποια συγκεκριμένη τιμή μπορεί να διαρκέσει και δέκα μέρες). Η τυποποίηση της λειτουργίας αυτής προσφέρει στους πράκτορες τη δυνατότητα μεταφοράς μεταξύ των συστημάτων και αυξάνει σημαντικά την απόδοση του συστήματος στην ολότητά του.

Ονομασία των πρακτόρων και των συστημάτων
Εκτός από τις λειτουργίες που πρέπει να τυποποιηθούν, πρέπει να τυποποιηθεί η σημασιολογία και η σύνταξη μερικών παραμέτρων. Στις παραμέτρους αυτές ανήκουν το όνομα πρακτόρων, συστημάτων και τοποθεσιών.

Η τυποποίηση των λειτουργιών της διαχείρισης των κινούμενων πρακτόρων καθιστά απαραίτητη και την τυποποίηση της ονοματολογίας των πρακτόρων και των συστημάτων βάση του γεγονότος ότι αποτελούν δομικά στοιχεία της ταυτότητας του υποκείμενου πράκτορα. Η τυποποίηση της ονοματολογίας επιτρέπει τα συστήματα κινούμενων πρακτόρων άμεσα να καθορίζουν εάν μπορούν να δεχτούν ή όχι έναν πράκτορα (συμβατότητα μεταξύ συστημάτων).

Ο τύπος του συστήματος πρακτόρων
Είναι σημαντικό να υπάρχει κάποιος οργανισμός που να παρέχει μοναδική ταυτότητα σε κάθε σύστημα κινούμενων πρακτόρων διαφορετικού κατασκευαστή. Η διασφάλιση της μοναδικότητας αποτρέπει το γεγονός της εμφάνισης συστημάτων με τον ίδιο τύπο, από διαφορετικές εταιρίες. Η τυποποίηση του τύπου αλλά και η χρήση του ως συστατικό του ονόματος των συστημάτων επιτρέπει την αναγνώριση συστημάτων και πρακτόρων με χρήση μόνο του ονόματός τους.

Η σύνταξη του ονόματος της τοποθεσίας
Οι πράκτορες κατά την διάρκεια της ζωής τους πρέπει να μπορούν να έχουν πρόσβαση σε στην πληροφορία του ονόματος του κάθε συστήματος. Αυτό καθίσταται δυνατόν μόνο όταν υπάρχει τυποποιημένος τρόπος δήλωσης της θέσης του κάθε συστήματος.

3.1.2 Συζήτηση

Στην προηγούμενη ενότητα παρουσιάστηκαν ορισμένα θέματα της τεχνολογίας των κινούμενων πρακτόρων που πρέπει να τυποποιηθούν για την διασφάλιση της διαλειτουργικότητας. Υπάρχουν και άλλα ζητήματα που πρέπει να τυποποιηθούν όπως το

θέμα της ασφάλειας και το θέμα της σειριοποίησης του κώδικα και της κατάστασης εκτέλεσης (execution state) των πρακτόρων. Η τυποποίηση των θεμάτων αυτών πρέπει να γίνει αφού πρώτα ωριμάσει η βιομηχανία και συγγενή πρότυπα αρχίσουν να ισχύουν.

Όταν ένας πράκτορας ταξιδεύει σε περισσότερο από δύο περιοχές ξεχωριστής ασφάλειας, το θέμα της ασφάλειας γίνεται περίπλοκο. Η πλειοψηφία των ασφαλών συστημάτων σήμερα ασχολείται μόνο με το θέμα της ασφάλειας μεταξύ δύο περιοχών. Η τυποποίηση του θέματος της ασφάλειας πρέπει να γίνει αφού πρώτα υπάρξει λύση του προβλήματος από τα ασφαλή συστήματα.

Τα υπάρχοντα συστήματα κινούμενων πρακτόρων έχουν υλοποιηθεί σε διαφορετικές γλώσσες προγραμματισμού (π.χ. Java, Tcl). Για τον λόγο αυτό θα ήταν περίπλοκη μια προσπάθεια μετατροπής της κωδικοποίησης ενός πράκτορα. Όταν η διαδικασίας της σειριοποίηση του κώδικα και της κατάστασης εκτέλεσης ενός πράκτορα γίνει κοινή για όλες τις γλώσσες προγραμματισμού που χρησιμοποιούνται θα είναι δυνατή η κατασκευή τυποποιημένων γεφυρών μεταξύ των συστημάτων πρακτόρων.

3.1.3 Διαλειτουργικότητα όπως διασφαλίζεται από το πρότυπο

Στην ενότητα αυτή παρουσιάζονται θέματα της διαλειτουργικότητας έτσι όπως τυποποιούνται από το πρότυπο καθώς και η πολυπλοκότητα που χρειάζεται για την υποστήριξή τους.

Η διαχείριση των πρακτόρων επιτρέπει ένα σύστημα να ελέγχει τους πράκτορες που προέρχονται από ένα άλλο σύστημα. Το πρότυπο διευθυνσιοδοτεί το θέμα αυτό παρέχοντας τυποποιημένο σύνολο από λειτουργίες που σχετίζονται με τον έλεγχο των πρακτόρων όπως αναστολή, επανέναρξη και τερματισμό της εκτέλεσης ενός πράκτορα. Η πολυπλοκότητα που επιβαρύνει το σύστημα είναι σχετικά μικρή.

Ο εντοπισμός ενός πράκτορα, που επιτυγχάνεται μέσω της τυποποίησης του ονόματος των πρακτόρων καθώς και της τυποποίησης του ονόματος της τοποθεσίας. Η πολυπλοκότητα που επιβαρύνει το σύστημα είναι και σε αυτήν την περίπτωση σχετικά μικρή.

Το πρότυπο τυποποιεί και το θέμα της μεταφοράς των πρακτόρων ορίζοντας μεθόδους για την παραλαβή πρακτόρων (παραλαβή κατάστασης εκτέλεσης ή και παραμέτρων εκτέλεσης) και για την παραλαβή των κλάσεών τους. Η διαφάνεια στη μεταφορά των πρακτόρων είναι αρκετά δύσκολο να επιτευχθεί και χρειάζεται την συντονισμένη προσπάθεια των διαφόρων κατασκευαστών συστημάτων κινούμενων πρακτόρων.

Το πρότυπο δεν ασχολείται καθόλου με θέματα που σχετίζονται με την επικοινωνία των πρακτόρων. Το θέμα αυτό καλύπτεται από την υποδομή πάνω στην οποία στηρίζεται το περιβάλλον κινούμενων πρακτόρων ή και από το περιβάλλον που χρησιμοποιείται για την ανάπτυξη των εφαρμογών κινούμενων πρακτόρων. Ο παρακάτω πίνακας περιγράφει τις μορφές διαλειτουργικότητας που διασφαλίζει το πρότυπο και μία εκτίμηση της πολυπλοκότητας, που απαιτήται από το σύστημα για την υποστήριξή τους.

Λειτουργία	Παρέχεται από το πρότυπο	πολυπλοκότητα
διαχείριση πρακτόρων	ΝΑΙ	μικρή
εντοπισμός πρακτόρων	ΝΑΙ	μικρή
μεταφορά πρακτόρων	ΝΑΙ	μεγάλη
επικοινωνία πρακτόρων	ΟΧΙ	-

Στις επόμενες ενότητες θα αναλυθεί με περισσότερη λεπτομέρεια ο τρόπος με τον οποίο διασφαλίζεται η διαλειτουργικότητα όταν είναι τυποποιημένες οι παραπάνω λειτουργίες.

3.2 Ορολογία

Πριν προχωρήσουμε στην περαιτέρω ανάλυση του προτύπου θα αναλύσουμε μερικές βασικές έννοιες που εισάγει η τεχνολογία των κινούμενων πρακτόρων.

Πράκτορας
Ο πράκτορας είναι ένα πρόγραμμα που δρα αυτόνομα με σκοπό την εξυπηρέτηση του δημιουργού του (άνθρωπος ή και πρόγραμμα). Την εκτέλεση του κάθε πράκτορα αναλαμβάνει μία ίνα (thread) και ανήκει αποκλειστικά σε αυτόν.

Στατικός πράκτορας
Είναι ο πράκτορας που εκτελείται μόνο στο σύστημα που άρχισε την εκτέλεση. Εάν χρειάζεται δεδομένα που δεν βρίσκονται στο σύστημα που εκτελείται ή εάν χρειάζεται να επικοινωνήσει με κάποιον άλλο πράκτορα που εκτελείται σε άλλο σύστημα ο πράκτορας αυτός κάνει χρήση κάποιου τρόπου επικοινωνίας όπως RPC.

Κινούμενος πράκτορας
Είναι ο πράκτορας που δεν περιορίζεται στο σύστημα που άρχισε την εκτέλεσή του. Έχει την ικανότητα να μεταφέρεται μεταξύ των υπαρχόντων συστημάτων. Η ικανότητα αυτή του επιτρέπει να επισκέπτεται τα αντικείμενα με τα οποία θέλει να αλληλεπιδράσει και να επωφελείται της τοπικότητας στην επικοινωνία. Παρόλο που τα σύγχρονα αντικειμενοστραφή κατανεμημένα συστήματα παρέχουν την απαραίτητη υποδομή για τις ανάγκες επικοινωνίας των στατικών πρακτόρων, δεν παρέχουν την υποδομή για τις ανάγκες των κινούμενων πρακτόρων.

Κατάσταση πράκτορα
Όταν ένας πράκτορας μετακινείται, μεταφέρει την κατάστασή (state) του και τον κώδικά του. Στον όρο κατάσταση ενός πράκτορα συμπεριλαμβάνεται τόσο η κατάσταση εκτέλεσης όσο και τιμές γνωρισμάτων που υποδηλώνουν την εργασία που έχει να διεκπεραιώσει ο πράκτορας όταν ξαναρχίσει την εκτέλεσή του στο σύστημα προορισμού.

Κατάσταση εκτέλεσης πράκτορα
Είναι η κατάσταση κατά τη διάρκεια εκτέλεσης ενός πράκτορα. Ο πράκτορας είναι ένα πρόγραμμα που εκτελείται πάνω σε ένα σύστημα. Για την εκτέλεση του προγράμματος χρησιμοποιείται μετρητής εντολών προγράμματος (program counter) και η στοίβα (frame stacks). Αυτή η πληροφορία που δείχνει την ακριβή κατάσταση εκτέλεσης του πράκτορα είναι η εν λόγω κατάσταση εκτέλεσης.

Δικαιοδοτική αρχή πράκτορα (agent authority)
Τον ρόλο της διαιοδοτικής αρχής ενός πράκτορα παίζει ο άνθρωπος ή ο οργανισμός τον οποίο ο πράκτορας εξυπηρετεί. Κάθε δικαιοδοτική αρχή πρέπει να μπορεί να πιστοποιεί την αυθεντικότητά της έτσι ώστε οι πράκτορες που δημιουργεί να δρουν σύμφωνα με τα προνόμια και τις άδειες της αρχής σε κάθε σύστημα που επισκέπτονται.

Όνομα πράκτορα
Οι πράκτορες είναι απαραίτητο να έχουν όνομα για να μπορούν να αναγνωρίζονται από τις λειτουργίες διαχείρισης αλλά και να εντοπίζονται μέσω υπηρεσιών ονοματολογίας (naming services). Το όνομα του πράκτορα είναι μία δομή που αποτελείται από το όνομα της δικαιοδοτικής αρχής, την ταυτότητα του πράκτορα (το πραγματικό όνομά του) και τον τύπο του συστήματος στον οποίο θα αναφερθούμε παρακάτω. Η ταυτότητα του πράκτορα πρέπει

να είναι μοναδική μέσα στα όρια της δικαιοδοτικής αρχής. Ο τρόπος αυτός ονομασίας των πρακτόρων, υπό ορισμένες συνθήκες, δίνει μια '1 προς 1' αντιστοίχηση του συνόλου των επιτρεπτών ονομάτων με το σύνολο των πρακτόρων. Για το λόγο αυτό το όνομα μπορεί να χρησιμοποιηθεί ως κλειδί σε λειτουργίες που αφορούν τον συγκεκριμένο πράκτορα.

Τοποθεσία πράκτορα
Η τοποθεσία ενός πράκτορα είναι η διεύθυνση του *μέρους* (αναλύεται παρακάτω) που εκτελείται ο πράκτορας. Το μέρος ανήκει σε ένα σύστημα πρακτόρων. Για το λόγο αυτό η τοποθεσία ενός πράκτορα πρέπει να είναι μία δομή που περιέχει το όνομα του συστήματος πρακτόρων και το όνομα του μέρους που διαμένει προσωρινά (κινούμενος πράκτορας) ή και μόνιμα (στατικός πράκτορας) ο πράκτορας. Εάν η τοποθεσία δεν περιέχει το όνομα του μέρους τότε εννοείται αυτόματα το προκαθορισμένο από το σύστημα μέρος.

Σύστημα πρακτόρων
Το σύστημα πρακτόρων είναι ένα περιβάλλον με τις δυνατότητες δημιουργίας, μετάφρασης, εκτέλεσης, μεταφοράς και τερματισμού πρακτόρων. Όπως οι πράκτορες έτσι και τα συστήματα πρακτόρων έχουν μία δικαιοδοτική αρχή. Για παράδειγμα, ένα σύστημα πρακτόρων που έχει ως δικαιοδοτική αρχή τον κύριο Μ. θα διανέμει του πόρους του συστήματος του κυρίου Μ. με τον τρόπο που εκείνος ορίζει.

Τύπος συστήματος πρακτόρων
Ο τύπος του συστήματος πρακτόρων περιγράφει το προφίλ του αλλά και το προφίλ των πρακτόρων μια και αποτελεί μέρος του ονόματος των τελευταίων. Για παράδειγμα, εάν ο τύπος του συστήματος είναι Concordia το σύστημα έχει υλοποιηθεί από τα εργαστήρια Horizon Systems της εταιρίας Mitsubishi Electric, υποστηρίζουν την Java σαν γλώσσα για πράκτορες και χρησιμοποιούν σαν μέθοδο σειριοποίησης την Java Object Serialization. Γνώση του τύπου του συστήματος σημαίνει αυτόματα και γνώση των ιδιομορφιών (τα είδη πρακτόρων που μπορούν να εκτελεστούν στο σύστημα) του συστήματος. Ο τύπος αυτός δηλαδή περιγράφει μοναδικά την λειτουργικότητά του κάθε συστήματος.

Μέρος
Όταν ο πράκτορας μεταφέρεται, ταξιδεύει μεταξύ περιβαλλόντων εκτέλεσης (execution environments) που ονομάζονται μέρη. Το μέρος είναι ένα γενικότερο πλαίσιο που υπάρχει στα συστήματα πρακτόρων για την εκτέλεση των πρακτόρων. Το σύστημα πρακτόρων μπορεί να έχει ένα ή και περισσότερα μέρη. Πάντα υπάρχει το προκαθορισμένο μέρος.

Η διεύθυνση (τοποθεσία) του μέρους αποτελείται από δύο τμήματα. Το όνομα του μέρους και τη διεύθυνση του συστήματος που παρέχει το μέρος. Όταν ένας πράκτορας ζητήσει να μάθει την τοποθεσία κάποιου άλλου πράκτορα παίρνει σαν απάντηση την τοποθεσία του μέρους στο οποίο ο πράκτορας αυτό εκτελείται.

Περιοχή
Η περιοχή είναι ένα σύνολο συστημάτων πρακτόρων που έχουν την ίδια δικαιοδοτική αρχή αλλά όχι απαραίτητα τον ίδιο τύπο συστήματος. Η έννοια της περιοχής επιτρέπει σε έναν άνθρωπο ή οργανισμό να παρέχει περισσότερα από ένα συστήματα πρακτόρων. Οι περιοχές συντελούν και στην επεκτασιμότητα (scalability) της τεχνολογίας των πρακτόρων μια και επιτρέπουν την εύκολη κατανομή του φορτίου στα συστήματα που τις αποτελούν.

Σειριοποίηση / Αποσειριοποίηση
Σειριοποίηση ονομάζεται η διαδικασία της αποθήκευσης ενός πράκτορα σε σειριοποιημένη μορφή. Αποσειριοποίηση είναι η διαδικασία της επαναφοράς ενός πράκτορα από την σειριοποιημένη μορφή. Η αποθήκευση και επιτυχής επαναφορά ενός πράκτορα γίνονται με αναπαράσταση της κατάστασης του πράκτορα σε μία τέτοια μορφή που να είναι επαρκής για

την ανακατασκευή του. Πρέπει να σημειωθεί ότι η σειριοποιημένη μορφή πρέπει να είναι ικανή να αναγνωρίζει τις κλάσεις από τις οποίες προέρχονται τα πεδία της.

Βασική τοποθεσία κώδικα (Codebase)
Η βασική τοποθεσία κώδικα καθορίζει το πού βρίσκονται οι κλάσεις που χρειάζεται ένας πράκτορας. Στην περίπτωση που ένα σύστημα πρακτόρων είναι υπεύθυνο για την παροχή των απαραίτητων κλάσεων το σύστημα αυτό ονομάζεται προμηθευτής κλάσεων (class provider).

Υποδομή για επικοινωνίες (communications infrastructure)
Με το όρο αυτό εννοείται η υποδομή που προσφέρει υπηρεσίες όπως μεταφορά μηνυμάτων, ονοματολογίας και ασφάλειας για τα συστήματα πρακτόρων.

Τοπικότητα
Η έννοιας της τοπικότητας όπως αυτή χρησιμοποιείται στην τεχνολογία των κινούμενων πρακτόρων σημαίνει το να είσαι *κοντά* στο σύστημα του ενδιαφέροντός σου είτε βρισκόμενος στο ίδιο μηχάνημα είτε στο ίδιο υποδίκτυο.

3.3 Αλληλεπίδραση πρακτόρων

Για τη διασφάλιση της διαλειτουργικότητας το πρότυπο ορίζει τους τρόπους αλληλεπίδρασης των πρακτόρων καθώς και τις λειτουργίες που πρέπει να παρέχει ένα σύστημα πρακτόρων. Όσον αφορά την αλληλεπίδραση των πρακτόρων ορίζει τρεις τύπους που είναι:

- απομακρυσμένη δημιουργία πράκτορα
- μεταφορά πράκτορα
- επίκληση μεθόδου του πράκτορα (agent method invocation)

3.3.1 Απομακρυσμένη δημιουργία πράκτορα

Η διαδικασία αυτή σχετίζεται με τον τρόπο που ένας πελάτης αλληλεπιδρά με το απομακρυσμένο σύστημα πρακτόρων με σκοπό τη δημιουργία ενός συγκεκριμένου πράκτορα και την έναρξη της εκτέλεσής του στο σύστημα αυτό. Στον ρόλο του πελάτη μπορεί να είναι είτε ένα πρόγραμμα ή ένας άλλος πράκτορας.

Ο πελάτης, κατ' αρχήν, πιστοποιεί τη γνησιότητά του στο απομακρυσμένο σύστημα και ορίζει τη δικαιοδοτική αρχή του πράκτορα που θέλει να δημιουργήσει. Επιπλέον, εάν χρειάζεται μπορεί να τροφοδοτήσει το σύστημα με τις παραμέτρους αρχικοποίησης του πράκτορα καθώς και τις κλάσεις που χρειάζονται για την εκτέλεσή του.

3.3.2 Μεταφορά πράκτορα

Όταν ένας πράκτορας θέλει να μετακομίσει σε ένα άλλο σύστημα πρακτόρων, εκδηλώνει την επιθυμία του στο σύστημα που τον φιλοξενεί, το οποίο με την σειρά του αποστέλλει μία αίτηση μετακόμισης στο σύστημα προορισμού. Ο πράκτορας παρέχει τις πληροφορίες που χρειάζονται για την εύρεση του συστήματος προορισμού όπως όνομα ή και διεύθυνση. Ο πράκτορας μπορεί να ζητήσει συγκεκριμένη ποιότητα στην εξυπηρέτηση της αίτησης μεταφοράς του. Αυτό βέβαια δεν παρέχεται από το πρότυπο αλλά υπάρχει η απαραίτητη υποδομή για την υποστήριξή του από τους υλοποιητές.

Εάν είναι επιτυχής η προσπάθεια του συστήματος, που φιλοξενεί τον πράκτορα, να επικοινωνήσει με το σύστημα προορισμού το τελευταίο αναλαμβάνει την εξυπηρέτηση της αίτησης μεταφοράς και την ευθύνη να απαντήσει σε περίπτωση αποτυχίας. Από την άλλη

μεριά, εάν δεν καταφέρει να επικοινωνήσει με το σύστημα προορισμού πρέπει το σύστημα να επιστρέψει μία ένδειξη αποτυχίας.

Κατά τη μεταφορά ενός πράκτορα η κατάσταση εκτέλεσης, η δικαιοδοτική αρχή καθώς και ο κώδικάς του μεταφέρεται στο σύστημα προορισμού. Η μεταφορά του κώδικα δεν είναι πάντοτε απαραίτητη. Είναι στην κρίση του συστήματος πρακτόρων αν θα στείλει τις κλάσεις που χρειάζονται για την εκτέλεση του πράκτορα ή αν θα τις στείλει αργότερα όταν αυτές ζητηθούν από το σύστημα προορισμού.

3.3.3 Επίκληση μεθόδων πράκτορα

Ο πράκτορας, εάν έχει ένα αναφορικό δείκτη σε κάποιο αντικείμενο και μπορεί να πιστοποιήσει την γνησιότητά του σε αυτό, έχει την δυνατότητα να καλέσει τις μεθόδους του αντικειμένου αυτού. Το πρότυπο δεν εμβαθύνει περισσότερο στο θέμα αυτό μια και στηρίζει την υποστήριξη του τρόπου αυτού αλληλεπίδρασης των πρακτόρων στη χρησιμοποιούμενη υποδομή για επικοινωνίες.

3.4 Λειτουργίες του συστήματος πρακτόρων

Κάθε σύστημα πρακτόρων πρέπει να προσφέρει ένα συγκεκριμένο σύνολο λειτουργιών ώστε να επιτυγχάνεται η διαφάνεια στη χρήση τους και η διαφοροποίηση στην υλοποίησή τους. Στο σύνολο των λειτουργιών αυτών περιέχονται:

- η μεταφορά των πρακτόρων, που είναι δυνατόν να περιλαμβάνει και την μεταφορά των κλάσεων
- δημιουργία ενός πράκτορα
- η παροχή μοναδικών ονομάτων πρακτόρων και τοποθεσιών
- η εύρεση πρακτόρων
- η εξασφάλιση ενός ασφαλούς περιβάλλοντος για τη δράση των πρακτόρων

Στην ενότητα αυτή θα αναλυθεί σε βάθος το σύνολο των παρεχόμενων λειτουργιών. Όπου είναι απαραίτητο να αναφερθούν νέοι ορισμοί και εκφράσεις, όπως η τεχνολογία των κινούμενων πρακτόρων το επιβάλλει, θα υπάρχει και αναλυτική περιγραφή τους.

3.4.1 Μεταφορά πράκτορα

Η λειτουργία της μεταφορά ενός πράκτορα υλοποιείται σε βήματα που είναι η αρχικοποίηση της μεταφοράς από το σύστημα που τον φιλοξενεί, η παραλαβή του από το σύστημα προορισμού και η μεταφορά των κλάσεών του. Στην ενότητα αυτή θα αναλύσουμε τα παραπάνω βήματα.

Κατά τη διαδικασία της αρχικοποίησης της μεταφοράς ο πράκτορας πρέπει να μπορεί να καθορίσει το σύστημα που επιθυμεί να επισκεφτεί. Εάν το όνομα του μέρους που θα εκτελεστεί δεν καθοριστεί τότε ο πράκτορας θα εκτελεστεί στο προκαθορισμένο, από το σύστημα προορισμού, μέρος. Όταν βρεθεί η διεύθυνση του συστήματος προορισμού ο πράκτορας ζητάει από το σύστημα τη μεταφορά του. Η αίτηση αυτή γίνεται με χρήση ενός εσωτερικού τρόπου επικοινωνίας μεταξύ του πράκτορα και του συστήματος που το φιλοξενεί. Αφότου, η αίτηση αυτή φτάσει επιτυχώς στο σύστημα προορισμού. Τοπικά γίνονται οι εξής διαδικασίες:

- Προσωρινή αναστολή λειτουργίας του πράκτορα

- Ανεύρεση των στοιχείων του πράκτορα που καθορίζουν την κατάσταση εκτέλεσής του και τα οποία πρέπει να ταξιδέψουν μαζί με αυτόν
- Σεριοποίηση του στιγμιότυπου του πράκτορα αλλά και της κατάστασής του
- Κωδικοποίηση της σειριοποιημένης μορφής του πράκτορα για το πρωτόκολλο επικοινωνίας που θα χρησιμοποιηθεί
- Πιστοποίηση της αυθεντικότητας του πράκτορα
- Μεταφορά του πράκτορα

Επόμενο βήμα της μεταφοράς ενός πράκτορα είναι η παραλαβή του από το σύστημα προορισμού. Το σύστημα αυτό είναι υπεύθυνο να αποφασίσει αν μπορεί να μεταφράσει τον μεταφερόμενο πράκτορα. Εάν είναι ικανό, δέχεται το μεταφερόμενο πράκτορα και γίνονται οι εξής διαδικασίες:

- πιστοποίηση της γνησιότητας του πράκτορα
- αποκωδικοποίηση για την παραλαβή των δεδομένων του πράκτορα
- αποσειριοποίηση της κλάσης και της κατάστασης του πράκτορα
- δημιουργία ενός νέου στιγμιότυπου του πράκτορα
- εγκατάσταση, στο στιγμιότυπο αυτό, της κατάστασης του πράκτορα που μεταφέρθηκε μαζί με αυτόν
- επανέναρξη της εκτέλεσης του πράκτορα

Το τελευταίο βήμα, της μετακίνησης αυτής, είναι η μεταφορά των κλάσεων. Η έκφραση, μεταφορά κλάσης σημαίνει την ικανότητα της μεταφοράς των δεδομένων των κλάσεων από το ένα σύστημα στο άλλο. Η ικανότητα αυτή είναι απαίτηση σε συστήματα πρακτόρων που υποστηρίζουν αντικειμενοστραφής πράκτορες. Λόγω του γεγονότος της ύπαρξης πολλών διαφορετικών κατασκευαστών συστημάτων πρακτόρων είναι απαραίτητο να υπάρχουν μοναδικά ονόματα στις κλάσεις. Την εξασφάλιση αυτή δεν μπορεί να την παρέχει το σύνολο των κατασκευαστών συστημάτων. Για το λόγο αυτό το πρότυπο χρησιμοποιεί για την ονομασία των κλάσεων μία δομή που αποτελείται από το πραγματικό όνομα της κλάσης (για την αναγνώρισή της από προγραμματιστές) και ένα πεδίο που παίζει το ρόλο του διαχωριστή (discriminator) και υλοποιείται από τους κατασκευαστές του συστήματος κινούμενων πρακτόρων.

3.4.2 Δημιουργία πρακτόρων

Για κάθε πράκτορα, που παρέχεται από διαφορετικό κατασκευαστή, υπάρχει μία κλάση από την οποία το σύστημα πρακτόρων δημιουργεί το στιγμιότυπο του πράκτορα. Η κλάση αυτή ονομάζεται κλάση του πράκτορα. Η δημιουργία του πράκτορα γίνεται στο μέρος που επέλεξε αυτός που έδωσε την εντολή (π.χ. εφαρμογή, άλλος πράκτορας, σύστημα πρακτόρων). Αν δεν υπάρχει η προεπιλογή του μέρους τότε ο πράκτορας δημιουργείται στο προκαθορισμένο μέρος του τοπικού συστήματος. Η κλάση του πράκτορας καθορίζει όχι μόνο το τρόπο επικοινωνίας και αλληλεπίδρασης με τον πράκτορα αλλά παρέχει και την υλοποίηση βασικών λειτουργιών του πράκτορα όπως τερματισμού, μετακόμισης και άλλων. Για τη δημιουργία του πράκτορα το σύστημα πρέπει να ακολουθήσει τις εξής διαδικασίες:

- έναρξη μιας ίνας εκτέλεσης για να αναλάβει την εκτέλεση του πράκτορα
- δημιουργία ενός νέου στιγμιότυπου της κλάσης του πράκτορας
- αν είναι απαραίτητο (την ευθύνη για την ανάθεση μοναδικών ονομάτων στους πράκτορες μπορεί να την έχει το σύστημα πρακτόρων ή ο προγραμματιστής των εφαρμογών) το σύστημα πρέπει να αναθέσει ένα μοναδικό όνομα στον πράκτορα που δημιουργείται
- έναρξη εκτέλεσης του πράκτορα μέσα από την ίνα εκτέλεσης

3.4.3 Παροχή μοναδικών ονομάτων πρακτόρων και τοποθεσιών

Το σύστημα πρακτόρων πρέπει να συνθέτει ένα μοναδικό όνομα για τον εαυτό του και για τα μέρη που δημιουργεί. Η σημασία της ύπαρξης μοναδικών ονομάτων, όπως αυτή αναλύθηκε για τα ονόματα πρακτόρων, επεκτείνεται και για τα ονόματα συστημάτων και περιβαλλόντων εκτέλεσης (μέρη). Το σύστημα πρέπει να μπορεί, αν είναι απαραίτητο, να δίνει μοναδικά ονόματα σε πράκτορες που δημιουργεί.

3.4.4 Εύρεση πρακτόρων

Όταν ένας πράκτορας θέλει να επικοινωνήσει με ένα άλλο πράκτορα πρέπει να μπορεί να βρει το σύστημα που το φιλοξενεί έτσι ώστε να εγκαταστήσει ένα κανάλι επικοινωνίας. Η ικανότητα της εύρεσης ενός συγκεκριμένου πράκτορα είναι σημαντική και για τις λειτουργίες διαχείρισης των πρακτόρων. Εφόσον τα ονόματα των πρακτόρων είναι μοναδικά μέσα στα όρια της υποδομής που εγγυάται την διαλειτουργικότητα μεταξύ των συστημάτων πρακτόρων, μπορούν τα τελευταία να παρέχουν μία υπηρεσία ονοματολογίας βασισμένη στα ονόματα πρακτόρων για την εύκολη εύρεση τους. Μία τέτοια υπηρεσία παρεχόμενη από κάθε δικαιοδοτική περιοχή συντελεί στην κατανομή του φόρτου της εύρεσης ενός πράκτορα.

3.4.5 Εξασφάλιση ενός ασφαλούς περιβάλλοντος για την δράση των πρακτόρων

Οι κινούμενοι πράκτορες είναι προγράμματα που μπορούν να ταξιδεύουν μεταξύ των συστημάτων πρακτόρων που βρίσκονται σε διαφορετικούς υπολογιστές. Η ικανότητά τους αυτή, τους κάνει να παρομοιάζονται συχνά με τους ιούς. Για το λόγο αυτό είναι επιτακτική η ανάγκη να παίρνονται μέτρα ασφάλειας από τα συστήματα πρακτόρων. Τα συστήματα πρέπει να είναι σε θέση να προστατεύουν το λειτουργικό σύστημα, τη χρήση του επεξεργαστή, τη μνήμη, το τοπικό σύστημα αρχείων, τους άλλους πράκτορες που τρέχουν στο ίδιο σύστημα αλλά και την πρόσβαση σε τοπικές εφαρμογές. Για την εξασφάλιση της προστασίας των πόρων του συστήματος αλλά και των άλλων πρακτόρων πρέπει το σύστημα πρακτόρων να πιστοποιεί τη γνησιότητα του κάθε πράκτορα που προτίθεται να φιλοξενήσει.

3.5 Παραδείγματα

Για την πληρέστερη κατανόηση των όρων που εισάγει η τεχνολογία των κινούμενων πρακτόρων, παραθέτονται μερικά σενάρια που μπορούν να εκτελεστούν στηριζόμενα στη διαλειτουργικότητα των συστημάτων έτσι όπως ορίζεται από το πρότυπο. Τα παραδείγματα αυτά, επιπλέον θα βοηθήσουν στο σχηματισμό μιας εικόνας του τρόπου συνεργασίας πρακτόρων με συστήματα πρακτόρων και συστημάτων πρακτόρων μεταξύ τους.

Σύμφωνα με το πρότυπο μπορούν να ειπωθούν οι εξής παρατηρήσεις:

- ένα σύστημα πρακτόρων με τύπο 007 μπορεί να δημιουργήσει και να φιλοξενήσει μόνο πράκτορες τύπου 007
- η διαλειτουργικότητα μεταξύ δύο συστημάτων πρακτόρων ίδιου τύπου υπάρχει εκ κατασκευής των συστημάτων

Οπότε εφαρμογή των ορισμών του προτύπου έχει νόημα σε περιπτώσεις συνεργασίας συστημάτων διαφορετικού τύπου. Σε αυτές τις περιπτώσεις γίνεται φανερή η ανάγκη για τυποποίηση λειτουργιών και μεθόδων αλληλεπίδρασης.

Σενάριο 1: Ο πράκτορας James θέλει να επικοινωνήσει με τον πράκτορα Molter. Ο πράκτορας James βρίσκεται σε ένα σύστημα πρακτόρων που ονομάζεται Bond, ενώ ο πράκτορας Molter βρίσκεται σε ένα σύστημα με το όνομα FBI. Τα δύο συστήματα είναι

συνδεδεμένα με ένα σχετικά φορτωμένο δίκτυο. Λόγω του γεγονότος αυτού, αλλά και επειδή η συνεργασία προβλέπεται να είναι μακροσκελής ο πράκτορας James θα ήθελε να επωφεληθεί της τοπικότητας στην επικοινωνία μετακομίζοντας στον ίδιο ξενιστή (host) ή στο ίδιο υποδίκτυο.

Εάν υποθέσουμε ότι τα συστήματα Bond και FBI υποστηρίζουν πράκτορες του ιδίου τύπου έχουν κατ' επέκταση σύμφωνα με τις παρατηρήσεις τον ίδιο τύπο και διασφαλισμένη τη διαλειτουργικότητα εκ κατασκευής τους. Άρα είναι εφικτή μία μετακόμιση του πράκτορα James στο σύστημα FBI, μία συνεργασία των πρακτόρων στη συνέχεια, και τέλος μια μετακόμιση του πράκτορα James στο σύστημα Bond, από το οποίο και ξεκίνησε.

Την εύρεση της πληροφορίας για τον τύπο του συστήματος αλλά και την κωδικοποίησή της, τα διευθετεί το πρότυπο ορίζοντας ως βασική παρεχόμενη λειτουργία ενός συστήματος την ανάκτηση του τύπου του. Οι μεταφορές του πράκτορα γίνονται βάση των λειτουργιών αποστολής και παραλαβής πράκτορα και κλάσεων. Ο ορισμός και η σημασιολογία όλων των παραπάνω λειτουργιών παρέχεται από το πρότυπο. Αν βέβαια τα δύο συστήματα πρακτόρων έχουν διαφορετικό τύπο υπάρχουν δύο τρόποι για να συνεργαστούν οι πράκτορες, που θα περιγραφθούν στα παρακάτω σενάρια.

Σενάριο 2: Στην περίπτωση αυτή τα συστήματα Bond και FBI δεν υποστηρίζουν τον ίδιο τύπο. Εάν όμως υποτεθεί η ύπαρξη ενός τρίτου συστήματος πρακτόρων με όνομα BritishSecretService που να υποστηρίζει τον ίδιο τύπο πρακτόρων με το σύστημα Bond και να είναι τοπικό ως προς το σύστημα FBI (στον ίδιο ξενιστή ή στο ίδιο υποδίκτυο) τότε επιτυγχάνεται η τοπικότητα στην επικοινωνία. Η πληροφορία της ύπαρξης, τοπικά, συστήματος συγκεκριμένου τύπου παρέχεται από κάθε σύστημα πρακτόρων. Λόγω της ύπαρξης του συστήματος BritishSecretService, ο πράκτορας James μετακομίζει στο σύστημα αυτό που μπορεί να τον φιλοξενήσει και που είναι τοπικό στο σύστημα FBI. Η συνεργασία μεταξύ των πρακτόρων θα γίνει με χρήση της απομακρυσμένης κλήσης διαδικασίας (Remote Procedure Call). Όσον αφορά τις υπόλοιπες ενέργειας που γίνονται είναι πανομοιότυπες με το πρώτο σενάριο, μια και γίνονται μεταξύ συστημάτων με τον ίδιο τύπο.

Σενάριο 3: Υπάρχει βέβαια και η περίπτωση να μη μπορεί να βρεθεί ένα σύστημα που να υποστηρίζει πράκτορες του ιδίου τύπου με αυτούς που υποστηρίζει το σύστημα Bond. Τότε αναγκαστικά ο πράκτορας James θα μείνει στο σύστημα Bond από όπου θα επικοινωνήσει μέσω απομακρυσμένης κλήσης διαδικασίας με τον πράκτορα Molter στο σύστημα FBI. Παρόλο που δε μπορεί να επωφεληθεί της τοπικότητας στην επικοινωνία ο πράκτορας James τουλάχιστον μπορεί να επικοινωνήσει με τον πράκτορα Molter.

3.6 Γλώσσα ορισμού διεπαφής (IDL[6]) του προτύπου

Το πρότυπο ορίζει ένα σύνολο από ορισμούς και διεπαφές (intefaces) που παρέχουν μία διαφανή προσέγγιση στα συστήματα πρακτόρων. Το πρότυπο ορίζει μόνο ένα μικρό σύνολο θέλοντας να παραμείνει όσο πιο απλό και γενικό γίνεται αλλά και θέλοντας να αφήσει περιθώρια για περαιτέρω εξειδίκευση στο μέλλον. Υπάρχουν δύο ορισμοί διεπαφών:

- ορισμός διεπαφής του συστήματος πρακτόρων (MAFAgentSystem Interface)
- ορισμός διεπαφής του ευρετηρίου (MAFFinder Interface)

Από τη διεπαφή του συστήματος πρακτόρων ορίζονται οι λειτουργίες που σχετίζονται με τους πράκτορες. Το σύνολο των λειτουργιών αυτών απαριθμεί τα εξής μέλη: παραλαβή, δημιουργία και τερματισμό, επανέναρξη και προσωρινή διακοπή λειτουργίας πράκτορα. Οι λειτουργίες αυτές αποτελούν το βασικό σύνολο λειτουργιών διαχείρισης που είναι αρκετό για τη διασφάλιση της μεταφοράς και ελέγχου πρακτόρων μεταξύ των συστημάτων. Από την

άλλη πλευρά, οι ορισμοί του ευρετηρίου σχετίζονται με λειτουργίες για καταχώρηση, διαγραφή καταχώρησης και εντοπισμό είτε πράκτορα ή μέρους ή συστήματος πρακτόρων.

Η ενότητα αυτή ξεκινάει με μία ανάλυση των προκαθορισμένων δομών και ορισμών που χρησιμοποιούνται και συνεχίζει με την παρουσίαση των εξαιρέσεων που άρονται σε περίπτωση εσφαλμένης λειτουργίας. Ακολουθούν, αναλύσεις του ορισμού διεπαφής του συστήματος πρακτόρων και του ευρετηρίου.

3.6.1 Ορισμοί & Δομές

Οι ορισμοί και οι δομές που χρησιμοποιεί το πρότυπο περιέχονται στο θαλαμίσκο (module) CfMAF. Υπάρχουν οι βασικοί ορισμοί, όπως "typedef OctetString Authority;" που είναι φανερή η σημασιολογία τους μια και συνεπάγεται αυτόματα από την σημασιολογία των ορισμών διεπαφών. Υπάρχουν και οι δομές, όπως του ονόματος πράκτορα (struct Name) και του ονόματος κλάσης (struct ClassName) που τη σημασιολογία τους την παρέχει το πρότυπο. Η ενότητα αυτή περιέχει μία ανάλυση των δομών αυτών ώστε να γίνει πιο εύκολη η προσέγγιση στην παρουσίαση των ορισμών διεπαφών που ακολουθούν.

```
module CfMAF
{
        typedef sequence<octet> OctetString;
        typedef sequence<OctetString> OctetStrings;
        typedef OctetString Authority;
        typedef OctetString Identity;

        typedef short LanguageID;
        typedef short AgentSystemType;
        typedef short Authenticator;
        typedef short SerializationID;
        typedef sequence<SerializationID> SerializationIDList;

        typedef any Property;
        typedef sequence<Property> PropertyList;

        typedef any InOut;
        typedef sequence<InOut> InOutList;

        struct Name {
                Authority authority;
                Identity identity;
                AgentSystemType agent_system_type;
        };
        typedef sequence<Name> NameList;

        struct AuthInfo {
                boolean is_auth;
                Authenticator authenticator;
        };

        struct LanguageMap {
                LanguageID language_id;
                SerializationIDList serializations;
        };
        typedef sequence<LanguageMap> LanguageMapList;

        struct AgentSystemInfo {
                Name agent_system_name;
                AgentSystemType agent_system_type;
```

```
            LanguageMapList language_maps;
            string agent_system_description;
            short major_version;
            short minor_version;
            PropertyList properties;
        };

        struct AgentProfile {
            LanguageID language_id;
            AgentSystemType agent_system_type;
            string agent_system_description;
            short major_version;
            short minor_version;
            SerializationID serialization;
            PropertyList properties;
        };

        struct ClassName {
            string name;
            OctetString descriminator;
        };

        typedef sequence<ClassName> ClassNameList;
        typedef sequence<any> Arguments;
        typedef string Location;
        typedef sequence<Location> Locations;

        enum AgentStatus {
            MAFRunning, MAFSuspended, MAFTerminated
        };

};
```

3.6.1.1 Δομή ονόματος (struct Name)

Το όνομα είναι ορισμένο σαν μία δομή με τρία γνωρίσματα (πεδία): ταυτότητα, δικαιοδοτική αρχή και τύπος συστήματος πρακτόρων. Η δομή ονόματος χρησιμοποιείται για την ονομασία πρακτόρων και συστημάτων πρακτόρων. Τα πεδία αυτά μπορούν να παρέχουν μοναδικό όνομα σε πράκτορες και συστήματα πρακτόρων.

Όταν η δομή χρησιμοποιείται για ονομασία πράκτορα, το πεδίο που περιέχει τον τύπο συστήματος κρατάει τον τύπο του συστήματος που δημιούργησε τον πράκτορα. Σε περίπτωση βέβαια που χρησιμοποιείται για όνομα συστήματος πρακτόρων κρατάει τον τύπο του συστήματος αυτού. Το πεδίο της δικαιοδοτικής αρχής ορίζει τον άνθρωπο ή τον οργανισμό τον οποίο ο πράκτορας ή το σύστημα αντιπροσωπεύει. Τέλος, το πεδίο της ταυτότητας περιέχει ένα ξεχωριστό όνομα μέσα στα όρια της δικαιοδοτικής αρχής.

Αυτό που επιτυγχάνεται χρησιμοποιώντας την δομή για την ονομασία των πρακτόρων είναι μία εξειδίκευση του περιεχομένου του κάθε πεδίου στο σύνολο που ορίζει το περιεχόμενο του προηγούμενου πεδίου. Η σειρά των πεδίων από το γενικότερο στο ειδικότερο είναι δικαιοδοτική αρχή, τύπος συστήματος πρακτόρων και ταυτότητα.

3.6.1.2 Δομή ονόματος κλάσης (struct ClassName)

Η δομή του ονόματος της κλάσης καθορίζει τον τρόπο σύνταξης του ονόματος των κλάσεων. Περιέχει δύο γνωρίσματα που είναι: το όνομα της κλάσης έτσι όπως βαφτίζεται κατά τη

δημιουργία της και ένα ορμαθό από χαρακτήρες που εξασφαλίζει τη μοναδικότητα του ονόματος. Το πρότυπο δεν παρέχει κανένα μηχανισμό για την εξασφάλιση της μοναδικότητας. Αφήνεται στους κατασκευαστές των συστημάτων να προσθέσουν την απαραίτητη αυτή συνθήκη.

3.6.1.3 Τοποθεσία (Location)

Η τοποθεσία καθορίζει το μονοπάτι που οδηγεί στο συγκεκριμένο αντικείμενο. Για παράδειγμα η τοποθεσία ενός πράκτορα καθορίζεται από τη διεύθυνση του συστήματος πρακτόρων που τον φιλοξενεί και το όνομα του μέρους που εκτελείται. Η τοποθεσία χρησιμοποιείται όχι μόνο για τους πράκτορες, αλλά και τα μέρη και τα συστήματα πρακτόρων.

Ο ορμαθός χαρακτήρων που ορίζει την τοποθεσία μπορεί να είναι είτε

- ένα URL που περιέχει μία διεύθυνση στο διαδίκτυο
- ένα URI που περιέχει ένα όνομα αντικειμένου όπως ορίζεται από την αρχιτεκτονική CORBA

Το πλεονέκτημα της χρήσης ονομάτων CORBA είναι ότι δεν εξαρτώνται τους από τα πρωτόκολλα επικοινωνίας που χρησιμοποιούνται. Από την άλλη πλευρά, η χρήση διευθύνσεων στο διαδίκτυο είναι καταλληλότερη για τη φύση των κινητών πρακτόρων. Το πρότυπο υποστηρίζει και τους δύο τρόπους. Ο διαχωρισμός γίνεται σύμφωνα με το πρόθεμα που χρησιμοποιείται. CosNaming για το όνομα CORBA και mafiiop για διεύθυνση στο διαδίκτυο.

Στα δομικά στοιχεία της τοποθεσίας, όταν γίνεται χρήση των διευθύνσεων διαδικτύου, ανήκουν το όνομα του ξενιστή και ο αριθμός της πόρτας επικοινωνίας με αυτόν. Το πρότυπο, με κωδικό RFC1738, που ορίζει την σύνταξη της διεύθυνσης επιτρέπει την προσθήκη επιπρόσθετης πληροφορίας σε ειδικά διαμορφωμένα πεδία, όπως όνομα μέρους εκτέλεσης πράκτορα. Με την πληροφορία που περιέχεται στην τοποθεσία είναι δυνατόν να παραχθεί ένα IOR, που είναι μία παραπομπή στο σύστημα πρακτόρων.

3.6.1.4 Χαρακτηριστικές τιμές

Οι τιμές μερικών παραμέτρων που χρησιμοποιούνται από το πρότυπο πρέπει να είναι μοναδικές για όλα τα συστήματα πρακτόρων που υλοποιούν το πρότυπο αυτό. Τις τιμές των παραμέτρων πρέπει να τις αναθέτει κάποια ομάδα ανθρώπων που θα είναι υπεύθυνοι και για τη συντήρηση τους. Οι παράμετροι αυτοί βρίσκονται στους ορισμούς:

typedef short LanguageID;
: Η γλώσσα προγραμματισμού του συστήματος. Γλώσσες που χρησιμοποιούνται είναι η Java, η Tcl, η Scheme, και η Perl.

typedef short AgentSystemType;
: Τύπος του συστήματος πρακτόρων που δηλώνει τον κατασκευαστή του συστήματος όπως Aglets και AgentTcl.

typedef short SerializationID;
: Η μέθοδος που χρησιμοποιείται για την σειριοποίηση των πρακτόρων όπως Java Object Serialization.

3.6.2 Τυποποιημένες εξαιρέσεις

Πριν προχωρήσουμε στην περιγραφή των διεπαφών θα αναφερθούμε στους ορισμούς των εξαιρέσεων. Οι εξαιρέσεις δημιουργούνται κατά την κλήση των μεθόδων για να δηλώσουν, σε όποιον κάλεσε τη συγκεκριμένη μέθοδο, το είδος του σφάλματος που συνέβηκε.

exception AgentNotFound { };	Δεν βρέθηκε πράκτορας που να αντιστοιχεί στο όνομα ή στο προφίλ που ζητήθηκε.
exception AgentIsRunning { };	Η πράκτορας ήδη βρίσκεται σε κατάσταση εκτέλεσης.
exception AgentIsSuspended { };	Η εκτέλεση του πράκτορα έχει ήδη διακοπεί προσωρινά.
exception ArgumentInvalid { };	Οι τύποι των παραμέτρων δεν ταιριάζουν με κανένα κατασκευαστή της κλάσης του πράκτορα.
exception ClassUnknown { };	Αποτυχία εύρεσης του ορισμού ή των δεδομένων της κλάσης.
exception EntryNotFound { };	Το ζητούμενο αντικείμενο (πράκτορας, σύστημα πρακτόρων, μέρος) δεν είναι γνωστό στο σύστημα.
exception FinderNotFound { };	Αποτυχία εύρεσης παραπομπής του συστήματος ευρετηρίου.
exception NameInvalid { };	Αποτυχία ενημέρωσης του συστήματος ευρετηρίου.
exception ResumeFailed { };	Αποτυχία επανέναρξης εκτέλεσης του πράκτορα.
exception DeserializationFailed { };	Αποτυχία αποσειριοποίησης της κατάστασης ή του αντικειμένου του πράκτορα.
exception SuspendFailed { };	Αποτυχία προσωρινής διακοπής της εκτέλεσης του πράκτορα.
exception TerminateFailed { };	Αποτυχία διακοπής της εκτέλεσης του πράκτορα.
exception MAFExtendedException { };	Οποιαδήποτε άλλη εξαίρεση που δεν συμπεριλαμβάνεται στην παραπάνω λίστα.

3.6.3 Σύστημα πρακτόρων (MAFAgentSystem)

Στον ορισμό διεπαφής του συστήματος πρακτόρων καθορίζονται μέθοδοι και αντικείμενα για την υποστήριξη των λειτουργιών διαχείρισης των πρακτόρων. Στους ορισμούς αυτούς συμπεριλαμβάνεται και το βασικό σύνολο των λειτουργιών για τη μετακίνηση του πράκτορα.

3.6.3.1 Ορισμός διεπαφής

```
interface MAFAgentSystem {

        CfMAF::Name create_agent(
                                in CfMAF::Name agent_name,
                                in CfMAF::AgentProfile agent_profile,
                                in CfMAF::OctetString agent,
                                in string place_name,
                                in CfMAF::Arguments arguments,
                                in CfMAF::ClassNameList class_names,
                                in string code_base,
```

```
                        in MAFAgentSystem class_provider)
        raises ( ClassUnknown, ArgumentInvalid,
                DeserializationFailed, MAFExtendedException);


CfMAF::OctetStrings fetch_class(
                        in CfMAF::ClassNameList class_name_list,
                        in string code_base,
                        in CfMAF::AgentProfile agent_profile)
        raises (ClassUnknown, MAFExtendedException);


CfMAF::Location find_nearby_agent_system_of_profile(
                        in CfMAF::AgentProfile profile)
        raises (EntryNotFound);


CfMAF::AgentStatus get_agent_status(in CfMAF::Name agent_name)
        raises (AgentNotFound);


CfMAF::AgentSystemInfo get_agent_system_info();


CfMAF::AuthInfo get_authinfo(in CfMAF::Name agent_name)
        raises (AgentNotFound);


MAFFinder get_MAFFinder() raises (FinderNotFound);


CfMAF::NameList list_all_agents();


CfMAF::NameList list_all_agents_of_authority (in MAF::Authority authority);


CfMAF::Locations list_all_places();


void receive_agent(     in CfMAF::Name agent_name,
                        in CfMAF::AgentProfile agent_profile,
                        in CfMAF::OctetString agent,
                        in string place_name,
                        in CfMAF::ClassNameList class_names,
                        in string code_base,
                        in MAFAgentSystem agent_sender)
        raises ( ClassUnknown, ArgumentInvalid,
                DeserializationFailed, MAFExtendedException);


void resume_agent(in CfMAF::Name agent_name)
        raises (AgentNotFound, ResumeFailed, AgentIsRunning);


void suspend_agent(in CfMAF::Name agent_name)
        raises (AgentNotFound, SuspendFailed, AgentIsSuspended);


void terminate_agent(in CfMAF::Name agent_name)
        raises (AgentNotFound, TerminateFailed);
```

};

3.6.3.2 Περιγραφή μεθόδων

Στην ενότητα αυτή περιέχεται μία αναφορά της σημασιολογίας των μεθόδων. Στο επομενο κεφάλαιο, όπου θα αναλυθεί η υλοποίηση που προσφέρει το σύστημα EasyAgent, θα υπάρχει αναλυτικότερη περιγραφή των μεθόδων καθώς και δικαιολόγηση των αποφάσεων για εφαρμογή συγκεκριμένων τρόπων υλοποίησης, όπου το πρότυπο το επιτρέπει στους κατασκευαστές.

create_agent():
Η μέθοδος αυτή δημιουργεί ένα καινούργιο στιγμιότυπο ενός πράκτορα. Ο νέος πράκτορας θα αρχίσει την εκτέλεσή του στο σύστημα που κάλεσε την μέθοδο. Πληροφορίες για τις κλάσεις και τις παραμέτρους που χρειάζονται κατά την αρχικοποίηση του πράκτορα δίνονται ως παράμετροι κλήσης της μεθόδου.

Επιστρέφει το όνομα του καινούργιου πράκτορα, όπως αυτό του ανατέθηκε από το σύστημα. Χρησιμοποιείται για τη δημιουργία πρακτόρων τοπικά, μετά από αίτηση απομακρυσμένου εξυπηρετούμενου.

fetch_class():
Η μέθοδος αυτή επιστρέφει τους ορισμούς των κλάσεων που της ζητήθηκαν.

find_nearby_agent_system_of_profile():
Το σενάριο που πράκτορες δύο διαφορετικών συστημάτων θέλουν να επικοινωνήσουν επωφελούμενοι την τοπικότητα στην επικοινωνία το είδαμε στην ενότητα 3.5. Με τη μέθοδο αυτή ο πράκτορας ζητάει από το σύστημα τη διεύθυνση ενός συστήματος πρακτόρων που μπορεί να τον φιλοξενήσει και που βρίσκεται *'κοντά'* στο σύστημα στο οποίο απευθύνεται η αίτηση. Η αίτηση διεκπεραιώνεται από το σύστημα σε συνεργασία με το σύστημα ευρετηρίου.

get_agent_status():
Επιστρέφει μία από τις κατάστασης που μπορεί να βρίσκεται ο πράκτορας κατά την διάρκεια της ζωής του. Οι καταστάσεις αυτές είναι: εκτελούμενος, προσωρινά ανενεργός, έχει τελειώσει την εκτέλεσή του.

get_agent_system_info():
Η μέθοδος αυτή επιστρέφει τη δομή που περιέχει όλες τις πληροφορίες για τις δυνατότητες του συστήματος. Η δομή αυτή είναι αναγνωριστική για κάθε σύστημα μια και περιέχει το όνομα του συστήματος, πλήρης περιγραφή του προϊόντος (από την πλευρά του λογισμικού) αλλά και των πρακτόρων που μπορεί να υποστηρίξει.

get_authinfo():
Παρέχει την πληροφορία της πιστοποίησης ή μη της γνησιότητάς ενός πράκτορα. Ακόμη, δίνει τη μέθοδο που χρησιμοποιήθηκε για την πιστοποίηση της γνησιότητας.

get_MAFFinder():
Επιστρέφει παραπομπή στο σύστημα ευρετηρίου της περιοχής.

list_all_agents():
Παρέχει τη λίστα με τους πράκτορες που βρίσκονται τη στιγμή εκείνη στο σύστημα. Η μέθοδος αυτή χρησιμοποιείται από τους διαχειριστές του συστήματος.

list_all_agents_of_authority():
Παρέχει τη λίστα με τους πράκτορες που ανήκουν στη ζητούμενη δικαιοδοτική αρχή. Η μέθοδος αυτή χρησιμοποιείται από τους διαχειριστές του συστήματος.

list_all_places():
Παρέχει τη λίστα με τα μέρη (περιβάλλοντα εκτέλεσης) που υπάρχουν στο σύστημα. Η μέθοδος αυτή χρησιμοποιείται από τους διαχειριστές του συστήματος.

receive_agent():
Η μέθοδος αυτή αναλαμβάνει τη μεταφορά των πρακτόρων. Παίρνει ένα στιγμιότυπο του πράκτορα σε σειριοποιημένη μορφή και αφού αποσειριοποίηση την κατάσταση και το αντικείμενο του πράκτορα επανεκκινεί την εκτέλεσή του στο σύστημα. Κατά τη διαδικασία της αποσειριοποίησης, το σύστημα μπορεί να ζητήσει κλάσεις που δεν του είναι γνωστές από κάποιο άλλο σύστημα, που δίνεται ως παράμετρος στη μέθοδο αυτή.

resume_agent():
Επανέναρξη εκτέλεσης πράκτορα. Η μέθοδος αυτή παρέχει τη λειτουργία της επανέναρξης εκτέλεσης ενός πράκτορα του οποίου η εκτέλεση είχε ανασταλεί από τη μέθοδο suspendAgent().

suspend_agent():
Αναστολή εκτέλεσης πράκτορα. Μπορεί κάλλιστα να χρησιμοποιηθεί για την εφαρμογή σειράς προτεραιότητας στην κατανάλωση των πόρων του συστήματος.

terminate_agent():
Τερματισμός εκτέλεσης πράκτορα.

terminate_agent_system():
Τερματισμός εκτέλεσης συστήματος πρακτόρων. Ανάλογα με την υλοποίηση το σύστημα μπορεί να σώζει την κατάστασή του και να πληροφορεί τα άλλα συστήματα πριν τερματίσει.

3.6.4 Σύστημα ευρετηρίου (MAFFinder)

Ο ορισμός διεπαφής του συστήματος ευρετηρίου προσφέρει μεθόδους για την υποστήριξη μίας δυναμικής βάσης δεδομένων. Στην βάση καταχωρείται και ανανεώνεται δυναμικά η πληροφορία της τοποθεσίας των πρακτόρων, των περιβαλλόντων εκτέλεσης (μέρη) και των συστημάτων πρακτόρων. Οι μέθοδοι που περιγράφει υποθέτουν μία βάση δεδομένων που υποστηρίζει την εγγραφή, διαγραφή και εντοπισμό των αντικειμένων τις.

3.6.4.1 Ορισμός διεπαφής

```
interface MAFFinder {

        void register_place(    in string place_name,
                                in CfMAF::Location place_location)
                raises (NameInvalid);


        void register_agent(    in CfMAF::Name agent_name,
                                in CfMAF::Location agent_location,
                                in CfMAF::AgentProfile agent_profile)
                raises (NameInvalid);
```

```
        void register_agent_system(
                        in CfMAF::Name agent_system_name,
                        in CfMAF::Location agent_system_location,
                        in CfMAF::AgentSystemInfo agent_system_info)
                raises (NameInvalid);


        CfMAF::Locations lookup_agent(in CfMAF::Name agent_name,
                                      in CfMAF::AgentProfile agent_profile)
                raises (EntryNotFound);


        CfMAF::Locations lookup_agent_system(
                        in CfMAF::Name agent_system_name,
                        in CfMAF::AgentSystemInfo agent_system_info)
                raises (EntryNotFound);


        CfMAF::Location lookup_place(in string place_name)
                raises (EntryNotFound);


        void unregister_agent(in CfMAF::Name agent_name)
                raises (EntryNotFound);


        void unregister_agent_system(in CfMAF::Name agent_system_name)
                raises (EntryNotFound);


        void unregister_place(in string place_name)
                raises (EntryNotFound);
};
```

3.6.4.2 Περιγραφή μεθόδων

Στην ενότητα αυτή περιέχεται μία αναφορά της σημασιολογίας των μεθόδων που ορίζονται στην διεπαφή του συστήματος ευρετηρίου. Στο επόμενο κεφάλαιο, όπου θα αναλυθεί η υλοποίηση που προσφέρει το σύστημα EasyAgent θα υπάρχει αναλυτικότερη περιγραφή των μεθόδων.

lookup_agent():
Επιστρέφει την τοποθεσία του συγκεκριμένου πράκτορα. Η μέθοδος αυτή μπορεί να αναζητήσει πράκτορα με συγκεκριμένο όνομα ή και σύνολο πρακτόρων που να ταιριάζουν με συγκεκριμένο προφίλ πράκτορα.

lookup_agent_system():
Επιστρέφει την τοποθεσία του συγκεκριμένου συστήματος πρακτόρων. Η μέθοδος αυτή μπορεί να αναζητήσει σύστημα με συγκεκριμένο όνομα ή και σύνολο συστημάτων που να ταιριάζουν στο συγκεκριμένο προφίλ συστήματος.

lookup_place():
Επιστρέφει την τοποθεσία του συγκεκριμένου περιβάλλοντος εκτέλεσης.

Όλες οι μέθοδοι εύρεσης ψάχνουν για αντικείμενα που έχουν ήδη εγγραφεί με τις αντίστοιχες μεθόδους. Επιπλέον οι μέθοδοι αυτοί δεν εγγυώνται την ύπαρξη των αντικειμένων στην

τοποθεσία που επιστρέφουν για οποιοδήποτε χρονικό περιθώριο από τη στιγμή που τελειώνει η κλήση τους. Η σωστή τους λειτουργία εξαρτάται σε μεγάλο βαθμό από τη συμπεριφορά των αντικειμένων που φιλοξενούνται στη βάση. Αν, δηλαδή, τα τελευταία φροντίζουν να εγγράφονται και να διαγράφονται έγκαιρα και όλες τις φορές.

register_agent():
Προσθέτει το όνομα του πράκτορα στην λίστα με τους εγγεγραμμένους πράκτορες. Λόγω του γεγονότος ότι οι πράκτορες ταξιδεύουν συχνά, η μέθοδος αυτή καλείται αρκετές φορές καθ' όλη τη διάρκεια ζωής του πράκτορα. Σε περίπτωση που υπάρχει ήδη εγγραφή με το συγκεκριμένο όνομα, η εγγραφή ανανεώνεται με την πληροφορία της πιο πρόσφατης κλήσης.

register_agent_system():
Προσθέτει το όνομα του συστήματος πρακτόρων στη λίστα με τα εγγεγραμμένα συστήματα πρακτόρων. Λόγω του γεγονότος ότι τα συστήματα πρακτόρων είναι στάσιμα αντικείμενα, προσπάθεια εγγραφής με ίδιο όνομα συστήματος αποτυγχάνει.

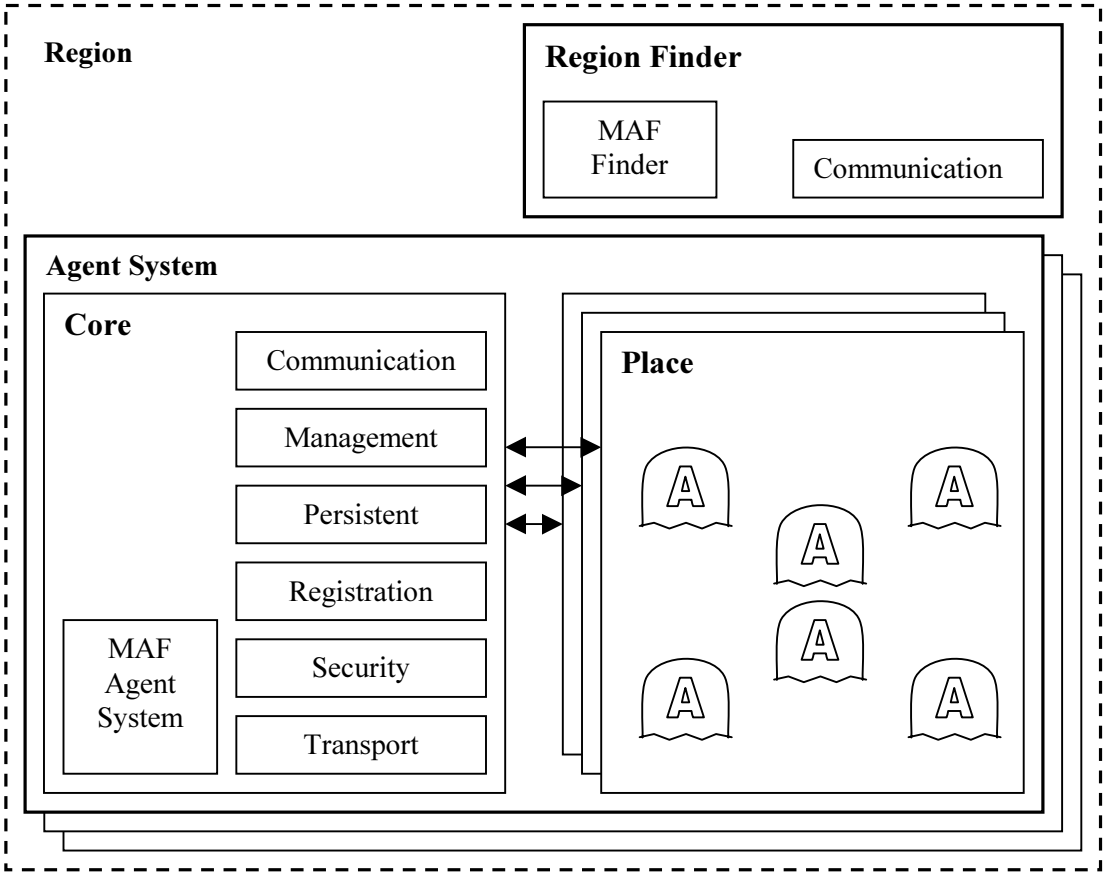
register_place():
Προσθέτει το όνομα του περιβάλλοντος εκτέλεσης πρακτόρων στην λίστα με τα εγγεγραμμένα περιβάλλοντα εκτέλεσης πρακτόρων.

unregister_agent():
unregister_agent_system():
unregister_place():
Αφαιρεί το όνομα του αντικειμένου (πράκτορας, σύστημα πρακτόρων, περιβάλλον εκτέλεσης) από το σύνολο των εγγεγραμμένων αντικειμένων

4 Υλοποίηση του προτύπου.

Το πρώτο σύστημα πρακτόρων βασισμένο πάνω στο πρότυπο έχει αρχίσει να υλοποιείται από την ομάδα IMA-CC (Intelligent Mobile Agent Center of Competence) του GMD (GMD National Research Center for Information Technology), φέρει το όνομα Grasshopper και υπάρχει ήδη η πρώτη του έκδοση. Η ομάδα αυτή είχε συμβάλει και στη δημιουργία του προτύπου. Η υλοποίηση που περιγράφεται σε αυτήν την εργασία ονομάζεται EasyAgent. Παρέχει ένα σύστημα πρακτόρων βασισμένο πάνω στο πρότυπο. Το σύστημα διαφοροποιείται από το πρότυπο λόγω της δυνατότητάς του να καλεί οποιαδήποτε μέθοδο του πράκτορα που δεν είναι από πριν γνωστή. Η προσθήκη της μεθόδου αυτής δεν επηρεάζει καθόλου τη διαλειτουργικότητα με άλλα συστήματα που βασίζονται στο πρότυπο.

Η υλοποίηση χωρίζεται σε δύο μέρη: την υλοποίηση του συστήματος πρακτόρων και την υλοποίηση του ευρετηρίου. Παρέχεται επιπλέον και μία διεπαφή χρήσης (user interface) για την οπτικοποίηση της δράσης των πρακτόρων αλλά και τη διαχείρισή τους από τους υπεύθυνους λειτουργίας του συστήματος. Η διεπαφή επιτρέπει την δημιουργία και τον τερματισμό νέων πρακτόρων και περιβαλλόντων εκτέλεσης πρακτόρων.



Σχήμα 4-1: Ιεραρχική δομή των οντοτήτων.

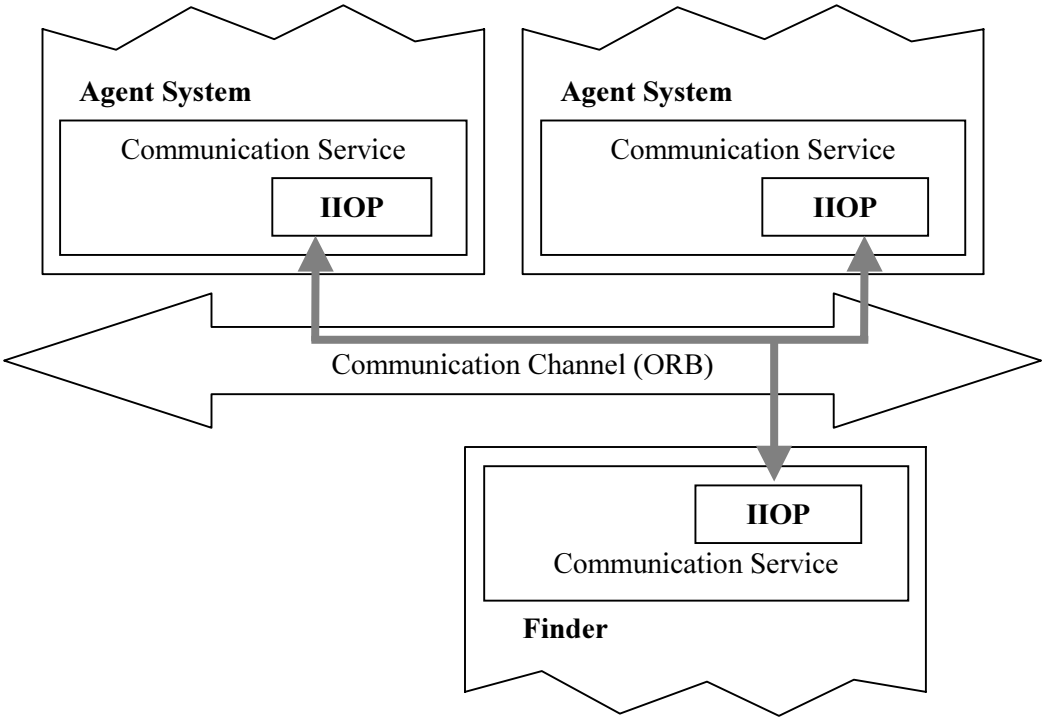
Στο κεφάλαιο αυτό, αφού έχουμε ήδη ασχοληθεί με την σημασιολογία του ορισμού των διεπαφών του προτύπου, θα περιγραφεί η υλοποίηση όπως αυτή παρέχετε από το σύστημα EasyAgent. Οι αποφάσεις και λόγοι που οδήγησαν σε αυτές θα αναλυθούν σε βάθος. Αρχίζει με την παρουσίαση μίας γενικής εικόνας των οντοτήτων που εισάγει η τεχνολογία των κινούμενων πρακτόρων αλλά και των συσχετίσεων μεταξύ τους. Στη συνέχεια, αναλύει τους λόγους που συνέβαλαν στην προσθήκη που έγινε στον ορισμό της διεπαφής του συστήματος πρακτόρων. Με τα θέματα που σχετίζονται με την υλοποίηση συγκεκριμένων τμημάτων του

συστήματος με ιδιαίτερη σημασία, συνεχίζει η ανάλυση, για να τελειώσει με την παρουσίαση των κλάσεων που υλοποιούν τις οντότητες του περιβάλλοντος EasyAgent.

4.1 Περίληψη

Μία αφαιρετική άποψη της δομής του κατανεμημένου συστήματος πρακτόρων θα παρουσιαστεί στην υποενότητα αυτή. Στο Σχήμα 4-1 διαφαίνονται όλες οι οντότητες που ορίζονται από το σύστημα EasyAgent. Η σημασιολογία των αντικειμένων αυτών είναι αυτή που τους δίνετε από το πρότυπο.

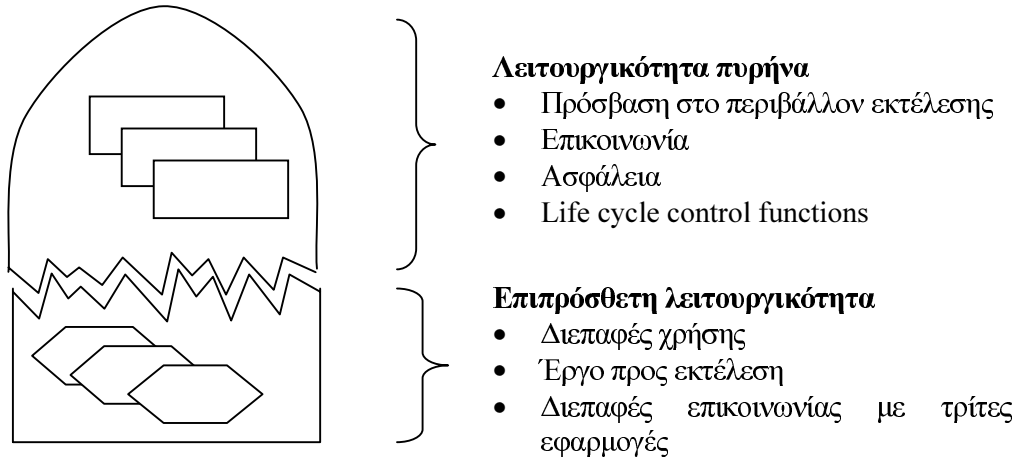
Η περιοχή είναι μία συλλογή από συστήματα πρακτόρων που έχουν την ίδια δικαιοδοτική αρχή. Τέτοιες περιοχές μπορούν να υπάρχουν πολλές σε ένα περιβάλλον πρακτόρων και αναγνωρίζονται μεταξύ τους μέσω της υπηρεσίας ονοματολογίας (Naming Service) που προσφέρεται από την υπάρχουσα υποδομή για επικοινωνίες. Κάθε περιοχή έχει ένα σύστημα ευρετηρίου που υλοποιεί τον ορισμό διεπαφής του ευρετηρίου (MAFFinder) έτσι όπως ορίζεται από το πρότυπο. Η τοποθεσία του συστήματος αυτού γνωστοποιείται μέσω της υπηρεσίας ονοματολογίας στα συστήματα πρακτόρων που ανήκουν στην περιοχή και μπορούν να είναι περισσότερα του ενός. Τα συστήματα αυτά υλοποιούν τον ορισμό διεπαφής του συστήματος πρακτόρων (MAFAgentSystem). Τα συστήματα πρακτόρων αλλά και ευρετηρίου είναι αντικείμενα έτσι όπως αυτά ορίζονται από την αρχιτεκτονική CORBA και η υλοποίησή τους έχει γίνει με τη γλώσσα προγραμματισμού Java (JDK1.2). Την επικοινωνία μεταξύ των συστημάτων πρακτόρων άλλα και μεταξύ αυτών με το σύστημα ευρετηρίου επιτυγχάνεται μέσω του πρωτοκόλλου επικοινωνίας για κινούμενους πράκτορες (MAFIIOP) που ορίζει το πρότυπο. Το πρωτόκολλο που χρησιμοποιείται, στηρίζεται πάνω στο πρωτόκολλο επικοινωνίας των αντικειμένων που ορίζει η αρχιτεκτονική CORBA με όνομα IIOP. Αυτό απεικονίζεται και στο below σχήμα.



Σχήμα 4-2: Υποστήριξη της επικοινωνίας των οντοτήτων.

Τα συστήματα πρακτόρων διατηρούν περιβάλλοντα εκτέλεσης πρακτόρων, που ονομάζονται μέρη, τα οποία παρέχουν το σύνολο των απαραίτητων πόρων για την εκτέλεση των πρακτόρων. Τέτοια μέρη μπορεί να υπάρχουν περισσότερα του ενός σε κάθε σύστημα πρακτόρων. Βέβαιη είναι η ύπαρξη του προκαθορισμένου μέρους που χρησιμεύει για την εκτέλεση των κινούμενων όντων όταν δεν υπάρχουν άλλα περιβάλλοντα εκτέλεσης διαθέσιμα στο συγκεκριμένο σύστημα. Χρησιμοποιείται επίσης, και όταν δεν καθορίζεται το όνομα του περιβάλλοντος εκτέλεσης κατά τις λειτουργίες της μεταφοράς ή της δημιουργίας των αυτόνομων όντων.

Πριν προχωρήσουμε σε αναλυτικότερη περιγραφή της υλοποίησης θα ήταν προτιμότερο να δοθεί μία αφαιρετική περιγραφή των πρακτόρων και των συστημάτων πρακτόρων έτσι ώστε να επιτευχθεί μια εισαγωγή στον τρόπο υλοποίησης, δράσης και αλληλεπίδρασης των οντοτήτων που διαδραματίζουν πρωταγωνιστικό ρόλο στην τεχνολογία των κινούμενων πρακτόρων. Τα συστήματα πρακτόρων αποτελούνται από δύο τμήματα, τον πυρήνα και το σύνολο των περιβαλλόντων εκτέλεσης για τα οποία αναφερθήκαμε παραπάνω. Ο πυρήνας των συστημάτων είναι αυτός που προσφέρει το ελάχιστο σύνολο λειτουργιών για τη φιλοξενία, εκτέλεση και αλληλεπίδραση των πρακτόρων. Οι πράκτορες, από την άλλη πλευρά, αποτελούνται και αυτοί από δύο τμήματα, τον πυρήνα και το τμήμα που ορίζεται από τον προγραμματιστή, όπως φαίνεται και στην παρακάτω εικόνα. Ο πυρήνας του πράκτορα είναι υπεύθυνος για τη διαφανή παροχή υπηρεσιών όπως ασφάλεια, επικοινωνία και πρόσβαση στο σύστημα πρακτόρων στο οποίο φιλοξενείται τη στιγμή εκείνη ο πράκτορας. Το τμήμα που ορίζεται από τον προγραμματιστή μπορεί να αναλάβει την ευθύνη του προγραμματισμού και της εκτέλεσης σε καθορισμένη σειρά των εργασιών της αυτόνομης οντότητας, την ευθύνη της επικοινωνίας με εφαρμογές ξένες ως προς το περιβάλλον πρακτόρων αλλά και την ευθύνη ενός παραθυρικού τρόπου επικοινωνίας με τον χρήστη. Η λίστα που παρατέθηκε για τις δυνατότητες του τμήματος του πράκτορα που ορίζεται από τον προγραμματιστή σίγουρα δεν είναι πλήρης. Αναφέρθηκαν μερικές λειτουργίες που μπορούν να εκτελεστούν μέσω του τμήματος αυτού ως ενδεικτικές των δυνατοτήτων που μπορεί να έχει ένα αυτόνομο ων.



Σχήμα 4-3: Τα δύο τμήματα ενός πράκτορα τύπου EasyAgent.

4.2 Ορισμός διεπαφής του συστήματος

Ο ορισμός διεπαφής του περιγραφόμενου συστήματος στηρίζεται πάνω στον ορισμό διεπαφής του συστήματος πρακτόρων του προτύπου. Ο ορισμός διεπαφής του συστήματος EasyAgent χρησιμοποιεί αυτούσιους τους ορισμούς, τις δομές και τις μεθόδους που περιγράφει η διεπαφή του προτύπου. Από το γεγονός αυτό συνεπάγεται αυτόματα η

διασφάλιση της διαλειτουργικότητας του συστήματος με συστήματα συμβατά με το πρότυπο. Είναι δηλαδή, το εν λόγω σύστημα ικανό να προσφέρει τις ακόλουθες λειτουργίες σε συνεργασία με άλλα συστήματα:

- δημιουργία, φιλοξενία, διαχείριση πρακτόρων του ιδίου τύπου
- να παρέχει πληροφορίες για το ίδιο το σύστημα
- να παρέχει ένα ασφαλές περιβάλλον για την εκτέλεση των πρακτόρων
- να παρέχει μοναδικά ονόματα για πράκτορες
- να επιτρέπει την αλληλεπίδραση με τους πράκτορες με χρήση ενός προκαθορισμένου συνόλου μεθόδων, που πρέπει να υλοποιούν οι πράκτορες

4.2.1 Επέκταση ορισμού διεπαφής

Το πρότυπο προσφέρει στο σύστημα πρακτόρων τη δυνατότητα κλήσης συγκεκριμένου συνόλου μεθόδων ενός πράκτορα με σκοπό τη διαχείριση του πράκτορα κατά τη διάρκεια παραμονής του στο σύστημα. Το σύνολο απαρτίζεται από τις μεθόδους, προσωρινής αναστολής λειτουργίας πράκτορα (suspend_agent()), τερματισμού λειτουργίας πράκτορα (terminate_agent()) και επανέναρξη εκτέλεσης πράκτορα (resume_agent()). Το ίδιο σύνολο μεθόδων μπορεί να χρησιμοποιηθεί και από άλλους πράκτορες ή συστήματα πρακτόρων που θέλουν να αλληλεπιδράσουν με τον πράκτορα αυτόν. Βέβαια, την αίτηση για την επίκληση των μεθόδων αυτών αναλαμβάνει να την εξυπηρετήσει το σύστημα στο οποίο φιλοξενείται την στιγμή εκείνη ο πράκτορας. Οποιαδήποτε άλλη αλληλεπίδραση μεταξύ πρακτόρων ή και συστημάτων με πράκτορες μπορεί να υποστηριχθεί από τον εκάστοτε κατασκευαστή με χρήση τεχνικών και μεθόδων ξένων ως προς το περιβάλλον πρακτόρων.

Δύο είναι οι προφανείς παρατηρήσεις που απορρέουν από την παραπάνω παράγραφο. Από τη μία ότι το πρότυπο δεν παρέχει ένα κοινό και διαφανή τρόπο για την επίκληση οποιασδήποτε μεθόδου του πράκτορα. Η πληροφορία που παρέχει το σύστημα για τους πράκτορες δεν είναι μορφοποιημένη ώστε να τους εφοδιάζει με τη δυνατότητα να δημοσιεύουν τον τρόπο με τον οποίο μπορεί κάποιος άλλος να επικοινωνήσει με αυτούς. Βέβαια, ο ορισμός του προτύπου δίνει ένα πεδίο στη δομή που περιγράφει το προφίλ του πράκτορα, για την προσθήκη επιπλέον χαρακτηριστικών (PropertyList properties;). Όμως χρήση του πεδίου αυτού θα σήμαινε ελευθερία επικοινωνίας μόνο σε πράκτορες του ιδίου τύπου που εκ κατασκευής γνωρίζουν την κωδικοποίηση των περιεχομένων του συγκεκριμένου πεδίου.

Η δεύτερη παρατήρηση αφορά τις επιπρόσθετες ιδιότητες που πρέπει να έχουν οι πράκτορες προκειμένου να μπορούν να αλληλεπιδράσουν. Εφόσον το υπάρχον σύστημα που φιλοξενεί τους πράκτορες δεν τους παρέχει καμία δυνατότητα για περαιτέρω αλληλεπίδραση (βασικοί τρόποι προσφέρονται μέσω του προκαθορισμένου συνόλου μεθόδων του πράκτορα) είναι αναγκασμένοι οι τελευταίοι να φέρουν ιδιαίτερα χαρίσματα ή και να αναλαμβάνουν την πλήρη ευθύνη για την εγκατάσταση ενός καναλιού επικοινωνίας με άλλους πράκτορες. Μπορούν δηλαδή οι πράκτορες να είναι αντικείμενα με τον τρόπο που ορίζονται αυτά από την αρχιτεκτονική CORBA οπότε να αφήνουν το θέμα της αλληλεπίδρασης με άλλους πράκτορες να το χειριστεί η υλοποίηση της αρχιτεκτονικής που φιλοξενεί τα αντικείμενα αυτά. Ακόμη είναι δυνατόν οι πράκτορες να εγκαθιστούν κανάλια επικοινωνίας (sockets) με σκοπό την αλληλεπίδραση με άλλους πράκτορες.

Η διεπαφή του συστήματος πρακτόρων του EasyAgent επεκτείνει τους τρόπους αλληλεπίδρασης των πρακτόρων εφοδιάζοντάς τους με την ικανότητα να καλούν οποιαδήποτε μέθοδο κάποιου άλλου πράκτορα. Έτσι, προσπαθεί να καλύψει το θέμα της επικοινωνίας των κινούμενων οντοτήτων που αφήνει ανοιχτό το πρότυπο, όπως είδαμε και στην ενότητα 3.1.3. Η ικανότητα αυτή δίνεται στους πράκτορες με την προσθήκη μίας ακόμη μεθόδου στον ορισμό διεπαφής του συστήματος πρακτόρων και υπακούει τις αρχές που επιβάλλονται και στις κλήσεις των μεθόδων πρακτόρων που ορίζονται από το πρότυπο. Με

την προσθήκη αυτή επιτυγχάνεται η διαφανής συνεργασία μεταξύ των πρακτόρων μια και οι τελευταίοι δε χρειάζεται να αναζητούν τρόπο επικοινωνίας που να υποστηρίζεται και από τους δύο συνομιλητές. Η επιπλέον μέθοδος που συμπληρώνει τον ορισμό διεπαφής περιγράφεται στον παρακάτω ορισμό

```
module CfMAF {

        exception ExecMethodFailed { };

        interface MAFAgentSystem{

                CfMAF OctetString exec_agent_method(
                                in CfMAF::Name agent_name,
                                in string method_name,
                                in CfMAF::OctetStrings arguments)
                        raises (AgentNotFound, ExecMethodFailed);
        };

};
```

Η μέθοδος αυτή χρησιμοποιείται για την επίκληση μεθόδων ενός πράκτορα παρέχοντας μόνο το όνομα του πράκτορα, το όνομα της μεθόδου και φυσικά και τις παραμέτρους της. Η προσθήκη της εξαίρεσης έρχεται να συμπληρώσει τον ορισμό της μεθόδου. Η εξαίρεση δηλώνει ότι συνέβηκε κάποιο σφάλμα κατά τη διάρκεια εκτέλεσης της μεθόδου. Είτε δεν βρέθηκε η ζητούμενη μέθοδος, είτε οι παράμετροι δεν ταιριάζουν με τους παραμέτρους της ζητούμενης μεθόδου.

Η επιθυμητή μέθοδος του πράκτορα εκτελείται στο περιβάλλον εκτέλεσης του πράκτορα που παρέχει την υλοποίησή της. Τα συστήματα πρακτόρων που φιλοξενούν τους συνεργαζόμενους πράκτορες (οι τελευταίοι μπορεί να είναι και στο ίδιο σύστημα) αναλαμβάνουν την κωδικοποίηση και αποκωδικοποίηση των πληροφοριών που ανταλλάσσονται για την επιτυχημένη κλήση της μεθόδου.

4.3 Θέματα υλοποίησης

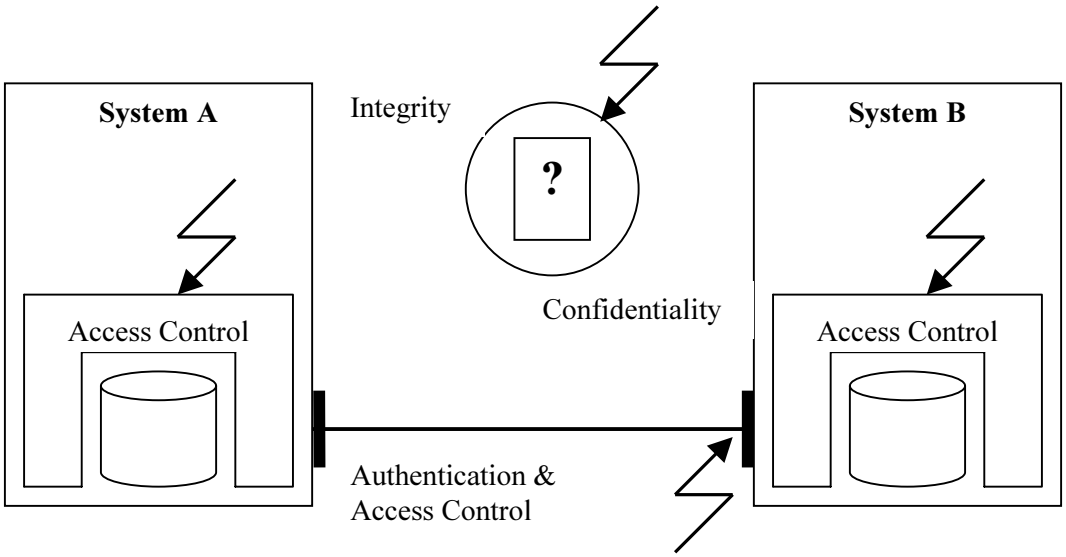
Κατά την αναλυτική περιγραφή του προτύπου αναφέρθηκαν μερικά θέματα που ο τρόπος υλοποίησης αφήνεται στην κρίση του συγκεκριμένου κατασκευαστή. Τα θέματα αυτά φυσικά δεν έχουν να κάνουν με τον τρόπο που τα συστήματα αλληλεπιδρούν. Ότι αφορά τέτοια θέματα το πρότυπο δεν αφήνει κανένα περιθώριο επιλογής στους υλοποιητές, μια και αποτελούν βάση για τη διασφάλιση της διαλειτουργικότητας. Ο τρόπος απόδοσης μοναδικών ονομάτων σε κλάσεις, η πολιτική που χρησιμοποιείται για τη μεταφορά τους και τη μεταφορά του πράκτορα, αλλά και θέματα ασφάλειας, είναι τα ζητήματα που απασχολούν την ενότητα αυτή. Επιπλέον, αναλύονται ο σχεδιασμός των συστημάτων πρακτόρων και ευρετηρίου.

4.3.1 Ασφάλεια

Στα ανοικτά κατανεμημένα περιβάλλοντα (open distributed environments) όπως είναι και το EasyAgent υπάρχουν πολλές απειλές για την ασφάλεια του συστήματος που πρέπει να ληφθούν υπ' όψιν για το σχεδιασμό μίας αποτελεσματικής τακτικής. Η ενότητα αυτή παρουσιάζει τα βασικά θέματα που αφορούν την ασφάλεια των συστημάτων και την υλοποίηση που προσφέρει το σύστημα της εργασίας αυτής.

Εμπιστευτικότητα (confidentiality): Οι πράκτορες κατά τη διάρκεια της ζωής τους μεταφέρονται μεταξύ των συστημάτων κουβαλώντας αρκετές φορές και εμπιστευτικά

δεδομένα. Τα δεδομένα αυτά πρέπει να μπορούν να χρησιμοποιηθούν μόνο από τα δύο συστήματα που παίρνουν μέρος στην μεταφορά του πράκτορα. Οποιαδήποτε διαρροή της πληροφορίας μπορεί να είναι καταστροφική. Όμως, για τη μετάδοση χρησιμοποιούνται μέσα μεταφοράς, φυσικά (π.χ. καλώδια) και τεχνητά (π.χ. δρομολογητές), που δεν ανήκουν στην δικαιοδοσία των συστημάτων που εμπλέκονται στη μεταφορά του πράκτορα. Για το λόγο αυτό, πρέπει ο μεταφερόμενος πράκτορας να είναι κωδικοποιημένος. Έτσι, το περιεχόμενο της μεταφοράς, του πράκτορα στην ουσία, δεν θα έχει καμία αξία εκτός εάν κάποιος ξέρει τον τρόπο αποκωδικοποίησής του (και αυτός πρέπει να είναι μόνο ο τελικός παραλήπτης).



Σχήμα 4-4: Πιθανές απειλές για την ασφάλεια ενός συστήματος.

Ακεραιότητα (Integrity):Όταν οι πράκτορες φτάνουν στον προορισμό τους πρέπει το σύστημα να είναι σε θέση να καταλάβει αν η πληροφορία που τον περιγράφει έχει αλλοιωθεί από οποιαδήποτε εσκεμμένη προσπάθεια. Ο επιτιθέμενος δεν πρέπει να καταφέρει να αλλάξει την πληροφορία του πράκτορα, κατά την μεταφορά του, χωρίς το σύστημα να το αντιληφθεί. Οποτεδήποτε το τελευταίο αμφιβάλει για την ακεραιότητα των δεδομένων πρέπει να ζητάει την επανάληψη της μετάδοσης.

Πιστοποίηση γνησιότητας (authentication): Κατά τη διάρκεια μιας συνομιλίας, η πιστοποίηση γνησιότητας των δεδομένων που ανταλλάσσονται προϋποθέτει την πιστοποίηση γνησιότητας των συνομιλητών. Εάν τα δεδομένα που ανταλλάσσονται είναι σημαντικά το κάθε σύστημα που παίρνει μέρος σε μία συνομιλία πρέπει να είναι σίγουρο για την αληθινή ταυτότητα του συνομιλητή του. Ο τρόπος κατασκευής των δικτύων από υπολογιστές ευνοεί τους 'απατεώνες' να προσποιούνται ότι είναι κάποιοι άλλοι με σκοπό την κλοπή πληροφορίας. Εάν αυτό το επίπεδο ασφάλειας ξεπεραστεί, μετά είναι πολύ ευκολότερη η διάσπαση των παρακάτω επιπέδων ασφάλειας. Για τους λόγους αυτούς η πιστοποίηση γνησιότητας είναι μία από τις βασικές απαιτήσεις, όσον αφορά την ασφάλεια, των σύγχρονων κατανεμημένων συστημάτων και κατ' επέκταση των συστημάτων πρακτόρων.

Έλεγχος πρόσβασης (access control): Το σύστημα πρακτόρων παρέχει στους πράκτορες πόρους όπως το σύστημα αρχείων, το δίκτυο, το χρόνο στον επεξεργαστή, την μνήμη και αρκετά άλλα που δεν χρειάζεται να τα απαριθμήσουμε για να επισημάνουμε την πολιτική που πρέπει να ακολουθεί για τον ασφαλή διαμοιρασμό τους. Η πρόσβαση στους πόρους αυτούς πρέπει να είναι περιορισμένη και το σύστημα είναι υπεύθυνο για την φύλαξη από πρόσβαση δίχως εξουσιοδότηση. Μία τέτοια πρόσβαση μπορεί, αναλόγως του είδους του

πόρου, είτε να καταστρέψει δεδομένα (σύστημα αρχείων), είτε να οδηγήσει στην εξάντληση του συγκεκριμένου πόρου (μνήμη, χρόνος στον επεξεργαστή). Η πρόσβαση σε ζωτικούς πόρους πρέπει λοιπόν να προστατεύεται από επίπεδα ασφάλειας ανάλογα με τα προνόμια του πράκτορα.

Ακρόαση (Auditing): Οι υπηρεσίες ασφάλειας που αναφέρθηκαν παραπάνω και ειδικότερα αυτή του ελέγχου πρόσβασης και της πιστοποίησης γνησιότητας, πρέπει να αποθηκεύουν πληροφορία σχετική με οποιαδήποτε δραστηριότητα γίνεται υπό την εποπτεία τους. Έτσι, δραστηριότητες όπως πρόσβαση σε πόρους του συστήματος και αποτυχημένες προσπάθειες πιστοποίησης γνησιότητας καταγράφονται σε αρχεία για να εξεταστούν αργότερα.

4.3.1.1 Κρυπτογραφία

Στα περισσότερα σενάρια που εμπλέκεται η κρυπτογραφία αφορούν έναν αποστολέα και ένα παραλήπτη. Ο αποστολέας επιθυμεί να στείλει το μήνυμα με ασφάλεια και θέλει να είναι σίγουρος ότι οποιοσδήποτε και αν παρακολουθεί τα δεδομένα που αποστέλλει δε μπορεί να καταλάβει το περιεχόμενό τους. Για το λόγο αυτό πριν αποστείλει τα δεδομένα τα κρυπτογραφεί. Από την άλλη πλευρά ο παραλήπτης τα αποκρυπτογραφεί με σκοπό να τα χρησιμοποιήσει.

Ο αλγόριθμος της κρυπτογράφησης είναι μία μαθηματική συνάρτηση που χρησιμοποιείται τόσο για την κρυπτογράφηση όσο και για την αποκρυπτογράφηση. Το να κρατάς όμως τη συνάρτηση μυστική είναι αναποτελεσματικό. Ως εκ τούτου, χρησιμοποιείται ένα κλειδί μαζί με τον αλγόριθμο και είναι αυτό που κρατείται μυστικό. Το ίδιο κλειδί χρησιμοποιείται για την κρυπτογράφηση και την αποκρυπτογράφηση. Έτσι ο αλγόριθμος μπορεί να δημοσιοποιηθεί, αναλυθεί και τελικά τυποποιηθεί. Το ίδιο ακριβώς φαινόμενο συμβαίνει και με τις κλειδαριές. Οποιοσδήποτε μπορεί να μάθει τα τεχνικά χαρακτηριστικά μίας κλειδαριάς. Αλλά χωρίς το κλειδί η κλειδαριά του είναι εντελώς άχρηστη.

Υπάρχουν πολλοί αλγόριθμοι για κρυπτογράφηση και χωρίζονται στις εξής κατηγορίες:

- symmetric algorithms [26]
- asymmetric or public-key algorithms [26]
- one-way hash functions

Με χρήση των αλγόριθμων κρυπτογράφησης μπορεί να διασφαλιστεί η ακεραιότητα των δεδομένων, αλλά και να αποφευχθεί οποιαδήποτε διαρροή πληροφορίας.

4.3.1.2 Πιστοποίηση γνησιότητας

Με χρήση της κρυπτογραφίας μπορεί να διασφαλιστεί και η πιστοποίηση της γνησιότητας. Ο περισσότερο διαδεδομένος τρόπος για τη διασφάλιση της πιστοποίησης είναι με τη χρήση public-key αλγορίθμων. Εάν ο αποστολέας επιθυμεί να αποδείξει την ταυτότητά του στον παραλήπτη, τοποθετεί μία ψηφιακή υπογραφή[27] στην πληροφορία πριν την στείλει. Τις περισσότερες φορές η ενέργεια αυτή είναι η ίδια η κρυπτογράφηση της πληροφορίας με το προσωπικό (private) κλειδί του αποστολέα. Έτσι, οποιοσδήποτε λαμβάνει την πληροφορία μπορεί να την αποκρυπτογραφήσει χρησιμοποιώντας το δημόσιο (public) κλειδί του αποστολέα. Εάν το καταφέρει ο παραλήπτης είναι σίγουρος ότι μόνο ο συγκεκριμένος αποστολέας μπορούσε να του είχε στείλει το μήνυμα αυτό. Η διαδικασία αυτή ονομάζεται πιστοποίηση της ψηφιακής υπογραφής.

4.3.1.3 Πιστοποιητικά τύπου X.509

Η χρήση των δημοσίων κλειδιών επιβάλλει την πιστοποίηση της γνησιότητάς τους. Πρέπει να υπάρχει κάποιος οργανισμός που να παρέχει τα δημόσια κλειδιά και να είναι έμπιστος[28]. Στο σενάριο που περιγράφηκε παραπάνω, ο παραλήπτης λαμβάνει το δημόσιο κλειδί του αποστολέα από μία δημόσια βάση δεδομένων για να πιστοποιήσει την υπογραφή

του τελευταίου. Εάν και οι δύο συνομιλητές ανήκουν στην ίδια δικαιοδοτική αρχή τότε δεν υφίσταται κανένα θέμα σχετικό με την ασφάλεια, μια και ο παραλήπτης γνωρίζει ότι το κλειδί του αποστολέα είναι γνήσιο αφού ανακτήθηκε από ένα έμπιστο σύστημα. Το πρόβλημα υφίσταται όταν οι δύο συνομιλητές έχουν διαφορετικές δικαιοδοτικές αρχές. Τότε ένας τρίτος , που αποκαλείται πιστοποιητής γνησιότητας (certification authority) και είναι έμπιστος και προς τους δύο συνομιλητές αναλαμβάνει την δημοσιοποίηση των κλειδιών.

4.3.1.4 Έλεγχος πρόσβασης

Ο έλεγχος πρόσβασης είναι η διαδικασία κατά την οποία ελέγχονται τα δικαιώματα όσον αφορά την πρόσβαση μίας οντότητας στο σύστημα ή σε πόρους του συστήματος. Ο έλεγχος πρόσβασης είναι στενά συνδεδεμένος με τη διαδικασία της πιστοποίησης γνησιότητας μια και η απόφαση του να επιτραπεί ή όχι η πρόσβαση βασίζεται στην ταυτότητα της οντότητας που την ζητάει. Το σύστημα, από την άλλη μεριά, συσχετίζει την ταυτότητα της οντότητας που πήρε την άδεια πρόσβασης με ένα σύνολο από δικαιώματα. Έτσι, εάν ο επιτιθέμενος καταφέρει να προσποιηθεί κάποιο χρήστη αυτόματα αποκτάει όλα τα δικαιώματα πρόσβασης του χρήστη αυτού.

Μεταξύ των πόρων ενός κατανεμημένου συστήματος που πρέπει να προστατευθούν είναι και

- ο χρόνος στον επεξεργαστή
- η κατανάλωση μνήμης
- το σύστημα αρχείων
- η ικανότητα να χρησιμοποιεί το δίκτυο

4.3.1.5 Ασφάλεια στο περιβάλλον EasyAgent

Το περιβάλλον EasyAgent υποστηρίζει δύο μηχανισμούς ασφάλειας, την εξωτερική και την εσωτερική ασφάλεια.

- Με τον όρο εξωτερική ασφάλεια εννοείται η προστασία των αλληλεπιδράσεων μεταξύ των κατανεμημένων συστατικών (συστήματα πρακτόρων και ευρετηρίου) του περιβάλλοντος κινούμενων πρακτόρων. Τέτοιου είδους προστασία μπορεί να προσφέρει η υπηρεσία ασφάλειας[5], που ορίζεται από την αρχιτεκτονική CORBA. Η υπηρεσία αυτή, όπως και όλες οι υπηρεσίες που ορίζονται στα πλαίσια της αρχιτεκτονικής βρίσκουν άμεση εφαρμογή στα αντικείμενα που φιλοξενούνται από τον εξυπηρετητή τύπου ORB. Δηλαδή, προσφέροντας μία υλοποίηση για την υπηρεσία ασφάλειας, που να συμφωνεί με τις προδιαγραφές που ορίζονται από το πρότυπο[5], αυτόματα η υπηρεσία αυτή χρησιμοποιείται από τα αντικείμενα.

  Οι ανάγκες επικοινωνίας μεταξύ των συστημάτων πρακτόρων και ευρετηρίου εξυπηρετούνται από τον εξυπηρετητή ORB. Οπότε, χρησιμοποίηση ενός ORB που παρέχει ταυτόχρονα και μία υλοποίηση της υπηρεσίας ασφάλειας συνεπάγεται αυτόματα την προστασία όλων των απομακρυσμένων αλληλεπιδράσεων από πιθανές προσπάθειες διαφθοράς.

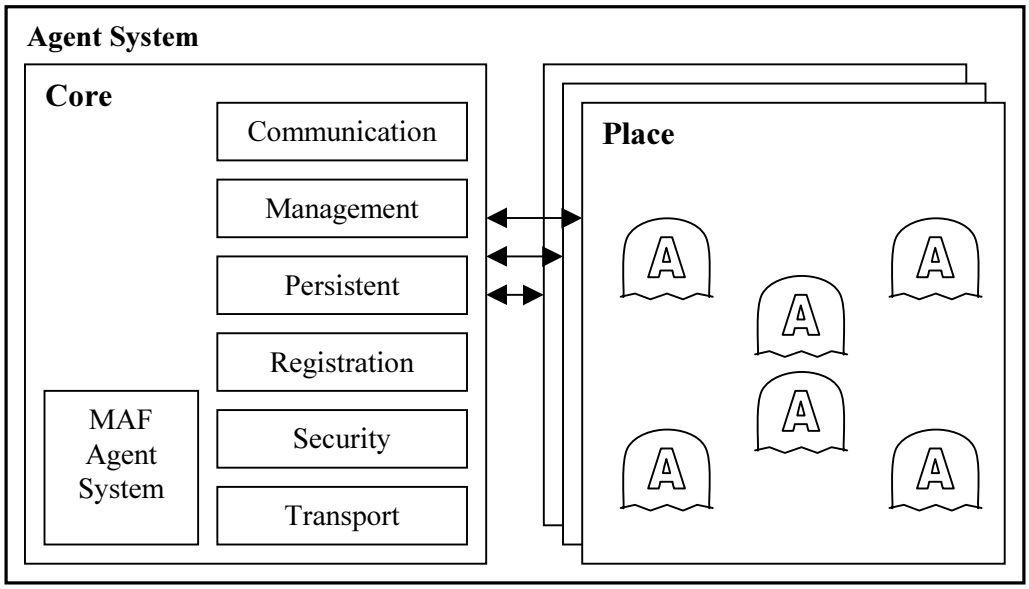
- Με τον όρο εσωτερική ασφάλεια εννοείται η προστασία των πόρων του συστήματος από πρόσβαση χωρίς τα απαραίτητα δικαιώματα από τους πράκτορες. Επιπρόσθετα, αναφέρεται και σε προστασία των πρακτόρων από άλλους πράκτορες. Αυτό επιτυγχάνεται πιστοποιώντας την ταυτότητα και εξουσιοδοτώντας το χρήστη για τον οποίο δραστηριοποιείται ο πράκτορας. Ανάλογα με το αποτέλεσμα της πιστοποίησης ταυτότητας και της εξουσιοδότησης ενεργοποιούνται οι πολιτικές ελέγχου πρόσβασης. Η εξωτερική ασφάλεια των συστημάτων πρακτόρων του EasyAgent βασίζεται στους μηχανισμούς ασφάλειας που παρέχει το περιβάλλον προγραμματισμού JDK1.2.

4.3.2 Σύστημα Πρακτόρων

Το σύστημα πρακτόρων είναι το περιβάλλον που βρίσκονται οι πράκτορες κατά τη διάρκεια της διαμονής τους σε κάποιο ξενιστή. Το σύστημα πρακτόρων του περιβάλλοντος EasyAgent (EasyAgentSystem) αποτελείται από δύο τμήματα: τον πυρήνα του συστήματος και το σύνολο από τα μέρη (περιβάλλοντα εκτέλεσης).

4.3.2.1 Πυρήνας του Συστήματος Πρακτόρων

Ο πυρήνας είναι το τμήμα του συστήματος που παρέχει την ελάχιστη απαιτούμενη λειτουργικότητα με σκοπό την υποστήριξη της εκτέλεσης ενός πράκτορα. Οι παρακάτω υπηρεσίες παρέχονται από τον πυρήνα του συστήματος πρακτόρων του EasyAgent.



Σχήμα 4-5: Τα δομικά συστατικά του συστήματος πρακτόρων.

Υπηρεσία επικοινωνίας (communication service)
Η υπηρεσία αυτή είναι υπεύθυνη για όλες τις αλληλεπιδράσεις που συμβαίνουν μεταξύ των οντοτήτων που συνθέτουν το περιβάλλον κινούμενων πρακτόρων. Τέτοιου είδους αλληλεπιδράσεις είναι η μεταφορά πρακτόρων μεταξύ συστημάτων, η εύρεση πρακτόρων μέσω του συστήματος ευρετηρίου, η επίκληση μεθόδων πράκτορα και η μεταφορά πληροφορίας για κλάσεις. Γενικότερα, όλες οι λειτουργίες που χρειάζονται επικοινωνία, συνεργασία, μεταξύ συστημάτων πρακτόρων χρησιμοποιούν την υπηρεσία επικοινωνίας. Η υπηρεσία αυτή στηρίζεται πάνω στην υπηρεσία επικοινωνίας που παρέχει η αρχιτεκτονική CORBA μια και οι οντότητες που επικοινωνούν (συστήματα πρακτόρων και συστήματα ευρετηρίου) είναι εξυπηρετητές που φιλοξενούνται από το περιβάλλον εκτέλεσης της CORBA.

Υπηρεσία ευρετηρίου (registration service)
Κάθε σύστημα πρακτόρων πρέπει να μπορεί να παρέχει πληροφορίες για όλους τους πράκτορες που φιλοξενεί την συγκεκριμένη στιγμή καθώς και για όλα τα περιβάλλοντα εκτέλεσης που υπάρχουν στο σύστημα. Οι πληροφορίες αυτές χρησιμοποιούνται από την μία πλευρά σε λειτουργίες διαχείρισης του συστήματος και από την άλλη με σκοπό την πληροφόρηση των πρακτόρων που φιλοξενούνται στο σύστημα για τις οντότητες που συνυπάρχουν στο σύστημα αυτό. Επιπρόσθετα, η υπηρεσία χρησιμοποιείται για την αποθήκευση πληροφοριών για τις κλάσεις που γνωρίζει το σύστημα. Κάθε νέα κλάση που εισάγεται στο σύστημα, είτε με τη λειτουργία της μεταφοράς κλάσεων, είτε φορτώνεται από κάποιο αποθηκευτικό μέσο, εγγράφεται στην υπηρεσία. Έτσι, οποιαδήποτε στιγμή στο

μέλλον μπορεί να χρησιμοποιηθεί από όλες τις οντότητες του τοπικού συστήματος, ή και να γνωστοποιηθεί σε άλλο σύστημα μετά από αίτηση. Τους σκοπούς αυτούς επιτυγχάνει η ενσωμάτωση της υπηρεσίας αυτής στο σύστημα πρακτόρων.

Υπηρεσία διαχείρισης (management service)
Οι λειτουργίες που σχετίζονται με τη διαχείριση αναπτύχθηκαν με σκοπό να επιτρέπουν την παρακολούθηση των πρακτόρων και των περιβαλλόντων εκτέλεσης ενός συστήματος πρακτόρων από εξωτερικούς χρήστες (ανθρώπους). Η λειτουργίες που προσφέρονται είναι:

- δημιουργία, τερματισμός, αναστολή και επανέναρξη εκτέλεσης πρακτόρων και περιβαλλόντων εκτέλεσης
- πληροφορίες για τους πράκτορες που φιλοξενούνται από το σύστημα
- εξαντλητικός κατάλογος πρακτόρων που βρίσκονται σε συγκεκριμένο μέρος
- εξαντλητικός κατάλογος των περιβαλλόντων εκτέλεσης ενός συστήματος πρακτόρων

Εκτός από αυτές τις λειτουργίες, παρέχεται και ένα περιβάλλον προγραμματισμού στους χρήστες που τους επιτρέπει να καθορίζουν παραμέτρους που αφορούν την πολιτική ασφάλειας του συστήματος.

Υπηρεσία μεταφοράς (transport service)
Η υπηρεσία αυτή υποστηρίζει τη μετακίνηση των πρακτόρων μεταξύ των συστημάτων. Στο σύστημα προορισμού ο πράκτορας συνεχίζει την εργασία, που του έχει ανατεθεί, από το σημείο ακριβώς που διέκοψε την εκτέλεση πριν τη μεταφορά. Επιπλέον, η υπηρεσία είναι υπεύθυνη για τη μεταφορά των ορισμών των κλάσεων. Παίζει τον ρόλο του συντονιστή της μεταφοράς, η οποία πραγματοποιείται από την υπηρεσία επικοινωνίας των συστημάτων. Κωδικοποιεί / Αποκωδικοποιεί την πληροφορία της κατάστασης, αν πρόκειται για μεταφορά πράκτορα και των κλάσεων στα πρότυπα της διεπαφής επικοινωνίας των συστημάτων πρακτόρων.

Υπηρεσία ασφάλειας (security service)
Το σύστημα EasyAgent, όπως διαφαίνεται αναγκαίο από την ενότητα 4.3.1 που ασχολείται με θέματα ασφάλειας των συστημάτων πρακτόρων, υποστηρίζει δύο είδη μηχανισμών ασφάλειας που είναι η εξωτερική και η εσωτερική ασφάλεια. Όσον αφορά την εξωτερική ασφάλεια, από την μία πλευρά, το σύστημα πρακτόρων ασχολείται με την προστασία των απομακρυσμένων αλληλεπιδράσεων των οντοτήτων. Για λόγους υλοποίησης αλλά και περαιτέρω εξέλιξης του συστήματος EasyAgent, οι μηχανισμοί ασφάλειας των αλληλεπιδράσεων μεταξύ των συστημάτων στηρίζονται σε αυτούς της ασφάλεια της αρχιτεκτονικής CORBA. Οι οντότητες που αλληλεπιδρούν (συστήματα πρακτόρων) είναι αντικείμενα CORBA οπότε οι μηχανισμοί ασφάλειας βρίσκουν άμεση εφαρμογή. Από την άλλη πλευρά, οι μηχανισμοί της εσωτερικής ασφάλειας των συστημάτων πρακτόρων προστατεύουν τους πόρους των συστημάτων από τους πράκτορες. Ελέγχουν δηλαδή τα προνόμια των πρακτόρων πριν τους δώσουν πρόσβαση όχι μόνο σε πόρους του συστήματος αλλά και σε άλλους πράκτορες. Αυτό γίνεται με την διαδικασία της πιστοποίησης γνησιότητας του χρήστη για τον οποίο ο πράκτορας δρα. Ανάλογα με το αποτέλεσμα της διαδικασίας αυτής ενεργοποιούνται τα προνόμια των πρακτόρων. Οι μηχανισμοί που ασχολούνται με την εσωτερική ασφάλεια των συστημάτων υλοποιούνται με χρήση των μηχανισμών ασφάλειας που προσφέρει το περιβάλλον προγραμματισμού της Java.

Υπηρεσία σταθερότητας (persistent service)
Η υπηρεσία σταθερότητας του συστήματος EasyAgent υποστηρίζει την αποθήκευση των πρακτόρων και των περιβαλλόντων εκτέλεσης σε μόνιμο αποθηκευτικό μέσο (persistent medium). Αποθήκευση πρακτόρων σημαίνει την αποθήκευση της κατάστασης και των κλάσεων του πράκτορα, ενώ με τον όρο αποθήκευση περιβάλλοντος εκτέλεσης εννοείται η αποθήκευση της πληροφορίας και της κατάστασης (πράκτορες που φιλοξενεί) του

περιβάλλοντος. Με τον τρόπο αυτό είναι δυνατή η ανάκτηση των πρακτόρων και των περιβαλλόντων εκτέλεσης όποτε αυτό είναι αναγκαίο (π.χ. μετά από μία διακοπή λειτουργίας του υπολογιστή που φιλοξενεί το σύστημα πρακτόρων).

4.3.2.2 Περιβάλλον εκτέλεσης πρακτόρων

Τα περιβάλλοντα εκτέλεσης προσφέρουν μία λογική ομαδοποίηση λειτουργιών στα συστήματα πρακτόρων τα οποία τα φιλοξενούν. Για παράδειγμα μπορεί να υπάρχει περιβάλλον εκτέλεσης το οποίο να εξειδικεύεται στην αγοροπωλησία πληροφοριών, δηλαδή οι πράκτορες που φιλοξενούνται σε αυτό να προσφέρουν ή να αγοράζουν πληροφορίες ή και πρόσβαση σε υπηρεσίες. Το όνομα του μέρους πρέπει να αντικατοπτρίζει τη λειτουργικότητά του μια και η αναφορά σε αυτό γίνεται μόνο μέσω του ονόματός του. Κάθε σύστημα πρακτόρων έχει ένα προκαθορισμένο μέρος εκτέλεσης στο οποίο φιλοξενούνται οι πράκτορες που δεν δηλώνουν ρητά το μέρος που επιθυμούν να μεταφερθούν. Το όνομα του περιβάλλοντος αυτού έχει δύο συνθετικά. Το πρώτο μέρος είναι το όνομα του συστήματος πρακτόρων που φιλοξενεί το σύστημα και το δεύτερο η λέξη *'DefaultPlace'*. Το χρησιμοποιούν οι πράκτορες για την απόκτηση πληροφοριών σχετικών με το σύστημα (λίστα φιλοξενούμενων περιβαλλόντων εκτέλεσης και πρακτόρων).

4.3.3 Σύστημα Ευρετηρίου

Το σύστημα ευρετηρίου διατηρεί πληροφορίες για όλες τις οντότητες που σχετίζονται με μία περιοχή (βλέπε ενότητα 3.2). Όποτε μία νέα οντότητα (πράκτορας, σύστημα πρακτόρων, περιβάλλον εκτέλεσης) δημιουργείται, εγγράφεται αυτόματα στο σύστημα ευρετηρίου της περιοχής εκείνης. Μεταξύ των πληροφοριών που διατηρούνται είναι και η τοποθεσία της οντότητας. Έτσι, πρέπει οι κινούμενες οντότητες του περιβάλλοντος να ενημερώνουν το σύστημα ευρετηρίου για κάθε κίνηση. Η πληροφορία που διατηρεί το σύστημα ευρετηρίου είναι διαθέσιμη για όλες τις οντότητες του περιβάλλοντος αλλά και για εξωτερικούς χρήστες (άλλες εφαρμογές, άνθρωποι) που μπορούν να επικοινωνήσουν με αυτό. Έτσι είναι οποιαδήποτε στιγμή εφικτή η παρακολούθηση όλων των οντοτήτων της περιοχής.

Η σημαντικότητα του ρόλου που διαδραματίζει το σύστημα ευρετηρίου στην περιοχή, που αντιπροσωπεύει, γίνεται φανερή αν αναλογιστεί κανείς ότι όλες οι αναφορές στις οντότητες του συστήματος EasyAgent γίνονται με το όνομά τους. Οι υπηρεσίες επικοινωνίας των συστημάτων πρακτόρων, που ανήκουν σε μία περιοχή, επικοινωνώντας με το αντίστοιχο σύστημα ευρετηρίου μαθαίνουν την τοποθεσία οποιασδήποτε οντότητας. Στη συνέχεια, κάνοντας χρήση της τοποθεσίας μπορούν να απευθύνουν αιτήσεις προς αυτήν.

Τέλος, το σύστημα ευρετηρίου προσφέρει, μέσω της διεπαφής του, μεθόδους για εύρεση οντοτήτων όχι μόνο με χρήση του ονόματός τους αλλά και με σύγκριση κάποιων χαρακτηριστικών τους. Σε όλες τις περιπτώσεις, φροντίζει να ενημερώνει τον εξυπηρετούμενο με την τοποθεσία της ζητούμενης οντότητας ή με ένδειξη λάθους αν η οντότητα δεν υπάρχει.

4.3.4 Πράκτορες

Κάθε πράκτορας είναι μία οντότητα μέσα στο σύστημα που προσφέρει κάποια καθορισμένη λειτουργικότητα στις υπόλοιπες οντότητες που απαρτίζουν το σύστημα. Ο πράκτορας χωρίζεται σε δύο μέρη: τον πυρήνα, που είναι κοινός για όλους τους πράκτορες του συστήματος EasyAgent, και το ιδιαίτερο μέρος, όπου ορίζεται η λειτουργικότητα του κάθε πράκτορα από τον κατασκευαστή του (βλέπε Σχήμα 4-3).

Το σύστημα EasyAgent δεν κάνει κανένα διαχωρισμό των πρακτόρων σε κινούμενους (mobile) και σταθερούς (stationary). Παρέχει ένα πυρήνα κοινό για όλους τους πράκτορες που τους προσφέρει τις λειτουργίες

- της επικοινωνίας με τις υπόλοιπες οντότητες του συστήματος ,
- της ασφάλειας,
- της πρόσβασης στο σύστημα πρακτόρων
- και της διαχείρισης κατά την διάρκεια της ζωής του.

Από την άλλη πλευρά, στο ιδιαίτερο μέρος του πράκτορα ο κατασκευαστής μπορεί να προσθέσει οποιαδήποτε επιπλέον λειτουργικότητα επιθυμεί. Δηλαδή, μπορεί να εφοδιασεί τον πράκτορα με:

- παραθυρικά περιβάλλοντα επικοινωνίας με τον χρήστη,
- λίστα από εργασίες που έχει να πραγματοποιήσει,
- υλοποίηση γεφυρών με εξωτερικές ως προς το σύστημα πρακτόρων εφαρμογές και περιφερειακά.

4.3.5 Κλάσεις

Η σημασιολογία ενός πράκτορα περικλείεται στις κλάσεις που αυτός χρησιμοποιεί. Οι κλάσεις περιέχουν όλη την πληροφορία της εκτέλεσης του πράκτορα, αν βέβαια ο τελευταίος είναι υλοποιημένος με οντοκεντρική γλώσσα προγραμματισμού. Στην περίπτωση του συστήματος EasyAgent οι κλάσεις που ορίζουν τους πράκτορες είναι κλάσεις όπως αυτές ορίζονται από τη γλώσσα προγραμματισμού Java. Αναφέρθηκε στο προηγούμενο κεφάλαιο (βλέπε ενότητα 3.6.1.2) ότι είναι αναγκαίο να έχουν οι κλάσεις που σχετίζονται με το σύστημα μοναδικά ονόματα. Το θέμα της μοναδικότητας μαζί με θέματα σχετικά με τη μεταφορά και αποθήκευση των κλάσεων θα αναλυθούν παρακάτω.

4.3.5.1 Μεταφορά κλάσεων

Υπάρχουν τρεις λόγοι που καθιστούν τη μεταφορά των κλάσεων ως απαραίτητη λειτουργία των συστημάτων πρακτόρων.

- *Η δημιουργία στιγμιότυπου ενός πράκτορα σε ένα απομακρυσμένο ξενιστή ως τμήμα της λειτουργίας της απομακρυσμένης δημιουργίας πράκτορα.*

Κατά τη δημιουργία του πράκτορα χρειάζεται να είναι γνωστή η κλάση του πράκτορα. Εάν η κλάση δεν είναι γνωστή στον ξενιστή, που στη συγκεκριμένη περίπτωση είναι το απομακρυσμένο σύστημα πρακτόρων, πρέπει να υπάρχει τρόπος μεταφοράς της πληροφορίας της κλάσης μεταξύ των συστημάτων για να δημιουργηθεί επιτυχώς το νέο στιγμιότυπο..

- *Η δημιουργία στιγμιότυπου ενός πράκτορα ως τμήμα της λειτουργίας της μεταφοράς πρακτόρων.*

Μετά την ολοκλήρωση της λειτουργίας της μεταφοράς του πράκτορα, η κλάση του πράκτορα είναι απαραίτητη για τη δημιουργία του στιγμιοτύπου του. Εάν η κλάση δεν είναι γνωστή στο σύστημα προορισμού πρέπει να υπάρχει τρόπος μεταφοράς της από το σύστημα που αναχώρησε ο συγκεκριμένος πράκτορας.

- *Η εκτέλεση του πράκτορα μετά τη δημιουργία του στιγμιοτύπου του.*

Μετά τη δημιουργία του νέου στιγμιότυπου του πράκτορα, είτε αυτό προήλθε από τη λειτουργία της απομακρυσμένης δημιουργίας, είτε από τη λειτουργία της μεταφοράς του πράκτορα ο τελευταίος συχνά δημιουργεί νέα αντικείμενα. Οι κλάσεις των αντικειμένων αυτών πρέπει να είναι γνωστές στο σύστημα για τη δημιουργία των

στιγμιότυπων τους. Εάν κάποια από τις κλάσεις αυτές δεν είναι γνωστή πρέπει το σύστημα να μπορεί να τις ζητήσει από το σύστημα που βρισκόταν ο πράκτορας πριν μετακομίσει σε αυτό.

Τα παραπάνω κάνουν φανερή την ανάγκη της υποστήριξης της μεταφοράς κλάσεων από τα συστήματα πρακτόρων, χωρίς αυτό βέβαια να σημαίνει ότι στο συγκεκριμένο σημείο εξαντλούνται τα θέματα που σχετίζονται με τη μεταφορά των κλάσεων. Οι μηχανισμοί της μεταφοράς των κλάσεων είναι ένα από τα ζητήματα που αναλύονται στη συνέχεια.

Η διαφοροποίηση στους μηχανισμούς μεταφοράς των κλάσεων προκύπτει από την τακτική που ακολουθείται κατά τη μεταφορά τους, αν δηλαδή θα αποστέλλονται όλες οι απαραίτητες κλάσεις ή αν θα στέλνονται μόνο μετά από απαίτηση. Ειδικότερα, οι εξής μηχανισμοί μπορούν να υποστηριχθούν και ο καθένας βέβαια έχει τα πλεονεκτήματά του καθώς και τα μειονεκτήματά του:

- *Αυτόματη μεταφορά όλων των απαραίτητων κλάσεων.*

  Το σύστημα πρακτόρων το οποίο, είτε αποστέλλει τους πράκτορες, είτε αιτεί για τη δημιουργία των πρακτόρων στο απομακρυσμένο σύστημα, αποστέλλει ταυτόχρονα και όλες τις απαραίτητες κλάσεις για τη δημιουργία του στιγμιότυπου του πράκτορα αλλά και για την εκτέλεσή του. Με την τεχνική αυτή δεν χρειάζεται το απομακρυσμένο σύστημα να ζητήσει αργότερα οποιαδήποτε επιπλέον κλάση. Αλλά το να στέλνει κανείς αυτόματα όλες τις απαιτούμενες κλάσεις, καταναλώνει περισσότερους πόρους δικτύου από ότι θα ήταν απαραίτητο, εάν μερικές από τις αποστελλόμενες κλάσεις είναι ήδη γνωστές στο σύστημα προορισμού.

- *Αυτόματη μεταφορά της κλάσης του πράκτορα, όλες οι υπόλοιπες κλάσεις μεταφέρονται μετά από αίτηση.*

  Το σύστημα πρακτόρων, που αρχικοποιεί τη λειτουργία της δημιουργίας ή της μεταφοράς του πράκτορα αποστέλλει μόνο την κλάση του πράκτορα για να μπορέσει το σύστημα προορισμού να δημιουργήσει το στιγμιότυπό του. Εάν το σύστημα προορισμού χρειάζεται επιπλέον κλάσεις για την εκτέλεση του πράκτορα, της ζητάει από το αρχικό σύστημα. Με την τεχνική αυτή οι πόροι του δικτύου χρησιμοποιούνται μόνο όταν κρίνεται εντελώς απαραίτητο. Βέβαια, υπάρχει η περίπτωση το σύστημα προορισμού να μην είναι δυνατόν να αποκτήσει πρόσβαση στο αρχικό σύστημα. Για παράδειγμα αυτό μπορεί να συμβεί όταν για κάποιο χρονικό διάστημα το αρχικό σύστημα παραμείνει αποσυνδεδεμένο λόγω βλάβης διασύνδεσης με το δίκτυο.

- *Αυτόματη μεταφορά της κλάσης του πράκτορα, οι υπόλοιπες κλάσεις είτε μεταφέρονται μετά από αίτηση (στην περίπτωση της μεταφοράς πράκτορα), είτε αποστέλλονται ταυτόχρονα με την κλάση του πράκτορα (στην περίπτωση της λειτουργίας της απομακρυσμένης δημιουργίας πράκτορα).*

  Η τεχνική αυτή συνδυάζει τις προηγούμενες δύο. Όταν το αρχικό σύστημα δεν έχει συνεχώς πρόσβαση στο δίκτυο (π.χ. ένας φορητός υπολογιστής) κατά την δημιουργία του πράκτορα σε ένα άλλο ξενιστή αποστέλλει ταυτόχρονα και όλες τις απαραίτητες κλάσεις. Έτσι αντιμετωπίζεται το πρόβλημα που παρουσίαζε η προηγούμενη τεχνική.

Το σύστημα που αναπτύχθηκε στα πλαίσια της εργασίας αυτής χρησιμοποιεί μία παραλλαγή της τρίτης τεχνικής. Εάν δεν οριστεί διαφορετικά, δεν αποστέλλει στο σύστημα προορισμού, μαζί με την αίτηση δημιουργίας ή και μεταφοράς του πράκτορα, την κλάση του πράκτορα. Ο λόγος είναι ότι η κλάση του πράκτορα, η κλάση δηλαδή από την οποία κληρονομείται η ιδιότητα της μετακίνησης, είναι μοναδική για κάθε παροχέα συστημάτων κινούμενων

πρακτόρων. Αυτό συνεπάγεται αυτόματα ότι κατά την πρώτη επίσκεψη ενός πράκτορα τύπου Α σε ένα σύστημα πρακτόρων η κλάση πράκτορα για τους πράκτορες τύπου Α γίνεται γνωστή (δεν χρειάζεται δηλαδή να ξαναζητηθεί στο μέλλον). Από την άλλη, ο αριθμός των διαφορετικών κατασκευαστών συστημάτων πρακτόρων είναι αρκετές τάξεις μεγέθους μικρότερος από τον αριθμό των πρακτόρων κάποιου συγκεκριμένου τύπου που επισκέπτονται το σύστημα κατά τη διάρκεια της λειτουργίας του. Η παρατήρηση αυτή σε συνδυασμό με την δυνατότητα της επαναχρησιμοποίησης της κλάσης του πράκτορα οδηγεί στην εξής παραδοχή, ότι είναι προτιμότερο να καθυστερεί χρονικά η έναρξη της εκτέλεσης του πρώτου πράκτορα κάποιου συγκεκριμένου τύπου που επισκέπτεται ένα σύστημα πρακτόρων παρά να επιβαρύνεται κάθε αίτηση για δημιουργία ή μεταφορά πράκτορα με την ταυτόχρονη μεταφορά της κλάσης του πράκτορα.

Αν κατά την δημιουργία του νέου στιγμιότυπου χρειάζονται κλάσεις που δεν υπάρχουν στο τοπικό σύστημα ζητείται από το σύστημα πρακτόρων που ξεκίνησε την διαδικασία να στείλει τους ορισμούς των άγνωστων γι' αυτό, κλάσεων. Βέβαια, μπορεί κάλλιστα, αν του ζητηθεί από τον προγραμματιστή των εφαρμογών κινούμενων πρακτόρων, να αποστέλλει στο σύστημα προορισμού ταυτόχρονα με την αίτηση δημιουργίας η μεταφοράς και όλες τις απαραίτητες κλάσεις, μαζί με την κλάση του πράκτορα, για την δημιουργία και την εκτέλεση του συγκεκριμένου πράκτορα.

4.3.5.2 Μοναδικότητα ονομάτων

Το πρότυπο, όπως έχει αναφερθεί, ορίζει μία δομή για την ονομασία των κλάσεων. Η δομή αυτή αποτελείται από το πραγματικό όνομα της κλάσης και ένα πεδίο για τη διατήρηση της μοναδικότητας του ονόματός της στα πλαίσια του περιβάλλοντος που δημιουργείται από τη συνεργασία των συστημάτων. Πριν προχωρήσουμε στην τεχνική που χρησιμοποιεί το σύστημα EasyAgent για τη διατήρηση της μοναδικότητας του ονόματος, θα σταθούμε στους λόγους που συμβάλουν στο να αποδίδονται μοναδικά ονόματα στις κλάσεις που μετακινούνται στο σύστημα.

Ας υποθέσουμε το εξής σενάριο. Έστω ότι το σύστημα πρακτόρων ΣύστημαΓ, ζητάει από το σύστημα ΣύστημαΑ την κλάση με όνομα Χ. Η κλάση Χ πρέπει να είναι μοναδική στο ΣύστημαΑ για να μπορέσει το σύστημα να εξυπηρετήσει την αίτηση της αποστολής της κλάσης. Ομοιοτρόπως, το ΣύστημαΓ μπορεί να ζητήσει την κλάση Χ από το ΣύστημαΒ που περιέχει και αυτό κλάση με το ίδιο όνομα αλλά μοναδική στα όρια του συστήματός του. Το ΣύστημαΓ πρέπει να είναι σε θέση να διαχωρίζει την κλάση Χ που πήρε από το ΣύστημαΑ και αύτη που πήρε από το ΣύστημαΒ. Αυτό είναι απαραίτητο για το γεγονός ότι ένα σύστημα με όνομα ΣύστημαΔ μπορεί να ζητήσει την κλάση Χ από το ΣύστημαΓ για να δημιουργήσει στιγμιότυπο ενός πράκτορα. Από το σενάριο γίνεται φανερός τουλάχιστον ένας λόγος για τον οποίο το σύστημα πρέπει να διασφαλίζει τη μοναδικότητα των ονομάτων των κλάσεων.

Το σύστημα EasyAgent παρέχει μία τεχνική για τη διατήρηση της μοναδικότητας των ονομάτων των κλάσεων που στηρίζεται στη χρήση του ονόματος της δικαιοδοτικής αρχής του πράκτορα. Η δικαιοδοτική αρχή, όπως έχει γίνει φανερό από τα ως τώρα γραφόμενα, είναι αυτή που δίνει το προνόμιο στους πράκτορες για να ζήσουν και να δραστηριοποιηθούν πάνω στα συστήματα πρακτόρων. Περιορίζοντας το πρόβλημα της μοναδικότητας των ονομάτων στα όρια της δικαιοδοτικής αρχής γίνεται ευκολότερη η λύση του για δύο λόγους. Από την μία, το όνομα της δικαιοδοτικής αρχής όπως ορίζει το πρότυπο πρέπει να αποδίδεται μοναδικά από μία ομάδα υπεύθυνη για τη συντήρηση της λειτουργίας του συστήματος. Και από την άλλη, η δικαιοδοτική αρχή μπορεί από μόνη της να είναι υπεύθυνη για την ονοματολογία των κλάσεων με τις οποίες τροφοδοτεί τα συστήματα πρακτόρων.

4.3.5.3 Αποθήκευση

Τα συστήματα πρακτόρων κατά τη διάρκεια της λειτουργίας τους φιλοξενούν πολλούς πράκτορες που πιθανότατα να χρησιμοποιούν κοινές κλάσεις. Για τη δράση τους οι πράκτορες χρειάζονται κλάσεις που το σύστημα πρακτόρων φροντίζει να τους τις παρέχει. Έτσι, το σύστημα είναι υποχρεωμένο είτε να αιτεί από άλλα συστήματα για αποστολή κλάσεων, είτε να δέχεται κλάσεις κατά τις λειτουργίες της μεταφοράς ή της δημιουργίας των πρακτόρων. Οι ορισμοί των κλάσεων που μεταφέρονται μπορούν μετά τη χρησιμοποίησή τους να αποθηκεύονται για χρήση από πράκτορες, που μπορεί να φιλοξενηθούν αργότερα από το ίδιο σύστημα. Αυτή η τεχνική συμβάλλει στην μείωση του όγκου των συναλλαγών που πρέπει να λάβουν χώρα για τη μεταφορά και την απομακρυσμένη εκτέλεση των πρακτόρων.

Το σύστημα πρακτόρων που παρέχει το EasyAgent δίνει σε κάθε μέρος (περιβάλλον εκτέλεσης) τη δυνατότητα να αποθηκεύει τους ορισμούς των κλάσεων σε μόνιμα αποθηκευτικό μέσο, για την επανεκκίνηση του συστήματος μετά από αποτυχία. Επιπλέον, σε αλληλεπίδραση με την υπηρεσία ευρετηρίου γνωστοποιεί τον ορισμό της κλάσης σε όλες τις τοπικές οντότητες.

4.4 Υλοποίηση

Στην ενότητα 4.3 συζητήσαμε θέματα που μας απασχόλησαν κατά τη διάρκεια του σχεδιασμού του συστήματος. Στην ενότητα αυτή θα παρουσιάσουμε την ακριβή διαδικασία που ακολουθήθηκε για την υλοποίηση του προτύπου. Η υλοποίηση έγινε χρησιμοποιώντας το περιβάλλον προγραμματισμού JDK1.2. Το περιβάλλον έχει ενσωματωμένο εξυπηρετητή για ανεύρεση αντικειμένων (Object Request Broker) όπως αυτός ορίζεται από την αρχιτεκτονική CORBA καθώς και εργαλείο για τη μεταγλώττιση των διεπαφών των αντικειμένων που θα φιλοξενούνται από τον εξυπηρέτη. Οπότε, το περιβάλλον εκτέλεσης που προσφέρει το JDK1.2, πληρεί το σύνολο των απαιτήσεων της υποδομής για επικοινωνίες που πρέπει να υπάρχει για την εκτέλεση των συστημάτων πρακτόρων. Επιπλέον, το περιβάλλον προγραμματισμού που χρησιμοποιείται, προσδίδει τη δυνατότητα εκτέλεσης των συστημάτων σε ξενιστές με λειτουργικά συστήματα τύπου Unix και Windows.

Η παρουσίαση της υλοποίησης είναι ενδεδειγμένο να βασιστεί στη δομή των πακέτων της γλώσσας προγραμματισμού Java που συνιστούν το περιβάλλον EasyAgent. Η χρήση της οντοκεντρικής γλώσσας προγραμματισμού είναι δυνατό όχι μόνο να μοντελοποιήσει το περιβάλλον, αλλά και να προσφέρει ένα μονοπάτι για την περιήγηση ανάμεσα στις οντότητες που ορίζει. Οι επόμενες ενότητες του κεφαλαίου αυτού ασχολούνται με την παρουσίαση του πακέτου που υλοποιεί τις διεπαφές επικοινωνίας των οντοτήτων. Αυτές διασφαλίζουν τη διαλειτουργικότητα, παρέχοντας καθορισμένο τρόπο πρόσβασης σε κάθε οντότητα, όπως ορίζει το πρότυπο MASIF. Το πακέτο αυτό ονομάζεται CfMAF. Ασχολούνται επίσης, με το πακέτο που περιέχει την υλοποίηση του συστήματος πρακτόρων και ευρετηρίου. Ονομάζεται EasyAgent, και εσωκλείει την υλοποίηση όλων των δομικών συστατικών του περιβάλλοντος.

4.4.1 Πακέτο CfMAF

Περιέχει τις κλάσεις που υλοποιούν τις διεπαφές του συστήματος πρακτόρων και του συστήματος ευρετηρίου. Οι διεπαφές αυτές καθορίζουν τον τρόπο επικοινωνίας με τις οντότητες που τις χρησιμοποιούν. Παρέχουν δηλαδή ένα κοινό τρόπο προσπέλασης στην πληροφορία που διατηρούν ώστε να είναι δυνατή η πρόσβαση από οντότητες διαφορετικών κατασκευαστών. Το σύνολο των ορισμών καταφέρνει να αποκρύψει τις λεπτομέρειες της υλοποίησης του συστήματος πρακτόρων και ευρετηρίου.

Πιο συγκεκριμένα, περιέχει μαζί με άλλες βοηθητικές κλάσεις, και κλάσεις που προέρχονται από τη μετάφραση του ορισμού των διεπαφών με το εργαλείο που παρέχει το περιβάλλον προγραμματισμού, τις κλάσεις

- MAFAgentSystemImplementation, για την υλοποίηση της διεπαφή του συστήματος πρακτόρων
- MAFFinderImplementation, για την υλοποίηση της διεπαφή του συστήματος ευρετηρίου
- και το σύνολο των κλάσεων που πραγματοποιούν τις ενδείξεις σφαλμάτων κατά την διάρκεια εκτέλεσης (runtime exceptions).

4.4.1.1 Κλάση MAFAgentSystemImplementation

Η κλάση αυτή υλοποιεί τον ορισμό διεπαφής του συστήματος πρακτόρων. Αναλαμβάνει την αποκωδικοποίηση της πληροφορίας που μεταφέρεται από τις οντότητες του περιβάλλοντος κινούμενων πρακτόρων (πράκτορες ή και άλλα συστήματα πρακτόρων) μέσω της κλήσης των μεθόδων που παρέχει. Στη συνεχεία συνεργάζεται με υπηρεσίες του συστήματος πρακτόρων για την εξυπηρέτηση των αιτήσεων. Τέλος, φροντίζει είτε να κωδικοποιήσει το αποτέλεσμα της εργασίας που του ανατέθηκε και να το γνωστοποιήσει στον εξυπηρετούμενο, είτε να ειδοποιήσει τον τελευταίο για τυχόν σφάλμα που προέκυψε στην εκτέλεση.

Οι μέθοδοι που περιέχονται στην κλάση, και που προσδίδουν την απαιτούμενη από το πρότυπο λειτουργικότητα στα συστήματα πρακτόρων, αναλύονται παρακάτω. Η σύνταξή τους καθορίζεται από την αντιστοίχηση (mapping) των ορισμών της διεπαφής, στη γλώσσα προγραμματισμού Java. Η μέθοδος της αντιστοίχησης είναι τυποποιημένη και γίνεται αυτόματα με τη χρήση ενός εργαλείου που παρέχεται από το περιβάλλον προγραμματισμού, το μεταγλωττιστή idltojava. Για την περιγραφή των μεθόδων, όπου θα παρουσιαστεί η αλληλεπίδρασή τους με τις υπόλοιπες οντότητες του συστήματος πρακτόρων, θα τις χωρίσουμε σε ομάδες ανάλογα με το σκοπό που εξυπηρετούν. Έτσι έχουμε:

- *Το σύνολο των μεθόδων για τη διαχείριση των πρακτόρων που απαρτίζεται από τις* **create_agent(), resume_agent(), suspend_agent(), terminate_agent()**.

Οι μέθοδοι αυτοί που υλοποιούν τις λειτουργίες της αρχικοποίησης, τερματισμού αλλά και αλλαγής κατάστασης εκτέλεσης των πρακτόρων συνεργάζονται με την υπηρεσία διαχείρισης του πυρήνα του συστήματος πρακτόρων. Η αλληλεπίδραση με την υπηρεσία συντελεί στην εξυπηρέτηση της εργασίας που καλούνται να εκπονήσουν. Η υπηρεσία διαχείρισης σε συνεργασία με τις υπηρεσίες ευρετηρίου και σταθερότητας και να ενημερώνει την κατάσταση του συστήματος πρακτόρων και να αποθηκεύει την απαραίτητη πληροφορία για το σύστημα ώστε να είναι δυνατή η αποκατάσταση της ορθής λειτουργίας μετά από τυχών σφάλματα ή αποτυχίες.

- *Το σύνολο των μεθόδων για την ανάκτηση πληροφορίας σχετική με τους πράκτορες που φιλοξενεί, που απαρτίζεται από τις,* **list_all_agents(), list_all_agents_of_authority(), get_agent_status()**.

Για την εξυπηρέτηση των αιτήσεων αυτών απαιτείτε η συνεργασία με την υπηρεσία ευρετηρίου που διατηρεί την κατάσταση όλων των οντοτήτων που δραστηριοποιούνται στο συγκεκριμένο σύστημα πρακτόρων.

- *Το σύνολο των μεθόδων για την ανάκτηση πληροφοριών σχετικών με το σύστημα πρακτόρων, όπως τη δικαιοδοτική αρχή, τα περιβάλλοντα εκτέλεσης που διαθέτει, την αναφορά για το συστήματος ευρετηρίου της περιοχής, τις διευθύνσεις άλλων συστημάτων πρακτόρων. Συγκεκριμένα, αποτελείται από τις* **list_all_places(),**

get_authinfo(), find_nearby_agent_system_of_profile(), get_MAFFinder(), get_agent_system_info().

Οι μέθοδοι αυτοί συνεργάζονται με τις υπηρεσίες ευρετηρίου και διαχείρισης του πυρήνα του συστήματος πρακτόρων. Αφορούν την ανάκτηση πληροφοριών που διατηρούν οι υπηρεσίες αυτές για την ορθή λειτουργία του συστήματος.

- *Τις μεθόδους για την παραλαβή πράκτορα και κλάσεων που είναι οι* **receive_agent()** *και* **fetch_class()**.

Για τη μεταφορά της κατάστασης του πράκτορα αλλά και των κλάσεών του, ώστε να είναι δυνατή η επανέναρξη της λειτουργίας του στο τοπικό σύστημα πρακτόρων οι παραπάνω αναφερόμενοι μέθοδοι αλληλεπιδρούν με την υπηρεσία μεταφοράς του συστήματος αυτού. Σε συνεργασία με την υπηρεσία διαχείρισης ενεργοποιείται ο πράκτορας που μεταφέρθηκε. Επιπλέον, ενημερώνεται με τα νέα δεδομένα που προκύπτουν για την κατάσταση του συστήματος η υπηρεσία ευρετηρίου και σταθερότητας.

- *Τη μέθοδο για τον τερματισμό του συστήματος πρακτόρων,* **terminate_agent_system()**.

Η μέθοδος αυτή συνεργάζεται με την υπηρεσία διαχείρισης για τον τερματισμό της εκτέλεσης των πρακτόρων και των περιβαλλόντων εκτέλεσης. Στη συνέχεια, επικοινωνεί με το σύστημα ευρετηρίου της περιοχής για να δηλώσει τον τερματισμό της λειτουργίας του, να γνωστοποιήσει δηλαδή, την κατάστασή του στις υπόλοιπες οντότητες της περιοχής.

- *Τη μέθοδο που προστέθηκε στον ορισμό του προτύπου στα πλαίσια της εργασίας αυτής και δίνει τη δυνατότητα στους πράκτορες να καλούν μεθόδους άλλων τοπικών ή απομακρυσμένων πρακτόρων,* **exec_agent_method()**.

Για την επίκληση μεθόδου ενός πράκτορα είναι απαραίτητη η συνεργασία της υπηρεσίας ευρετηρίου, για τον εντοπισμό του ζητούμενου πράκτορα και της υπηρεσίας επικοινωνίας, για την ανταλλαγή των απαραίτητων πληροφοριών.

4.4.1.2 Κλάση MAFFinderImplementation

Η κλάση αυτή υλοποιεί τον ορισμό διεπαφής του συστήματος ευρετηρίου. Αναλαμβάνει την αποθήκευση και διανομή της πληροφορίας για την τρέχουσα κατάσταση του συστήματος. Υλοποιεί τη διεπαφή επικοινωνίας του συστήματος ευρετηρίου που ορίζει το πρότυπο.

Οι μέθοδοι που περιέχονται στην κλάση, διασφαλίζουν τη λειτουργικότητα και αφορούν την ενημέρωση του συστήματος για την τρέχουσα κατάσταση της περιοχής. Η κατάσταση της περιοχής καθορίζεται από τις οντότητες που ανήκουν στην περιοχή και είναι: τα συστήματα πρακτόρων, τα περιβάλλοντα εκτέλεσης πρακτόρων, που φιλοξενούν τα συστήματα της περιοχής και οι πράκτορες που δραστηριοποιούνται σε αυτά. Οπότε, παρέχονται τρόποι εισαγωγής μίας νέας οντότητας, διαγραφή της, ανανέωσης της πληροφορίας που διατηρείτε γι' αυτήν (μόνο στην περίπτωση των πρακτόρων, μια και είναι η μόνη οντότητα που αλλάζει δυναμικά τα χαρακτηριστικά της λόγο της ιδιότητας της μετακίνησης) αλλά και γνωστοποίηση της κατάστασης σε οποιονδήποτε εξυπηρετούμενο. Η σημασιολογία των μεθόδων έχει αναλυθεί στο τρίτο κεφάλαιο, και αντικατοπτρίζει τη σημασιολογία των μεθόδων που απαρτίζουν την κλάση MAFFinderImplementation, η οποία περιέχει την υλοποίηση του ευρετηρίου. Τα ονόματα παραθέτονται σε κατηγορίες ανάλογα με την οντότητα που αφορούν. Ταυτόχρονα με την παράθεση των ονομάτων των μεθόδων περιγράφεται και το είδος της πληροφορίας που διατηρείται για κάθε οντότητα.

- *Οι μέθοδοι που αφορούν τις οντότητες πρακτόρων είναι:* **register_agent(), unregister_agent(), lookup_agent()**.

Η πληροφορία που διατηρείται για κάθε πράκτορα σχετίζεται με τη δικαιοδοτική του αρχή, την τοποθεσία του, τον τύπο του αλλά και τα ιδιαίτερα χαρακτηριστικά του που μπορούν να οριστούν από τον κατασκευαστή του.

- *Οι μέθοδοι που αφορούν τα περιβάλλοντα εκτέλεσης είναι:* **register_place(), unregister_place(), lookup_place()**.

Για τα περιβάλλοντα εκτέλεσης χρειάζεται μόνο η πληροφορία της τοποθεσίας του.

- *Οι μέθοδοι που αφορούν τα συστήματα πρακτόρων είναι:* **register_agent_system(), unregister_agent_system(), lookup_agent_sytem()**.

Τα συστήματα πρακτόρων εκτός της πληροφορίας της τοποθεσίας τους, σχετίζονται με τη δικαιοδοτική τους αρχή, τον τύπο τους αλλά και τα ιδιαίτερα χαρακτηριστικά που μπορεί να τους αποδώσει ο κατασκευαστής τους.

4.4.1.3 Κλάσεις ενδείξεων σφαλμάτων (exceptions)

Όλες οι μέθοδοι που περιγράφονται στις διεπαφές του συστήματος πρακτόρων και ευρετηρίου αλληλεπιδρούν με δομικά στοιχεία των συστημάτων αυτών με σκοπό να εκπληρώσουν την εργασία που τους ορίζει το πρότυπο. Κατά τη διάρκεια της εκτέλεσης μπορεί

- να συμβούν σφάλματα που υπεύθυνος μπορεί να είναι είτε ο εξυπηρετούμενος, παρέχοντας λανθασμένες πληροφορίες, είτε το σύστημα εξυπηρετητής, αδυνατώντας να εκτελεστεί ορθά,
- να προκύψει αδυναμία εκτέλεσης της συγκεκριμένης εργασίας λόγω έλλειψης των προνομίων από τον εξυπηρετούμενο,
- να μην υπάρχει η ζητούμενη πληροφορία.

Οπότε, το πρότυπο εισάγει τις κλάσεις ένδειξης σφαλμάτων για τη γνωστοποίηση των λόγων του αιφνιδιαστικού τερματισμού της εκτέλεσης. Τα ονόματα των κλάσεων αυτών είναι: AgentNotFound, AgentIsRunning, AgentIsSuspended, ArgumentInvalid, ClassUnknown, DeserializationFailed, EntryNotFound, FinderNotFound, MAFExtendedException, NameInvalid, ResumeFailed, SuspsendFailed, TerminateFailed, ExecMethodFailed και είναι υποκλάσεις της γενικής κλάσης για ένδειξη σφάλματος της γλώσσας προγραμματισμού Java.

4.4.2 Πακέτο EasyAgent.AgentSystem

Το πακέτο αυτό περιέχει τις κλάσεις που υλοποιούν το σύστημα πρακτόρων EasyAgent και τις λειτουργίες του. Η οντότητα του συστήματος πρακτόρων συνίσταται από τα δομικά στοιχεία που ορίζονται από το πρότυπο και υλοποιούνται μοναδικά από τον εκάστοτε κατασκευαστή. Ο τρόπος αλληλεπίδρασης του συστήματος πρακτόρων με τα υπόλοιπα συστήματα είναι προκαθορισμένος, μέσω των διεπαφών που έχουν τυποποιηθεί από το πρότυπο, αλλά όχι μοναδικός. Μπορούν, δηλαδή, τα συστήματα πρακτόρων να παρέχουν επιπρόσθετα μονοπάτια πρόσβασης σε αυτά. Η χρήση τέτοιων εναλλακτικών τρόπων αλληλεπίδρασης δίνει από τη μία περισσότερη ευελιξία στο εκάστοτε σύστημα, αλλά από την άλλη επιβάλλει την αναλυτική περιγραφή των μεθόδων αυτών για να είναι δυνατή η χρήση τους από οντότητες προερχόμενες από διαφορετικούς κατασκευαστές. Το σύστημα που αναπτύχθηκε στα πλαίσια της εργασίας αυτής χρησιμοποιεί ως μοναδικό τρόπο αλληλεπίδρασης, με ξένες ως προς αυτό οντότητες, αυτόν που καθορίζεται από τις διεπαφές του προτύπου.

Τα δομικά στοιχεία του συστήματος πρακτόρων υλοποιούνται στις κλάσεις, που περιέχονται στο πακέτο, για το οποίο γίνεται λόγος στην ενότητα αυτή. Αναλυτικότερα είναι

- το σύστημα πρακτόρων, κλάση AgentSystemServer.

- τον πυρήνα του συστήματος, κλάση AgentSystemCore.
- το περιβάλλον εκτέλεσης των πρακτόρων, δηλαδή το μέρος, κλάση Place.
- η ίνα που αναλαμβάνει την εκτέλεση του πράκτορα, κλάση AgentThread.
- οι υπηρεσίες που προσφέρει το σύστημα πρακτόρων, μεταφοράς, ασφάλειας, σταθερότητας, διαχείρισης, ευρετηρίου, και επικοινωνίας, οι οποίες αντιστοιχούν στις κλάσεις ASTransport, ASSecurity, ASPersistent, ASManagement, ASRegistration και ASCommunication.

Η λειτουργικότητα της κάθε οντότητας, το σύνολο των οποίων συνιστά το σύστημα πρακτόρων, καθορίζεται πλήρως από τις μεθόδους που παρέχει η αντίστοιχη κλάση που την υλοποιεί. Για το λόγο αυτό στις επόμενες υποενότητες παράλληλα με την περιγραφή της κλάσης παραθέτονται και οι μέθοδοι που εσωκλείουν. Θα ξεκινήσουμε από την περιγραφή των κλάσεων που ορίζουν τις υπηρεσίες του πυρήνα του συστήματος πρακτόρων μια και οι οντότητες αυτές διαδραματίζουν ίσως τον πιο καθοριστικό ρόλο στη λειτουργία του συστήματος. Στη συνέχεια, θα ασχοληθούμε ξεχωριστά με κάθε μία από τις υπόλοιπες οντότητες που απαρτίζουν το σύστημα πρακτόρων.

4.4.2.1 Κλάση AgentSystemServer

Είναι η κλάση που περιέχει την κύρια (main) μέθοδο του συστήματος πρακτόρων. Αναλαμβάνει την έναρξη της λειτουργίας του ενεργοποιώντας πρώτα έναν εξυπηρετητή για ανεύρεση αντικειμένων (ORB) και καταχωρώντας σε αυτόν το σύστημα πρακτόρων ως αντικείμενο τύπου CORBA. Φροντίζει να επικοινωνήσει με την υπηρεσία ονοματολογίας για την εύρεση του συστήματος ευρετηρίου της περιοχής. Στη συνέχεια εγγράφει το σύστημα πρακτόρων στην υπηρεσία ευρετηρίου και αρχικοποιεί τον πυρήνα του συστήματος πρακτόρων.

4.4.2.2 Κλάση AgentSystemCore

Η κλάση AgentSystemCore εσωκλείει την υλοποίηση του πυρήνα του συστήματος πρακτόρων. Ο πυρήνας, όπως έχει αναφερθεί, είναι η οντότητα που παρέχει την ελάχιστη απαιτούμενη λειτουργικότητα για τη φιλοξενία των πρακτόρων. Συνίσταται από ένα σύνολο υπηρεσιών που αρχικοποιεί κατά την έναρξη της λειτουργίας του συστήματος πρακτόρων. Εφοδιάζει τις υπηρεσίες, κατά την φάση της αρχικοποίησης, με όλες τις απαραίτητες πληροφορίες. Συνάμα παρέχει και πληροφορίες για το σύστημα πρακτόρων με τις παρακάτω μεθόδους

Μέθοδος	Περιγραφή
getName()	επιστρέφει το όνομα του συστήματος πρακτόρων
getAuthority()	επιστρέφει τη δικαιοδοτική αρχή του συστήματος πρακτόρων
getDescription()	επιστρέφει την περιγραφή του συστήματος πρακτόρων
getLocation()	επιστρέφει την τοποθεσία του συστήματος πρακτόρων.
getMAFFinder()	επιστρέφει μία οντότητα, αντιπρόσωπο της οντότητας που υλοποιεί το σύστημα ευρετηρίου. Η δυνατότητα αυτή παρέχεται από το γεγονός ότι τόσο το σύστημα πρακτόρων όσο και το σύστημα ευρετηρίου είναι ορισμένα ως αντικείμενα μέσα σε ένα εξυπηρετητή τύπου ORB. Μέσω του αντιπροσώπου αυτού είναι δυνατή η κλήση των μεθόδων που περιγράφονται στη διεπαφή του.

Πίνακας 4-1: Μέθοδοι της κλάσης AgentSystemCore

4.4.2.3 Κλάσεις υλοποίησης των υπηρεσιών

Οι υπηρεσίες που αναλύονται στην ενότητα αυτή, συνεργάζονται συνεχώς μεταξύ τους κατά τη διάρκεια της εκτέλεσης του συστήματος πρακτόρων με τελικό στόχο την παροχή ενός ασφαλούς περιβάλλοντος δράσης των κινούμενων πρακτόρων. Οι υπηρεσίες ανήκουν στον πυρήνα του συστήματος πρακτόρων και οποιαδήποτε αναφορά σε αυτές γίνεται μέσω του πυρήνα. Επιπλέον, χρησιμοποιούνται από τα περιβάλλοντα εκτέλεσης πρακτόρων που φιλοξενεί το σύστημα πρακτόρων αλλά και από τη διεπαφή χρήσης του συστήματος.

Κλάση ASRegistration (Υπηρεσία ευρετηρίου)
Η υλοποίηση της υπηρεσίας ευρετηρίου εσωκλείνεται στην κλάση ASRegistration. Η υπηρεσία είναι υπεύθυνη για την παροχή πληροφοριών σχετικών με τους πράκτορες που δραστηριοποιούνται στο τοπικό σύστημα πρακτόρων αλλά και για τα περιβάλλοντα εκτέλεσης που υπάρχουν στο εν λόγω σύστημα. Επειδή όμως τόσο οι κινούμενοι πράκτορες όσο και τα περιβάλλοντα εκτέλεσης (μέρη) είναι οντότητες που μεταβάλλονται κατά την διάρκεια εκτέλεσης του συστήματος, παρέχονται μέθοδοι για τη δυναμική ενημέρωση της πληροφορίας που σχετίζεται με αυτές. Οι κινούμενοι πράκτορες, από την μία, αλλάζουν κατά τη διάρκεια της ζωής τους μέρος εκτέλεσης, ενώ τα περιβάλλοντα εκτέλεσης δημιουργούνται και τερματίζονται με πρωτοβουλία του χειριστή του συστήματος πρακτόρων. Η υπηρεσία επιπλέον, είναι υπεύθυνη για την ενημέρωση του συστήματος ευρετηρίου της περιοχής για όλες τις αλλαγές που συμβαίνουν στο τοπικό σύστημα πρακτόρων. Η υπηρεσία αυτή, θα μπορούσε κανείς να πει ότι, διατηρεί τοπικά ένα αντίγραφο της πληροφορίας, που υπάρχει στο σύστημα ευρετηρίου της περιοχής για το εν λόγω σύστημα πρακτόρων.

Πρόσβαση στην υπηρεσία ευρετηρίου έχουν μόνο οι υπηρεσίες διαχείρισης και σταθερότητας, μια και αυτές έχουν άμεση ανάγκη τις πληροφορίας που διατηρεί η εν λόγω υπηρεσία. Οι μέθοδοι που παρέχονται από την κλάση ASRegistration περιγράφονται από τον παρακάτω πίνακα.

Μέθοδος	Περιγραφή
registerPlace()	καλείται οποτεδήποτε δημιουργείται ένα νέο περιβάλλον εκτέλεσης (μέρος). Αναλαμβάνει την ενσωμάτωση όλων των πληροφοριών (όνομα, μέγιστος αριθμός φιλοξενούμενων πρακτόρων, κατάλογος αποθήκευσης κλάσεων) με το νέο μέρος, καθώς και την αποθήκευσή τους. Επίσης ενημερώνει το σύστημα ευρετηρίου της περιοχής για την έναρξη της εκτέλεσης του νέου περιβάλλοντος.
unregisterPlace()	καλείται οποτεδήποτε τερματίζεται η λειτουργία ενός περιβάλλοντος εκτέλεσης. Εκτός της αποδέσμευσης της πληροφορίας που κρατούταν τοπικά για το εν λόγω μέρος, αναλαμβάνει και την ενημέρωση του συστήματος ευρετηρίου της περιοχής.
registerAgent()	συσχετίζει την έναρξη εκτέλεσης ενός πράκτορα (είτε μετά από μετακίνησή του είτε μετά από εκ νέου δημιουργία του) με το μέρος στο οποίο λαμβάνει χώρα η διαδικασία αυτή. Αποθηκεύει τοπικά τις πληροφορίες που σχετίζονται με τον κινούμενο πράκτορα και τον εγγράφει στο σύστημα ευρετηρίου της περιοχής.
unregisterAgent()	καλείται όταν τερματιστεί η εκτέλεση ενός πράκτορα. Ενημερώνει, και αυτή με τη σειρά της, το σύστημα ευρετηρίου της περιοχής

Πίνακας 4-2: Μέθοδοι της κλάσης ASRegistration

Κλάση ASManagement (Υπηρεσία διαχείρισης)
Η υλοποίηση της υπηρεσίας διαχείρισης βρίσκεται στην κλάση ASManagement. Η υπηρεσία είναι υπεύθυνη για τη δημιουργία και τον τερματισμό πρακτόρων και περιβαλλόντων εκτέλεσης. Επιπρόσθετα, αναλαμβάνει τη διαχείριση των πρακτόρων που φιλοξενούνται από το σύστημα πρακτόρων. Σε συνεργασία με την υπηρεσία ευρετηρίου του τοπικού συστήματος προσφέρει εξαντλητικούς καταλόγους των πρακτόρων και περιβαλλόντων εκτέλεσης που δραστηριοποιούνται στο σύστημα. Οι μέθοδοι που προσφέρονται από την κλάση υλοποίησης είναι προσβάσιμες από εξωτερικούς χρήστες (άλλες εφαρμογές ή και ανθρώπους) μια και αποτελούν το σύνολο των βασικών λειτουργιών που προσφέρει το σύστημα πρακτόρων. Επιπλέον, είναι η υπηρεσία που προσφέρει τη δυνατότητα παρακολούθησης της τρέχουσας κατάστασης του συστήματος μια και παρέχει τρόπους ανάκτησης οποιασδήποτε πληροφορίας σχετίζεται με οντότητες που δραστηριοποιούνται στο σύστημα.

Στο σύστημα EasyAgent η υπηρεσία αυτή αναλαμβάνει την εξυπηρέτηση όλων των αιτήσεων που εισέρχονται στο σύστημα από την διεπαφή χρήσης (user interface). Οι αιτήσεις σχετίζονται όχι μόνο με την εισαγωγή και διαγραφή πρακτόρων και περιβαλλόντων εκτέλεσης αλλά και με την ανάκτηση των ιδιαίτερων χαρακτηριστικών των οντοτήτων αυτών. Στον παρακάτω πίνακα περιγράφονται οι μέθοδοι που παρέχονται από την κλάση ASManagement.

Μέθοδος	Περιγραφή
createPlace()	χρησιμοποιείται για την δημιουργία ενός νέου περιβάλλοντος εκτέλεσης στο τοπικό σύστημα πρακτόρων. Αναλαμβάνει όλες τις αρχικοποιήσεις που σχετίζονται με την λειτουργία αυτή όπως καθορισμός καταλόγου για αποθήκευση των ορισμών των κλάσεων, δημιουργία μίας ίνας που θα αναλάβει την εκτέλεση του περιβάλλοντος και αρκετές άλλες που θα δούμε αναλυτικά στην περιγραφή της κλάσης Place. Επίσης, ενημερώνει την υπηρεσία ευρετηρίου.
suspendPlace()	χρησιμοποιείται για την προσωρινή αναστολή λειτουργίας ενός περιβάλλοντος εκτέλεσης. Αναλαμβάνει την αναστολή λειτουργίας όλων των φιλοξενούμενων πρακτόρων.
resumePlace()	επανεκκινεί την λειτουργία ενός περιβάλλοντος εκτέλεσης μετά από αναστολή με χρήση της suspendPlace(). Αναλαμβάνει την επανέναρξη της εκτέλεσης των πρακτόρων που εκτελούνταν πριν την αναστολή.
terminatePlace()	χρησιμοποιείται για τον τερματισμό λειτουργίας ενός περιβάλλοντος εκτέλεσης κινούμενων πρακτόρων. Αναλαμβάνει να τερματίσει τυχών ακόμα εκτελούμενους πράκτορες στο εν λόγω μέρος, να μεταφέρει σε σταθερό αποθηκευτικό μέσο τους ορισμούς των κλάσεων που μεταφέρθηκαν για την εκτέλεση των πρακτόρων, και να ενημερώσει την υπηρεσία ευρετηρίου.
suspendAgent()	χρησιμοποιείται για την προσωρινή αναστολή της εκτέλεσης της ίνας που έχει αναλάβει την εκτέλεση του πράκτορα.
resumeAgent()	χρησιμοποιείται για την επανέναρξη της εκτέλεσης της ίνας που έχει αναλάβει την εκτέλεση του πράκτορα. Η αναστολή της εκτέλεσης έχει γίνει με χρήση της μεθόδου suspendAgent().

terminateAgent()	χρησιμοποιείται για τον τερματισμό της εκτέλεσης ενός πράκτορα. Αναλαμβάνει όχι μόνο να ελευθερώσει τυχών πόρους που χρησιμοποιούσε ο πράκτορα αλλά και να ενημερώσει την υπηρεσία ευρετηρίου.
getAgentStatus()	επιστρέφει την κατάσταση του πράκτορα. Το σύνολο των πιθανών καταστάσεων απαρτίζεται από τις: εκτελέσιμη (running), αναστολής εκτέλεσης (suspended), και τερματισμού εκτέλεσης (terminated).
listPlaces()	επιστρέφει ένα πλήρες κατάλογο που περιέχει τα ονόματα όλων των περιβαλλόντων εκτέλεσης που φιλοξενεί το εν λόγω σύστημα πρακτόρων.
listAgents()	επιστρέφει ένα πλήρες κατάλογο που περιέχει τα ονόματα όλων των πρακτόρων που δραστηριοποιούνται στα περιβάλλοντα εκτέλεσης του εν λόγω συστήματος πρακτόρων.
createAgentFromClass()	χρησιμοποιείται για την δημιουργία ενός νέου πράκτορα μετά από αίτηση που μπορεί να προκύψει, είτε από την διεπαφή χρήσης του συστήματος, είτε από κάποιο άλλο πράκτορα (που βρίσκεται σε απομακρυσμένο ή στο τοπικό σύστημα) μέσω της διεπαφής επικοινωνίας με το σύστημα πρακτόρων. Η ολοκλήρωση της διαδικασίας της δημιουργίας ενός νέου πράκτορα περνάει από τα εξής στάδια: δημιουργία μίας ίνας που θα αναλάβει την εκτέλεση του πράκτορα, δημιουργία ενός νέου στιγμιότυπου της κλάσης του πράκτορα, αίτηση προς την υπηρεσία μεταφοράς για την μεταφορά άγνωστων, μέχρι την στιγμή εκείνη για το τοπικό σύστημα, κλάσεων που χρειάζονται για την δημιουργία του στιγμιότυπου (όποτε είναι απαραίτητο), γένεση ενός μοναδικού ονόματος για τον πράκτορα και έναρξη εκτέλεσης της ίνας στην οποία ανατέθηκε το στιγμιότυπο του πράκτορα. Εφόσον σε κάθε νέο πράκτορα αποδίδεται μία ξεχωριστή ίνα για να εκτελεστεί, αυτός εκτελείται ανεξάρτητα από όλους τους άλλους πράκτορες. Η μέθοδος αυτή αναλαμβάνει να ενημερώσει και την υπηρεσία ευρετηρίου του τοπικού συστήματος.
createAgentFromByteArray()	χρησιμοποιείται για την δημιουργία ενός πράκτορα μετά από αίτηση που μπορεί να προκύψει, είτε κατά την διαδικασία μεταφοράς του πράκτορα στο τοπικό σύστημα, είτε κατά την διαδικασία αποκατάστασης της ορθής λειτουργίας του συστήματος πρακτόρων μετά από κάποιο σφάλμα στην εκτέλεσή του. Ο λόγος είναι ότι και στις δύο περιπτώσεις έχει ήδη προηγηθεί αναστολή της εκτέλεσης του πράκτορα και σειριοποίηση των χρησιμοποιούμενων κλάσεών του για την ορθή διατήρηση της κατάστασής του. Τα στάδια που αποτελούν την διαδικασία αυτή είναι: αποσειριοποίηση της κατάστασης του πράκτορα, αίτηση προς την υπηρεσία μεταφοράς για την μεταφορά άγνωστων, μέχρι την στιγμή εκείνη για το τοπικό σύστημα, κλάσεων που χρειάζονται στην διαδικασία της αποσειριοποίησης (όποτε είναι απαραίτητο), δημιουργία μίας ίνας που θα αναλάβει την εκτέλεση του πράκτορα, ανάθεση της τρέχουσας κατάστασης στην ίνα που θα αναλάβει την εκτέλεση του πράκτορα και επανέναρξη της εκτέλεσης του πράκτορα. Τελικά, η μέθοδος αυτή αναλαμβάνει να ενημερώσει και την υπηρεσία ευρετηρίου του τοπικού συστήματος.

Πίνακας 4-3: Μέθοδοι της κλάσης ASManagement

Κλάση ASTransport (Υπηρεσία μεταφοράς)
Η κλάση ASTransport υλοποιεί την υπηρεσία της μεταφοράς. Η υπηρεσία αυτή αναλαμβάνει την μετακίνηση των πρακτόρων μεταξύ των συστημάτων, αλλά και την μεταφορά των ορισμών των κλάσεων που χρειάζονται για την εκτέλεσή τους. Η υπηρεσία διατηρεί ένα κατάλογο από τις μεταφερόμενες προς το σύστημα κλάσεις στην προσπάθειά της να μειώσει τον όγκο των ορισμών των κλάσεων που μεταφέρονται πάνω από το δίκτυο. Η λειτουργία αυτή της υπηρεσίας μεταφοράς έχει τις ιδιότητες μίας κρυφής μνήμης δεύτερου επιπέδου. Η κρυφή μνήμη πρώτου επιπέδου υποστηρίζεται από το περιβάλλον εκτέλεσης των πρακτόρων. Όπως θα δούμε στην περιγραφή της κλάσης που το υλοποιεί, κάθε μέρος κατά την διάρκεια της λειτουργίας του αποθηκεύει τους ορισμούς των κλάσεων που έχει ζητήσει για την ικανοποίηση των αναγκών εκτέλεσης των πρακτόρων που φιλοξενεί. Η υπηρεσία μεταφοράς, αναλαμβάνοντας να εξυπηρετήσει τις αιτήσεις μεταφοράς των κλάσεων που προέρχονται από όλα τα περιβάλλοντα εκτέλεσης που δραστηριοποιούνται στο τοπικό σύστημα πρακτόρων, γνωρίζει όλες τις κλάσεις που χρησιμοποιήθηκαν και χρησιμοποιούνται από τους πράκτορες. Έτσι, μία αίτηση για μεταφορά κλάσης είναι πολύ πιθανό να εξυπηρετηθεί από το τοπικό σύστημα με κόστος μίας αναζήτησης σε ένα ευρετήριο. Με την τεχνική που χρησιμοποιείται, η πάροδος του χρόνου συνεπάγεται σε αύξηση του αριθμού των γνωστών στο σύστημα κλάσεων άρα και μεγαλύτερη πιθανότητα αποφυγής της χρήσης του δικτύου. Ενώ, από την άλλη μεριά, το κόστος της αναζήτησης στο ευρετήριο παραμένει σχεδόν αμετάβλητο.

Μέθοδος	Περιγραφή
fetchClass()	επιστρέφει τον ορισμό της ζητούμενης κλάσης. Η κλάση, κατ' αρχήν, αναζητάτε στο τοπικό σύστημα πρακτόρων. Εάν αποτύχει η αναζήτηση, προωθείται η αίτηση στο σύστημα πρακτόρων που ξεκίνησε την διαδικασία της μεταφοράς ή που αιτήθηκε της δημιουργίας του πράκτορα στο τοπικό σύστημα. Πριν η κλάση παραδοθεί στο περιβάλλον εκτέλεσης που την ζήτησε ενημερώνεται το ευρετήριο των γνωστών για το σύστημα κλάσεων.
receiveAgent()	χρησιμοποιείται για την εξυπηρέτηση της λειτουργίας της μεταφοράς ενός πράκτορα στο τοπικό σύστημα πρακτόρων. Αναλαμβάνει την αποκωδικοποίηση των δεδομένων του πράκτορα. Στην συνέχεια τροφοδοτώντας με τις απαραίτητες πληροφορίες την υπηρεσία διαχείρισης αιτεί την επανέναρξη της εκτέλεσής του.

Πίνακας 4-4: Μέθοδοι της κλάσης ASTransport

Κλάση ASPersistent (Υπηρεσία σταθερότητας)
Η κλάση ASPersistent εσωκλείει την υλοποίηση της υπηρεσίας σταθερότητας του συστήματος πρακτόρων. Η υπηρεσία αυτή σε συνεργασία με την υπηρεσία διαχείρισης υποστηρίζει την επανέναρξη της λειτουργίας του συστήματος πρακτόρων, χωρίς την απώλεια δεδομένων, μετά από διακοπή λόγω σφάλματος στην εκτέλεσή του. Για να είναι δυνατή η επανέναρξη πρέπει η υπηρεσία να αποθηκεύει την κατάσταση τόσο των πρακτόρων όσο και των περιβαλλόντων εκτέλεσης σε μόνιμο αποθηκευτικό μέσο (δίσκος). Με τον τρόπο αυτό όμως δεν αντιμετωπίζονται περιπτώσεις διακοπής της λειτουργίας του συστήματος πρακτόρων που οφείλονται σε εξωτερικούς ως προς το σύστημα παράγοντες (π.χ. διακοπή τροφοδοσίας). Για την προφύλαξη του συστήματος πρακτόρων από τέτοιες δυσμενείς περιπτώσεις, η υπηρεσία σταθερότητας υποστηρίζει την αυτόματη αποθήκευση της κατάστασης του συστήματος ανά τακτά χρονικά διαστήματα που μπορούν να καθοριστούν από τον διαχειριστή του εν λόγω συστήματος.

Η κατάσταση του συστήματος πρακτόρων αποτελείται από την κατάσταση των οντοτήτων που αποτελούν το σύστημα πρακτόρων. Αναλυτικότερα,

- από την κατάσταση εκτέλεσης των πρακτόρων που φιλοξενούνται,
- από την πληροφορία τόσο για τους πράκτορες όσο και για τις κλάσεις που διατηρούν τα περιβάλλοντα εκτέλεσης,
- και από την πληροφορία που διατηρούν οι υπηρεσίες του ευρετηρίου και μεταφοράς του τοπικού συστήματος πρακτόρων.

Μέθοδοι	Περιγραφή
saveStateToDisk()	η μέθοδος αυτή αναλαμβάνει να αποθηκεύσει όλα τα στοιχεία που συνιστούν την κατάσταση του συστήματος σε μόνιμο αποθηκευτικό μέσο. Έτσι αποθηκεύει εκτός από τα δεδομένα που προέχοντα από τις υπηρεσίες και τα περιβάλλοντα εκτέλεσης, τις κλάσεις και την σειριοποιημένη μορφή των κλάσεων.
readStateFromDisk()	η μέθοδος αυτή αναλαμβάνει την επανέναρξη της λειτουργίας του συστήματος αποκαταστώντας την κατάσταση που αυτό είχε πριν αναστείλει την λειτουργία του. Διαβάζει την σωσμένη στο μόνιμο αποθηκευτικό μέσω κατάσταση και αρχικοποιεί τις υπηρεσίες του ευρετηρίου και της μεταφοράς. Στην συνέχεια, σε συνεργασία με την υπηρεσία διαχείρισης ξεκινάει την λειτουργία των περιβαλλόντων εκτέλεσης και των πρακτόρων που δραστηριοποιούνταν στα περιβάλλοντα αυτά.
setAutoSave()	δίνει τη δυνατότητα στο χειριστή του συστήματος να αρχικοποιήσει την διαδικασία της αυτόματης αποθήκευσης της κατάστασης του συστήματος ανά τακτά χρονικά διαστήματα. Μετά την αρχικοποίηση της διαδικασίας ένας δαίμονας (deamon) αναλαμβάνει το σώσιμο της τρέχουσας κατάστασης σε μόνιμο μέσο. Η περίοδος της λειτουργίας αυτής καθορίζεται από τον χειριστή του συστήματος. Τα στάδια που περιλαμβάνει είναι, προσωρινή διακοπή της εκτέλεσης των περιβαλλόντων εκτέλεσης, άρα και των κινούμενων πρακτόρων που φιλοξενούν, σειριοποίηση του κάθε πράκτορα, αποθήκευση των κωδικοποιημένων δεδομένων που προκύπτουν, και τέλος επανέναρξη της εκτέλεσης.
shutdown()	που αναλαμβάνει τη διακοπή λειτουργίας του συστήματος πρακτόρων. Η διαδικασία αυτή περιλαμβάνει τον τερματισμό εκτέλεσης των πρακτόρων και των περιβαλλόντων εκτέλεσης που δραστηριοποιούνται στο σύστημα. Την ενημέρωση του συστήματος ευρετηρίου για τον τερματισμό του εν λόγω συστήματος. Την αποθήκευση της κατάστασης του συστήματος, εάν είναι απαραίτητο.

Πίνακας 4-5: Μέθοδοι της κλάσης ASPersistent

Κλάση ASCommunication (Υπηρεσία επικοινωνίας)

Η κλάση ASCommunication εσωκλείει την υλοποίηση της υπηρεσίας επικοινωνίας. Η υπηρεσία αναλαμβάνει να εξυπηρετήσει τις ανάγκες επικοινωνίας του τοπικού συστήματος πρακτόρων με τα υπόλοιπα συστήματα πρακτόρων και ευρετηρίου. Οποιαδήποτε μορφή επικοινωνίας των συστημάτων που ορίζει το περιβάλλον εξυπηρετείται από το γενικότερο πλαίσιο πάνω από το οποίο εκτελούνται τα συστήματα, το περιβάλλον κατανεμημένων οντοτήτων της CORBA. Η υπηρεσία επικοινωνίας λοιπόν, κωδικοποιεί τα μηνύματα σύμφωνα με τις απαιτήσεις που θέτει η υλοποίηση του ORB και ζητά από τον τελευταίο να

διεκπεραιώσει τη συναλλαγή. Από την άλλη πλευρά, αποκωδικοποιεί το μήνυμα και το αποδίδει στο σύστημα για περαιτέρω επεξεργασία.

Κλάση ASSecurity (Υπηρεσία ασφάλειας)
Η κλάση ASSecurity εσωκλείει την υλοποίηση της υπηρεσίας ασφάλειας Το περιβάλλον EasyAgent είναι υπεύθυνο για την εσωτερική ασφάλεια των συστημάτων πρακτόρων (βλέπε ενότητα 4.3.1). Η ασφάλεια στηρίζεται στους μηχανισμούς ασφάλειας που παρέχει το περιβάλλον προγραμματισμού JDK1.2. Χρησιμοποιεί τα ονόματα των πρακτόρων με σκοπό να τους παρέχει πρόσβαση στους πόρους του συστήματος, όπως χρήση υπολογιστικής ισχύς και πρόσβαση στα αρχεία του τοπικού συστήματος αρχείων. Ο έλεγχος, με σκοπό την αποφυγή περιπτώσεων παραβίασης ασφάλειας, γίνεται σε κάθε αλληλεπίδραση των κινούμενων οντοτήτων με το σύστημα. Κάθε αίτηση κινούμενου πράκτορα εμπεριέχει και την ταυτότητά του. Έτσι, είναι δυνατό να ελέγχονται τα δικαιώματα πρόσβασης του πράκτορα αυτού.

4.4.2.4 Κλάση Place

Η κλάση Place υλοποιεί το περιβάλλον εκτέλεσης των πρακτόρων. Το περιβάλλον εκτέλεσης αποτελεί το μοναδικό παράθυρο επικοινωνίας του πράκτορα με το υπόλοιπο σύστημα αλλά και αντίστροφα. Περιέχει λοιπόν μεθόδους που επιτρέπουν την διαχείριση των πρακτόρων καθώς και μεθόδους που παρέχουν πληροφορίες σχετικές με το περιβάλλον εκτέλεσης. Αναλυτικότερα, στην κλάση περιέχονται, οι μέθοδοι:

- **createAgent(), suspendAgent(), resumeAgent, terminateAgent(), cloneAgent(), saveAgent(),** για την δημιουργία, αναστολή εκτέλεσης, επανέναρξη και τερματισμό εκτέλεσης, κλωνοποίησης, και αποθήκευσης ενός πράκτορα αντίστοιχα. Οι μέθοδοι αυτοί εξυπηρετούν όλο το σύνολο των λειτουργιών διαχείρισης που προσφέρει το σύστημα πρακτόρων σύμφωνα με το πρότυπο.
- **serialiazeAgent(), deserializeAgent(),** για την σειριοποίηση και αποσειριοποίηση των κλάσεων ενός πράκτορα. Χρησιμοποιούνται κατά την μετακίνηση των πρακτόρων άλλα και από την υπηρεσία σταθερότητας για την αποθήκευση και επαναφορά της κατάστασης του συστήματος.
- **execMethodAgent()** για την κλήση μεθόδου ενός πράκτορα που δραστηριοποιείται στο εν λόγο περιβάλλον εκτέλεσης. η μέθοδος αυτή υπάρχει λόγο της προσθήκης που έγινε στην διεπαφή του συστήματος πρακτόρων που ορίζει το πρότυπο.
- **getName(), getLocation(), getAgentList(),** είναι μέθοδοι που παρέχουν πληροφορίες για το περιβάλλον εκτέλεσης. Η πρώτη επιστρέφει το όνομα του μέρους. Το όνομα αποδίδεται στο περιβάλλον εκτέλεσης από τον διαχειριστή του συστήματος πρακτόρων. Η δεύτερη επιστρέφει την τοποθεσία του περιβάλλοντος εκτέλεσης. Η τοποθεσία αυτή χρησιμοποιείται για να δηλώσει και την θέση των πρακτόρων. Η τρίτη επιστρέφει μία λίστα που περιέχει τους πράκτορες που εκτελούνται την στιγμή εκείνη στο περιβάλλον.
- **getFinder(),** που με την σειρά της καλεί την getMAFFinder() της κλάσης AgentSystemCore.

4.4.2.5 Κλάση AgentThread

Πολλές φορές μέχρι το σημείο αυτό έχουμε αναφέρει ότι μία ίνα αναλαμβάνει την εκτέλεση του πράκτορα. Η υλοποίηση της ίνας αυτής περιέχεται στην κλάση AgentThread που κληρονομεί από την κλάση Thread, που παρέχεται από το περιβάλλον προγραμματισμού JDK1.2, την συμπεριφορά και τις ιδιότητες εκτέλεσης ως ίνα.

4.4.3 Πακέτο EasyAgent.Finder

Το πακέτο αυτό περιέχει τις κλάσεις που υλοποιούν το σύστημα ευρετηρίου. Πιο συγκεκριμένα, περιέχει μαζί με άλλες βοηθητικές κλάσεις

- την κλάση που ορίζει το σύστημα ευρετηρίου, κλάση FinderServer.
- την κλάση που υλοποιεί την μοναδική υπηρεσία που προσφέρει το σύστημα, αυτή του ευρετηρίου, κλάση FRegistration

4.4.3.1 Κλάση FRegistration

Στην κλάση αυτή εσωκλείεται η υλοποίηση της υπηρεσίας ευρετηρίου του συστήματος. Είναι ανάλογη της υπηρεσίας ευρετηρίου που χρησιμοποιείται στα συστήματα πρακτόρων, μόνο που έχει αυξημένη λειτουργικότητα. Χρησιμοποιείται για την εγράφη όλων των οντοτήτων μιας περιοχής (πράκτορες, συστήματα πρακτόρων, περιβάλλοντα εκτέλεσης). Παρέχει μεθόδου για την αναζήτηση κάποιας οντότητας όχι μόνο βάση του ονόματός της αλλά και βάση κάποιων χαρακτηριστικών που πιθανότατα να έχει. Αυτή η δυνατότητα δίνεται μια και η εγγραφή περιλαμβάνει παράλληλα με την καταχώρηση του ονόματος και κάποιων ιδιαίτερων χαρακτηριστικών τις οντότητας. Οι μέθοδοι που περιέχονται στην κλάση είναι:

- **registerAgent(), registerPlace(), registerAgentSystem(),** για την εγγραφή πρακτόρων, περιβαλλόντων εκτέλεσης και συστημάτων πρακτόρων αντίστοιχα.
- **unregisterAgent(), unregisterPlace(), unregisterAgentSystem(),** για την διαγραφή πρακτόρων, περιβαλλόντων εκτέλεσης και συστημάτων πρακτόρων αντίστοιχα.
- **lookupAgent(), lookupPlace(), lookupAgentSystem,** για την εύρεση οντοτήτων βάση του ονόματός τους. Οι μέθοδοι αυτές επιστρέφουν την τοποθεσία της οντότητας, εφόσον η τελευταία είναι καταχωρημένη στο ευρετήριο. Αλλιώς, γεννάνε εξαίρεση τύπου NotFoundException.
- **findAgent(), findAgentSystem(),** για την εύρεση οντοτήτων βάση των χαρακτηριστικών τους. Οι μέθοδοι αυτές επιστρέφουν μία λίστα από τοποθεσίες οντοτήτων που ταιριάζουν στα επιθυμητά χαρακτηριστικά.

4.4.3.2 Κλάση FinderServer

Η κλάση αυτή περιέχει την κύρια (main) μέθοδο του συστήματος ευρετηρίου. Αναλαμβάνει την έναρξη της λειτουργίας του ενεργοποιώντας πρώτα έναν εξυπηρετητή για ανεύρεση αντικειμένων (ORB) και καταχωρώντας σε αυτόν το σύστημα ευρετηρίου ως αντικείμενο τύπου CORBA. Φροντίζει ώστε να ενημερώσει την υπηρεσία ονοματολογίας για το νέο σύστημα, δίνοντας έτσι την δυνατότητα χρησιμοποίησής του από τις υπόλοιπες οντότητες. Αρχικοποιεί την υπηρεσία ευρετηρίου του συστήματος.

5 Το περιβάλλον εκτέλεσης εφαρμογών κινούμενων πρακτόρων.

Κατά την περιγραφή της υλοποίησης του συστήματος πρακτόρων και ευρετηρίου που παρέχει το σύστημα EasyAgent, αναφέραμε και τον τρόπο αλληλεπίδρασης των οντοτήτων αυτών για την εξυπηρέτηση των αναγκών των κινούμενων πρακτόρων. Η έναρξη λειτουργίας, η αρχικοποίηση των παραμέτρων και ο τρόπος αλληλεπίδρασης με τους χρήστες των συστημάτων είναι κάποια από τα θέματα που δεν απασχόλησαν το προηγούμενο κεφάλαιο και θα παρουσιαστούν παρακάτω.

Οι κλάσεις που απασχολούν το κεφάλαιο βρίσκονται στο πακέτο EasyAgent.Gui.

Στο κεφάλαιο αυτό θα αναφερθούμε κατ' αρχήν στις τρεις οντότητες που συνιστούν το περιβάλλον και αποτελούν την βάση για ανάπτυξη και εκτέλεση εφαρμογών κινούμενων πρακτόρων. Στην συνέχεια θα περιγραφεί η διεπαφή χρήσης του συστήματος πρακτόρων και αυτή του συστήματος ευρετηρίου.

5.1 Οι οντότητες που απαρτίζουν το περιβάλλον.

Το περιβάλλον για την δραστηριοποίηση των κινούμενων πρακτόρων συνίσταται από οντότητες τριών τύπων,

- την υπηρεσία ονοματολογίας, όπως ορίζεται από την αρχιτεκτονική CORBA.
- το σύστημα πρακτόρων
- το σύστημα ευρετηρίου

Χαρακτηριστικά των οντοτήτων αυτών, που δεν έχουν αναφερθεί μέχρι τώρα, καθώς και τρόπος χρήσης τους στο περιβάλλον κινούμενων πρακτόρων θα παρουσιαστούν στις επόμενες υποενότητες.

5.1.1 Υπηρεσία ονοματολογίας (CORBA Naming Service)

Η υπηρεσία ονοματολογίας ανήκει στο σύνολο των υπηρεσιών που παρέχονται για τα αντικείμενα που ορίζονται από την αρχιτεκτονική CORBA. Το σύνολο των ορισμών των υπηρεσιών αυτών ονομάζεται Common Object Services Specification (COSS). Σκοπός της υπηρεσίας ονοματολογίας είναι ο καθορισμός μιας αντιστοίχησης μεταξύ του ονόματος ενός αντικειμένου και του αναφορικού του δείκτη τύπου IOR (Inter-Orb Reference). Αυτού του είδους η συσχέτιση, του ονόματος με το ίδιο το αντικείμενο ονομάζεται δέσιμο ονόματος (name binding). Η υπηρεσία ονοματολογίας έχει την ικανότητα να δημιουργεί τέτοιες συσχετίσεις, κατά την εγγραφή ενός αντικειμένου, καθώς και να τις αναλύει στα επιμέρους στοιχεία (resolve a name), δηλαδή να καθορίζει τον αναφορικό δείκτη τύπου IOR που σχετίζεται με ένα συγκεκριμένο όνομα.

Η υπηρεσία ονοματολογίας είναι αναπόσπαστο κομμάτι κάθε εφαρμογής που παρέχει περιβάλλον φιλοξενίας αντικειμένων, όπως αυτό ορίζεται στην δεύτερη έκδοση του ορισμού της αρχιτεκτονικής CORBA. Η αναγκαιότητά της απορρέει από το γεγονός ότι είναι η μοναδική υπηρεσία που παρέχει την δυνατότητα ανάκτησης του αναφορικού δείκτη τύπου IOR ενός αντικειμένου τύπου CORBA. Υπάρχουν και άλλοι μηχανισμοί για την ανάκτηση της ίδιας πληροφορίας, όπως η μέθοδος bind(), αλλά δεν είναι συμβατοί με την δεύτερη έκδοση του προτύπου της αρχιτεκτονικής CORBA.

Στα πλαίσια του συστήματος EasyAgent, η υπηρεσία ονοματολογίας είναι απαραίτητη, μια και διατηρεί τις αναφορές για τα αντικείμενα που υλοποιούν τα συστήματα πρακτόρων και ευρετηρίου. Τα αντικείμενα MAFAgentSystem και MAFFinder αποτελούν τα σημεία πρόσβασης για τα περιβάλλοντα εκτέλεσης και για τις περιοχές αντίστοιχα, που ορίζονται στο περιβάλλον κινούμενων πρακτόρων. Τα συστήματα πρακτόρων, κατά την έναρξη της λειτουργίας τους χρειάζονται να 'βρουν' το τοπικό σύστημα ευρετηρίου για να ενταχθούν στην περιοχή. Τα συστήματα ευρετηρίου πρέπει να είναι εγγεγραμμένα σε μια κοινή υπηρεσία για να εντοπίζονται μεταξύ τους, αλλά και από τα νέα συστήματα πρακτόρων. Έτσι, η μόνη πληροφορία που είναι απαραίτητη κατά την εκκίνηση οποιασδήποτε οντότητας (συστήματα πρακτόρων και ευρετηρίου) είναι η τοποθεσία της υπηρεσίας ονοματολογίας.

Η υπηρεσία ονοματολογίας που χρησιμοποιείται από το σύστημα EasyAgent παρέχεται από το περιβάλλον προγραμματισμού τύπου JDK1.2 μια και από το ίδιο περιβάλλον προέρχεται και ο εξυπηρετητής για ανεύρεση αντικειμένων (JDK1.2 ORB). Σε ορισμένα περιβάλλοντα η υπηρεσία ονοματολογίας εκκινείτε αυτόματα με την πρώτη εγγραφή ενός αντικειμένου σε αυτήν. Στο συγκεκριμένο περιβάλλον, ο χρήστης είναι υπεύθυνος να ξεκινήσει την λειτουργία της υπηρεσίας, πριν κάποιο αντικείμενο ζητήσει να εγγραφεί σε αυτήν. Οι παράμετροι για την έναρξη της υπηρεσίας, καθώς και αναλυτικότερες πληροφορίες μπορούν να βρεθούν στα εγχειρίδια χρήσης που παρέχονται μαζί με το περιβάλλον JDK1.2.

5.1.2 Σύστημα πρακτόρων

Το σύστημα πρακτόρων, που παρέχει το περιβάλλον EasyAgent, είναι υλοποιημένο όπως το πρότυπο OMG - MASIF ορίζει. Είναι υλοποιημένο ως αντικείμενο τύπου CORBA με διεπαφή επικοινωνίας που ορίζει το πρότυπο. Παρέχει στα περιβάλλοντα εκτέλεσης που φιλοξενεί την δυνατότητα πρόσβασης μέσω προκαθορισμένων μεθόδων. Εκτελείται στα πλαίσια ενός εξυπηρετητή για ανεύρεση αντικειμένων (ORB). Ο εξυπηρετητής εκκινείτε αυτόματα με το σύστημα πρακτόρων και αναλαμβάνει την εξυπηρέτηση των αναγκών επικοινωνίας με οποιαδήποτε άλλο σύστημα πρακτόρων και ευρετηρίου.

Το σύστημα πρακτόρων, κατά την έναρξη της λειτουργίας του, είναι απαραίτητο να γνωρίζει τον τρόπο με τον οποίο θα επικοινωνήσει με την υπηρεσία ονοματολογίας. Από την υπηρεσία αυτή θα συλλέξει όλες τις βασικές πληροφορίες (π.χ. τοποθεσία συγκεκριμένου συστήματος ευρετηρίου) για να συνεχίσει την εκτέλεσή του. Η τοποθεσία της υπηρεσίας επικοινωνίας λοιπόν, δίνεται ως όρισμα κατά την εκκίνηση του συστήματος πρακτόρων. Στην συνέχεια εγγράφεται στο σύστημα ευρετηρίου και έτσι γίνεται ενεργό μέλος μιας περιοχής. Στον παρακάτω πίνακα παραθέτονται όλοι οι παράμετροι για την ενεργοποίηση του συστήματος πρακτόρων από την γραμμή εντολών.

Παράμετρος	Περιγραφή
-r	(recover) Το σύστημα πρακτόρων αρχίζει την εκτέλεση προσπαθώντας να επαναφέρει την τελευταία κατάσταση, όπως αυτή αποτυπόθηκε στο μίνομα αποθηκευτικό μέσο.
-host <name>	Το όνομα του ξενιστή που παρέχει την υπηρεσία ονοματολογίας. Αν δεν παρέχεται θεωρείται ότι η υπηρεσία βρίσκεται στο τοπικό μηχάνημα.
-p <port number>	Ο αριθμός της πόρτας στο μηχάνημα του ξενιστή, που παρέχει την υπηρεσία ονοματολογίας, για την εγκατάσταση καναλιού επικοινωνίας. Ο προκαθορισμένος αριθμός είναι 20004.
-f <path>	Το μονοπάτι προς το αρχείο που περιέχει απαραίτητες πληροφορίες για τη λειτουργία του συστήματος. Το αρχείο ονομάζεται **__ASI.cnf**. Όταν δεν παρέχεται, το αρχείο αναζητάτε

	στον κατάλογο που έχει εγκατασταθεί το περιβάλλον EasyAgent. Αυτό βρίσκεται από την μεταβλητή $EA_HOME.
-c <path>	Το μοναπάτι προς τον κατάλογο που περιέχει τις κλάσεις υλοποίησης των πρακτόρων.
-h	(help) Τυπώνει την επεξήγηση των αποδεκτών παραμέτρων.

Πίνακας 5-1: Παράμετροι εκκίνησης του συστήματος πρακτόρων από τη γραμμή εντολών.

5.1.3 Σύστημα ευρετηρίου

Το σύστημα ευρετηρίου, που παρέχει το περιβάλλον EasyAgent, είναι και αυτό υλοποιημένο κατά τους ορισμούς του προτύπου. Αποτελεί το σημείο πρόσβασης για την περιοχή που θεωρητικά ορίζεται από τα συστήματα πρακτόρων που είναι εγγεγραμμένα στο εν λόγω σύστημα ευρετηρίου. Είναι υλοποιημένο ως αντικείμενο CORBA και εκτελείται στα πλαίσια ενός εξυπηρετητή τύπου ORB που εκκινείτε αυτόματα με το σύστημα ευρετηρίου. Κατά την έναρξη της λειτουργίας του επικοινωνεί με την υπηρεσία ονοματολογίας για να γνωστοποιήσει την ύπαρξή του. Από την στιγμή εκείνη και μετά οποιαδήποτε οντότητα ζητάει πληροφορίες καταλόγου από την υπηρεσία ονοματολογίας πληροφορείται και για το εν λόγω σύστημα.

Άλλες παράμετροι που μπορούν να χρησιμοποιηθούν για την εκκίνηση του συστήματος ευρετηρίου παραθέτονται στον παρακάτω πίνακα.

Παράμετρος	Περιγραφή
-host <name>	Το όνομα του ξενιστή που παρέχει την υπηρεσία ονοματολογίας. Αν δεν παρέχεται θεωρείται ότι η υπηρεσία βρίσκεται στο τοπικό μηχάνημα.
-p <port number>	Ο αριθμός της πόρτας στο μηχάνημα του ξενιστή, που παρέχει την υπηρεσία ονοματολογίας, για την εγκατάσταση καναλιού επικοινωνίας. Ο προκαθορισμένος αριθμός είναι 20004.

Πίνακας 5-2: Παράμετροι εκκίνησης του συστήματος ευρετηρίου από την γραμμή εντολών.

5.2 Διεπαφές χρήσης (User Interfaces)

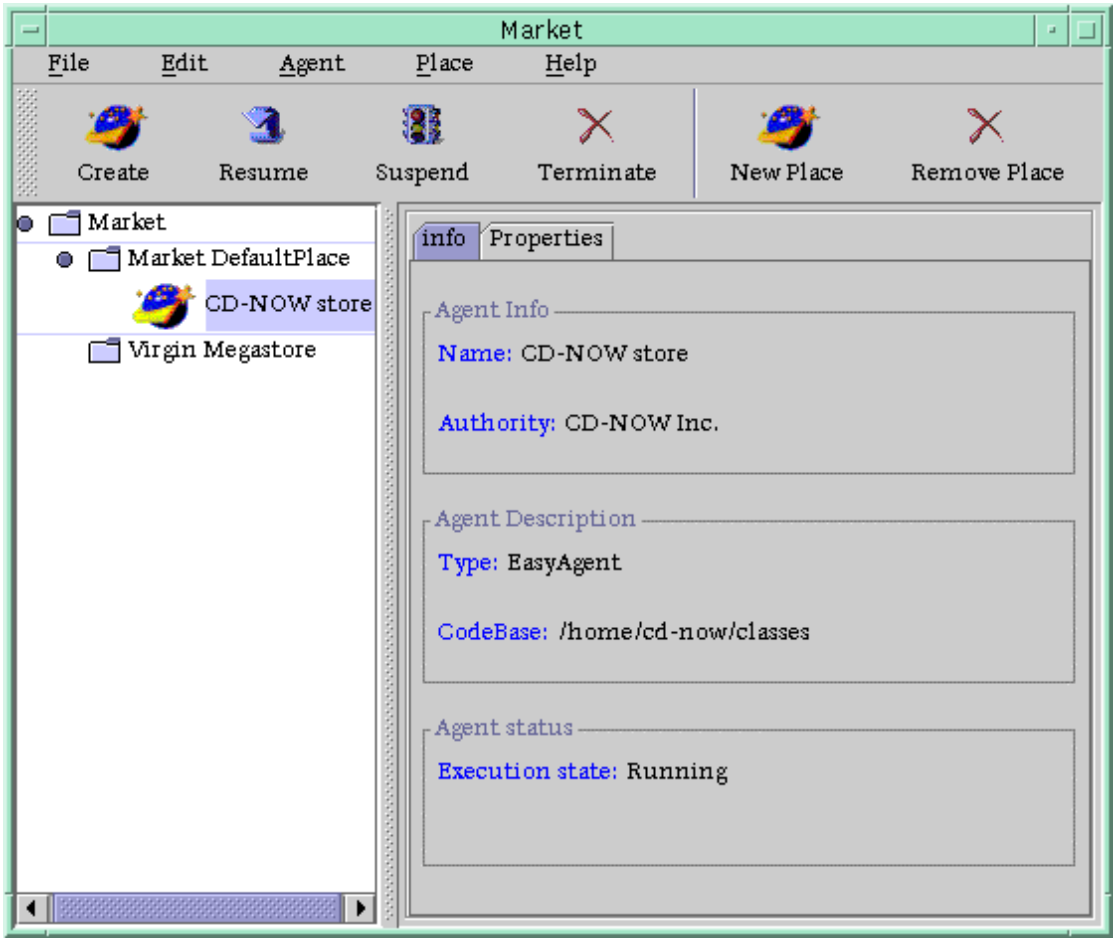
Στο προηγούμενο κεφάλαιο αναφέραμε ότι ο διαχειριστής του συστήματος πρακτόρων μπορεί να ξεκινήσει ή και να τερματίσει την εκτέλεση ενός πράκτορα. Αναφέραμε ακόμη ότι μπορεί να παρακολουθεί την κατάσταση τόσο του συστήματος πρακτόρων όσο και του συστήματος ευρετηρίου. Οι δυνατότητες αυτές παρέχονται στους χρήστες μέσω των διαπεφών χρήσης που έχουν αναπτυχθεί για το σύστημα EasyAgent.

Οι κλάσεις που υλοποιούν τις διεπαφές εσωκλείονται στο πακέτο EasyAgent.Gui, μαζί με άλλες βοηθητικές κλάσεις για την εξυπηρετικότερη παρουσίαση των πληροφοριών. Η υλοποίηση των διεπαφών έγινε με χρήση της βιβλιοθήκης Swing που εμπεριέχεται στο περιβάλλον προγραμματισμού τύπου JDK1.2. Οι βασικές κλάσεις του πακέτου EasyAgent.Gui που παρέχουν τις διεπαφές του συστήματος πρακτόρων και ευρετηρίου είναι οι AgentSystemUI και FinderUI αντίστοιχα.

Μία παρατήρηση που είναι απαραίτητη κατά την ανάγνωση των παρακάτω υποενοτήτων είναι ότι, οι ακολουθίες λέξεων που παρουσιάζονται μέσα σε πλαίσιο περιγράφουν το μονοπάτι που πρέπει να ακολουθήσει ο χρήστης με σκοπό την ενεργοποίηση μιας λειτουργίας. Το όνομα της λειτουργίας εμφανίζεται στο τέλος της ακολουθίας. Οι λέξεις που προηγούνται εκφράζουν την σειρά των ενεργειών πάνω στην διεπαφή χρήσης.

5.2.1 Διεπαφή χρήσης του συστήματος πρακτόρων.

Η διεπαφή χρήσης του συστήματος πρακτόρων είναι το οπτικό αποτέλεσμα της έναρξης εκτέλεσης του συστήματος πρακτόρων από την γραμμή εντολών (βλέπε Εικόνα 5-1). Χρησιμοποιώντας την διεπαφή ο χρήστης μπορεί να εκμεταλλευθεί όλο το σύνολο των λειτουργιών που παρέχει το σύστημα πρακτόρων. Δηλαδή, μπορεί να διαχειρίζεται τους τοπικούς πράκτορες και τα περιβάλλοντα εκτέλεσης και το ίδιο το σύστημα πρακτόρων. Το σύνολο των λειτουργιών που παρέχεται στον χρήστη περιγράφεται εξαντλητικά σε μία 'εκ των έσω' παρουσίαση της λειτουργικότητας στην ενότητα που αναλύονται οι κλάσεις που υλοποιούν το σύστημα πρακτόρων.



Εικόνα 5-1: Η διεπαφή χρήσης του συστήματος πρακτόρων.

Η διεπαφή χρήσης χωρίζεται σε δύο τμήματα (frames) που παρέχουν στον χρήστη πληροφορίες για της οντότητες που φιλοξενεί το σύστημα πρακτόρων. Το αριστερό τμήμα αναπαριστά σε δενδρική μορφή τα περιβάλλοντα εκτέλεσης και τους κινούμενους πράκτορες που δραστηριοποιούνται στο εν λόγω σύστημα. Στην κορυφή του δένδρου βρίσκεται το σύστημα πρακτόρων. Το δεξί τμήμα παρέχει τις πληροφορίες που σχετίζονται με την επιλεγμένη οντότητα του αριστερού τμήματος. Για παράδειγμα, στην Εικόνα 5-1 βλέπουμε στο αριστερό τμήμα πως το σύστημα πρακτόρων με όνομα *Market* έχει δύο περιβάλλοντα εκτέλεσης. Το προκαθορισμένο περιβάλλον με όνομα *Market DefaultPlace* και το περιβάλλον με όνομα *Virgin Megastore*. Στο πρώτο υπάρχει ο πράκτορας με όνομα *CD-NOW store*. Στο δεξί τμήμα της διεπαφής παρουσιάζονται λεπτομέρειες για την οντότητα που είναι επιλεγμένη στο αριστερό τμήμα, δηλαδή τον πράκτορα.

Όλες οι λειτουργίες που συνοδεύουν την διεπαφή χρήσης μπορούν να επιλεχθούν από τη μπάρα επιλογών (menu bar) στο πάνω μέρος της διεπαφής χρήσης. Οι περισσότερο διαδεδομένες λειτουργίες μπορούν να ενεργοποιηθούν και από μία λίστα από εικονίδια, αντιπροσώπους κάποιας λειτουργίας. Η λίστα βρίσκεται κάτω από τη μπάρα επιλογών. Όταν ο δείκτης του ποντικιού σταματήσει πάνω από κάποιο συγκεκριμένο τμήμα της διεπαφής (τμήμα του δένδρου, εικονίδιο) ένα παραθυράκι με μία λιτή περιγραφή εμφανίζεται στην οθόνη στη θέση του δείκτη.

Οι παρακάτω υποενότητες εξηγούν όλες τις λειτουργίες που προσφέρει η διεπαφή του συστήματος πρακτόρων με λεπτομέρεια. Η παρουσίαση των λειτουργιών θα χωριστεί σε δύο τμήματα. Το ένα θα σχετίζεται με λειτουργίες που αφορούν την διαχείριση του συστήματος πρακτόρων και το άλλο αυτές που σχετίζονται με την διαχείριση των κινούμενων πρακτόρων.

5.2.1.1 Λειτουργίες διαχείρισης του συστήματος πρακτόρων.

Οι λειτουργίες που ανήκουν στο σύνολο αυτό, σχετίζονται με τη συντήρηση του καταλόγου των υλοποιημένων πρακτόρων, την διαμόρφωση των χαρακτηριστικών του συστήματος, την παρακολούθηση των μηνυμάτων και τον τερματισμό του συστήματος.

5.2.1.1.1 Κατάλογος υλοποιημένων πρακτόρων

File / Bookmarks

Στην Εικόνα 5-4 παρουσιάζεται το παράθυρο που εμφανίζεται όταν ο χρήστης επιλέξει να δει τον κατάλογο από τους υλοποιημένους πράκτορες που υπάρχουν στο σύστημα πρακτόρων. Οι πράκτορες αυτοί μπορούν να επιλεχθούν και να ζητηθεί η έναρξη της εκτέλεσής τους.

5.2.1.1.2 Διαμόρφωση χαρακτηριστικών συστήματος

Edit / Preferences

5.2.1.1.3 Τερματισμός συστήματος πρακτόρων

File / Exit

Η επιλογή Exit τερματίζει την εκτέλεση του συστήματος πρακτόρων. Το σύστημα ζητάει επιβεβαίωση από το χρήστη για τον τερματισμό της εκτέλεσης. Επιπλέον, του δίνει τη δυνατότητα να σώσει την κατάσταση του συστήματος (πράκτορες και περιβάλλοντα εκτέλεσης) πριν τερματίσει τη λειτουργία του. Έτσι, είναι δυνατή η επανέναρξη της εκτέλεσης, χωρίς απώλειες σημαντικών πληροφοριών. Το μενού επιλογών που εμφανίζεται κατά τη λειτουργία του τερματισμού φαίνεται στην below εικόνα.

Εικόνα 5-2: Η διαλογική διεπαφή χρήσης για τον τερματισμό του συστήματος πρακτόρων.

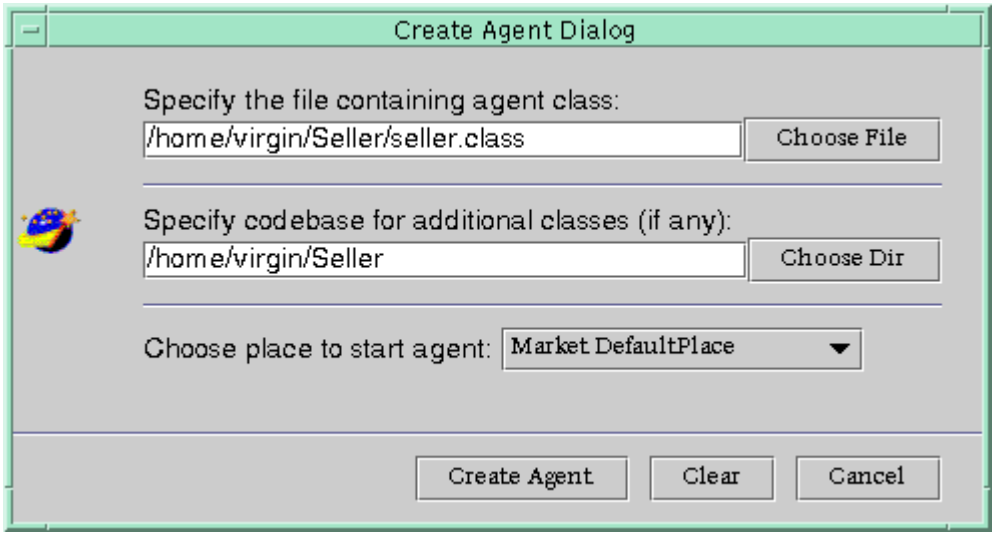
5.2.1.2 Λειτουργίες διαχείρισης των κινούμενων πρακτόρων.

Στο σύνολο αυτό ανήκουν οι λειτουργίες της δημιουργία, τερματισμού, αναστολής και επανέναρξης εκτέλεσης, και αποθήκευσης του πράκτορα.

5.2.1.2.1 Δημιουργία

Η διεπαφή παρέχει δύο τρόπους για τη δημιουργία πρακτόρων:

- μέσω του καταλόγου υλοποιημένων πρακτόρων
- μέσω της επιλογής δημιουργίας πράκτορα, που παρέχεται από τη μπάρα επιλογών (ή από το αντίστοιχο εικονίδιο)



Εικόνα 5-3: Η διαλογική διεπαφή χρήσης για τη δημιουργία πράκτορα.

Δημιουργία μέσω της επιλογής

Edit / Create

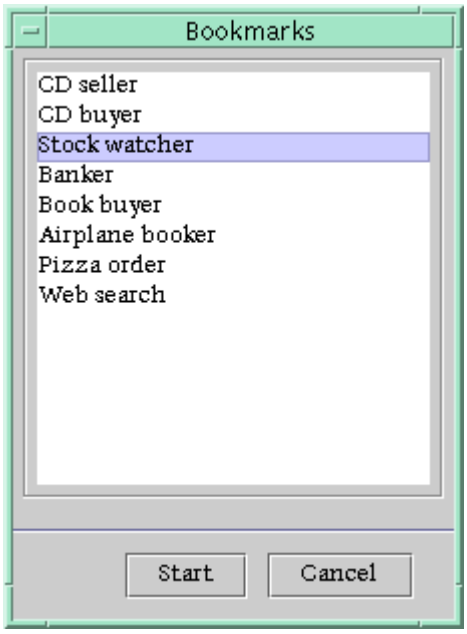
Στην Εικόνα 5-3 φαίνεται το παράθυρο που εμφανίζεται στην οθόνη όταν ο χρήστης επιλέξει τη λειτουργία της δημιουργίας πράκτορα.

Ζητείται από το χρήστη να καθορίσει την κλάση του πράκτορα, το μονοπάτι του καταλόγου που περιέχει τις απαραίτητες κλάσεις για την εκτέλεση, και το περιβάλλον εκτέλεσης που θα ξεκινήσει την εκτέλεσή του. Η κλάση του πράκτορα και το μονοπάτι μπορούν να καθοριστούν μέσω του πεδίου για εισαγωγή κειμένου (text field) αλλά και μέσω διεπαφής επιλογής αρχείου (file dialog). Η διεπαφή εμφανίζεται στην οθόνη πατώντας το κουμπί που βρίσκεται στη δεξιά πλευρά των πεδίων για εισαγωγή κειμένου. Το περιβάλλον εκτέλεσης επιλέγεται μέσω μίας λίστας με τα υπάρχοντα, τη στιγμή εκείνη, στο σύστημα περιβάλλοντα. Αφού επιλεχθεί η λειτουργία της δημιουργίας (*Create* button) θα εκκινηθεί η διαδικασία της δημιουργίας πράκτορα. Σε οποιαδήποτε περίπτωση αποτυχίας εμφανίζεται παράθυρο που ειδοποιεί το χρήστη ενημερώνοντάς τον ταυτόχρονα με μία περιγραφή του σφάλματος που συνέβη. Στην αντίθετη περίπτωση ο πράκτορας θα εμφανιστεί στο κύριο παράθυρο του συστήματος πρακτόρων.

Δημιουργία μέσω καταλόγου

File / Bookmarks

Στην Εικόνα 5-4 φαίνεται το παράθυρο που εμφανίζεται στην οθόνη όταν ο χρήστης επιλέξει τη δημιουργία πράκτορα μέσω καταλόγου.



Εικόνα 5-4: Διεπαφή χρήσης για την επιλογή εκτέλεσης ενός πράκτορα.

Ο χρήστης μπορεί να επιλέξει τον πράκτορα από την λίστα των υλοποιημένων πρακτόρων που εμφανίζεται στην οθόνη μαζί με το παράθυρο του καταλόγου. Επιλέγει το περιβάλλον εκτέλεσης και ξεκινάει την εκτέλεση του πράκτορα επιλέγοντας την έναρξη της εκτέλεσης (*Start* button).

5.2.1.2.2 Τερματισμός

Η διεπαφή παρέχει δύο τρόπους για τον τερματισμό πρακτόρων:

- επιλέγοντας τον πράκτορα από τη δενδρική αναπαράσταση των οντοτήτων του συστήματος πρακτόρων
- μέσω της επιλογής τερματισμού πράκτορα, που παρέχεται από τη μπάρα επιλογών (ή από το αντίστοιχο εικονίδιο)

Τερματισμός μέσω της επιλογής

Agent / Terminate

Στην Εικόνα 5-5 φαίνεται το παράθυρο που εμφανίζεται στην οθόνη όταν ο χρήστης επιλέξει τη λειτουργία του τερματισμού της εκτέλεσης του πράκτορα.

Εικόνα 5-5: Η διαλογική διεπαφή χρήσης για τον τερματισμό εκτέλεσης ενός πράκτορα.

ζητείται από το χρήστη να επιλέξει τον πράκτορα ακολουθώντας τα εξής βήματα πάνω στη διεπαφή επιλογής πράκτορα. Πρώτα πρέπει να επιλέξει το περιβάλλον εκτέλεσης από τη λίστα με τα υπάρχοντα στο σύστημα περιβάλλοντα. Στη συνέχεια επιλέγει τον πράκτορα από την λίστα των πρακτόρων που απαρτίζεται από αυτούς που δραστηριοποιούνται στο επιλεγμένο περιβάλλον. Το σύστημα πρακτόρων θα ζητήσει επιβεβαίωση προκειμένου να τερματίσει την εκτέλεση του πράκτορα. Σε περίπτωση αποτυχίας θα ενημερώσει το χρήστη με το ανάλογο μήνυμα. Στην αντίθετη περίπτωση, ο πράκτορας εξαφανίζεται από τη δενδρική αναπαράσταση του συστήματος.

Τερματισμός μέσω της δενδρικής αναπαράστασης
Ο χρήστης μπορεί να επιλέξει τον πράκτορα από το αριστερό τμήμα της διεπαφής του συστήματος πρακτόρων. Στη συνέχεια επιλέγοντας το εικονίδιο που αντιπροσωπεύει την λειτουργία του τερματισμού αρχικοποιεί την διαδικασία. Τα μηνύματα λάθους και η ενημέρωση της διεπαφής ισχύουν και στην περίπτωση αυτή.

5.2.1.2.3 Αναστολή εκτέλεσης

Η διεπαφή παρέχει δύο τρόπους για την αναστολή εκτέλεσης των πρακτόρων:

- επιλέγοντας τον πράκτορα από τη δενδρική αναπαράσταση των οντοτήτων του συστήματος πρακτόρων
- μέσω της επιλογής αναστολής εκτέλεσης πράκτορα, που παρέχεται από τη μπάρα επιλογών (ή από το αντίστοιχο εικονίδιο)

Αναστολή μέσω της επιλογής

Agent / Suspend

Στην Εικόνα 5-6 φαίνεται το παράθυρο που εμφανίζεται στην οθόνη όταν ο χρήστης επιλέξει τη λειτουργία της αναστολής εκτέλεσης του πράκτορα.

Εικόνα 5-6: Η διεπαφή χρήσης για την αναστολή εκτέλεσης ενός πράκτορα.

ζητείται από το χρήστη να επιλέξει τον πράκτορα ακολουθώντας τα εξής βήματα πάνω στη διεπαφή επιλογής πράκτορα. Πρώτα πρέπει να επιλέξει το περιβάλλον εκτέλεσης από τη λίστα με τα υπάρχοντα στο σύστημα περιβάλλοντα. Από το σύνολο των περιβαλλόντων παρουσιάζονται αυτά που φιλοξενούν εκτελούμενους, εκείνη τη χρονική στιγμή, πράκτορες. Στη συνέχεια επιλέγει τον πράκτορα από την λίστα των πρακτόρων που απαρτίζεται από αυτούς που δραστηριοποιούνται στο επιλεγμένο περιβάλλον. Το σύστημα πρακτόρων θα ζητήσει επιβεβαίωση προκειμένου να αναστείλει την εκτέλεση του πράκτορα. Σε περίπτωση αποτυχίας θα ενημερώσει το χρήστη με το ανάλογο μήνυμα. Στην αντίθετη περίπτωση, αλλάζει η κατάσταση του πράκτορα στη δενδρική αναπαράσταση του συστήματος.

Αναστολή μέσω της δενδρικής αναπαράστασης
Ο χρήστης μπορεί να επιλέξει τον πράκτορα από το αριστερό τμήμα της διεπαφής του συστήματος πρακτόρων. Δεν μπορούν να επιλεγεί οποιοσδήποτε πράκτορας. Ο πράκτορας πρέπει να εκτελείται τη στιγμή εκείνη. Στη συνέχεια επιλέγοντας το εικονίδιο που αντιπροσωπεύει την λειτουργία της αναστολής εκτέλεσης, αρχικοποιεί την διαδικασία. Τα μηνύματα λάθους και η ενημέρωση της διεπαφής ισχύουν και στην περίπτωση αυτή.

5.2.1.2.4 Επανέναρξη εκτέλεσης

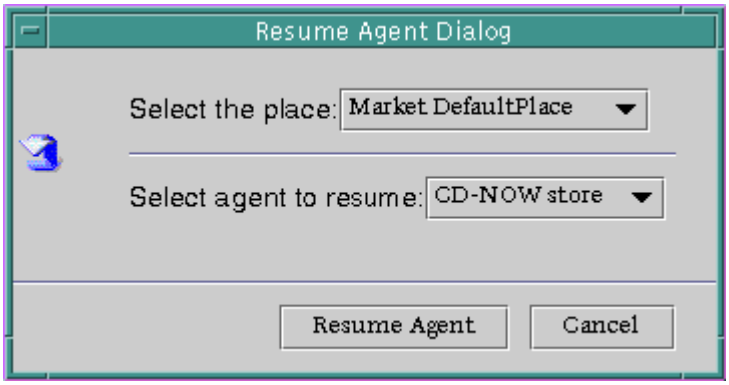
Η διεπαφή παρέχει δύο τρόπους για την επανέναρξη εκτέλεσης των πρακτόρων:

- επιλέγοντας τον πράκτορα από τη δενδρική αναπαράσταση των οντοτήτων του συστήματος πρακτόρων
- μέσω της επιλογής επανέναρξης εκτέλεσης πράκτορα, που παρέχεται από τη μπάρα επιλογών (ή από το αντίστοιχο εικονίδιο)

Επανέναρξη μέσω της επιλογής

Agent / Resume

Στην Εικόνα 5-7 φαίνεται το παράθυρο που εμφανίζεται στην οθόνη όταν ο χρήστης επιλέξει τη λειτουργία της επανέναρξης εκτέλεσης του πράκτορα.



Εικόνα 5-7: Η διεπαφή χρήσης για την επανέναρξη εκτέλεσης ενός πράκτορα.

ζητείται από το χρήστη να επιλέξει τον πράκτορα ακολουθώντας τα εξής βήματα πάνω στη διεπαφή επιλογής πράκτορα. Πρώτα πρέπει να επιλέξει το περιβάλλον εκτέλεσης από τη λίστα με τα υπάρχοντα στο σύστημα περιβάλλοντα. Από το σύνολο των περιβαλλόντων παρουσιάζονται αυτά που φιλοξενούν πράκτορες που έχει ανασταλεί η εκτέλεσή τους. Στη συνέχεια επιλέγει τον πράκτορα από την λίστα των πρακτόρων που απαρτίζεται από αυτούς που φιλοξενούνται στο επιλεγμένο περιβάλλον. Το σύστημα πρακτόρων θα ζητήσει επιβεβαίωση προκειμένου να επανεκκινήσει την εκτέλεση του πράκτορα. Σε περίπτωση αποτυχίας θα ενημερώσει το χρήστη με το ανάλογο μήνυμα. Στην αντίθετη περίπτωση, αλλάζει η κατάσταση του πράκτορα στη δενδρική αναπαράσταση του συστήματος.

Επανέναρξη μέσω της δενδρικής αναπαράστασης
Ο χρήστης μπορεί να επιλέξει τον πράκτορα από το αριστερό τμήμα της διεπαφής του συστήματος πρακτόρων. Δεν μπορεί να επιλεγεί οποιοσδήποτε πράκτορας. Ο πράκτορας πρέπει να είναι σε κατάσταση αναστολής εκτέλεσης. Στη συνέχεια επιλέγοντας το εικονίδιο που αντιπροσωπεύει την λειτουργία της επανέναρξης εκτέλεσης, αρχικοποιεί την διαδικασία. Τα μηνύματα λάθους και η ενημέρωση της διεπαφής ισχύουν και στην περίπτωση αυτή.

5.2.1.2.5 Κλωνοποίηση πράκτορα

Η διεπαφή παρέχει δύο τρόπους για την κλωνοποίηση των πρακτόρων:

- επιλέγοντας τον πράκτορα από τη δενδρική αναπαράσταση των οντοτήτων του συστήματος πρακτόρων
- μέσω της επιλογής κλωνοποίησης πράκτορα, που παρέχεται από τη μπάρα επιλογών (ή από το αντίστοιχο εικονίδιο)

Κλωνοποίηση μέσω της επιλογής

Agent / Clone

Στο παράθυρο που εμφανίζεται όταν επιλεχθεί η λειτουργία της κλωνοποίησης, ζητείται από το χρήστη να επιλέξει τον πράκτορα από τη λίστα των υπαρχόντων στο σύστημα πρακτόρων. Το σύστημα πρακτόρων θα ενημερώσει την δενδρική αναπαράσταση που διατηρεί όταν ο πράκτορας ολοκληρώσει την διαδικασία της κλωνοποίησης και εκκινήσει την εκτέλεσή του. Σε περίπτωση αποτυχίας θα ενημερωθεί ο χρήστης με το ανάλογο μήνυμα που παρουσιάζεται στην οθόνη.

Κλωνοποίηση μέσω της δενδρικής αναπαράστασης
Ο χρήστης μπορεί να επιλέξει τον πράκτορα από το αριστερό τμήμα της διεπαφής του συστήματος πρακτόρων. Στη συνέχεια επιλέγοντας το εικονίδιο που αντιπροσωπεύει την λειτουργία της κλωνοποίησης, αρχικοποιεί την διαδικασία. Τα μηνύματα λάθους και η ενημέρωση της διεπαφής ισχύουν και στην περίπτωση αυτή.

5.2.1.2.6 Αποθήκευση κατάστασης πράκτορα

Η αποθήκευση της κατάστασης των πρακτόρων είναι μια λειτουργία στενά συνδεδεμένη με την υπηρεσία σταθερότητας του συστήματος πρακτόρων. Ο χρήστης του συστήματος πρακτόρων όχι μόνο μπορεί να παραγγείλει στο σύστημα να αποθηκεύει την κατάστασή του μπορεί, επιπλέον, να ζητήσει την αποθήκευση ενός συγκεκριμένου πράκτορα.

- επιλέγοντας τον πράκτορα από τη δενδρική αναπαράσταση των οντοτήτων του συστήματος πρακτόρων
- μέσω της επιλογής αποθήκευσης πράκτορα, που παρέχεται από τη μπάρα επιλογών (ή από το αντίστοιχο εικονίδιο)

Ο πράκτορας μπορεί είτε να εκτελείται είτε να έχει ανασταλεί η εκτέλεσή του. Μετά το τέλος της διαδικασίας της αποθήκευσης είτε συνεχίζει την εκτέλεσή του εάν εκτελούταν, είτε παραμένει ανενεργός εάν ήταν σε κατάσταση αναστολής εκτέλεσης.

Αποθήκευση μέσω της επιλογής

Agent / Save

Στο παράθυρο που εμφανίζεται όταν επιλεχθεί η λειτουργία της αποθήκευσης, ζητείται από το χρήστη να επιλέξει τον πράκτορα από τη λίστα των υπαρχόντων στο σύστημα πρακτόρων. Σε περίπτωση αποτυχίας θα ενημερωθεί ο χρήστης με το ανάλογο μήνυμα που παρουσιάζεται στην οθόνη.

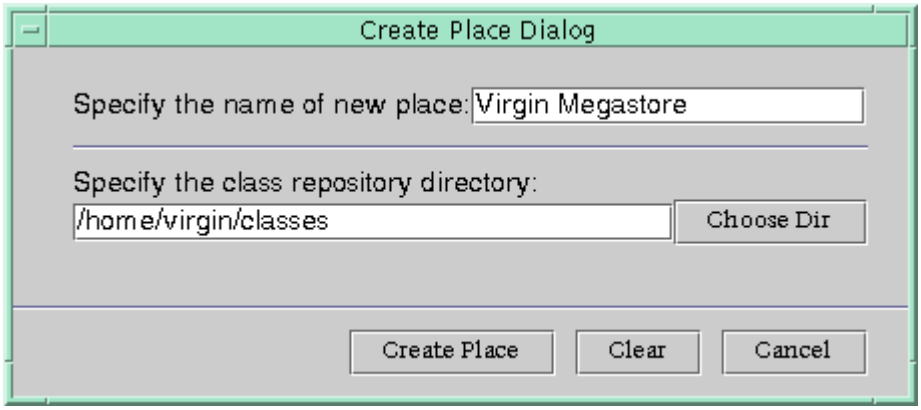
Αποθήκευση μέσω της δενδρικής αναπαράστασης
Ο χρήστης μπορεί να επιλέξει τον πράκτορα από το αριστερό τμήμα της διεπαφής του συστήματος πρακτόρων. Στη συνέχεια επιλέγοντας το εικονίδιο που αντιπροσωπεύει τη λειτουργία της αποθήκευσης, αρχικοποιεί την διαδικασία. Τα μηνύματα λάθους και η ενημέρωση της διεπαφής ισχύουν και στην περίπτωση αυτή.

5.2.1.3 Λειτουργίες διαχείρισης των περιβαλλόντων εκτέλεσης.

Στο σύστημα πρακτόρων υπάρχουν ένα ή περισσότερα περιβάλλοντα εκτέλεσης. Η διαχείριση των περιβαλλόντων γίνεται μέσω της διεπαφής. Το σύνολο των λειτουργιών που προσφέρονται απαρτίζεται από τις: δημιουργία, τερματισμό, αναστολή και επανέναρξη λειτουργίας ενός περιβάλλοντος εκτέλεσης.

5.2.1.3.1 Δημιουργία

Στην Εικόνα 5-8 φαίνεται το παράθυρο που εμφανίζει η διεπαφή χρήσης του συστήματος όταν ο χρήστης επιλέξει την λειτουργία της δημιουργίας ενός περιβάλλοντος εκτέλεσης.



Εικόνα 5-8: Η διαλογική διεπαφή χρήσης για την δημιουργία ενός περιβάλλοντος εκτέλεσης.

Ο χρήστης πρέπει να εισάγει το όνομα του περιβάλλοντος, μία λιτή περιγραφή του και τον κατάλογο που θα χρησιμοποιεί για την αποθήκευση των κλάσεων. Δεν επιτρέπετε η δημιουργία δύο περιβαλλόντων εκτέλεσης με το ίδιο όνομα σε ένα σύστημα πρακτόρων. Η διεπαφή χρήσης ειδοποιεί το χρήστη για οποιοδήποτε σφάλμα συμβεί κατά την δημιουργία του νέου περιβάλλοντος. Εάν η δημιουργία είναι επιτυχής, ενημερώνεται η δενδρική δομή στο αριστερό τμήμα της διεπαφής του συστήματος.

5.2.1.3.2 Τερματισμός

Ο τερματισμός της λειτουργίας ενός περιβάλλοντος εκτέλεσης συνεπάγεται αυτόματα και τον αιφνιδιαστικό τερματισμό της εκτέλεσης των πρακτόρων που φιλοξενούνται στο περιβάλλον αυτό. Στο προκαθορισμένο από το σύστημα πρακτόρων EasyAgent περιβάλλον εκτέλεσης, που φέρει το όνομα DefaultPlace, δεν μπορεί να τερματιστεί η λειτουργία. Η διεπαφή προσφέρει δύο τρόπους για τον τερματισμό της λειτουργίας ενός περιβάλλοντος εκτέλεσης.

- Επιλέγοντας τον πράκτορα από τη δενδρική αναπαράσταση των οντοτήτων του συστήματος πρακτόρων
- Μέσω της επιλογής τερματισμού περιβάλλοντος εκτέλεσης, που παρέχεται από τη μπάρα επιλογών

Τερματισμός μέσω της επιλογής

Place / Terminate

Στην Εικόνα 5-9 φαίνεται το παράθυρο που εμφανίζεται στην οθόνη όταν ο χρήστης επιλέξει τη λειτουργία της διακοπής εκτέλεσης ενός περιβάλλοντος.

Εικόνα 5-9: Η διαλογική διεπαφή για τον τερματισμό ενός περιβάλλοντος εκτέλεσης.

ζητείται από το χρήστη να επιλέξει το περιβάλλον εκτέλεσης από την λίστα των υπάρχοντων στο σύστημα περιβαλλόντων και να αρχικοποιήσει την διαδικασία. Σε περίπτωση αποτυχίας θα ενημερωθεί ο χρήστης με το ανάλογο μήνυμα που παρουσιάζεται στην οθόνη. Αλλιώς, ενημερώνεται το αριστερό τμήμα της διεπαφής του συστήματος πρακτόρων.

Τερματισμός μέσω της δενδρικής αναπαράστασης
Ο χρήστης μπορεί να επιλέξει περιβάλλον εκτέλεσης από το αριστερό τμήμα της διεπαφής του συστήματος πρακτόρων. Στη συνέχεια επιλέγοντας το εικονίδιο που αντιπροσωπεύει τη λειτουργία του τερματισμού, αρχικοποιεί την διαδικασία. Τα μηνύματα λάθους και η ενημέρωση της διεπαφής ισχύουν και στην περίπτωση αυτή.

5.2.1.3.3 Αναστολή λειτουργίας

Η αναστολή λειτουργίας ενός περιβάλλοντος εκτέλεσης συνεπάγεται αυτόματα και την αναστολή της εκτέλεσης των πρακτόρων που φιλοξενούνται στο περιβάλλον αυτό. Μόνο στα ενεργά περιβάλλοντα εκτέλεσης μπορεί να ανασταλεί η λειτουργία. Η επανέναρξη γίνεται μόνο με την αντίστοιχη λειτουργία. Η διεπαφή προσφέρει δύο τρόπους για την αναστολή λειτουργίας ενός περιβάλλοντος εκτέλεσης.

- Επιλέγοντας τον πράκτορα από τη δενδρική αναπαράσταση των οντοτήτων του συστήματος πρακτόρων
- Μέσω της επιλογής αναστολής λειτουργίας του περιβάλλοντος εκτέλεσης, που παρέχεται από τη μπάρα επιλογών

Αναστολή μέσω της επιλογής

Place / Suspend

Στην Εικόνα 5-10 φαίνεται το παράθυρο που εμφανίζεται στην οθόνη όταν ο χρήστης επιλέξει τη λειτουργία της αναστολής εκτέλεσης ενός περιβάλλοντος.

Εικόνα 5-10: Η διαλογική διεπαφή χρήσης για την αναστολή λειτουργίας ενός περιβάλλοντος εκτέλεσης.

ζητείται από το χρήστη να επιλέξει το περιβάλλον εκτέλεσης από την λίστα των υπάρχοντων στο σύστημα περιβαλλόντων και να αρχικοποιήσει την διαδικασία. Σε περίπτωση αποτυχίας θα ενημερωθεί ο χρήστης με το ανάλογο μήνυμα που παρουσιάζεται στην οθόνη.

Αναστολή μέσω της δενδρικής αναπαράστασης
Ο χρήστης μπορεί να επιλέξει περιβάλλον εκτέλεσης από το αριστερό τμήμα της διεπαφής του συστήματος πρακτόρων. Στη συνέχεια επιλέγοντας το εικονίδιο που αντιπροσωπεύει τη λειτουργία της αναστολής εκτέλεσης, αρχικοποιεί την διαδικασία. Τα μηνύματα λάθους ισχύουν και στην περίπτωση αυτή.

5.2.1.3.4 Επανέναρξη λειτουργίας

Η επανέναρξη λειτουργίας ενός περιβάλλοντος εκτέλεσης συνεπάγεται αυτόματα και την επανέναρξη της εκτέλεσης των πρακτόρων που φιλοξενούνται στο περιβάλλον αυτό. Η διεπαφή προσφέρει δύο τρόπους για την επανέναρξη λειτουργίας ενός περιβάλλοντος εκτέλεσης.

- Επιλέγοντας τον πράκτορα από τη δενδρική αναπαράσταση των οντοτήτων του συστήματος πρακτόρων
- Μέσω της επιλογής επανέναρξης λειτουργίας του περιβάλλοντος εκτέλεσης, που παρέχεται από τη μπάρα επιλογών

Επανέναρξη μέσω της επιλογής

Place / Resume

Στην Εικόνα 5-11 φαίνεται το παράθυρο που εμφανίζεται στην οθόνη όταν ο χρήστης επιλέξει τη λειτουργία της επανέναρξης εκτέλεσης ενός περιβάλλοντος.

Εικόνα 5-11: Η διαλογική διεπαφή χρήσης για την επανέναρξη λειτουργίας ενός περιβάλλοντος εκτέλεσης.

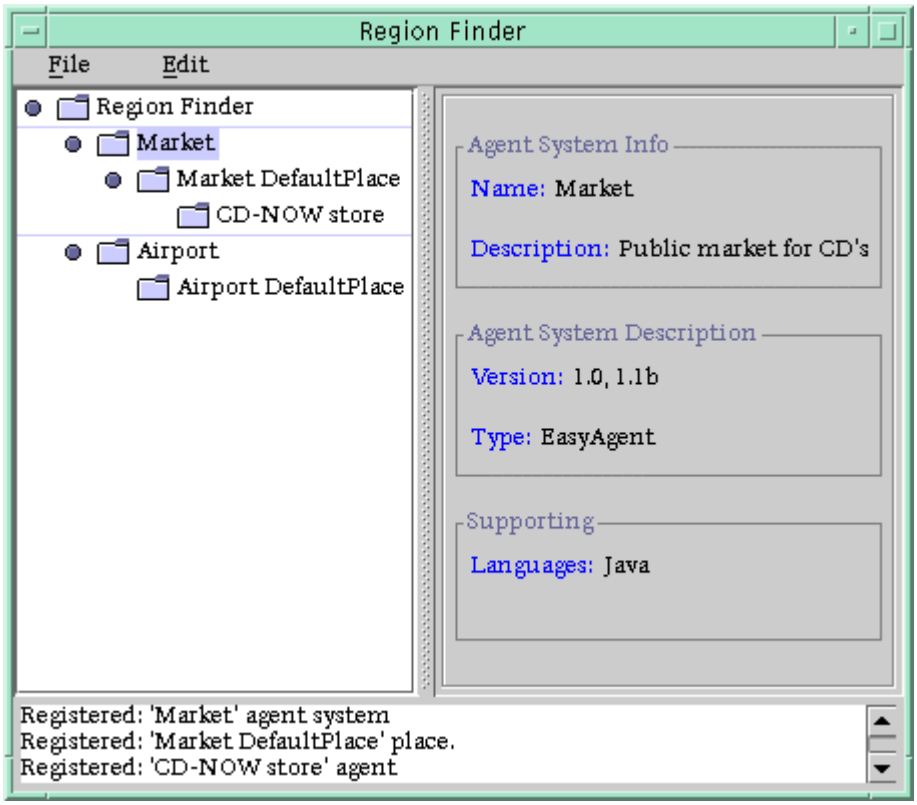
ζητείται από το χρήστη να επιλέξει το περιβάλλον εκτέλεσης από την λίστα των απενεργοποιημένων περιβαλλόντων εκτέλεσης που υπάρχουν στο σύστημα, και να αρχικοποιήσει την διαδικασία. Σε περίπτωση αποτυχίας θα ενημερωθεί ο χρήστης με το ανάλογο μήνυμα που παρουσιάζεται στην οθόνη.

Επανέναρξη μέσω της δενδρικής αναπαράστασης
Ο χρήστης μπορεί να επιλέξει περιβάλλον εκτέλεσης από το αριστερό τμήμα της διεπαφής του συστήματος πρακτόρων. Στη συνέχεια επιλέγοντας το εικονίδιο που αντιπροσωπεύει τη λειτουργία της επανέναρξη εκτέλεσης, αρχικοποιεί την διαδικασία. Τα μηνύματα λάθους ισχύουν και στην περίπτωση αυτή.

5.2.2 Διεπαφή χρήσης του συστήματος ευρετηρίου.

Μετά την έναρξη της εκτέλεσης του συστήματος ευρετηρίου από την γραμμή εντολών, η διεπαφή χρήσης του συστήματος εμφανίζεται στην οθόνη (βλέπε Εικόνα 5-12). Από την

οπτική γωνία του χρήστη η διεπαφή του ευρετηρίου αποτελεί το κύριο σημείο πρόσβασης σε αυτό. Παρέχει πρόσβαση στην πληροφορία που διατηρεί όσον αφορά τις οντότητες που ανήκουν στην περιοχή που αντιπροσωπεύει. Στο σύνολο των λειτουργιών του ανήκει και η πρόσβαση σε ιδιαίτερα χαρακτηριστικά του συστήματος καθώς και η αλλαγή αυτών.



Εικόνα 5-12: Η διεπαφή χρήσης του συστήματος ευρετηρίου.

Η διεπαφή χωρίζεται σε δύο τμήματα (frames) που τροφοδοτούν δυναμικά τον χρήστη με πληροφορίες που αφορούν τις οντότητες που είναι εγγεγραμμένες στο σύστημα. Παρουσιάζει δυναμικά πληροφορία που σχετίζεται με τα συστήματα πρακτόρων, τα περιβάλλοντα εκτέλεσης και τους πράκτορες που δραστηριοποιούνται στην περιοχή. Το αριστερό τμήμα αναπαριστά τα περιεχόμενα του ευρετηρίου σε δενδρική μορφή. Στην ρίζα του δένδρου είναι το εν λόγω σύστημα ευρετηρίου. Οι πρώτοι απόγονοι είναι τα συστήματα πρακτόρων που απαρτίζουν την περιοχή που αντιπροσωπεύει το ευρετήριο αυτό. Μετά έρχονται τα περιβάλλοντα εκτέλεσης και ως φύλλα οι πράκτορες που δρουν στην περιοχή αυτή. Στο δεξί τμήμα της διεπαφής χρήσης αναπαρίστανται οι πληροφορίες που διατηρεί το σύστημα για την επιλεγμένη στο αριστερό τμήμα οντότητα. Στην Εικόνα 5-1 φαίνεται η δομή της περιοχής τη συγκεκριμένη χρονική στιγμή. Υπάρχουν δύο συστήματα πρακτόρων, το Market και το Airport. Το καθένα από αυτά έχει το προκαθορισμένο περιβάλλον εκτέλεσης, όπως έχει αναλυθεί στην ενότητα 4.3.2.2. Στο 'Market DefaultPlace' φαίνεται ότι φιλοξενείται ο πράκτορας με όνομα 'CD-NOW store'. Στο δεξί τμήμα της διεπαφής παρουσιάζονται οι πληροφορίες που σχετίζονται με το σύστημα πρακτόρων Market, μια και αυτό έχει επιλεχθεί από τη δενδρική αναπαράσταση.

Η λειτουργικότητα που παρέχει το σύστημα ευρετηρίου είναι προσβάσιμη από τη μπάρα επιλογών στο πάνω μέρος της διεπαφής του συστήματος. Όταν ο δείκτης του ποντικιού σταματάει πάνω από κάποιο συγκεκριμένο τμήμα της διεπαφής χρήσης (κουμπί, εικονίδιο, περιεχόμενο του δένδρου) ένα παραθυράκι με μία λιτή περιγραφή εμφανίζεται στην οθόνη στη θέση του δείκτη.

6 Η διεπαφή προγραμματισμού πρακτόρων (The EasyAgent API)

Στο τέταρτο κεφάλαιο αναλύσαμε την υλοποίηση του συστήματος πρακτόρων που αναπτύχθηκε στα πλαίσια της εργασίας αυτής. Το σύστημα πρακτόρων EasyAgent προσφέρει και ένα περιβάλλον για την ανάπτυξη εφαρμογών κινούμενων πρακτόρων. Οι διεπαφές προγραμματισμού που παρέχονται στους κατασκευαστές των εφαρμογών, αφορούν τόσο την υλοποίηση των κινούμενων πρακτόρων όσο και την απόδοση συγκεκριμένων χαρακτηριστικών στο σύστημα πρακτόρων και στο σύστημα ευρετηρίου.

Οι κλάσεις που παρέχονται στον προγραμματιστή, ανήκουν στο πακέτο EasyAgent.Agent.

Η ανάλυση αρχίζει με την κλάση του πράκτορα. Περιγράφονται οι μέθοδοι που παρέχονται καθώς και τεχνικές υλοποίησης της της μεθόδου run(). Τα λάθη στην εκτέλεση σημειώνονται από τις κλάσεις ενδείξεως σφάλματος που περιγράφονται στο σημείο αυτό.

6.1 Η κλάση του πράκτορα (class Agent)

Η κλάση του πράκτορα αποτελεί την πατρική κλάση όλων των πρακτόρων που υλοποιούνται με χρήση του περιβάλλοντος προγραμματισμού που προσφέρει το σύστημα EasyAgent. Οι μέθοδοι που παρέχει η κλάση του πράκτορα μπορεί να κατηγοριοποιηθούν ως:

1. μέθοδοι, των οποίων η υλοποίηση μπορούν να επανοριστεί (overridden), από τις κλάσεις που παρέχει ο προγραμματιστής
2. μέθοδοι, των οποίων η υλοποίηση δε μπορεί να επανοριστεί. Οι μέθοδοι αυτοί, στην γλώσσα προγραμματισμού Java, ορίζονται με το όνομα τελική μέθοδος (final method).

6.1.1 Τελικές μέθοδοι (final methods)

Στην κατηγορία αυτή ανήκουν οι μέθοδοι που σχετίζονται άμεσα με το στιγμιότυπο του πράκτορα από το οποίο εκτελούνται. Στο τέταρτο κεφάλαιο (βλέπε ενότητα ‘4.3.4. Πράκτορας’) αναφέραμε πως ο πράκτορας αποτελείται από δύο τμήματα, τον πυρήνα, που είναι κοινός για όλους τους πράκτορες, και το ιδιαίτερο μέρος. Οι τελικές μέθοδοι υλοποιούν το τμήμα του πυρήνα και για το λόγο αυτό δεν επιτρέπεται στον προγραμματιστή να επανορίσει την υλοποίησή τους. Οι μέθοδοι παρουσιάζονται αναλυτικά παρακάτω.

move()
Ο πράκτορας χρησιμοποιεί τη μέθοδο αυτή με σκοπό τη μετακίνησή του σε ένα άλλο περιβάλλον εκτέλεσης. Το περιβάλλον εκτέλεσης του προορισμού μπορεί να είναι είτε στο ίδιο ή σε διαφορετικό σύστημα πρακτόρων αρκεί μόνο να υποστηρίζει την εκτέλεση πρακτόρων του ιδίου τύπου (με τον πράκτορα που καλεί την μέθοδο αυτή). Φτάνοντας στον προορισμό του ο πράκτορας επανέρχεται στην κατάσταση που είχε πριν εκκινηθεί η διαδικασία της μεταφοράς. Καλείται η μέθοδος run() του πράκτορα, που σημαίνει την έναρξη της εκτέλεσής του στο νέο περιβάλλον.

Μετά από κάθε κλήση της μεθόδου move() η επανεκκίνηση εκτέλεσης του πράκτορα στο περιβάλλον προορισμού γίνεται με κλήση της μεθόδου run(). Κατά την υλοποίηση του πράκτορα λοιπόν πρέπει να λαμβάνεται υπ’ όψιν το γεγονός, εάν ο προγραμματιστής δεν επιθυμεί την επανεκτέλεση των εντολών.

Σύνταξη
```
public final void move(Location location)
        throws EntryNotFoundException, MalformedLocationException,
        SerializationException, EAException, DeserializationException,
```

```
        TerminateException

public final void move(String place_name)
        throws EntryNotFoundException, MalformedLocationException,
        SerializationException, EAException, DeserializationException,
        TerminateException
```

Παράμετροι

location	Η τοποθεσία του περιβάλλοντος εκτέλεσης που επιθυμεί να μεταφερθεί ο πράκτορας. Ο προγραμματιστής δεν μπορεί να ορίσει μεταβλητές τέτοιου είδους. Επιστρέφονται μόνο από συναρτήσεις όπως getLocation().
place_name	Το όνομα του περιβάλλοντος εκτέλεσης στο οποίο επιθυμεί να μεταφερθεί ο πράκτορας. Εάν δεν υπάρχει μέρος με το όνομα αυτό δημιουργείται μία εξαίρεση τύπου EntryNotFound.

Υλοποίηση

Ο εξής αλγόριθμος χρησιμοποιείται για την υλοποίηση της μεθόδου:

1. Αναστολή εκτέλεσης του πράκτορα αναστέλλοντας την εκτέλεση της ίνας που έχει αναλάβει την εκτέλεσή του.
2. Εύρεση της τοποθεσίας του περιβάλλοντος προορισμού, αν αυτό είναι απαραίτητο.
3. Αποθήκευση της κατάστασης του πράκτορα (με σειριοποίηση των στιγμιοτύπων των κλάσεών του) και αποστολή στο σύστημα προορισμού της πληροφορίας αυτής, μαζί με δεδομένα απαραίτητα για την εγκατάσταση και επανεκκίνηση της εκτέλεσης στο νέο περιβάλλον.
4. Στο σύστημα προορισμού πλέον οι εξής διαδικασίες λαμβάνουν χώρα: γίνεται αποσειριοποίηση των κλάσεων και πιθανή μεταφορά ορισνών κλάσεων εφόσον δεν υπάρχουν στο σύστημα.
5. Δημιουργία μίας ίνας για να αναλάβει την εκτέλεση του πράκτορα.
6. Εάν είναι επιτυχής η έναρξη της εκτέλεσης, ενημέρωση του συστήματος ευρετηρίου για τη μετακίνηση του πράκτορα. Αποστολή μηνύματος στο αρχικό σύστημα για τον τερματισμό της εκτέλεσης του εκεί αντιγράφου του πράκτορα (το αντίγραφο αυτό είναι σε κατάσταση ανεσταλμένης εκτέλεσης). Σε αντίθετη περίπτωση επανεκκινείται η εκτέλεση του εκεί αντιγράφου.
7. Εάν, κατά τη διάρκεια δραστηριοποίησης στο νέο περιβάλλον, ο πράκτορας χρειαστεί κλάση(εις) που πρέπει να μεταφερθούν από το προηγούμενο μέρος εκτέλεσης, η εκτέλεση του πράκτορα αναστέλλεται και αμέσως εκκινείτε η διαδικασία της μεταφοράς της(των). Μετά την ολοκλήρωση της διαδικασίας συνεχίζεται η εκτέλεση του πράκτορα.

createAgent()

Η μέθοδος αυτή χρησιμοποιείται με σκοπό τη δημιουργία ενός νέου πράκτορα. Ο νέος πράκτορας κληρονομεί από τον πράκτορα που κάλεσε τη μέθοδο αυτή, μόνο τη δικαιοδοτική αρχή. Το περιβάλλον εκτέλεσης που θα ξεκινήσει η εκτέλεση του πράκτορα μπορεί να είναι είτε στο τοπικό, είτε σε απομακρυσμένο σύστημα πρακτόρων. Το γεγονός που έχει ιδιαίτερη σημασία αφορά την κλάση υλοποίησης του νέου πράκτορα. Αυτή πρέπει οπωσδήποτε να είναι γνωστή στο περιβάλλον που εκτελείται ο πράκτορας που κάλεσε την μέθοδο. Έτσι, είτε θα χρησιμοποιηθεί από το τοπικό σύστημα αν ο νέος πράκτορας εκτελεστεί τοπικά, είτε θα αποσταλεί στο σύστημα που ξεκίνησε η διαδικασία της εκτέλεση του πράκτορα μετά από αίτηση του τελευταίου.

Η μέθοδος επιστρέφει το όνομα του νέου πράκτορα. Το όνομά του είναι ενσωματωμένο με τον κώδικά του, οπότε γνωστοποιείται μόνο κατά τη διάρκεια της δημιουργίας του.

Σύνταξη

```
public final String createAgent(String class_name, String place_name, String[] args)
        throws MalformedLocationException, EntryNotFoundException,
        ClassUnknownException, HaMaException;

public final String createAgent(String class_name, String class_base,
        String place_name, String[] args)
        throws MalformedLocationException, EntryNotFoundException,
        ClassUnknownException, HaMaException;
```

Παράμετροι

class_name	Το όνομα της κλάσης που εσωκλείει την υλοποίηση του πράκτορα. Η κλάση αυτή πρέπει οπωσδήποτε να κληρονομεί από την κλάση του πράκτορα (Agent). Είναι η κλάση που θα περιέχει την υλοποίηση της μεθόδου run() του νέου πράκτορα.
class_base	Το μονοπάτι στο οποίο βρίσκεται το αρχείο του ορισμού της κλάσης υλοποίησης του πράκτορα. Όταν δεν ορίζεται εννοείται πως η κλάση βρίσκεται στο κατάλογο με τις μόνιμες, στο σύστημα πρακτόρων, κλάσεις.
place_name	Το μέρος που θα ξεκινήσει η εκτέλεση του πράκτορα. Μπορεί να είναι είτε στο τοπικό είτε σε απομακρυσμένο σύστημα πρακτόρων.
args	Η λίστα των παραμέτρων για την κλήση του κατασκευαστή (constructor) της κλάσης υλοποίησης. Μπορεί να είναι κενή.

Υλοποίηση

Τα στάδια που περιλαμβάνει ο αλγόριθμος της δημιουργίας ενός νέου πράκτορα είναι:

1. Εύρεση της τοποθεσία του μέρους εκτέλεσης που θα δημιουργηθεί ο νέος πράκτορας.
2. Αποστολή αίτησης προς το σύστημα προορισμού για τη δημιουργία του πράκτορα (αν χρειάζεται).
3. Στο σύστημα προορισμού δημιουργείται μία ίνα για να αναλάβει την εκτέλεση του πράκτορα. Η εκτέλεση εκκινείται καλώντας τη μέθοδο run(). Επιστρέφεται το όνομα του νέου πράκτορα.
4. Το περιβάλλον που αναλαμβάνει τη δημιουργία του πράκτορα ελέγχει αν έχει τοπικά τις απαραίτητες κλάσεις. Αλλιώς τις ζητάει από το σύστημα που αιτήθηκε την έναρξη της διαδικασίας. Αυτό μπορεί να γίνεται κατά τη διάρκεια της αρχικοποίησης ή της εκτέλεσης του πράκτορα.

execMethod()

Χρησιμοποιείται για την επίκληση μεθόδου ενός πράκτορα. Ο πράκτορας καθορίζεται με το όνομά του και μπορεί να βρίσκεται είτε στο ίδιο, είτε σε διαφορετικό περιβάλλον εκτέλεσης με τον πράκτορα που αιτείται της επίκλησης της μεθόδου. Βέβαια, η εκτέλεση της μεθόδου λαμβάνει χώρα στο περιβάλλον εκτέλεσης του πράκτορα. Το όνομα της μεθόδου καθώς και ο τύπος των ορισμάτων καθορίζουν την μέθοδο που θα κληθεί. Η execMethod() επιστρέφει στον καλών πράκτορα το αποτέλεσμα της μεθόδου που εκτελέστηκε. Για να καλέστει η μέθοδος ο αιτών πρέπει να πληροί τις προϋποθέσεις που του δίνουν το προνόμιο αυτό. Δεν επιτρέπεται η κλήση όλων των μεθόδων του πράκτορα. Συγκεκριμένα απαγορεύεται η κλήση των move() γιατί μόνο ο ίδιος ο πράκτορας έχει την ιδιότητα να καλεί την μέθοδο αυτή, suspend(), resume(), και terminate() γιατί μπορούν να καλεστούν από τις αντίστοιχες μεθόδους.

Σύνταξη

```
java.lang.Object execMethod(String agent_name,
        String method_name,
```

```
        java.lang.Object[] arguments)
        throws MalformedLocationException, EntryNotFoundException,
        ExecMethodException, EAException,
        ForbiddenMethodException;
```

Παράμετροι

agent_name	Το όνομα του πράκτορα που παρέχει την υλοποίηση της μεθόδου.
method_name	Το όνομα της μεθόδου που θα καλεστεί.
arguments	Λίστα με τα ορίσματα της μεθόδου που θα καλεστεί. Ο τύπος των ορισμάτων χρησιμοποιείται για την εύρεση της μεθόδου λόγο της ιδιότητας της επαναχρησιμοποίησης των ονομάτων των μεθόδων στη γλώσσα προγραμματισμού Java. Οι τιμές των ορισμάτων χρησιμοποιούνται για την κλήση της επιθυμητής μεθόδου.

Υλοποίηση

Τα εξής βήματα περιλαμβάνει ο αλγόριθμος που χρησιμοποιείται από την μέθοδο αυτή:

1. Έλεγχος για το εάν η επιθυμητή μέθοδος ανήκει στο σύνολο των απαγορευμένων μεθόδων.
2. Εύρεση του περιβάλλοντος εκτέλεσης που φιλοξενεί τον πράκτορα.
3. Αποστολή αίτησης στο απομακρυσμένο σύστημα πρακτόρων για την επίκληση της συγκεκριμένης μεθόδου του πράκτορα, αν χρειάζεται. Ο πράκτορας θα συνεχίσει την εκτέλεσή του ή όταν τελειώσει η κλήση της απομακρυσμένης μεθόδου ή όταν γεννηθεί κάποια εξαίρεση λόγω σφάλματος.
4. Εκτέλεση της μεθόδου στο περιβάλλον εκτέλεσης του πράκτορα *'ιδιοκτήτη'*. Επιστροφή του αποτελέσματος της εργασίας ή γέννηση εξαίρεσης για την υπόδειξη του σφάλματος.

suspend()

Η μέθοδος αυτή χρησιμοποιείται για την προσωρινή διακοπή της εκτέλεσης ενός πράκτορα. Όταν ζητείται η διακοπή της εκτέλεσης κάποιου άλλου πράκτορα ελέγχεται αν ο αιτών πράκτορας έχει το προνόμιο να ξεκινήσει τη διαδικασία αυτή.

Σύνταξη

```
public final void suspend()
        throws SuspendException

public final void suspend(String agent_name)
        throws SuspendException, EntryNotFoundException, EAException;
```

Παράμετροι

agent_name	Το όνομα του πράκτορα του οποίου ζητείται η προσωρινή αναστολή της εκτέλεσης.

resume()

Η μέθοδος αυτή χρησιμοποιείται για την επανεκκίνηση της εκτέλεσης ενός πράκτορα μετά από προσωρινή αναστολή της εκτέλεσης του με χρήση της μεθόδου suspend(). Όταν ζητείται η επανεκκίνηση κάποιου άλλου πράκτορα ελέγχεται αν ο αιτών πράκτορας έχει το προνόμιο να ξεκινήσει τη διαδικασία αυτή.

Σύνταξη

```
public final void resume()
        throws ResumeException

public final void resume(String agent_name)
```

```
        throws ResumeException, EntryNotFoundException, EAException
```

Παράμετροι

agent_name	Το όνομα του πράκτορα του οποίου ζητείται η επανεκκίνηση της εκτέλεσης.

terminate()

Χρησιμοποιείται για την οριστική διακοπή της λειτουργίας ενός πράκτορα. Σε περίπτωση που ζητείται ο τερματισμός εκτέλεσης κάποιου άλλου πράκτορα ελέγχεται αν ο αιτών έχει το προνόμιο να τερματίσει την εκτέλεσή του.

Σύνταξη

```
public final void terminate()
        throws TerminateException;

public final void terminate(String agent_name)
        throws TerminateException, EntryNotFoundException, EAException;
```

Παράμετροι

agent_name	Το όνομα του πράκτορα του οποίου ζητείται ο τερματισμός της εκτέλεσης.

clone()

Η μέθοδος αυτή χρησιμοποιείται για την δημιουργία ενός νέου στιγμιότυπου του πράκτορα στον οποίον καλείται. Ο νέος πράκτορας δημιουργείται στο ίδιο περιβάλλον εκτέλεσης με τον κλώνο του. Το όνομα του νέου κλώνου που επιστρέφεται από την μέθοδο αποτελείται από δύο συστατικά. Ως πρώτο είναι το όνομα του πράκτορα από τον οποίο προέρχεται και δεύτερο είναι ένας αύξων αριθμός που αντιπροσωπεύει το σύνολο των κλώνων.

Σύνταξη

```
public final String clone()
        throws CloneException;

public final String clone(String agent_name)
        throws EntryNotFoundException, CloneException, EAException;
```

Παράμετροι

agent_name	Το όνομα του πράκτορα του οποίου ζητείται η κλωνοποίησή του.

Υλοποίηση

Τα βήματα του αλγορίθμου για την κλωνοποίηση ενός πράκτορα περιλαμβάνουν τα στάδια:

1. Αναστολή εκτέλεσης του πράκτορα, εάν ο πράκτορας εκτελείται.
2. Κλωνοποίηση των στιγμιοτύπων των κλάσεων που χρησιμοποιεί ο πράκτορας.
3. Το περιβάλλον εκτέλεσης αρχικοποιεί μία ίνα για να αναλάβει την εκτέλεση του πράκτορα κλώνου. Ανάθεση των δεδομένων στην ίνα, και έναρξη εκτέλεσής της, εάν ο αρχικός πράκτορας ήταν σε κατάσταση εκτέλεσης.
4. Επανέναρξη της εκτέλεσης του αρχικού πράκτορα (όποτε είναι απαραίτητο).

save()

Η μέθοδος αυτή χρησιμοποιείται για την αποθήκευση της κατάστασης του πράκτορα σε μόνιμο μέσο. Η αποθήκευση γίνεται σειριοποιώντας τις κλάσεις που χρησιμοποιεί ο πράκτορα. Ο πράκτορας συνεχίζει κανονικά την εκτέλεσή του, μετά το τέλος της λειτουργίας αυτής.

Σύνταξη

```
public final void save()
        throws SerializeException;

public final void save(String agent_name)
        throws EntryNotFoundException, SerializeException, EAException;
```

Παράμετροι

agent_name	Το όνομα του πράκτορα, του οποίου η κατάσταση θα αποθηκευτεί.

Υλοποίηση

Τα βήματα του αλγορίθμου για την αποθήκευση της κατάστασης ενός πράκτορα περιλαμβάνουν τα στάδια:

1. Αναστολή εκτέλεσης του πράκτορα, εάν ο πράκτορας εκτελείται.
2. Σειριοποίηση των στιγμιοτύπων των κλάσεων που χρησιμοποιεί ο πράκτορας.
3. Αποθήκευση της πληροφορίας σε μόνιμο μέσο.
4. Επανέναρξη της εκτέλεσης του πράκτορα (όποτε είναι απαραίτητο)

getAgentProperties() / setAgentProperties()

Μέθοδοι που δίνουν τη δυνατότητα στο προγραμματιστή να θέτει και να ανακτά τις ιδιαίτερες ιδιότητες ενός πράκτορα.

Σύνταξη

```
public final Property getAgentProperties();

public final void setAgentProperties(Property new_property);
```

Παράμετροι

new_property	Τα νέα ιδιαίτερα χαρακτηριστικά του πράκτορα.

getAgentSystemName() / getAgentSystemDescription()

Μέθοδοι για την ανάκτηση του ονόματος και της περιγραφής του συστήματος πρακτόρων στο οποίο φιλοξενείται τη στιγμή εκείνη ο πράκτορας. Η εξαίρεση τύπου MalformedLocationException γεννιέται αν για οποιονδήποτε λόγο δε μπορεί να επικοινωνήσει ο πράκτορας με το τοπικό σύστημα, συγκεκριμένα με τον πυρήνα του συστήματος πρακτόρων.

Σύνταξη

```
public final String getAgentSystemName()
        throws MalformedLocationException;

public final String getAgentSystemDescription()
        throws MalformedLocationException;
```

getPlaceList() / getPlaceName()

Η πρώτη επιστρέφει μία λίστα με ονόματα περιβαλλόντων εκτέλεσης που δραστηριοποιούνται στο τοπικό σύστημα πρακτόρων. Η δεύτερη επιστρέφει το όνομα του περιβάλλοντος εκτέλεσης στο οποίο εκτελείται την στιγμή εκείνη. Η εξαίρεση τύπου MalformedLocationException γεννιέται αν για οποιονδήποτε λόγο δεν μπορεί να επικοινωνήσει ο πράκτορας με την υπηρεσία ευρετηρίου του τοπικού συστήματος.

Σύνταξη

```
public final String[] getPlaceList()
        throws MalformedLocationException;
```

```
public final String getPlaceName()
        throws MalformedLocationException;
```

getAgentName() / getAgentList() / getAllAgentList()
Η πρώτη επιστρέφει το όνομα του πράκτορα. Η δεύτερη επιστρέφει μία λίστα με τα ονόματα των πρακτόρων που δραστηριοποιούνται στο ίδιο περιβάλλον εκτέλεσης την στιγμή της κλήσης. Και η τρίτη, όπως και η προηγούμενη, επιστρέφει μία λίστα που περιέχει τους πράκτορες που εκτελούνται στο ίδιο σύστημα πρακτόρων τη στιγμή της κλήσης. Η εξαίρεση τύπου MalformedLocationException γεννιέται αν για οποιονδήποτε λόγο δεν μπορεί να επικοινωνήσει ο πράκτορας με το τοπικό σύστημα, συγκεκριμένα με την υπηρεσία ευρετηρίου.

Σύνταξη
```
public final String getAgentName()

public final String[] getAgentList()
        throws MalformedLocationException;

public final String[] getAllAgentList()
        throws MalformedLocationException;
```

lookupAgent() / lookupPlace() / lookupAgentSystem()
Χρησιμοποιείται για την εύρεση της τοποθεσίας της οντότητας που αποτελεί το δεύτερο συνθετικό του ονόματος της μεθόδου. Η αναζήτησή της μπορεί να γίνει βάση του ονόματός του, σε όλες τις περιπτώσεις, ή και βάση κάποιων ιδιοτήτων του (ισχύει μόνο για τους πράκτορες και τα συστήματα πρακτόρων). Η τοποθεσία που επιστρέφει η μέθοδος αυτή μπορεί να χρησιμοποιηθεί ως όρισμα στην μέθοδο move(). Σε περίπτωση που δεν βρεθεί καμία οντότητα που να ικανοποιεί τα κριτήρια της αναζήτησης γεννιέται μία εξαίρεση τύπου EntryNotFoundException.

Σύνταξη
```
public final Location lookupAgent(String agent_name)
        throws MalformedLocationException, EntryNotFoundException;

public final Location[] lookupAgent(Property property)
        throws MalformedLocationException, EntryNotFoundException;

public final Location lookupPlace(String place_name)
        throws MalformedLocationException, EntryNotFoundException;

public final Location lookupAgentSystem(String agent_system_name)
        throws MalformedLocationException, EntryNotFoundException;

public final Location[] lookupAgentSystem(Property property)
        throws MalformedLocationException, EntryNotFoundException;
```

getLocation()
Επιστρέφει την τρέχουσα τοποθεσία του πράκτορα.

Σύνταξη
```
public final Location getLocation();
```

isSuspended() / isRunning()
Γνωστοποιούν την κατάσταση εκτέλεσης του πράκτορα. Επιστρέφουν αληθές όταν ο πράκτορας του οποίου η μέθοδος κλήθηκε έχει διακόψει την εκτέλεσή του ή εκτελείται κανονικά, αντίστοιχα.

Σύνταξη
```
public final boolean isSuspended();

public final boolean isRunning();
```

Μία σημείωση που κρίνεται απαραίτητη είναι ότι οι μέθοδοι resume(), suspend(), terminate(), clone(), και save() όταν χρησιμοποιούνται χωρίς παράμετρο αναφέρονται στον πράκτορα από τον οποίο καλούνται. Ενώ, όταν χρησιμοποιούνται με την παράμετρο τύπου String, αναφέρονται στον πράκτορα του οποίου το όνομα δίνεται ως τιμή στην παράμετρο κατά την κλήση. Ο πράκτορας μπορεί να βρίσκεται είτε στο τοπικό, είτε σε απομακρυσμένο σύστημα πρακτόρων. Η διαδικασία που εκκινείτε για τη διεκπεραίωση της αίτησης αποτελείται από τα εξής βήματα:

1. Εύρεση του πράκτορα χρησιμοποιώντας το σύστημα ευρετηρίου της περιοχής
2. Προετοιμασία και αποστολή της αίτησης στο απομακρυσμένο σύστημα που φιλοξενεί τον επιθυμητό πράκτορα.
3. Το απομακρυσμένο σύστημα αναλαμβάνει να εκκινήσει όλες τις λειτουργίες που απαιτούνται για την εξυπηρέτηση της εκάστοτε αίτησης.
4. Επίσης, αναλαμβάνει να ειδοποιήσει το αρχικό σύστημα για τυχών σφάλματα στην εκτέλεση, ή να επιστρέψει την απάντηση (όποτε είναι απαραίτητο).

6.1.2 Μέθοδοι που μπορούν να επανοριστούν

Το σύνολο αυτό απαρτίζεται από τις μεθόδους που προσδίδουν στον πράκτορα την λειτουργικότητα που επιθυμεί ο προγραμματιστής. Ορισμένες από τις μεθόδους αυτές πρέπει οπωσδήποτε να υλοποιηθούν από τον εκάστοτε προγραμματιστή, όπως η μέθοδος run(). Για ορισμένες άλλες παρέχεται μία τυποποιημένη υλοποίηση σε περίπτωση που ο προγραμματιστής δεν επιθυμεί να προσθέσει επιπλέον λειτουργικότητα. Ακολουθούν οι περιγραφές όλων των μεθόδων που αποτελούν το σύνολο.

run()
Είναι η μέθοδος που εσωκλείει το έργο που έχει να περατώσει ο πράκτορας. Πρέπει οπωσδήποτε να υλοποιείται από τους προγραμματιστές στην κλάση που κληρονομεί από την κλάση του πράκτορα (class Agent). Ο προγραμματιστής πρέπει να έχει υπ' όψιν του ότι η μέθοδος αυτή καλείται αυτόματα μετά από κάθε δημιουργία ενός καινούργιου στιγμιότυπου του πράκτορα. Καινούργιο στιγμιότυπο ενός πράκτορα έχουμε κατά τις λειτουργίες της δημιουργίας νέου πράκτορα και της μεταφοράς πράκτορα. Οπότε, η μέθοδος run(), που πρέπει να υλοποιήσει ο προγραμματιστής, καλείται αυτόματα για να εκκινήσει την εκτέλεση του πράκτορα μετά από την μέθοδο move() και την μέθοδο createAgent().

Σύνταξη
```
public abstract void run();
```

setAuthority() / setIdentity() / setDescription()
Είναι οι μέθοδοι που δίνουν τη δυνατότητα στον προγραμματιστή να θέσει τα βασικά χαρακτηριστικά του πράκτορα. Το σύνολο των χαρακτηριστικών, που χρησιμοποιούνται από τα συστήματα πρακτόρων μια και καθορίζονται από το πρότυπο, είναι το όνομα, η δικαιοδοτική αρχή, και μία περιγραφή. Η πληροφορία αυτή είναι συνυφασμένη με τον κώδικα του πράκτορα και ταξιδεύει μαζί του στα περιβάλλοντα εκτέλεσης. Μπορεί να αλλαχθεί μόνο από τον ίδιο τον πράκτορα. Ο προγραμματιστής μπορεί, παρέχοντας την υλοποίησή τους να αποδώσει να χαρακτηριστικά που επιθυμεί στον πράκτορα αυτό. Το περιβάλλον προγραμματισμού παρέχει ένα προκαθορισμένο τρόπο υλοποίησης των μεθόδων.

Σύνταξη
```
protected String setAuthority();

protected String setIdentity();

protected String setDescription();
```

loaded() / saved() / terminated()
Με τις μεθόδους αυτές δίνεται η δυνατότητα στον προγραμματιστή να καθορίσει επιπρόσθετη εργασία στον πράκτορα παράλληλα με την εκτέλεση κάποιων προκαθορισμένων διαδικασιών. Ο προγραμματιστής επανορίζοντας την υλοποίηση των μεθόδων μπορεί προσθέσει ενέργειες που πρέπει να εκτελέσει

- ένας αποθηκευμένος σε μόνιμο μέσο πράκτορας αμέσως μόλις επανεκκινηθεί η εκτέλεσή του, μέθοδος loaded(),
- ένας πράκτορας ακριβώς πριν ενεργοποιηθεί ο μηχανισμός που θα τον αποθηκεύσει σε μόνιμο μέσο, μέθοδος saved(),
- ένας πράκτορας ακριβώς πριν ενεργοποιηθεί ο μηχανισμός που θα τερματίσει την λειτουργία του, μέθοδος terminated().

Σύνταξη
```
protected void loaded();

protected void saved();

protected void terminated();
```

6.1.3 Η μέθοδος run() των κινούμενων πρακτόρων.

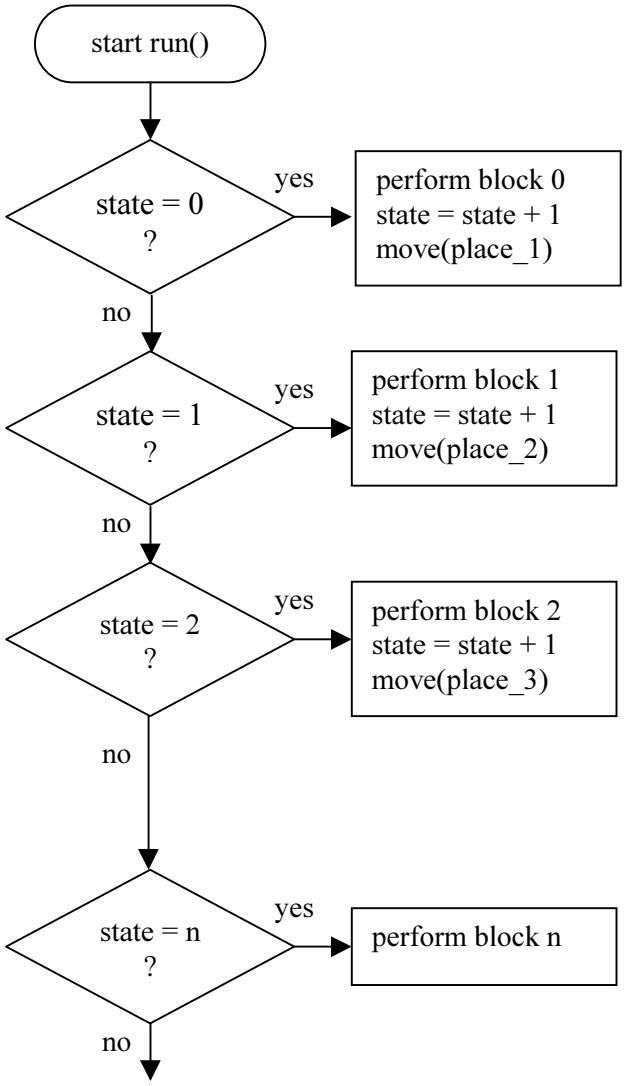
Αναφέραμε προηγουμένως ότι η μέθοδος run() περικλείει όλο το έργο του πράκτορα, δηλαδή όλες τις δραστηριότητες (δηλώσεις σε γλώσσα προγραμματισμού Java) που πρέπει να εκπληρωθούν κατά τη διάρκεια ζωής του πράκτορα. Η παρούσα ενότητα εξηγεί τους κανόνες που πρέπει να ληφθούν υπ' όψιν κατά την υλοποίηση της μεθόδου.

Εξ ορισμού, οι κινούμενοι πράκτορες μπορούν να μετακινούνται από τον ένα ξενιστή στον άλλο. Υπάρχουν όμως δύο διαφορετικοί τύποι μετακίνησης πρέπει να διαχωριστούν:

- *Απομακρυσμένη εκτέλεση*, που σημαίνει ότι το αντικείμενο, που μετακινείται, ξεκινάει την εκτέλεση της εργασίας του από την αρχή. Η τρόπος αυτός δεν σχετίζεται άμεσα με την τεχνολογία των κινούμενων πρακτόρων. Τα αντικείμενα που παρέχουν την δυνατότητα της απομακρυσμένης εκτέλεσης δεν πρέπει να θεωρούνται κινούμενοι πράκτορες αλλά κινούμενος κώδικας.
- *Μετανάστευση* που σημαίνει ότι το αντικείμενο (πράκτορας), μετά την μετακίνησή του σε ένα νέο ξενιστή (περιβάλλον εκτέλεσης), συνεχίζει την εργασία του από το σημείο που την είχε διακόψει. Η μετανάστευση απαιτεί, εκτός από την μεταφορά του κώδικα του πράκτορα και την μεταφορά της κατάστασης εκτέλεσης. Η κατάσταση εκτέλεσης σημειώνει το τρέχων σημείο της εκτέλεσης της εργασίας του πράκτορα.

Η γλώσσα προγραμματισμού Java, δεν επιτρέπει την αποθήκευση της στοίβας που χρησιμοποιείται για την εκτέλεση του προγράμματος (πράκτορα). Γι' αυτό, η κατάσταση εκτέλεσης του πράκτορα πρέπει να αναπαρίσταται με κάποιον άλλον τρόπο, ώστε να προσωμοιώνεται όσο το συνατό καλύτερα η μετακίνηση του αντικειμένου. Παρακάτω παρουσιάζεται μία τεχνική που μπορεί να χρησιμοποιηθεί από τους προγραμματιστές με σκοπό την επίτευξη της μετανάστευσης των πρακτόρων.

Οι δηλώσεις της γλώσσας προγραμματισμού Java που περιέχονται στη μέθοδο run() του πράκτορα πρέπει να είναι χωρισμένες σε ομάδες, σε κομμάτια εκτελούμενου κώδικα (execution blocks). Κάθε τέτοιο κομμάτι εκτελείται εξ' ολοκλήρου σε ένα περιβάλλον εκτέλεσης, και η τελευταία μέθοδός του είναι η μέθοδος move(). Έτσι, μετά την ολοκλήρωση της εργασίας, που όριζε το κομμάτι του εκτελούμενου κώδικα στο συγκεκριμένο μέρος, ο πράκτορας μετακομίζει στο επόμενο μέρος. Βέβαια, μετά τη μετακόμιση ο πράκτορας πρέπει να αποφασίσει ποιο κομμάτι εκτελούμενου κώδικα θα εκτελέσει. Αυτό μπορεί να καθοριστεί εσωκλείοντας τα κομμάτια εκτελούμενου κώδικα μέσα σε δηλώσεις καθορισμού της ροής εκτέλεσης του προγράμματος, δηλαδή διακλαδώσεις (π.χ. switch, if-else-is). Όμως, πρέπει επιπλέον να αποθηκεύει και σε μία μεταβλητή η τιμή που θα καθορίζει το επόμενο κομμάτι εντολών προς εκτέλεση. Η μεταβλητή που αποτελεί την κατάσταση εκτέλεσης του πράκτορα, πρέπει να ενημερώνεται πριν τη μετακόμιση του πράκτορα. Αλλιώς, το ίδιο κομμάτι εντολών θα εκτελείται, από την αρχή, μετά από κάθε μετακίνησή του, που σημαίνει ότι η όλη εκτέλεση θα είναι ένας ατελείωτος βρόγχος. Στην παρακάτω εικόνα παρουσιάζεται γραφικά η δομή της μεθόδου run().



Σχήμα 6-1: Η δομή που πρέπει να έχει η μέθοδος run().

6.2 Τοποθεσία (Location)

Είναι η κλάση που χρησιμοποιείται για να περιγράψει τη διεύθυνση οποιασδήποτε οντότητας του περιβάλλοντος πρακτόρων. Επιστρέφεται από κλήσεις μεθόδων που σχετίζονται με εύρεση οντοτήτων, όπως lookup_agent(), lookup_place().

6.3 Κλάσεις ενδείξεως σφάλματος

Το περιβάλλον ανάπτυξης εφαρμογών κινούμενων πρακτόρων του EasyAgent παρέχει και κλάσεις για τη δήλωση σφαλμάτων στην εκτέλεση. Οι κλάσεις κληρονομούν από την γενική κλάση για ένδειξη σφάλματος της γλώσσας προγραμματισμού Java, την java.lang.RuntimeException. Επανορίζουν την υλοποίηση της μεθόδου toString() για να ενημερώνουν τον προγραμματιστή με τον ακριβή λόγο του σφάλματος. Οι παρακάτω κλάσεις ορίζουν το σύνολο των σφαλμάτων στην εκτέλεση που διαχωρίζει το σύστημα EasyAgent. Οποιοδήποτε άλλο σφάλμα παρουσιάζεται ως σφάλμα τύπου EAException.

TerminateException	δηλώνει ότι συνέβηκε κάποιο σφάλμα κατά τη διαδικασία του τερματισμού της εκτέλεσης. Ο πράκτορας συνεχίζει να εκτελείται.
SuspendException	δηλώνει ότι συνέβηκε κάποιο σφάλμα κατά τη διαδικασία της αναστολής λειτουργίας του πράκτορα. Ο εν λόγω πράκτορας συνεχίζει κανονικά την εκτέλεσή του.
ResumeException	δηλώνει ότι συνέβηκε κάποιο σφάλμα κατά τη διαδικασία της επανέναρξης λειτουργίας του πράκτορα. Ο εν λόγω πράκτορας παραμένει ανενεργός.
ExecMethodException	δηλώνει σφάλμα κατά την επίκληση μεθόδου ενός πράκτορα. Η γένεση της εξαίρεσης μπορεί να σημαίνει σφάλμα κατά την επίκληση της μεθόδου, σφάλμα κατά την εκτέλεση της μεθόδου, ή και σφάλμα κατά την επιστροφή του αποτελέσματος. Οπότε, μετά την εμφάνιση της εξαίρεσης, ο καλών δε μπορεί να γνωρίζει το αν ή όχι έχει περατωθεί κάποιο μέρος ή και ολόκληρη η μέθοδος.
MalformedLocationException	δηλώνει σφάλμα κατά την προσπάθεια επικοινωνίας με κάποια οντότητα. Το σφάλμα μπορεί να προέρχεται είτε λόγο λανθασμένης σύνταξης της διεύθυνσης, είτε λόγο απώλειας του αναφορικού δείκτη προς την οντότητα αυτή.
EntryNotFoundException	δηλώνει σφάλμα στη διαδικασία αναζήτησης του ευρετηρίου. Δηλαδή, δε βρέθηκε οντότητα που να εκπληρεί τις προϋποθέσεις της αναζήτησης.
SerializationException	δηλώνει σφάλμα κατά τη διαδικασία σειριοποίησης του πράκτορα. Μπορεί να γεννηθεί και κατά την εκτέλεση της μεθόδου move() και σημαίνει ότι οι κλάσεις που χρησιμοποιεί ο πράκτορας δεν μπορούν να σειριοποιηθούν για να αποσταλούν στο σύστημα προορισμού.
DeserializationException	δηλώνει σφάλμα κατά την διαδικασία αποσειριοποίησης του πράκτορα.

ClassUnknownException	δηλώνει πως ο πράκτορας δε μπορεί να βρει τον ορισμό μίας κλάσης απαραίτητης για τη λειτουργία του. Δηλαδή, από τη μία, ο ορισμός της κλάσης δεν είναι γνωστός στο τοπικό σύστημα πρακτόρων. Από την άλλη, εάν ο πράκτορας σχετίζεται με κάποιο απομακρυσμένο σύστημα πρακτόρων (createAgent(), move()), το σύστημα αυτό δεν εξυπηρετεί την αίτηση μεταφοράς κλάσης που του έχει αποσταλεί. Η αίτηση αποστέλεται αυτόματα από το σύστημα πρακτόρων μετά τη διαπίστωση της άγνοιας του ορισμού της κλάσης.
EAException	υπάρχει για οποιαδήποτε άλλη εξαίρεση δε μπορεί να καλυφθεί από τις παραπάνω αναφερόμενες περιπτώσεις.

7 Ένα σενάριο χρήσης

Στα προηγούμενα κεφάλαια ασχοληθήκαμε με την περιγραφή του συστήματος πρακτόρων που αναπτύχθηκε στα πλαίσια της εργασίας αυτής, καθώς και με το περιβάλλον προγραμματισμού που παρέχεται για την ανάπτυξη εφαρμογών κινούμενων πρακτόρων. Αυτό που δεν έχουμε δει είναι ο τρόπος που λειτουργεί το σύστημα πρακτόρων όπως επίσης και τον τρόπο που οι εφαρμογές κινούμενων πρακτόρων χρησιμοποιούν τις παρεχόμενες οντότητες με σκοπό τη διεκπεραίωση της εργασίας που τους ανατέθηκε. Μερικοί από τους αναγνώστες ίσως να μην έχουν πειστεί ακόμα για τη χρησιμότητα των συστημάτων πρακτόρων. Στα ζητήματα αυτά βασίζεται το περιεχόμενο του κεφαλαίου αυτού.

Πρώτο μέλημα του κεφαλαίου είναι να παρουσιάσει τις ανάγκες της αγοράς που οδηγούν στη χρήση της τεχνολογίας των κινούμενων πρακτόρων. Στη συνέχεια, περιγράφει αναλυτικά ένα σενάριο χρήσης των συστημάτων πρακτόρων για την επίλυση ενός προβλήματος συχνά απαντούμενου στο κομμάτι της αγοράς που απευθύνεται η τεχνολογία αυτή.

7.1 Εισαγωγή

Οι απαιτήσεις τις πλειοψηφίας των καταναλωτών όσον αφορά τα αγαθά που πρόκειται να αγοράσουν, είναι η ποιότητα και η χαμηλή τιμή και όσον αφορά την αλληλεπίδρασή τους με τους πωλητές ή και τους κατασκευαστές είναι η εξυπηρέτηση. Επιχειρήσεις, μεγάλες σε έκταση, που διατηρούν πολλαπλά υποκαταστήματα μπορούν να παρέχουν μεγάλη γκάμα προϊόντων, σε χαμηλή τιμή. Το πόσο καλή είναι η εξυπηρέτηση εξαρτάται από το πόσο καλά συνεργάζονται τα συστήματα που υπάρχουν στα εν λόγω υποκαταστήματα. Αν υποθέσουμε ότι οι επιχειρήσεις αυτές προέρχονται από συγχωνεύσεις μικρότερων εταιριών, από εξαγορές αλλά και επεκτάσεις των υπό έλεγχο επιχειρήσεων, προκύπτει ένα σύνολο ανομοιογενών στην οργάνωσή τους επιχειρήσεων που πρέπει να συνεργαστούν όχι μόνο για την ευκολότερη και γρηγορότερη ανάπτυξη της επιχείρησης, αλλά και για την καλύτερη δυνατή εξυπηρέτηση του καταναλωτή. Ο όρος ανομοιογένεια στην οργάνωση περικλείει τους διαφορετικούς τύπους συστημάτων που χρησιμοποιούνται από τις επιμέρους επιχειρήσεις, τις διαφορετικές εφαρμογές που υποστηρίζουν τη λειτουργία τους, τους διαφορετικούς τρόπους κωδικοποίησης και αποθήκευσης των δεδομένων, ακόμα και τις διαφορετικές εκδόσεις μίας εφαρμογής που χρησιμοποιούνται. Σε ένα τέτοιο περιβάλλον η αλληλεπίδραση των επιμέρους συστημάτων φαίνεται να είναι δύσκολο πρόβλημα που πρέπει όμως να διευθετηθεί.

Ανάλογο είναι και το πρόβλημα της συνεργασίας πάνω από το διαδίκτυο. Και αυτό απαρτίζεται από πολλά διαφορετικά συστήματα που το καθένα εξυπηρετεί συγκεκριμένους σκοπούς. Το μέχρι τώρα χρησιμοποιούμενο μοντέλο για την αλληλεπίδραση των συστημάτων, αυτό του εξυπηρετούμενου - εξυπηρέτη (client - server), δε μπορεί να καλύψει από μόνο του τις ανάγκες που προκύπτουν από τη συνεχώς αυξανόμενη εξάπλωση του Παγκόσμιου Ιστού.

Από τα παραπάνω είναι εμφανές ότι η λύση του προβήματος πρέπει να μπορεί, από την μία πλευρά να επιτρέπει την ανάπτυξη συστημάτων διαφορετικών μεταξύ τους και από την άλλη να τους παρέχει τρόπους για αλληλεπίδραση. Ας δούμε λοιπόν πως η τεχνολογία των κινούμενων πρακτόρων και πιο συγκεκριμένα τα συστήματα πρακτόρων που είναι συμβατά με το πρότυπο μπορούν να συνεισφέρουν στην λύση του προβλήματος.

7.2 Το σενάριο

Έχοντας ήδη ορίσει μία περιοχή που θα μπορούσε να βοηθήσει η τεχνολογία των κινούμενων πρακτόρων μπορούμε να θέσουμε ένα ρεαλιστικό πρόβλημα στο οποίο η ανάγκη για συνεργασία των συστημάτων είναι καθοριστική. Ας υποθέσουμε λοιπόν ότι η εταιρία EasyMusic ασχολείται με την εμπορία μέσων αναπαραγωγής εικόνας και ήχου (οπτικοί δίσκοι, βυνίλιο, κασέτες) και έχει υποκαταστήματα σε πολλές περιοχές της χώρας. Έχει επιπλέον και αποθήκες υλικού σε μερικές μεγαλουπόλεις. Όλα τα υποκαταστήματα και οι αποθήκες είναι μηχανογραφημένες αλλά όχι με το ίδιο σύστημα μηχανογράφησης. Για την καλύτερη εξυπηρέτηση του πελάτη τα επιμέρους υποκαταστήματα φροντίζουν να είναι όσο το δυνατό ενήμερα για την κατάσταση της αποθήκης όλων των αντιπροσώπων της εταιρίας. Έτσι, εάν τους ζητηθεί ένα προϊόν που δεν υπάρχει στην αποθήκη τους να μπορούν να απαντήσουν υπεύθυνα στον πελάτη την ακριβή ποσότητα που έχει ετοιμοπαράδοτη η εταιρία αλλά και το χρόνο που θα μεσολαβήσει έως ότου μεταφερθούν στο τοπικό κατάστημα. Βέβαια, ο μηχανισμός της ενημέρωσης της αποθήκης δεν είναι τόσο απλός. Χρειάζεται επικοινωνία μεταξύ των συστημάτων που υπάρχουν στα υποκαταστήματα αλλά και συνεργασία συστημάτων διαφορετικών μεταξύ τους. Δηλαδή, από την μία εισάγεται πολύ κίνηση στο δίκτυο της εταιρίας λόγω της καθημερινής αντιγραφής των δεδομένων και από την άλλη είναι απαραίτητη μερικές φορές η κωδικοποποίηση των δεδομένων προκειμένου να αλληλεπιδράσουν τα συστήματα διαφορετικών κατασκευαστών.

Το αντίστοιχο σενάριο μπορεί να σκηνοθετηθεί και πάνω από το διαδίκτυο. Εδώ το ρόλο του πωλητή παίζουν οι ιστοσελίδες του κάθε υποκαταστήματος. Αυτές παρέχουν άμεσα στον ενδιαφερόμενο οποιαδήποτε πληροφορία χρειάζεται για το προϊόν που επιθυμεί να αγοράσει. Όμως δύο είναι τα προβλήματα που καλείται να αντιμετωπίσει ο αγοραστής. Πρώτο και κυριότερο, είναι η εύρεση ιδεατών καταστημάτων που εμπορεύονται τα προϊόντα του ενδιαφέροντός του (εύρεση δηλαδή των διευθύνσεών τους στο διαδίκτυο). Και δεύτερο είναι η αλληλεπίδραση με τη διεπαφή χρήσης που προσφέρει το κάθε κατάστημα (ο τρόπος παρουσίασης και αναζήτησης των προϊόντων μέσα από τις ιστοσελίδες των καταστημάτων δεν είναι ποτέ ο ίδιος).

7.3 Λύση βασισμένη στην υπάρχουσα τεχνολογία

Ένας πελάτης πηγαίνει στο υποκατάστημα Α της εταιρίας στο Ηράκλειο και ζητάει 50 οπτικούς δίσκους του δημοφιλούς τραγουδιστή Χ. Ο πωλητής συμβουλεύεται το τερματικό του για την κατάσταση της αποθήκης του. Το προϊόν δεν υπάρχει. Με το σύστημα που έχει μπορεί να ελέγξει αν υπάρχει η απαιτούμενη ποσότητα στα υπόλοιπα υποκαταστήματα της ίδιας πόλης μια και αυτά χρησιμοποιούν το ίδιο σύστημα. Η προσπάθειά του όμως είναι αποτυχημένη. Για να εξυπηρετήσει τον πελάτη προσπαθεί να συνδεθεί με το σύστημα αποθήκης της γειτονικής πόλης (π.χ. Ρέθυμνο) αλλά υπάρχει πρόβλημα στο διασυνδετικό δίκτυο. Έτσι, τηλεφωνεί στο κεντρικό υποκατάστημα των Χανίων, μια και το τοπικό σύστημα δεν είναι πλήρως ενημερωμένο για την κατάσταση της αποθήκης των Χανίων, όπου ο υπάλληλος τον πληροφορεί ότι έχει μόνο 20 οπτικούς δίσκους και 15 δίσκους βυνιλίου. Ο πωλητής ενημερώνει τον πελάτη για τους διαθέσιμους οπτικούς δίσκους και τον ρωτάει αν οι δίσκοι βυνιλίου είναι επιθυμητοί. Ταυτόχρονα, προσπαθεί να επικοινωνήσει με άλλα καταστήματα στην υπόλοιπη χώρα αλλά και με τις κεντρικές αποθήκες. Επειδή όμως περιμένουν και άλλοι πελάτες, λέει στον ενδιαφερόμενο να επικοινωνήσει ξανά μαζί του την επόμενη μέρα που θα έχει περισσότερες πληροφορίες. Ο πελάτης φεύγει και δεν ξαναπαίρνει τηλέφωνο.

Στην περίπτωση του διαδικτύου, ο ενδιαφερόμενος επισκέπτεται μερικά καταστήματα που είτε έχει τύχει να συνεργαστεί μαζί τους στο παρελθόν είτε έχει ακούσει για αυτά (διαφήμιση, φιλική συμβουλή). Από το σύνολο των καταστημάτων αυτών θα επιλέξει κάποιο

με κριτήρια, το χαμηλό μέσο όρο τιμών (όχι αυτό με το χαμηλότερο κόστος των αγαθών που επιθυμεί να αγοράσει) και την περισσότερο εύχρηστη διεπαφή χρήσης. Αρκετές φορές θα ξεχάσει να συγκρίνει το κόστος της αποστολής των προϊόντων που χρεώνουν τα καταστήματα.

7.4 Η επιθυμητή λύση

Ένας πελάτης πηγαίνει στο υποκατάστημα Α της εταιρίας στο Ηράκλειο και ζητάει 50 οπτικούς δίσκους του δημοφιλούς τραγουδιστή Χ. Ο πωλητής συμβουλεύεται το τερματικό του για την κατάσταση της αποθήκης του. Το προϊόν δεν υπάρχει. Το σύστημα τον ρωτάει αν θέλει να αναζητήσει το προϊόν σε άλλα υποκαταστήματα. Επίσης, τον ρωτάει αν ενδιαφέρεται για παρόμοια αγαθά όπως δίσκους βυνιλίου και κασέτες. Ο πωλητής εισάγει τα δεδομένα και ξεκινάει τη διαδικασία της αναζήτησης.

Ένα λεπτό αργότερα, το σύστημα τον πληροφορεί ότι βρήκε μέρος της επιθυμητής ποσότητας (π.χ. 20) στο υποκατάστημα στα Χανιά αλλά και 15 δίσκους βυνιλίου και 10 κασέτες. Ο πελάτης πληροφορείτε για τα πρώτα αποτελέσματα της αναζήτησης και λέει ότι οι δίσκοι βυνιλίου είναι επιθυμητοί όχι όμως και οι κασέτες. Ο πωλητής αλλάζει τα κριτήρια της αναζήτησης και ζητάει από το σύστημα να επανεκκινήσει τη διαδικασία από το σημείο που σταμάτησε.

Μερικά λεπτά αργότερα, το τερματικό πληροφορεί τον πωλητή ότι βρέθηκαν 25 οπτικοί δίσκοι σε κάποιο υποκατάστημα στην Θεσσαλονίκη. Ο πελάτης ζητάει να του αποσταλούν στο σπίτι του οι 45 οπτικοί δίσκοι και 5 δίσκοι βυνιλίου με το ταχυδρομείο. Ο πωλητής ετοιμάζει την παραγγελία που θα εξυπηρετηθεί από τα υποκαταστήματα των Χανίων και της Θεσσαλονίκης.

Όσον αφορά το σενάριο πάνω από το διαδίκτυο, θα μπορούσε ο ενδιαφερόμενος αγοραστής να ενεργοποιήσει μία διαδικασία αυτόματης ανεύρεσης των ιδεατών καταστημάτων εμπορίας μέσων αναπαραγωγής ήχου και εικόνας. Στη συνέχεια, αρχικοποιώντας με τα χαρακτηριστικά του επιθυμητού προϊόντος τη διαδικασία της αναζήτησης, δίνει εντολή για την έναρξή της. Η διεπαφή χρήσης του συστήματος που χρησιμοποιεί τον ειδοποιεί για οποιαδήποτε εμφάνιση του προϊόντος στη λίστα των καταστημάτων που ψάχνει. Ο ενδιαφερόμενος μέσω της διεπαφής μπορεί να επιλέξει την οικονομικότερη γι' αυτόν λύση, αποφεύγοντας την πολυπλοκότητα που επιβάλλει η αλληλεπίδραση με τα συστήματα του εκάστοτε ιδεατού καταστήματος, αλλά και αποταμιεύοντας χρόνο που θα σπαταλούσε εάν από μόνος του αναζητούσε εξαντλητικά τον πλήρη κατάλογο των πιθανών πωλητών.

7.5 Η απαιτούμενη υποδομή

Η απαιτούμενη υποδομή για την πραγματοποίηση τόσο του σεναρίου πάνω από το διαδίκτυο όσο και αυτού που σχετίζεται με τα πολλαπλά υποκαταστήματα της εταιρίας περιλαμβάνει την χρήση της τεχνολογίας των κινούμενων πρακτόρων σε συνδυασμό με καλύτερες τεχνικές ομαδοποίησης και ενοποίησης των δεδομένων. Κάθε κατάστημα, ιδεατό (στην περίπτωση του διαδικτύου) ή πραγματικό (στην περίπτωση της εταιρίας) έχει εγκαταστημένο ένα σύστημα πρακτόρων. Παρόλο που το σύνολο των μηχανημάτων είναι ετερογενές (οι ξενιστές που φιλοξενούν τα συστήματα πρακτόρων μπορεί να διαφέρουν σε θέματα αρχιτεκτονικής της μηχανής ή και σε θέματα λογισμικού που υποστηρίζουν), το περιβάλλον που συνίσταται από τη χρησιμοποίηση των συστημάτων πρακτόρων προσφέρει την υποδομή για την υποστήριξη των κινούμενων πρακτόρων. Μέσα σε ένα τέτοιο περιβάλλον οι πράκτορες μπορούν να αλληλεπιδρούν μεταξύ τους, να συντονίζουν τις ενέργειές τους και να εντοπίζουν την πληροφορία που αναζητούν.

Οι υπηρεσίες των συστημάτων πρακτόρων και ευρετηρίου που χρησιμοποιούνται από σενάρια όπως αυτό που περιγράφεται στην προηγούμενη ενότητα είναι:

- εντοπισμός ενός συστήματος πρακτόρων για τη δημιουργία πράκτορα (με χρήση της διεπαφής του συστήματος ευρετηρίου).
- δημιουργία πράκτορα για την αναζήτηση του επιθυμητού προϊόντος (χρησιμοποιώντας τη διεπαφή του συστήματος πρακτόρων).
- εγγραφή του πράκτορα στο σύστημα ευρετηρίου καθώς μετακινείται, ώστε να μπορούν οι υπόλοιπες οντότητες να τον εντοπίζουν.
- μετακίνηση του πράκτορα πάνω από τα περιβάλλοντα εκτέλεσης για αναζήτηση των αγαθών (με χρήση της διεπαφής του συστήματος πρακτόρων).
- εντοπισμός του πράκτορα για την απόκτηση πληροφοριών που αφορούν την κατάστασή του (με χρήση της διεπαφής του συστήματος ευρετηρίου).
- αναστολή ή επανέναρξη της εκτέλεσης πράκτορα (με χρήση της διεπαφής του συστήματος πρακτόρων).
- ενεργοποίηση υπηρεσιών που προσφέρει ο πράκτορας μέσω της κλήσης μεθόδων του πράκτορα (με χρήση της διεπαφής του συστήματος πρακτόρων).

7.6 Η αλληλεπίδραση των οντοτήτων

Η επιθυμία του αγοραστή για την απόκτηση των συγκεκριμένων οπτικών δίσκων αρχικοποιεί μια σειρά από δραστηριότητες. Στην ενότητα αυτή παρουσιάζεται λεπτομερώς η αλληλεπίδραση των οντοτήτων (συστήματα πρακτόρων, συστήματα ευρετηρίου, πρακτόρων) με στόχο την εύρεση και αγορά οπτικών δίσκων από τα ιδεατά καταστήματα του διαδικτύου. Περιγράφεται η αλληλεπίδραση του χρήστη με το σύστημα πρακτόρων και το στατικό πράκτορα. Σε παραγράφους που αφήνεται λίγο μεγαλύτερο περιθώριο αναφέρονται οι ενέργειες των πρακτόρων και οι μέθοδοι της κλάσης του πράκτορα που χρησιμοποιούνται. Επιπλέον λεπτομέρειες περιγράφονται στις παραγράφους με διπλό περιθώριο.

Πριν ξεκινήσουμε την αναλυτική περιγραφή και προς αποφυγή παρεξηγήσεων και σκοτεινών σημείων αναφέρουμε τα εξής:

- *ο αγοραστής αλληλεπιδρά με το περιβάλλον που συνθέτουν τα συστήματα πρακτόρων μέσω ενός στατικού πράκτορα.* Αν και το σύστημα πρακτόρων EasyAgent δεν κάνει διάκριση μεταξύ στατικών και κινούμενων πρακτόρων, του αποδίδεται η ονομασία αυτή γιατί ο εν λόγω πράκτορας δε μετακινείται καθ' όλη τη διάρκεια της ζωής του. Εκτελείται στο τοπικό σύστημα πρακτόρων (στο μηχάνημα του ενδιαφερόμενου αγοραστή) και παίζει το ρόλο του μεσάζοντα που επιτρέπει τη μεταφορά πληροφοριών από τον αγοραστή στο σύστημα και αντίστροφα. Παρέχει μία διεπαφή χρήσης μέσω της οποίας ο χρήστης μπορεί να εκμεταλλευθεί τις παρεχόμενες από τα συστήματα πρακτόρων υπηρεσίες.
- *οποτεδήποτε είναι αναγκαίος ο εντοπισμός κάποιας οντότητας (σύστημα πρακτόρων, πράκτορας, περιβάλλον εκτέλεσης) χρησιμοποιείται το σύστημα ευρετηρίου.* Ο εντοπισμός του ίδιου του συστήματος ευρετηρίου γίνεται μέσω του συστήματος πρακτόρων, μια και το τελευταίο κατά την έναρξη της λειτουργίας του αποκτά ένα αναφορικό δείκτη για το σύστημα ευρετηρίου μέσω της υπηρεσίας ονοματολογίας της αρχιτεκτονικής τύπου CORBA.

Ο ενδιαφερόμενος αγοραστής θέτει σε λειτουργία ένα σύστημα πρακτόρων EasyAgent, πάνω στο οποίο θα ξεκινήσει τον εξειδικευμένο πράκτορα για την αγορά οπτικών δίσκων αναπαραγωγής μουσικής. Ταυτόχρονα εκκινείτε και το προκαθορισμένο περιβάλλον εκτέλεσης που θα αναλάβει την εκτέλεση του πράκτορα, το DefaultPlace. Ο χρήστης επιλέγει από τη λίστα των υλοποιημένων πρακτόρων, αυτόν που ταιριάζει καλύτερα στις

ανάγκες του, και τον ενεργοποιεί. Ο στατικός πράκτορας ζητάει από το χρήστη να του εισάγει πληροφορίες για τα προϊόντα που επιθυμεί να αγοράσει και η αναζήτηση αρχίζει…

[Εντοπισμός]. Ο στατικός πράκτορας καλώντας την Place.getFinder() παίρνει ένα αντικείμενο, αντιπρόσωπο του συστήματος ευρετηρίου. Έτσι, μπορεί να βρει τις τοποθεσίες των υπάρχοντων οντοτήτων στην περιοχή καλώντας τις lookupAgent(), lookupPlace() και lookupAgentSystem() της κλάσης Agent. Χρησιμοποιώντας την πληροφορία αυτή θα αποφασίσει για το μέρος που θα ξεκινήσει την εκτέλεση του πράκτορα αλλά και το δρομολόγιό του.

[Δημιουργία]. Ο στατικός πράκτορας δημιουργεί ένα νέο πράκτορα χρησιμοποιώντας την createAgent(). Αν το περιβάλλον εκτέλεσης του νέου πράκτορα δεν είναι το ίδιο με αυτό του πατρικού, η αίτηση δημιουργίας προωθείται στο ξενιστή του περιβάλλοντος προορισμού. Γίνεται κλήση της create_agent() της κλάσης CfMAF.MAFAgentSystem.

Ο πράκτορας αρχίζει την εκτέλεσή του στο περιβάλλον προορισμού. Μέρος της αποστολής του είναι να συλλέξει πληροφορίες. Πηγές των πληροφοριών μπορεί να είναι άλλοι πράκτορες, το ίδιο το σύστημα πρακτόρων ή και εφαρμογές ξένες προς το περιβάλλον που υποστηρίζει τους κινούμενους πράκτορες. Όσον αφορά την αλληλεπίδραση με τρίτες εφαρμογές εξαρτώνται αποκλειστικά από την φύση των εφαρμογών αυτών.

[Αλληλεπίδραση]. Ο κινούμενος πράκτορας μπορεί να αντλήσει πληροφορίες από το περιβάλλον που τον φιλοξενεί χρησιμοποιώντας τις getAgentSystemName(), getAgentSystemDescription(), getAgentList(), getAllAgentList() κ.α. Μπορεί επιπλέον, να συνεργαστεί με άλλους πράκτορες καλώντας μεθόδους των πρακτόρων αυτών. Αυτό επιτυγχάνεται με χρήση της execMethod(). Οι μέθοδοι αυτοί παρέχονται από την κλάση του πράκτορα.

Αφού συλλέξει τις απαραίτητες πληροφορίες, ο πράκτορας μπορεί να αποφασίσει αν θα μετακομίσει σε άλλο περιβάλλον ή αν θα ειδοποιήσει το στατικό πράκτορα για τα αποτελέσματα της έρευνάς του. Αν υποθέσουμε ότι αποστέλλει πληροφορίες στο στατικό πράκτορα, το κάνει με χρήση της μεθόδου execMethod().

[Αναστολή / Επανεκκίνηση]. Ο κινούμενος πράκτορας αναστέλλει τη λειτουργία του καλώντας την suspend() και περιμένει να τον επανεκκινήσει ο στατικός πράκτορας αφού επεξεργαστεί τα δεδομένα. Ας υποθέσουμε, για τη συνέχεια του παραδείγματος, ότι ο στατικός πράκτορας επανεκκινεί τον κινούμενο πράκτορα καλώντας την resume() με όρισμα το όνομα του πράκτορα αυτού.

[Μετακίνηση]. Αφού ο πράκτορας ολοκλήρωσε το έργο του στο πρώτο περιβάλλον ετοιμάζεται να μετακομίσει στον επόμενό του προορισμό. Ο ίδιος ο πράκτορας μόνο μπορεί να αρχικοποιήσει τη διαδικασία της μεταφοράς του. Ο πράκτορας αρχικοποιεί τη μεταφορά του επικοινωνώντας με το τοπικό σύστημα πρακτόρων. Αυτό με τη σειρά του, αφού κάνει τις απαραίτητες προετοιμασίες, επικοινωνεί με το σύστημα προορισμού.

Τα συστήματα πρακτόρων που εμπλέκονται στην μετακίνηση του πράκτορα ανταλλάσσουν πληροφορίες και δεδομένα. Συγκεκριμένα, το αρχικό σύστημα καλώντας την CfMAF.MAFAgentSystem.receive_agent() αποστέλλει όλα τα απαραίτητα δεδομένα για τη μεταφορά του πράκτορα. Αν το σύστημα προορισμού χρειάζεται επιπλέον κλάσεις για την εκτέλεση του

πράκτορα, τις αιτεί κάνοντας χρήση της μεθόδου fetch_class() του αρχικού συστήματος.

Στο νέο περιβάλλον πλέον, ο πράκτορας εκτελεί το τμήμα του έργου που σχετίζεται με το περιβάλλον αυτό. Αν υποθέσουμε ότι βρίσκει τα επιθυμητά αντικείμενα, τα αγοράζει εκ μέρος του χρήστη. Ενημερώνει το στατικό πράκτορα καλώντας την execMethod() και περιμένει καινούργιες διαταγές από τον αγοραστή μέσω αυτού.

[Τερματισμός]. Ο πράκτορας τερματίζει την εκτέλεσή του χρησιμοποιώντας την μέθοδο terminate(). Μπορεί βέβαια να αναγκαστεί να τερματίσει την εκτέλεσή του αν κάποια άλλη οντότητα, που έχει τα απαραίτητα προνόμια, ζητήσει τον τερματισμό της εκτέλεσής του.

8 Συμπεράσματα και προεκτάσεις

Καθώς ο Παγκόσμιος Ιστός επεκτείνεται, η διαθέσιμη πληροφορία που είναι πολύτιμη για τον οποιοδήποτε χρήστη ολοένα και αυξάνεται. Για να επωφεληθεί ο χρήστης, όσο το δυνατό περισσότερο από την πληροφορία αυτή, απαιτείται, η τελευταία, να γίνει όσο το δυνατόν ευκολότερα προσβάσιμη, ύστερα από αίτηση. Βέβαια, πρέπει να σημειωθεί ότι απευθυνόμαστε σε χρήστες με διαφορετικό επίπεδο εμπειριών όσον αφορά την αλληλεπίδραση με τον Παγκόσμιο Ιστό. Επίσης, η χρήσιμη πληροφορία πρέπει να διαμορφώνεται σύμφωνα με τις ανάγκες του χρήστη, πριν παρουσιαστεί σε αυτόν. Είναι επιτακτική ανάγκη λοιπόν να αυτοματοποιηθεί ο μηχανισμός ανάκτησης της πληροφορίας. Η τεχνολογία των κινούμενων πρακτόρων υπόσχεται ότι μπορεί να προσφέρει την υποδομή για το νέο αυτό περιβάλλον, εύκολης πρόσβασης στην επιθυμητή πληροφορία.

Στα πλαίσια της εργασίας αυτής, ασχολήθηκα με το σχεδιασμό και την υλοποίηση ενός περιβάλλοντος κινούμενων πρακτόρων που μπορεί να στηρίξει μία τέτοια υποδομή. Το περιβάλλον EasyAgent παρέχει υλοποίηση του συστήματος πρακτόρων και του συστήματος ευρετηρίου. Η διεπαφή επικοινωνίας που χρησιμοποιούν τα συστήματα είναι αυτή που καθορίζεται από το πρότυπο OMG MASIF. Αυτό δίνει τη δυνατότητα στα παρεχόμενα συστήματα να αλληλεπιδρούν με συστήματα διαφορετικών κατασκευαστών αλλά και με τρίτες εφαρμογές. Στη διεπαφή επικοινωνίας του συστήματος πρακτόρων πρόσθεσα ένα ακόμη ορισμό, ο οποίος τυποποιεί τον τρόπο αλληλεπίδρασης των κινούμενων πρακτόρων. Συγκεκριμένα, στην περιγραφή της διεπαφής προστέθηκε μία μέθοδος που επιτρέπει τους πράκτορες να καλούν μέθοδους άλλων πρακτόρων με διαφανή τρόπο.

Το σύστημα πρακτόρων είναι ικανό να φιλοξενήσει πράκτορες τύπου EasyAgent, που μπορούν να αναπτυχθούν με χρήση της διεπαφής προγραμματισμού που παρέχει το περιβάλλον. Η κλάση, η οποία εσωκλείει το έργο του κινούμενου πράκτορα και υλοποιείται από τον εκάστοτε προγραμματιστή εφαρμογών κινούμενων πρακτόρων, είναι υποκλάση της κλάσης του Πρακτόρα, που προσφέρει η διεπαφή προγραμματισμού του EasyAgent. Έτσι, η κλάση που περιγράφει την εργασία του πράκτορα με χρήση της γλώσσας προγραμματισμού Java, εξοπλίζεται με ένα σύνολο λειτουργιών και δυνατοτήτων. Στο σημείο αυτό είναι απαραίτητο να επιστήσουμε την προσοχή μας στο γεγονός ότι στην ενσωματοποίηση αυτή μπορούμε να έχουμε και δεδομένα οποιασδήποτε μορφής (κείμενο, εικόνα, μεταδεδομένα). Μέσω της κλάσης του Πράκτορα δίνεται η δυνατότητα σε δεδομένα και κώδικα, που περιγράφει την εργασία, να μετακινούνται μεταξύ των ξενιστών που τους προσφέρουν το κατάλληλο περιβάλλον εκτέλεσης (σύστημα πρακτόρων). Επιπλεόν, τους δίνεται η δυνατότητα να αλληλεπιδρούν με οποιοδήποτε άλλο σύστημα πρακτόρων αλλά και σύστημα ευρετηρίου υλοποιεί τη διεπαφή επικοινωνίας που ορίζει το πρότυπο. Έτσι, επιτυγχάνεται η διαφάνεια στην αλληλεπίδραση μεταξύ οντοτήτων διαφορετικών κατασκευαστών.

Η αλληλεπίδραση με άλλους πράκτορες που υποστηρίζεται από το πρότυπο αφορά μόνο τις λειτουργίες διαχείρισής τους. Οι κινούμενοι πράκτορες μπορούν χρησιμοποιώντας τρόπους αλληλεπίδρασης (sockets, rmi), που υποστηρίζονται από την γλώσσα υλοποίησής τους, να συνεργάζονται με άλλους πράκτορες και εξωτερικές εφαρμογές. Έτσι, έχουν την αποκλειστική ευθύνη εντοπισμού των οντοτήτων αυτών αλλά και εγκατάστασης του καναλιού επικοινωνίας με αυτές. Το περιβάλλον EasyAgent επεκτείνοντας τη διεπαφή του συστήματος πρακτόρων δίνει επιπλέον ένα τυποποιημένο τρόπο για τη συνεργασία των οντοτήτων. Επιτρέποντας στους κινούμενους πράκτορες να εκτελούν μεθόδους άλλων πρακτόρων, μέσω του συστήματος που τους φιλοξενεί, δίνει τη βάση για την υποστήριξη οποιουδήποτε μηχανισμού επικοινωνίας των οντοτήτων.

Η επέκταση αυτή μπορεί να αποτελέσει τη βάση για τη δημιουργία ενός μηχανισμού δημοσιοποίησης, των προσφερόμενων από την εκάστοτε οντότητα, λειτουργιών (μεθόδων).

Με χρήση των ιδιαίτερων χαρακτηριστικών των οντοτήτων μπορούμε να περιγράψουμε το σύνολο των προσφερόμενων λειτουργιών. Όμως, τα ιδιαίτερα χαρακτηριστικά του πράκτορα μπορούν να χρησιμοποιηθούν και για την εύρεσή του. Οπότε, καθίστατε δυνατός ο εντοπισμός της οντότητας αλλά και η εκτέλεση του έργου που προσφέρει μέσω των μεθόδων της, στο σύστημα που την φιλοξενεί. Με τον τρόπο αυτό, η πληροφορία που μεταφέρει ο πράκτορας ή και που έχει αποκτησεί από τη μέχρι τώρα ζωή του, μπορεί να αλληλεπιδράσει με την πληροφορία που υπάρχει στο περιβάλλον μέσα στο οποιό δραστηριοποιείται. Ο νεοτερισμός είναι ότι η έναρξη της αλληλεπίδρασης αυτής δίνεται μέσω μίας τυποποιημένης λειτουργίας από οποιαδήποτε οντότητα που ανήκει ή όχι στο περιβάλλον των κινούμενων πρακτόρων.

Από την άλλη, ο ίδιος μηχανισμός μπορεί να χρησιμοποιηθεί για την ομαδοποίηση των υπηρεσιών που προσφέρουν οι κινούμενες οντότητες. Μπορούν να υλοποιηθούν πράκτορες που να προσφέρουν μία λογική συγκέντρωση συναφών λειτουργιών. Οι πράκτορες αυτοί θα έχουν την ιδιότητα να προσφέρουν διαφανή πρόσβαση σε ένα σύνολο από συγκενείς υπηρεσίες που υλοποιούνται από οντότητες διαφορετικών κατασκευαστών. Επιπλεόν, λόγω της δυνατότητας της μετακίνησης με την οποία είναι εξοπλισμένοι οι πράκτορες, μπορούν να προσφέρουν τη διεπαφή χρήσης των υπηρεσίων στον υπολογιστή του κάθε χρήστη, μετά από αίτηση του τελευταίου.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Crystaliz Inc, General Magic Inc, GMD FOKUS, IBM, TOG: OMG Joint Submission "Mobile Agent System Interoperability Facility". November 1997

[2] The Common Object Request Broker: Architecture and Specification. Revision 2.2. February 1998.

[3] Arnold, K. and Gosling, J. The Java programming language, Second Edition. Addison-Wesley. 1998.

[4] David Curtis. "Java, RMI and CORBA, A White Paper". May 1997.

[5] CORBAservices: Common Object Services Specification, Revised Edition, OMG TC Document 95-3-31.

[6] IDL Type Extensions RFP, March 1995. OMG TC Document 95-1-35.

[7] Mike Bradley. "IIOP: OMG's Internet Inter-ORB Protocol: A Brief Description". May 97.

[8] The Component Object Model Specification

[9] S. Vinoski, CORBA: Integrating diverse applications within distributed heterogeneous environments, in IEEE Communications, vol. 14, no. 2, February 1997.

[10] D. Rogerson, Inside COM, Redmond, Washington: Microsoft Press, 1996.

[11] Java, Remote Method Invocation Specification. Revision 1.50. JDK1.2. October 1998.

[12] Alan O. Freier, Philip Karlton, Paul C. Kocher. The Secure Socket Layer Protocol. Version 3.0. November 1996.

[13] David Kotz, Robert Gray, and Daniela Rus. Transportable agents support worldwide applications. In Proceedings of the Seventh ACM SIGOPS European Workshop, pages 41-48. ACM Press, September 1996.

[14] R. Rivest, A. Shamir, and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, v. 21, n. 2, pp. 120 – 126, February 1978.

[15] Chess, D. M.; Harrison, C. G. and Kershenbaum, A. "Mobile Agents: Are they a good idea?" Mobile object systems: toward the programmable Internet, Springer-Verlag, Lecture Notes in Computer Science No. 1222: 25-45. 1997.

[16] Birrell, A. D. and Nelson, B. J. Implementing remote procedure calls. ACM Transactions on Computer Systems 2(1): 39-59. 1984.

[17] ISI for DARPA, RFC 793: Transport Control Protocol, September 1981.

[18] Daniela Rus, Robert Gray, and David Kotz. Transportable information agents. In Michael Huhns and Munindar Singh, editors, Readings in Agents, chapter 3.3, pages 283-291. Morgan Kaufmann Publishers, San Francisco, October 1997.

[19] IKV++ GmbH. Grasshopper Technical Overview (Revision 1.0). November 1998.

[20] D. T. Chang, D. B. Lange, "Mobile Agents: A New Paradigm for Distributed Object Computing on the WWW", In *Proceedings of the OOPSLA96 Workshop: Toward the Integration of WWW and Distributed Object Technology*, October 1996.

[21] Mitsubishi Electric ITA, Horizon Systems Laboratory. Mobile Agent Computing, A White Paper. January 1998

[22] Giovanni Vigna (Ed.): Mobile Agents and Security 257 pp, ISBN 3-540-64792-9, Springer-Verlag, Germany, 1998

[23] IBM Aglets Workbench, D. B. Lange and D. T. Chang. Programming Mobile Agents in Java, A White Paper. September 1996.

[24] Jonathan Bredin, David Kotz, and Daniela Rus. Market-based resource control for mobile agents. Technical Report PCS-TR97-326, Dept. of Computer Science, Dartmouth College, December 1997.

[25] Joseph Kiniry and Daniel Zimmerman. "A Hands-On Look at Java Mobile Agents". IEEE Internet Computing, Vol. 1, No. 4, July - August 1997.

[26] Donald Ervin Knuth. "The Art of Computer Programming : Fundamental Algorithms", Vol 1, 3rd Ed. July 1997.

[27] Gail L. Grant. "Understanding digital signatures: Establishing trust over the Internet and other netwoks". Computing McGraw-Hill. ISBN: 0070125546. November 1997.

[28] Jalal Feghhi, Peter Williams, Jalil Feghhi. "Digital Certificates: Applied Internet Security". Addison-Wesley Pub Co. ISBN: 0201309807. October 1998.

[29] Robert Gray, David Kotz, Saurab Nog, Daniela Rus, and George Cybenko. Mobile agents: The next generation in distributed computing. In *Proceedings of the Second Aizu International Symposium on Parallel Algorithms/Architectures Synthesis (pAs '97)*, pages 8-24, Fukushima, Japan. IEEE Computer Society Press. March 1997.

[30] Robert Gray, David Kotz, Saurab Nog, Daniela Rus, and George Cybenko. Mobile agents for mobile computing. Technical Report PCS-TR96-285, Dept. of Computer Science, Dartmouth College, May 1996.

[31] "Secure Internet Programming: Security Issues for Distributed and Mobile Objects". (to appear 1999).

[32] Dave Zeltserman and Gerard Puoplo. "Building Network Management Tools With Tcl/Tk". Prentice Hall. ISBN: 0130807273. April 1998.

[33] R. Kent Dybvig and Kent Dybbig. "The Scheme Programming Language : ANSI Scheme". Prentice Hall. ISBN: 0134546466. April 1996.

[34] ObjectSpace, Inc. The ObjectSpace Voyager Core Package Technical Overview. December 1997.

[35] General Magic Inc. Mobile Agents White Paper. 1997.

[36] Bradshaw, Jeffrey (ed.) “Software Agents”. Menlo Park, California: AAAI Press / The MIT Press, 1996.

ΔΙΕΥΘΥΝΣΕΙΣ ΣΤΟ ΔΙΑΔΙΚΤΥΟ

- GeneralMagic – Odyssey: http://www.genmagic.com/technology/odyssey.html
- IKV++ GmbH – Grasshopper: http://www.ikv.de/products/grasshopper/
- Object Space – Voyager: http://www.objectspace.com/products/voyager/index.html
- Open Management Group (OMG): http://www.omg.org
- The Agent Society: http://www.agent.org
- Mitsubishi Electric's Horizon Systems Labs – Concordia: http://www.meitca.com/HSL/Projects/Concordia/
- IBM Tokyo Research Lab – Aglets, Mobile Java Agents: http://www.trl.ibm.co.jp/aglets/index.html
- The Java language: http://java.sun.com
- 4th Workshop On Mobile Object Systems: Secure Internet Mobile Computations: http://cuiwww.unige.ch/~ecoopws/ws98/index.html