ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ
ΤΟΜΕΑΣ ΟΡΓΑΝΩΣΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑ ΕΝΟΣ ΒΑΣΙΣΜΕΝΟΥ ΣΤΗΝ ΤΕΧΝΟΛΟΓΙΑ ΠΡΑΚΤΟΡΩΝ ΣΥΣΤΗΜΑΤΟΣ ΥΠΟΣΤΗΡΙΞΗΣ ΑΠΟΦΑΣΕΩΝ

Διατριβή που υπεβλήθη για την μερική ικανοποίηση των απαιτήσεων για την απόκτηση Μεταπτυχιακού Διπλώματος Ειδίκευσης υπό

ΝΙΚΟΛΑΟΣ Ι. ΣΠΑΝΟΥΔΑΚΗΣ

ΧΑΝΙΑ, 2001

Η διατριβή του Νικόλαου Ι. Σπανουδάκη εγκρίνεται:

Επίκ. Καθηγ.	Νικόλαος Ματσατσίνης
Καθηγητής	Κίμωνας Βαλαβάνης
Καθηγητής	Αναστάσιος Πουλιέζος

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΡΟΛΟΓΟΣ

Η διατριβή αυτή είναι το αποτέλεσμα μιας δουλειάς η οποία είχε την αφετηρία της σε μια καταπληκτική συγκυρία για μένα. Η γνωριμία μου με τον κ. Ματσατσίνη, εξαιρετικό άνθρωπο και επιστήμονα τον οποίο ευχαριστώ θερμά για την ευκαιρία που μου έδωσε να ασχοληθώ με το αντικείμενο της κατανεμημένης τεχνητής νοημοσύνης και της τεχνολογίας πρακτόρων, ήταν καθοριστική καθώς αυτός για πρώτη φορά με ώθησε στο αντικείμενο αυτό. Εδώ θέλω να συμπληρώσω πως ο τομέας της τεχνητής νοημοσύνης πάντα με γοήτευε, από παιδί ακόμα διάβαζα τα βιβλία του Ασίμωφ σχετικά με τα ρομπότ, και ήταν μέσα στα όνειρά μου και τις φιλοδοξίες μου το να βοηθήσω στην υλοποίηση των σχεδίων που έκανε ο τελευταίος από τις επάλξεις της επιστημονικής φαντασίας. Η γνωριμία μου, ακολούθως, με τον κ. Μωραΐτη, επίσης εξαιρετικό επιστήμονα και μέντορά μου στο πεδίο της τεχνολογίας των πρακτόρων τον οποίο ευχαριστώ για την καθοδήγησή μου στα διάφορα θέματα που ενέκυψαν, για την αμέριστη βοήθεια που πρόσφερε στα θέματα που με προβλημάτισαν, για τις ατέλειωτες συζητήσεις και την υπομονή του, συμπληρώνει την ευτυχή αυτή συγκυρία που ανέφερα προηγουμένως. Μαζί λοιπόν με τους κυρίους Ματσατσίνη και Μωραΐτη και τον κύριο Ψωματάκη θέσαμε τα θεμέλια μιας συνεργασίας, η οποία ήταν αρκετά προσοδοφόρα και ένα από τα αποτελέσματα της οποίας ήταν αυτή η εργασία.

Θέλω ακόμα να ευχαριστήσω την οικογένειά μου για την ηθική υποστήριξη που μου παρείχε, τους φίλους που με παρότρυναν στην ολοκλήρωση του έργου αυτού και τα στελέχη και τους ιθύνοντες του εργαστηρίου Σχεδιασμού και Ανάπτυξης Συστημάτων Υποστήριξης Αποφάσεων του Τμήματος Μηχανικών Παραγωγής και Διοίκησης Πολυτεχνείου Κρήτης για την υλική υποδομή και την βοήθειά τους όποτε χρειάστηκε.

Αφιερώνω την εργασία αυτή α) στην Αρχοντία, ευχόμενος να δημιουργήσω πολλά περισσότερα πράγματα στο πλάι της και β) στα θύματα της τρομοκρατικής επίθεσης στους δίδυμους πύργους της Νέας Υόρκης, ευχόμενος επιτέλους ο κόσμος να σταματήσει να γκρεμίζει και να στραφεί στη δημιουργία και το χτίσιμο ενός καλύτερου αύριο.

ΒΙΟΓΡΑΦΙΚΟ ΣΗΜΕΙΩΜΑ

Ο Νικόλαος Ι. Σπανουδάκης γεννήθηκε το Φλεβάρη του 1974 στα Χανιά Κρήτης. Σε μικρή ηλικία έζησε στην Αθήνα όπου και τέλειωσε το Δημοτικό. Γυρίζοντας στα Χανιά τέλειωσε το Γυμνάσιο και το Λύκειο. Σε αυτό το χρονικό διάστημα πήρε επίσης τα πτυχία Αγγλικών LOWER (First Certificate in English of Cambridge University) και HIGHER του PALSO όπως και βραβεύτηκε από την Μαθηματική Εταιρία. Το 1991 εισήχθηκε στο τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής του Πανεπιστημίου Πατρών. Το 1997 αποφοίτησε από την σχολή του με πολύ καλή επίδοση. Παράλληλα, στον επαγγελματικό τομέα, ασχολήθηκε με τον έλεγχο ασφάλειας κρίσιμων, όσον αφορά την ασφάλειά τους, εφαρμογών πραγματικού χρόνου (safety analysis of Safety Critical Real-Time Applications) συμμετέχοντας σε ανάλογα ESPRIT προγράμματα. Το 1997 επέστρεψε στα Χανιά, καθώς εισήχθηκε στο Μεταπτυχιακό Πρόγραμμα Σπουδών του Πολυτεχνείου Κρήτης και συγκεκριμένα στον κύκλο σπουδών με τίτλο “Οργάνωση και Διοίκηση”. Το 1999 απέκτησε το πτυχίο PROFICIENCY (CPE) του Πανεπιστημίου Cambridge. Από το 1998 έως το 2000 εργάστηκε στα ερευνητικά προγράμματα: α) DIMITRA της Commission of the European communities directorate – general for agriculture DGVI FII.3 με αριθμό συμβολαίου FAIR-PL95-844 και β) ADAPT "Επιχειρηματικός ανασχεδιασμός με χρήση τεχνολογιών πληροφορικής, ανάπτυξη εκπαιδευτικού λογισμικού για την κατάρτιση προσωπικού και ανάπτυξη τηλεργασίας” κωδικός προγράμματος: 192/251-Υπουργείο Εργασίας.

ΠΕΡΙΛΗΨΗ

Στην εργασία αυτή χρησιμοποιείται η τεχνολογία πρακτόρων για τον εκσυγχρονισμό και λειτουργία σε κατανεμημένη βάση ενός συστήματος υποστήριξης αποφάσεων. Στόχος της εργασίας είναι ο ορισμός και υλοποίηση μιας αρχιτεκτονικής πρακτόρων η οποία τους διαχωρίζει σύμφωνα με τη πολυπλοκότητά τους σε σύνθετους πράκτορες και απλούς πράκτορες ενώ σύμφωνα με τη λειτουργικότητά τους σε πράκτορες εκτέλεσης εργασιών, πράκτορες αλληλεπίδρασης με τον χρήστη και πράκτορες πληροφοριοδότες. Στην ίδια διάσταση, της αρχιτεκτονικής, μελετάται το θέμα της οργάνωσης της κοινωνίας των πρακτόρων. Στόχος της εργασίας επίσης είναι ο σχεδιασμός και υλοποίηση του τρόπου επικοινωνίας των πρακτόρων μέσω του διαδικτύου με έναν μηχανισμό ανταλλαγής μηνυμάτων.

1ο ΚΕΦΑΛΑΙΟ

ΕΙΣΑΓΩΓΗ

Η προσομοίωση της στρατηγικής της διείσδυσης ενός προϊόντος σε μια αγορά είναι μια πολύπλοκη κατανεμημένη διαδικασία υποστήριξης αποφάσεων, η οποία περιλαμβάνει πολλούς δράστες, οι οποίοι, με τη σειρά τους, ανήκουν σε διαφορετικά επίπεδα ευθύνης και επιτελούν συμπληρωματικές λειτουργίες μέσα σε έναν οργανισμό. Πολλές δουλειές έχουν προταθεί, οι οποίες υπάρχουν στη βιβλιογραφία υποστήριξης αποφάσεων στο μάρκετινγκ Kotler (1994), Liberatore και Stylianou (1995), Van Bruggen (1992), Wierenga (1992) και παρουσιάζουν διαφορετικές προσεγγίσεις στην υποστήριξη της διαδικασίας ανάπτυξης προϊόντων. Μέσα σε αυτό το πλαίσιο κινείται και μια πρωτότυπη βασισμένη στον καταναλωτή μεθοδολογία την οποία παρουσιάζουν οι Matsatsinis και Siskos (1999). Όλες αυτές οι προσεγγίσεις χρησιμοποίησαν διαφορετικές τεχνικές από την τεχνολογία της πληροφορίας, όπως έμπειρα συστήματα διοίκησης (Management Expert Systems), ευφυή συστήματα υποστήριξης αποφάσεων (Intelligent Decision Support Systems), κλπ., για να υλοποιήσουν τις μεθοδολογίες τους. Παρόλα αυτά σε καμία από αυτές τις δουλειές δεν δοκιμάστηκε η κατανεμημένη διάσταση της διαδικασίας ανάπτυξης προϊόντων σε έναν πραγματικό κόσμο. Ο τελικός στόχος της διαδικασίας αυτής σε έναν οργανισμό είναι να βρεθεί η καταλληλότερη στρατηγική της διείσδυσης ενός νέου προϊόντος (προϊόν υπό σχεδίαση) σε μια αγορά και ο στόχος αυτός θα επιτευχθεί καλύτερα αν οι απόψεις όλων των αποφασιζόντων ληφθούν υπόψη και αν αυτός είναι όσο το δυνατό πιο κοντά στις προτιμήσεις των καταναλωτών. Για την επίτευξη του στόχου αυτού κρίσιμος είναι ο ρόλος εργασιών εύρεσης, πρόσβασης, φιλτραρίσματος και

ενσωμάτωσης πηγών πληροφορίας οι οποίες θα υποστηρίξουν μια συμπαγή και ταυτόχρονα κατανεμημένη διαδικασία λήψης απόφασης.

Σύμφωνα με αυτή την εργασία θα χρησιμοποιηθούν επαναχρησιμοποιήσιμοι ευφυείς πράκτορες λογισμικού (intelligent software agents) για την υλοποίηση της μεθοδολογίας που προτείνουν οι Matsatsinis και Siskos (1999) σε τέτοιο περιεχόμενο όπου δίνεται έμφαση στη διάσταση του πραγματικού κόσμου. Αυτή η χρήση της τεχνολογίας πρακτόρων για την υλοποίηση μιας πολυκριτήριας ολοκληρωμένης μεθοδολογίας υποστήριξης αποφάσεων στο μάρκετινγκ είναι και η πρώτη σημαντική καινοτομία της εργασίας αυτής.

Για την επιτυχή υλοποίηση του στόχου που τέθηκε προηγουμένως χρειάζονται τρεις κυρίως εργασίες να έρθουν σε πέρας. Η πρώτη εργασία συνιστά την επιλογή και εφαρμογή κατάλληλης μεθοδολογίας ανάπτυξης του συστήματος πολλαπλών πρακτόρων. Η δεύτερη αφορά την ανάπτυξη μιας πλατφόρμας υποστήριξης εύκολης ανάπτυξης συστημάτων πολλαπλών πρακτόρων (ΣΠΠ) όσον αφορά την αρχιτεκτονική των πρακτόρων και τον τρόπο ανταλλαγής μηνυμάτων και είναι το κυρίως αντικείμενο της εργασίας αυτής. Η τρίτη αναγκαία εργασία αφορά την κατάλληλη παραμετροποίηση των πρακτόρων, την ανάπτυξη κατάλληλων βιβλιοθηκών σχεδίων και μεθόδων ώστε να δημιουργηθούν οι κατάλληλοι πράκτορες για την λύση του συγκεκριμένου προβλήματος. Η τρίτη εργασία αυτή μαζί με κάποια θέματα της πρώτης είναι το αντικείμενο της διατριβής του κ. Ψωματάκη (2001).

Η εφαρμογή όλων των χαρακτηριστικών εξεταζόμενων στη λογοτεχνία Jennings et al. (1996a), Sycara και Zeng (1996) ως αναγκαίων για τη χρήση τεχνολογίας πρακτόρων, οδήγησε προς μια προσανατολισμένη στους πράκτορες προσέγγιση. Αυτά τα χαρακτηριστικά είναι:

α) η φυσική κατανομή των δυνατοτήτων επίλυσης προβλήματος (οι πράκτορες εκτελούν ανάλυση δεδομένων, μεθόδους ανάλυσης συμπεριφοράς καταναλωτών και πολυκριτήριες μεθόδους ανάλυσης, δηλαδή εργασίες ανεξάρτητες μεταξύ τους), των χρησιμοποιούμενων δεδομένων και πληροφοριών,

β) οι ανάγκες ευελιξίας (ταυτόχρονη εκτέλεση πολλών εργασιών διαφορετικής φύσης και διαθεσιμότητα του συστήματος στο παγκόσμιο δίκτυο), τμηματικότητας (οι πράκτορες μπορούν να εμφανιστούν και να εξαφανιστούν στο σύστημα χωρίς διατάραξη της λειτουργίας του) και επαναχρησιμοποίησης (προσαρμογή των πρακτόρων σε νέους αποφασίζοντες) και

γ) η πολυπλοκότητα επίλυσης προβλήματος που περιλαμβάνει το συντονισμό μεταξύ των δραστών που εκφράζουν τις διαφορετικές απόψεις.

Στην εργασία αυτή οι πράκτορες εξετάζονται ταυτόχρονα σε σχέση με δύο διαφορετικά επίπεδα: ένα λειτουργικό επίπεδο και ένα δομικό επίπεδο. Στο λειτουργικό επίπεδο, έχουμε μια φυσική διάκριση μεταξύ τριών διαφορετικών λειτουργιών των πρακτόρων: λειτουργία συλλογής πληροφοριών, η εκπλήρωση εργασιών από τους διαφορετικούς τύπους συνεργαζόμενων ειδικών και η μεσολάβηση μεταξύ των χρηστών και των τεχνητών πρακτόρων, προκειμένου να επιτραπούν οι χρήστες να ελέγξουν τις ενέργειες των πρακτόρων τους. Επομένως, εξετάζουμε τρεις τύπους πρακτόρων (Sycara και Zeng, 1996) πράκτορες διεπαφών (interface agents), πράκτορες πληροφοριών (information agents) και πράκτορες έργου (task agents).

Στο δομικό επίπεδο οι πράκτορες θεωρούνται ως στοιχειώδεις πράκτορες (elementary agents) και σύνθετοι πράκτορες (όρος: complex agents), η οποία θεώρηση είναι η δεύτερη σημαντικότερη ερευνητική καινοτομία αυτής της εργασίας. Σύμφωνα με αυτή την θεώρηση, οι σύνθετες εργασίες μπορούν να αποσυντεθούν με έναν επαναλαμβανόμενο τρόπο σε διάφορες στοιχειώδεις εργασίες. Με παρόμοιο τρόπο μια δομή πρακτόρων μπορεί να εξεταστεί σύμφωνα με τα διαφορετικά τοποθετημένα στρώματα που δημιουργούνται με έναν επαναλαμβανόμενο τρόπο. Κάποιος μπορεί να φανταστεί έναν σύνθετο πράκτορα ως μια «ρωσική κούκλα». Τα στρώματα του πράκτορα συσχετίζονται με τα υποέργα που πραγματοποιούνται. Αυτή η σύλληψη εμπνέεται από την αντιπροσώπευση ενός σύνθετου συστήματος μέσω των πολλαπλάσιων στρωμάτων προτεινόμενων στη θεωρία ελέγχου (Mesarovic et al., 1970). Επομένως, ένας πράκτορας θεωρείται ως σύνθετος όταν

πραγματοποιεί έναν στόχο (π.χ. μια παραγωγή σεναρίου) που περιλαμβάνει διάφορους πράκτορες τουλάχιστον ενός χαμηλότερου στρώματος. Ένας πράκτορας θεωρείται ως στοιχειώδης, εάν πραγματοποιεί έναν στοιχειώδη στόχο. Βλέποντάς τη από μεθοδολογική άποψη, αυτή η εκτίμηση διευκολύνει τη σύλληψη και την ανάπτυξη σύνθετων συστημάτων με χρήση πολλαπλών ευφυών πρακτόρων λογισμικού, προκειμένου να επιτευχθεί η διαμόρφωση των σύνθετων πραγματικών εφαρμογών όπως αυτή που παρουσιάζουμε σε αυτό το έγγραφο.

1.1. Προβληματική

Το πρόβλημα που λύνεται σε αυτή την εργασία και την παράλληλη του κ. Ψωματάκη (2001) είναι η σύλληψη, ο σχεδιασμός και η υλοποίηση ενός συστήματος πολλαπλών πρακτόρων το οποίο θα υλοποιεί με κατανεμημένο τρόπο την μεθοδολογία των Ματσατσίνη και Σίσκου (1999). Ο Gasser (1991) παρουσίασε τα παρακάτω έξι προβλήματα, οι λύσεις των οποίων είναι απαραίτητες αν κάποιος θέλει να σχεδιάσει και να υλοποιήσει ένα ΣΠΠ:

1) Πως θα διατυπωθεί, θα περιγραφεί, θα χωριστεί σε κομμάτια το πρόβλημα, πως αυτά τα κομμάτια θα ανατεθούν στους ευφυείς πράκτορες (ΕΠ) και πως θα συντεθούν τα αποτελέσματα των εργασιών των τελευταίων;

2) Πως οι ΕΠ θα επικοινωνούν και θα αλληλεπιδρούν, τι επικοινωνιακές γλώσσες και πρωτόκολλα θα χρησιμοποιηθούν, πότε θα επικοινωνούν, τι πληροφορίες θα ανταλλάσσουν;

3) Πως θα εξασφαλιστεί η συλλογικότητα στις αποφάσεις που θα πάρουν οι ΕΠ, συμβιβάζοντας τις μη τοπικές επιδράσεις τοπικών αποφάσεων και αποφεύγοντας επιβλαβής αλληλεπιδράσεις;

4) Πως οι ΕΠ θα μπορέσουν να παρουσιάσουν και να κατανοήσουν δράσεις, σχέδια και τη γνώση των άλλων πρακτόρων ώστε να συγχρονιστούν μεταξύ τους, πως θα κατανοούν σε πιο στάδιο της συγχρονισμένης αυτής διεργασίας βρίσκονται (π.χ. έναρξη, ολοκλήρωση κλπ);

5) Πως θα αναγνωριστούν και θα συμβιβαστούν οι ανόμοιες απόψεις για το περιβάλλον που μπορεί να έχουν οι ΕΠ (π.χ. ένας ΕΠ έχει μόνο όραση ενώ κάποιος άλλος έχει μόνο ακοή) όπως και οι τυχόν συγκρουόμενες προθέσεις τους (π.χ. θέλουν τον ίδιο πόρο), ώστε να καταφέρουν να συγχρονίσουν τις δράσεις τους;

6) Πως θα σχεδιαστούν και θα περιοριστούν πρακτικά ΣΠΠ; Πως θα σχεδιαστούν τεχνολογικές πλατφόρμες και μεθοδολογίες ανάπτυξης τέτοιων συστημάτων;

Η εργασία αυτή μαζί με την παράλληλη εργασία του κ. Ψωματάκη απαντούν σε όλα τα παραπάνω θέματα. Για να απαντηθούν τα ερωτήματα αυτά χρειάστηκε η επισταμένη μελέτη του προβλήματος που καλείται να λύσει το ΣΠΠ όπως και η μελέτη αρκετών σύγχρονων αλλά και παλιότερων state of the art αρχιτεκτονικών ΣΠΠ. Ακόμα λήφθηκε υπόψη η μέχρι τώρα δουλειά της OMG στην τεχνολογία πρακτόρων (Agent Working Group, 2000) Έτσι τα διάφορα στάδια της μεθοδολογίας αναλύθηκαν, χωρίστηκαν σε εργασίες οι οποίες είναι ανεξάρτητες μεταξύ τους και σύμφωνα με κριτήρια που θεσπίστηκαν αναγνωρίστηκαν οι πράκτορες που θα συμμετείχαν στο σύστημα.

Τα ερευνητικά θέματα που απασχόλησαν τον συγγραφέα και στα οποία προτείνει σημαντικές καινοτομίες είναι:

- Ο σχεδιασμός και η ανάπτυξη μιας αρχιτεκτονικής και πλατφόρμας για την εύκολη ανάπτυξη συστημάτων πολλαπλών πρακτόρων στο μέλλον
- Ο συντονισμός και η οργάνωση της ομάδας των πρακτόρων, τα οποία θα είναι επίσης ανεξάρτητα τομέα εφαρμογής του συστήματος

Και οι δύο στόχοι αυτοί καλύφθηκαν ικανοποιητικά καθώς αναπτύχθηκε μια πλατφόρμα ανάπτυξης συστημάτων πολλαπλών πρακτόρων η οποία είναι ανεξάρτητη της εφαρμογής του συστήματος και επίσης ανεξάρτητης των τμημάτων που θα χρειαστεί να χρησιμοποιηθούν για την υλοποίηση κάθε πράκτορα (μπορεί να υποστηρίξει απλούς-αντιδραστικούς ή πολύπλοκους-κοινωνικούς πράκτορες). Ακόμα με την ιδέα του πολύπλοκου πράκτορα και

την ανάπτυξη του κατάλληλου πρωτοκόλλου αλληλεπίδρασης των πρακτόρων λύθηκε και το θέμα του συντονισμού με την δημιουργία δυναμικών ιεραρχικών δομών ελέγχου, οι οποίες δομές προβλέπουν πράκτορες υπεύθυνους για την αποσύνθεση της εργασίας σε εργασίες χαμηλότερου επιπέδου επαναληπτικά μέχρι να απαιτείται η ολοκλήρωση στοιχειωδών εργασιών και την ακόλουθη σύνθεση των αποτελεσμάτων των εκτελέσεων των στοιχειωδών εργασιών πάλι ανά επίπεδο ελέγχου.

1.2. Οργάνωση της διατριβής

Στο δεύτερο κεφάλαιο παρουσιάζονται θεωρητικά όλες οι πτυχές ανάπτυξης ενός ΣΠΠ, πολλές υπάρχουσες πλατφόρμες ανάπτυξης ΣΠΠ και τέλος οι πιο σύγχρονες μεθοδολογίες ανάπτυξης ΣΠΠ. Το κεφάλαιο αρχίζει με μια συνοπτική παρουσίαση της μεθοδολογίας στρατηγικής διείσδυσης ενός προϊόντος σε μια αγορά (καθώς δεν είναι μέσα στους στόχους της διατριβής αυτής). Ακολουθούν οι πτυχές ανάπτυξης ενός ΣΠΠ, οι οποίες ήταν απαραίτητο να μελετηθούν ώστε να μπορέσουμε να λυθούν όλα τα τεχνικά και τεχνολογικά θέματα που ανακύπτουν κατά τη διάρκεια ανάπτυξης ενός ΣΠΠ. Οι υπάρχουσες πλατφόρμες που μελετήθηκαν για την αναγνώριση των χαρακτηριστικών τα οποία συνθέτουν μια πετυχημένη πλατφόρμα ανάπτυξης ΣΠΠ παρουσιάζονται στην επόμενη παράγραφο του κεφαλαίου. Το κεφάλαιο κλείνει με την παρουσίαση των πιο ανταγωνιστικών μεθοδολογιών ανάπτυξης ΣΠΠ ώστε να αναπτυχθεί με σωστά και σίγουρα βήματα ο διττός στόχος της εργασίας αυτής: α) μια πλατφόρμα ανάπτυξης ΣΠΠ, β) με εφαρμογή στο στρατηγικό μάρκετινγκ και συγκεκριμένα στην επιλογή στρατηγικής διείσδυσης ενός προϊόντος σε μια αγορά.

Στο τρίτο κεφάλαιο περιγράφεται η διαδικασία σχεδίασης του προτεινόμενου ΣΠΠ. Αρχικά παρουσιάζεται η μεθοδολογία ανάπτυξης που χρησιμοποιήθηκε ακολουθούμενη από τον σχεδιασμό του ΣΠΠ. Ακολούθως περιγράφεται η αρχιτεκτονική των στοιχειωδών και πολύπλοκων πρακτόρων καθώς και ο τρόπος επικοινωνίας των πρακτόρων (πρωτόκολλα, τρόπος ανταλλαγής

μηνυμάτων). Το κεφάλαιο κλείνει με ένα σενάριο αρχικοποίησης του προτεινόμενου ΣΠΠ.

Το τέταρτο κεφάλαιο παρουσιάζει θέματα υλοποίησης των σχεδίων που παρουσιάστηκαν στο προηγούμενο κεφάλαιο. Τέτοια θέματα είναι η γλώσσα προγραμματισμού και οι τεχνολογίες που χρησιμοποιήθηκαν ακολουθούμενα από την περιγραφή της οργάνωσης του έργου.

Τα συμπεράσματα και οι μελλοντική εργασία του συγγραφέα της διατριβής αυτής παρουσιάζονται στο πέμπτο κεφάλαιο. Ακολουθούν η βιβλιογραφία και τα παραρτήματα:

- συντομογραφίες που χρησιμοποιούνται σε όλη την έκταση αυτού του εγγράφου,
- ευρετήρια των εικόνων και των πινάκων του εγγράφου.

2ο ΚΕΦΑΛΑΙΟ

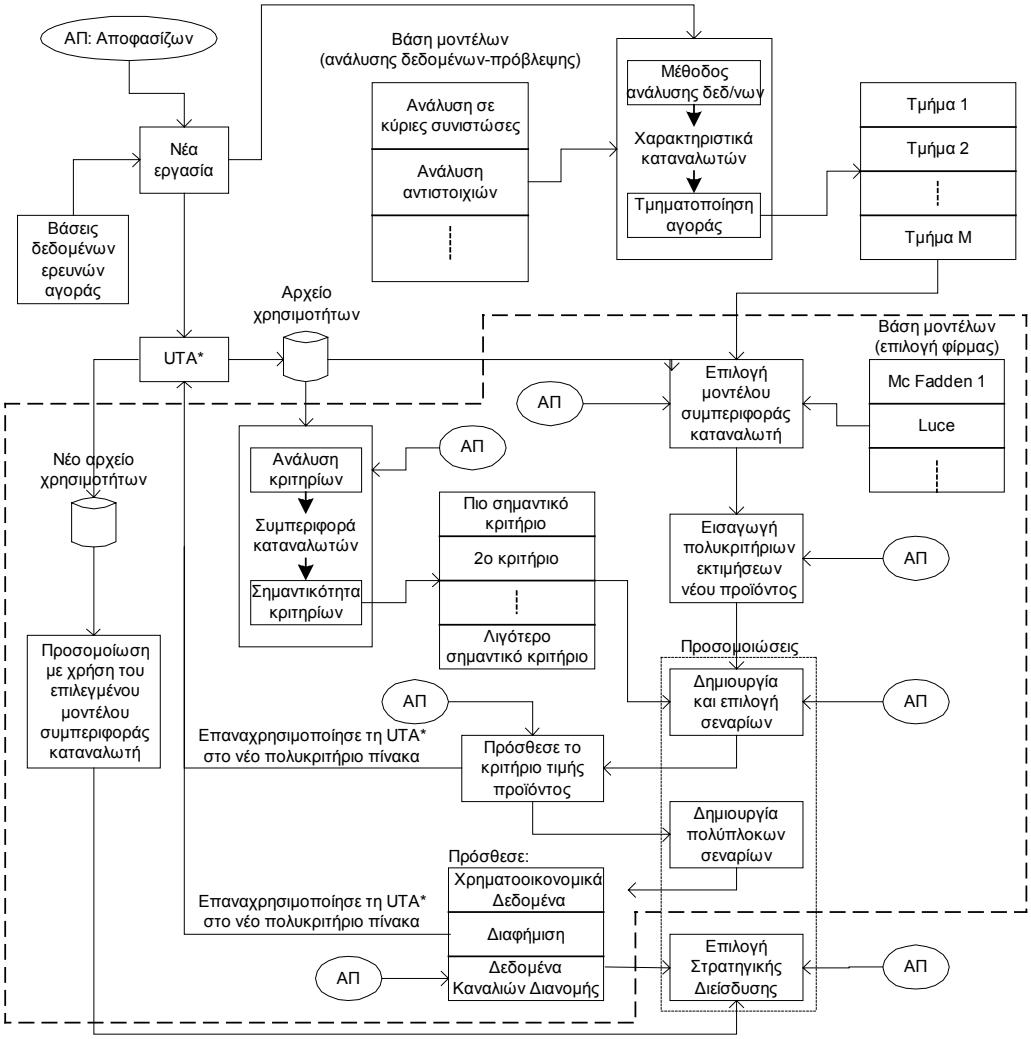
ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Στα πλαίσια αναζήτησης του τρόπου ανάπτυξης του προτεινόμενου ΣΠΠ μελετήθηκε καταρχήν η μεθοδολογία επιλογής της στρατηγικής διείσδυσης ενός προϊόντος στην αγορά. Μετά μελετήθηκαν τα διάφορα θέματα-πτυχές που αφορούν την ανάπτυξη ενός ΣΠΠ. Σημαντική βοήθεια στην αναγνώριση και κατηγοριοποίηση αυτών των θεμάτων προσέφερε η δουλειά των Moulin και Chaib-draa (1996), όπως και η πιο πρόσφατη των Jennings, Sycara και Wooldridge (1998). Ακολούθως παρουσιάζονται διάφορες πλατφόρμες και αρχιτεκτονικές ανάπτυξης ΣΠΠ. Τέλος μελετήθηκαν οι διάφορες μεθοδολογίες ανάπτυξης ΣΠΠ. Έτσι αφού κατανοήθηκαν οι εργασίες που θα εκτελεί το προτεινόμενο ΣΠΠ, μελετήθηκαν οι επιλογές που υπήρχαν στην βιβλιογραφία στα διάφορα θέματα που αφορούν την ανάπτυξη ΣΠΠ, προστέθηκαν καινοτομίες και τέλος αναπτύχθηκε το ΣΠΠ σύμφωνα με μια πρωτότυπη μεθοδολογία. Οι παράγραφοι που ακολουθούν άπτονται όλων αυτών των θεμάτων και τα περιγράφουν σε βάθος.

2.1. Η Βασισμένη στον Καταναλωτή Μεθοδολογία για την Επιλογή Στρατηγικής Διείσδυσης Προϊόντων

Για να υποστηριχθεί η διαδικασία ανάπτυξης προϊόντων οι Matsatsinis και Siskos (1999) πρότειναν μια πρωτότυπη βασισμένη στον καταναλωτή μεθοδολογία (Εικόνα 2-1). Η μεθοδολογία αυτή είναι βασισμένη στη χρήση διαφορετικών μοντέλων ανάλυσης δεδομένων, πολυκριτήριας ανάλυσης και προσωπικής επιλογής προϊόντων από τους καταναλωτές. Κατά τη διάρκεια της

έρευνας αγοράς, κάθε καταναλωτής εκφράζει τις εκτιμήσεις του σε ένα σύνολο προϊόντων αναφοράς που περιλαμβάνονται στην έρευνα, για τη βάση μιας ομάδας κριτηρίων. Τέλος, καλείται να ταξινομήσει τα προϊόντα σύμφωνα με τη σειρά προτίμησής τους από αυτόν. Η συλλογή αυτού του είδους στοιχείων απαιτεί ένα συγκεκριμένο ερωτηματολόγιο.



Εικόνα 2-1: Το μεθοδολογικό διάγραμμα ροής (Matsatsinis και Siskos, 1999)

Η αρχική φάση αυτής της μεθοδολογίας στοχεύει να αποκτήσει μια γενική εικόνα της έρευνας αγοράς. Αυτό πετυχαίνεται με τη χρήση μεθόδων

περιγραφικής στατιστικής και απλών μοντέλων πρόβλεψης (αν υπάρχουν ιστορικά στοιχεία πωλήσεων για τις εταιρίες που συμμετέχουν στην έρευνα αγοράς).

Ακολουθεί η τμηματοποίηση της αγοράς και η εξεύρεση της αγοράς στόχου για το προϊόν υπό ανάπτυξη. Αυτό επιτυγχάνεται αφενός με τη χρήση μεθόδων ανάλυσης δεδομένων και αφετέρου με την ανάλυση κριτηρίων. Σύμφωνα με την τελευταία, αφού τρέξει η μέθοδος UTA* (Siskos και Yannacopoulos, 1985) μπορούν να ταξινομηθούν οι καταναλωτές σύμφωνα με τη βαρύτητα που δίνουν σε κάποια κριτήρια (δηλαδή σύμφωνα με τις πολυκριτήριες εκτιμήσεις που έκαναν στα προϊόντα).

Ύστερα από την επιλογή της αγοράς στόχου πρέπει να προσδιοριστεί το μοντέλο επιλογής φίρμας (brand choice model) που εκφράζει αυτή την αγορά για να γίνουν προσομοιώσεις. Στη διάρκεια των προσομοιώσεων μπορούν να αγνοηθούν προϊόντα από την αγορά ή να δημιουργηθεί ένα νέο προϊόν.

Ύστερα από την επιλογή του μοντέλου συμπεριφοράς των καταναλωτών έρχεται η δημιουργία σεναρίων διείσδυσης. Αυτά μπορούν να είναι απλά ή σύνθετα σενάρια στα οποία ο αποφασίζων μπορεί να επηρεάζει τις πολυκριτήριες εκτιμήσεις των καταναλωτών ύστερα από υποθετικές κινήσεις του ή υποθετικές κινήσεις των ανταγωνιστών.

Τέλος ο αποφασίζοντας μπορεί να επιλέξει τη στρατηγική διείσδυσης του νέου προϊόντος του αφού εισαγάγει πληροφορίες ειδικών, όπως εκτιμήσεις για τα κανάλια διανομής του κάθε προϊόντος, την χρηματοοικονομική κατάσταση των εταιριών και την διαφήμιση, ξαναεκτελέσει τη UTA* με τα νέα κριτήρια αυτά και τελικά να εμπλουτίσει με αυτά τα εναλλακτικά σενάρια.

2.2. Οι Πτυχές Ανάπτυξης ΣΠΠ

2.2.1. Κατηγορίες Πρακτόρων

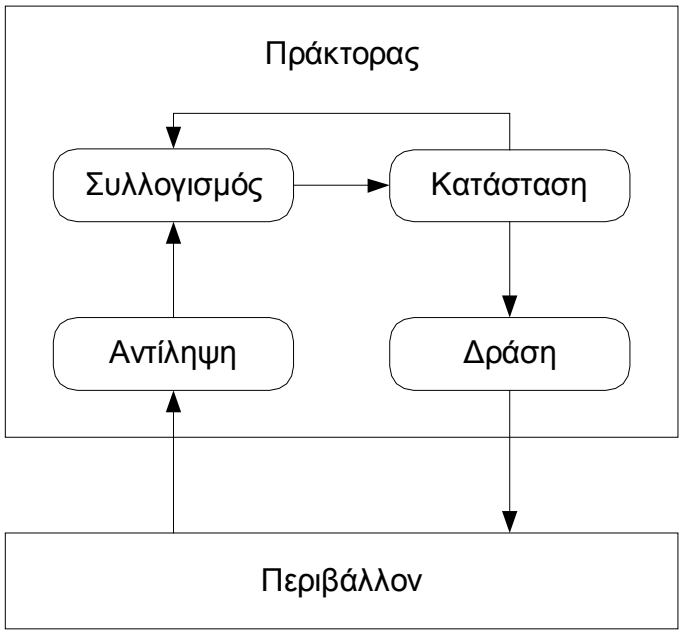
Οι ΕΠ χωρίζονται σε δύο βασικές κατηγορίες (Μωραΐτης, 2001) ανάλογα με την αντίληψη που έχουν για τον κόσμο και τους άλλους πράκτορες. Η πρώτη κατηγορία είναι οι πράκτορες που αντιδρούν σε ερεθίσματα (reactive) και η δεύτερη περιλαμβάνει αυτούς οι οποίοι διαθέτουν την ικανότητα «σκέψης», είναι διαβουλευτικοί (deliberative). Οι Moulin και Chaib-Draa (1996) παλιότερα χώριζαν τους ΕΠ σε κατηγορίες ανάλογα με την ικανότητά τους να λύνουν προβλήματα σε πράκτορες που αντιδρούν σε ερεθίσματα (reactive agents), σε πράκτορες που έχουν προθέσεις (intentional agents) και σε κοινωνικούς πράκτορες (social agents). Οι δύο τελευταίοι ανήκουν στους σύγχρονους διαβουλευτικούς πράκτορες, αλλά αξίζει να αναφερθούν αυτές οι κατηγορίες αναλυτικά:

- Ένας *πράκτορας που αντιδρά σε ερεθίσματα* ουσιαστικά αντιδρά σε αλλαγές στο περιβάλλον του ή σε μηνύματα άλλων πρακτόρων. Οι δράσεις του είναι αποτελέσματα των ερεθισμάτων που δέχεται και μόνο (δηλαδή ο πράκτορας αυτός είναι αυτόματο). Σε ένα ΣΠΠ αυτό το είδος του πράκτορα μπορεί να στέλνει ή να δέχεται μηνύματα ανάλογα με την τρέχουσα κατάσταση (θα μπορούσε να είναι ένας πράκτορας που παρακολουθεί μια τοποθεσία και εκπέμπει τις αλλαγές που παρατηρεί, ή ειδοποιεί για την παρουσία κάποιου στην τοποθεσία αυτή).
- Ένας *πράκτορας που έχει προθέσεις* είναι ικανός να κρίνει τις προθέσεις του και τις πεποιθήσεις του, να σχεδιάσει τις δράσεις του και να πραγματοποιήσει τα σχέδιά του. Σε ένα ΣΠΠ αυτό το είδος του πράκτορα συγχρονίζεται με τους άλλους πράκτορες ανταλλάσσοντας πληροφορίες σχετικές με τις πεποιθήσεις τους, τους στόχους τους ή τις δράσεις τους. Αυτές οι πληροφορίες τελικά ενσωματώνονται στα σχέδια του πράκτορα.

- Ένας *κοινωνικός πράκτορας,* εκτός από τις ικανότητες των πρακτόρων με προθέσεις, έχει την ικανότητα να γνωρίζει σαφώς τα μοντέλα των άλλων πρακτόρων. Έτσι ο κοινωνικός πράκτορας πρέπει να διατηρεί τα μοντέλα (ενημερώνεται συνέχεια για τις προθέσεις, τους στόχους και τελικά τα σχέδια του κάθε πράκτορα) και να κρίνει τη γνώση που αυτά περιέχουν (προθέσεις, υποχρεώσεις, προσδοκίες, αναμενόμενες αντιδράσεις και υποθετικές συμπεριφορές), ώστε να μπορεί να πάρει τις αποφάσεις του και να σχεδιάσει τις δράσεις του ανάλογα με τα μοντέλα αυτά.

2.2.2. Αρχιτεκτονική ΕΠ

Ο Μωραΐτης (2001) προτείνει την αρχιτεκτονική διαβουλευτικού πράκτορα, η οποία παρατίθεται στην Εικόνα 2-2. Η αρχιτεκτονική του reactive πράκτορα είναι παρόμοια αλλά δεν περιέχει τις διεργασίες «συλλογισμός» και «Κατάσταση».



Εικόνα 2-2: Αρχιτεκτονική Διαβουλευτικού Πράκτορα

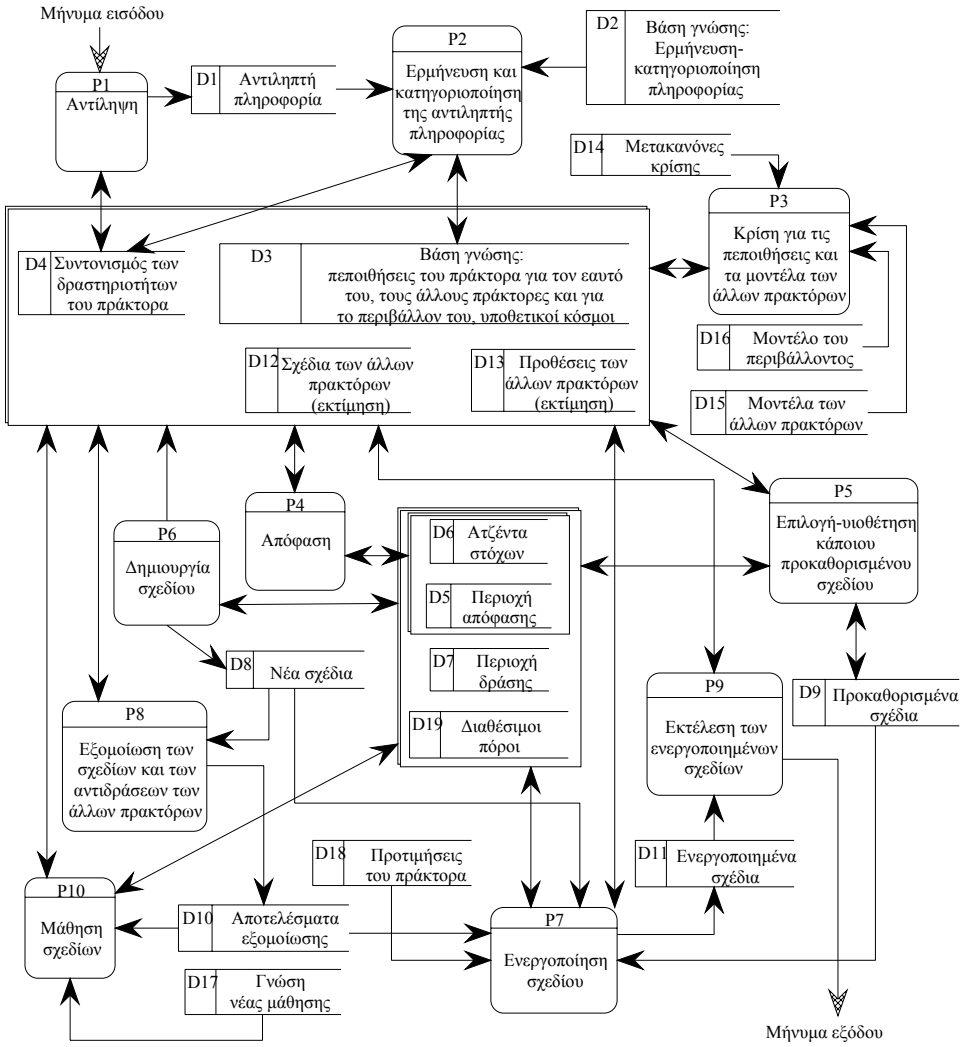
Σύμφωνα με τον Μωραΐτη η εσωτερική δομή ενός πράκτορα περιλαμβάνει:

- Δομές δεδομένων

- Λειτουργίες
- Ροή ελέγχου και δεδομένων μεταξύ των διεργασιών

Στην Εικόνα 2-3 φαίνεται η γενική αρχιτεκτονική του πιο πολύπλοκου πράκτορα, του κοινωνικού πράκτορα, σύμφωνα με τους Moulin και Chaib-draa (1996). Η αρχιτεκτονική εκφράζεται σαν ένα διάγραμμα ροής δεδομένων (ΔΡΔ). Οι αρχιτεκτονικές των πρακτόρων που διερευνήθηκαν σε αυτή την εργασία είναι υποσύνολα αυτής. Διαφοροποιούνται στο πόσο περίπλοκος είναι κάθε φορά ο πράκτορας, στο πόσες από τις διεργασίες του αφορούν ξεχωριστά νήματα εκτέλεσης και στο πως υλοποιείται κάθε φορά η αρχιτεκτονική αυτή. Καθώς φαίνεται από την Εικόνα 2-3 η λειτουργία του πράκτορα αποτελείται από κάποιες διεργασίες (Pν) οι οποίες έχουν σαν είσοδο και σαν έξοδο διάφορες πηγές δεδομένων (Dν).

Η λεπτομερής περιγραφή του ΔΡΔ αυτού είναι έξω από τους στόχους της εργασίας αυτής, όμως το παραπάνω μοντέλο προτείνεται σαν μια πλατφόρμα για τη συζήτηση των βασικών θεμάτων που άπτονται ενός ΣΠΠ. Για παράδειγμα ένα σύστημα εμπειρογνώμονα βασισμένο σε κανόνες θα μπορούσε να θεωρηθεί σαν ένας πράκτορας που αντιδρά σε ερεθίσματα και η αρχιτεκτονική του θα περιελάμβανε σε απλές εκδόσεις τις διεργασίες: Οι διεργασίες αντίληψη (P1) και ερμηνεία και κατηγοριοποίηση της αντιληπτής πληροφορίας (P2) αντιστοιχούν στην απόκτηση νέων γεγονότων για τη βάση γεγονότων (D3). Η κρίση των πεποιθήσεων του πράκτορα και των μοντέλων των άλλων πρακτόρων (P3) χρησιμοποιείται μόνο για τη συμπερασματική δημιουργία νέων γεγονότων, η απόφαση (P4) αντιστοιχεί στην επιλογή κάποιας υπόθεσης ή γεγονότος το οποίο πρέπει να επιδειχτεί. Ακολουθεί η επιλογή και εκτέλεση των σχετικών κανόνων (αυτό το κάνουν απλοποιημένες εκδόσεις των P5, P7 και P9 διεργασιών), το οποίο σημαίνει ότι ενημερώνεται η βάση γεγονότων (D3) και στέλνονται τα κατάλληλα μηνύματα προς τους άλλους πράκτορες.

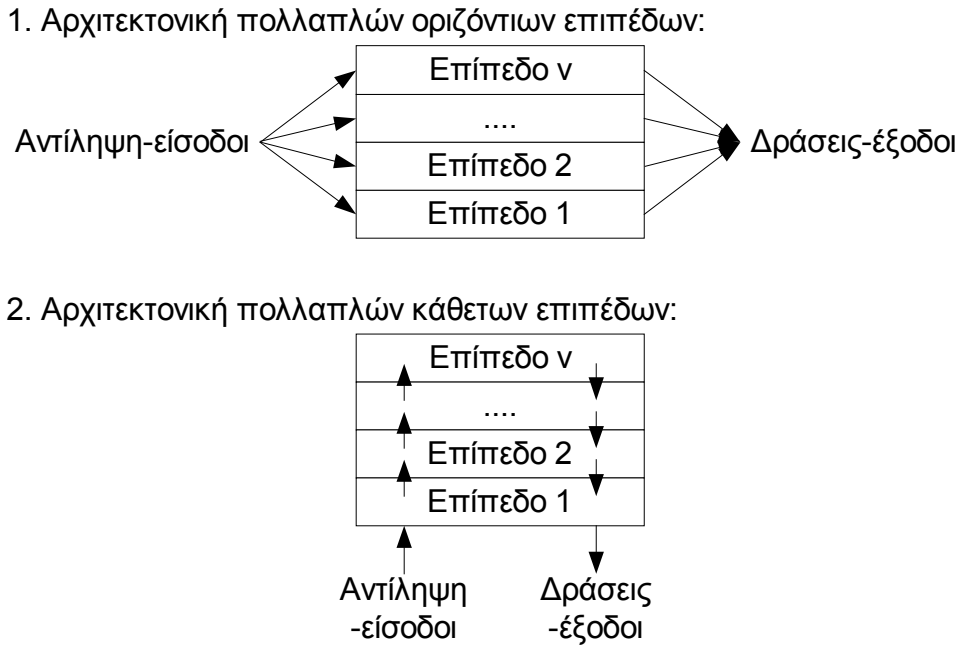


Εικόνα 2-3: Γενική αρχιτεκτονική ενός κοινωνικού πράκτορα (Moulin και Chaib-draa, 1996)

Πολλές αρχιτεκτονικές υιοθέτησαν συστήματα μαυροπίνακα (blackboard systems) (Nii, 1986). Για το project DVMT οι Lesser και Corkill (1983) χρησιμοποίησαν δύο μαυροπίνακες (έναν για τα δεδομένα και έναν για τους στόχους των πρακτόρων). Οι μαυροπίνακες παρουσιάζονται με λεπτομέρεια στην παράγραφο 2.2.5 καθώς ανήκουν στο θέμα της αλληλεπίδρασης των πρακτόρων περισσότερο παρά στην αρχιτεκτονική τους. Στο project MINDS οι Huhns et al. (1987) χρησιμοποίησαν επίσης δύο ειδικευμένους μαυροπίνακες.

Σε άλλες δουλειές, οι τύποι των πρακτόρων συνέθεταν την αρχιτεκτονική του συστήματος (σε αυτή την περίπτωση δεν έχουμε αυτόνομους πράκτορες αλλά κατανεμημένους λυτές προβλημάτων). Έτσι, η Hayes-Roth και οι συνεργάτες της (1989) πρότειναν μια αρχιτεκτονική η οποία συντίθεντο από τρεις αλληλεπιδρώντες υποπράκτορες (subagents): τον πράκτορα αντίληψης, τον πράκτορα ελέγχου και τον κριτικό πράκτορα. Οι πράκτορες ελέγχου και κρίσης έχουν μια παρόμοια αρχιτεκτονική η οποία αποτελείται από τρία κύρια μέρη: ένα διευθυντή ατζέντας, ένα δημιουργό προγράμματος δράσης και έναν εκτελεστή. Οι Nierenburg και Lesser (1986) δημιούργησαν το OFFICE, ένα σύστημα υποστήριξης γραφειοκρατικών εργασιών σε γραφεία. Σε αυτό το σύστημα ορίζονται ξεκάθαρα οι διεργασίες αντίληψης, δημιουργίας στόχου, χρονοπρογραμματισμού, σχεδιασμού δράσης και εκτέλεσης, όπως και οι κύριες δομές γνώσης που χρειάζεται ένας πράκτορας (στόχοι, σχέδια, γνώση χρονοπρογραμματισμού, ατζέντας, σχήματα αντικειμένων και εμφανίσεις αυτών, αναπαράσταση των ατζεντών των άλλων πρακτόρων). Οι πράκτορες προσωπικοί βοηθοί (personal assistants) αποτελούν σημαντικό τομέα έρευνας της κατανεμημένης τεχνητής νοημοσύνης.

Τέλος εμφανίστηκαν αρχιτεκτονικές πολλαπλών οριζόντιων ή κάθετων επιπέδων δράσης (Εικόνα 2-4). Ο Ferguson (1992) παρουσίασε μία πολυεπίπεδη αρχιτεκτονική για τον έλεγχο αυτόνομων, κινητών πρακτόρων που επιτρέπει σε έναν περιορισμένο από πόρους και κινούμενο προς την ικανοποίηση κάποιου στόχου πράκτορα να αντιδρά σωστά σε απρόσμενες αλλαγές στο περιβάλλον του. Αυτή η αρχιτεκτονική προτείνει τρία παράλληλα (οριζόντια) επίπεδα που λειτουργούν ταυτόχρονα: επίπεδο αντίδρασης, επίπεδο σχεδιασμού δράσης και σκεπτικό/ προφητικό επίπεδο. Ο Kiss (1992) επίσης πρότεινε μια πολυεπίπεδη αρχιτεκτονική (κάθετων επιπέδων) όπου τα υψηλά επίπεδα είναι αποσυνδεδεμένα με τον κόσμο (επίπεδα σκέψης) ενώ τα χαμηλότερα επίπεδα είναι άμεσα συνδεδεμένα με τον κόσμο (επίπεδα δράσης).



Εικόνα 2-4: Αρχιτεκτονικές πολλαπλών επιπέδων

2.2.3. Δομή και Οργάνωση του ΣΠΠ

Επειδή πολλά ΣΠΠ στο παρελθόν ασχολούνταν κυρίως με τη λύση προβλημάτων όπως η κατανομή των εργασιών στους ΕΠ, οργάνωση της ομάδας, επικοινωνίες και διαπραγματεύσεις, είχαν περιορισμένη προσαρμοστικότητα σε νέες συμπεριφορές. Σήμερα είναι απαραίτητο να υπάρχει η κατάλληλη υποδομή ώστε το ΣΠΠ να ακολουθεί κάποια βασικά στοιχεία οργάνωσης, έτσι ώστε να είναι πιο προσαρμόσιμο σε νέες απαιτήσεις.

Σε ένα περιβάλλον πολλαπλών πρακτόρων *δομή* καλείται το πρότυπο της σχέσης πληροφορίας και ελέγχου μεταξύ των πρακτόρων και η κατανομή των ικανοτήτων λύσης προβλημάτων αναμεταξύ τους. Έτσι, η δομή δίνει σε κάθε πράκτορα μια υψηλού επιπέδου άποψη του πως η ομάδα λύνει προβλήματα, όπως και τον ρόλο του κάθε πράκτορα μέσα στην ομάδα. Μέσω αυτής της άποψης οι πράκτορες μπορούν να σιγουρευτούν ότι ικανοποιούν τις προϋποθέσεις οι οποίες είναι απαραίτητες για την επιτυχή λύση ενός προβλήματος, περιλαμβάνοντας τα παρακάτω (Corkill και Lesser, 1983):

- *Κάλυψη*: οποιοδήποτε κομμάτι του ολικού προβλήματος πρέπει να είναι μέσα στις ικανότητες τουλάχιστον ενός πράκτορα.
- *Συνδεσιμότητα*: οι πράκτορες πρέπει να μπορούν να αλληλεπιδρούν με τέτοιο τρόπο ώστε οι καλυπτόμενες δραστηριότητες να μπορούν να συντεθούν σε μία ολοκληρωμένη λύση.
- *Ικανότητα*: η κάλυψη και η συνδεσιμότητα πρέπει να είναι δυνατές μέσα στα υπολογιστικά και επικοινωνιακά περιθώρια, όπως και τις προδιαγραφές αξιοπιστίας της ομάδας.

Γενικά, η δομή πρέπει να ορίζει *ρόλους* και *σχέσεις* για να πετύχει τις παραπάνω προϋποθέσεις. Για παράδειγμα η κάλυψη πετυχαίνεται όταν η κατάλληλη δομή μοιράσει ρόλους στους πράκτορες ανάλογα με την ανταγωνιστικότητά τους και την γνώση τους για ένα δεδομένο υποπρόβλημα. Περισσότερα για τους ρόλους θα αναφερθούν στην παράγραφο 2.4.1.

Η *οργάνωση* σε ένα ΣΠΠ είναι λιγότερο δομημένη στην προοπτική της και περισσότερο σχετίζεται με τη σύγχρονη οργανωτική θεωρία (Maines, 1984, Strauss, 1978). Ο Gasser (1986) όρισε την οργάνωση σαν «ένα ιδιαίτερο σύνολο από ορισμένα και αόριστα ζητήματα τα οποία αφορούν τις πεποιθήσεις και τις δράσεις μέσα από τις οποίες οι πράκτορες έχουν άποψη για τους άλλους πράκτορες». Σύμφωνα με τον παραπάνω ορισμό, οργανωτική αλλαγή σημαίνει άνοιγμα ή/και ορισμός κάποιου διαφορετικού συνόλου ζητημάτων με κάποιο διαφορετικό τρόπο, δίνοντας σε συγκεκριμένους πράκτορες νέα προβλήματα προς λύση όπως και μια διαφορετική βάση υποθέσεων για τις πεποιθήσεις και τις δράσεις των άλλων πρακτόρων. Έτσι, σε ένα πρόβλημα ένας πράκτορας μπορεί να συνεργάζεται με κάποιον άλλο πράκτορα, σε κάποιο επόμενο πρόβλημα όμως να μην είναι καν στην ίδια ομάδα.

Ο Malone (1990) ασχολείται με την *οργάνωση ομάδας*. "Μια ομάδα πρακτόρων είναι οργανωμένη αν οι τελευταίοι είναι συνδεδεμένοι με κάποιο τρόπο όπου οι συνδυασμένες ενέργειές τους αποδίδουν καλύτερα από το να μην υπήρχε αυτή η σύνδεση. Η οργανωμένη ομάδα αποτελείται από: μια

ομάδα πρακτόρων, ένα σύνολο ενεργειών των πρακτόρων, ένα σύνολο συνδέσεων μεταξύ των πρακτόρων, και ένα σύνολο από στόχους ή κριτήρια εκτίμησης με τα οποία οι συνδυασμένες ενέργειες των πρακτόρων εκτιμούνται".

Οι ΕΠ μπορούν να είναι γεωγραφικά κατανεμημένοι επιτρέποντας την αλληλεπίδραση με το σύστημα χρηστών με διαφορετικά επίπεδα ευθύνης-πρόσβασης. Ακόμα, διαβαθμισμένες βάσεις γνώσεις που χρησιμοποιούνται από συγκεκριμένους τύπους πρακτόρων μπορούν να φυλάσσονται σε συγκεκριμένους υπολογιστές.

2.2.4. Συντονισμός, Συνεργασία και Διαπραγματεύσεις

Οι τρεις αυτές έννοιες βρίσκονται πολύ κοντά εννοιολογικά για αυτό και περικλείονται στην ίδια παράγραφο. Με τον *συντονισμό* οι πράκτορες συγχρονίζουν τις πράξεις τους έτσι ώστε να είναι καλή η *συνεργασία* τους και να επιτευχθούν οι στόχοι που συμφώνησαν στις *διαπραγματεύσεις* τους.

Οι Jennings, Sycara και Wooldridge (1998) βλέπουν τις τρεις αυτές έννοιες ως τύπους αλληλεπίδρασης των ΕΠ. Οι ΕΠ *συνεργάζονται* για να κατακτήσουν ένα κοινό στόχο, *συντονίζονται* ώστε να οργανώσουν τις δραστηριότητές τους αποφεύγοντας τις συγκρούσεις και επιτυγχάνοντας βέλτιστο αποτέλεσμα και τέλος *διαπραγματεύονται* ώστε να συμφωνήσουν όλοι μαζί στην πορεία προς το συλλογικό στόχο.

2.2.4.1. Συντονισμός

Ο Malone, 1990 αναγνωρίζει δύο τρόπους συντονισμού: ιεραρχίες και αγορές. Οι ιεραρχίες βασίζονται στην άμεση επίβλεψη εργασιών. Οι αγορές από την άλλη βασίζονται στην επίτευξη συμφωνίας μεταξύ πρακτόρων οι οποίοι έχουν πρόσβαση σε διαφορετικούς πόρους, λειτουργίες. Και οι δύο αυτοί τρόποι συντονισμού γίνονται απαιτητικοί σε πόρους και επικοινωνία όσο αυξάνεται ο πληθυσμός της ομάδας.

Ένας άλλος τρόπος συντονισμού είναι μέσω ενός κεντρικού ελεγκτή ο οποίος θα συντονίζει τις πράξεις όλων των ΕΠ και ο οποίος θα μπορεί να έχει μια

ευρύτερη άποψη για την κοινωνία των πρακτόρων (Moulin και Chaib-draa, 1996, Artificial Intelligence Center, 2000). Αυτός ο τρόπος συντονισμού αναιρεί το πλεονέκτημα της DAI όπου η αστοχία ενός λυτή δεν επηρεάζει την απόδοση των υπολοίπων. Αν αστοχήσει ο κεντρικός ελεγκτής όλο το σύστημα αδρανεί.

Το 1993 ο Jennings μίλησε για τις συμβάσεις και δεσμεύσεις. Σύμφωνα με αυτή τη θεωρία οι πράκτορες συντονίζονται μέσω συμβάσεων που στέλνουν ο ένας στον άλλο και δεσμεύσεων (όταν κάποιος «υπογράψει» μια σύμβαση δεσμεύεται ως προς το περιεχόμενό της). Αυτή η θεωρία έχει πρακτική εφαρμογή σε συστήματα όπου οι πράκτορες είναι «ειλικρινείς» και αυτόνομοι.

Τις περισσότερες φορές ο συντονισμός γίνεται πριν οι πράκτορες αναλάβουν δράση – στη φάση σχεδιασμού της δράσης τους – σε ορισμένες περιπτώσεις όμως είναι δυνατή και αργότερα (π.χ. αλλάζουν οι συνθήκες στις οποίες εκτελείται ένα έργο). Ο επιμέρους σφαιρικός σχεδιασμός (Partial Global Planning) των Durfee και Lesser (1987) προβλέπει την αναδιαπραγμάτευση μεταξύ των πρακτόρων αφού έχει αρχίσει να εκτελείται μια εργασία.

Το contract-net είναι ένα πολύ σημαντικό πρωτόκολλο συντονισμού για τα ΣΠΠ, προτάθηκε το 1981 από τους Smith και Davis και χρησιμοποιείται ακόμα και στα πιο σύγχρονα ΣΠΠ, το ZEUS (Collis και Ndumu, 1999b) επί παραδείγματι. Σύμφωνα με αυτό το πρωτόκολλο διαπραγμάτευσης κάθε πράκτορας συντονίζεται με τους άλλους μέσω συμβολαίων. Κάθε πράκτορας που αναλαμβάνει ένα συμβόλαιο το σπάει σε μικρότερα και τα αναθέτει σε πράκτορες που γνωρίζει ότι μπορούν να τα φέρουν σε πέρας. Στέλνει λοιπόν σε όλους το συμβόλαιο με τα υποέργα αναλαμβάνοντας το ρόλο του μάνατζερ. Οι πράκτορες εκτιμούν αν μπορούν να το φέρουν σε πέρας και αν ναι αποστέλλουν τις προσφορές τους στο μάνατζερ. Ο τελευταίος αξιολογεί τις προσφορές και επιλέγει σε ποιον θα κάνει την ανάθεση. Από κει και πέρα ο μάνατζερ ολοκληρώνει το συμβόλαιό του επικοινωνώντας με τους πράκτορες στους οποίους ανέθεσε τα υποέργα. Το πρωτόκολλο αυτό είναι και μέρος του

κεφαλαίου της διαπραγμάτευσης (σε κάποιες εργασίες αναφέρεται ως πρωτόκολλο διαπραγμάτευσης, π.χ. Moulin και Chaib-Draa, 1996).

2.2.4.2. Συνεργασία

Σύμφωνα με τους Durfee et al. (1989) η συνεργασία των πρακτόρων που ανήκουν σε ένα ΣΠΠ πρέπει να έχει τέσσερις στόχους:

- Αύξηση του ρυθμού ολοκλήρωσης των εργασιών μέσω του παραλληλισμού εκτέλεσης υπο-εργασιών
- Αύξηση του συνόλου ή του εύρους των εργασιών που μπορούν να εκτελεστούν μέσω του διαμοιρασμού πόρων
- Αύξηση της πιθανότητας να ολοκληρωθούν οι εργασίες μέσω της διπλής εκτέλεσής τους[1]
- Μείωση της αλληλεπίδρασης εργασιών

Τέτοιοι πράκτορες είναι συνήθως οι συνεργατικοί κατανεμημένοι λυτές προβλημάτων (cooperative distributed problem-solving, CDPS). Αυτοί λύνουν προβλήματα τα οποία δεν είναι ποτέ μέσα στις δυνατότητες ενός μόνο ΕΠ. Έτσι, το πρόβλημα χωρίζεται σε υποπροβλήματα/ υποέργα των οποίων η σύνθεση δίνει τη λύση.

Οι Cammarata et al. (1983) πρότειναν μια στρατηγική συνεργασίας για αποφυγή συγκρούσεων μεταξύ πρακτόρων οι οποίοι εκτελούν τα σχέδιά τους. Σύμφωνα με αυτή τη στρατηγική, στην περίπτωση που οι πράκτορες δουν μια πιθανή σύγκρουση συμφερόντων πρέπει να επιλέξουν κάποιον ικανότερο από τους άλλους για να δημιουργήσει ένα κοινό σχέδιο δράσης. Η απόφαση στη δουλειά των Cammarata et al. αφορούσε το ποιος από τους πράκτορες που κανονίζουν τις πορείες αεροσκαφών που κινδυνεύουν να συγκρουστούν θα σχεδιάσει τις νέες πορείες των αεροσκαφών. Για να αποφασιστεί από την ομάδα ποιος είναι ο ικανότερος λαμβάνονταν υπόψη συγκεκριμένα κριτήρια

όπως επάρκεια πληροφόρησης, πόσο περιορισμένος στη δράση του (ποιος έχει περισσότερους περιορισμούς, π.χ. ποιος έχει τα λιγότερα καύσιμα) είναι κάποιος πράκτορας, κλπ.

Ο Decker (1987) θέτει το θέμα της συνεργασίας στο επίπεδο του βαθμού στον οποίο είναι συνεργάσιμοι οι ΕΠ από πλήρως συνεργάσιμοι μέχρι ανταγωνιστικοί. Οι πλήρως συνεργάσιμοι πράκτορες εκτελούν τις εργασίες τους συνήθως με μεγάλο επικοινωνιακό κόστος και με καθυστέρηση καθώς είναι πάντα έτοιμοι να αλλάξουν τους στόχους τους για να ικανοποιήσουν τις ανάγκες άλλων πρακτόρων λειτουργώντας με συμπαγή τρόπο και συντονιζόμενοι μαζί τους. Οι ανταγωνιστικοί πράκτορες από την άλλη μπορεί να μην είναι συνεργάσιμοι αν δεν τους συμφέρει, ακόμα και μπλοκάρουν ο ένας τους στόχους των άλλων.

Τέλος αξίζει να αναφερθεί η δουλειά των Levesque et al. (1990) πάνω στη δημιουργία ομάδων. Σύμφωνα με αυτή τη δουλειά μια ομάδα χαρακτηρίζεται από μια κοινή πρόθεση (joint intention). Η κοινή πρόθεση αυτή γίνεται πράξη όταν υπάρχει μια κοινή δέσμευση (joint commitment) από όλα τα μέλη της ομάδας για την ανάληψη κοινής δράσης. Η κοινή δέσμευση αυτή ορίζεται ως ένας κοινός επίμονος στόχος (joint persistent goal). Για να γίνει κάποιος κοινωνός μιας κοινής δέσμευσης πρέπει να ενστερνιστεί τις κοινές πεποιθήσεις (beliefs) και δεσμεύσεις της ομάδας.

2.2.4.3. Διαπραγμάτευση

Η διαπραγμάτευση είναι ένας μηχανισμός που χρησιμεύει συνήθως στον συντονισμό μιας ομάδας ΕΠ. Οι Durfee, Lesser και Corkill (1989) περιγράφουν τη διαπραγμάτευση ως ένα τρόπο βελτίωσης μιας συμφωνίας (μειώνοντας την ασάφεια και την αβεβαιότητα). Οι Jennings, Sycara και Wooldridge (1998) ορίζουν την διαπραγμάτευση σαν μια μέθοδο συντονισμού

---

[1] βρίσκει εφαρμογή κατά τη γνώμη του γράφοντα μόνο στην περίπτωση των συστημάτων πραγματικού χρόνου (real time systems)

και επίλυσης συγκρούσεων. Σύμφωνα με τους ίδιους τα χαρακτηριστικά ενός ΣΠΠ τα οποία κάνουν αναγκαία τη χρήση διαπραγμάτευσης είναι τα:

- Η ύπαρξη κάποιας μορφής σύγκρουσης η οποία πρέπει να επιλυθεί με έναν αποκεντρωμένο τρόπο.
- Η λειτουργία εγωκεντρικών πρακτόρων οι οποίοι έχουν:
    - πεπερασμένη λογική και
    - τοπική-ελλιπή πληροφόρηση

Όταν η διαπραγμάτευση χρειάζεται να αυτοματοποιείται (για χρήση σε πληροφοριακά συστήματα όπως τα ΣΠΠ) τρία είναι τα βασικά θέματα που πρέπει να οριστούν καλά:

- *Πρωτόκολλα διαπραγμάτευσης* (ένα σύνολο κανόνων σύμφωνα με τους οποίους θα αλληλεπιδράσουν οι διαπραγματευόμενοι ΕΠ, π.χ. ποιοι θα διαπραγματευτούν, ποιες είναι οι πιθανές καταστάσεις της διαπραγμάτευσης, κλπ),
- *αντικείμενα διαπραγμάτευσης* (τα θέματα για τα οποία θα γίνει διαπραγμάτευση, π.χ. τιμή αγοράς προϊόντος) και
- *μοντέλα λήψης αποφάσεων των πρακτόρων* (τα μοντέλα αυτά χρησιμοποιούνται παράλληλα με το πρωτόκολλο από τους ΕΠ ώστε καλύτερα να υπερασπιστούν τα συμφέροντά τους)

Η διαδικασία της διαπραγμάτευσης συνήθως προβλέπει την αποστολή από έναν πράκτορα σε έναν άλλο μιας επιχειρηματικής πρότασης-προσφοράς (proposal). Δύο είναι συνήθως οι απαντήσεις του πράκτορα που δέχεται την προσφορά:

- Η διατύπωση μιας κριτικής (critique) για την προσφορά ή/και
- η διατύπωση μιας νέας προσφοράς (counter-proposal)

Οι συνηθισμένοι τρόποι διαπραγμάτευσης είναι οι (Jennings et al., 2001):

- *Πλειστηριασμός* (Αγγλικού τύπου, Ολλανδικού τύπου κλπ). Σε αυτή την περίπτωση κάποιος πράκτορας κάνει προσφορές σε μια ομάδα πρακτόρων ή δέχεται προτάσεις πάνω σε ένα συγκεκριμένο αντικείμενο διαπραγμάτευσης από αυτούς.

- *Προσέγγιση με τη θεωρία παιγνίων* (game theory). Σύμφωνα ,με αυτή την προσέγγιση οι πράκτορες κάνουν την κίνησή τους προσπαθώντας να απαντήσουν στην ερώτηση: «Ποιο είναι η καλύτερη-περισσότερο λογική κίνηση που μπορεί να κάνει ο πράκτορας;». Η θεωρία παιγνίων προκύπτει από τη μελέτη παιχνιδιών όπως το σκάκι. Στην περίπτωση της διαπραγμάτευσης πρακτόρων χρησιμοποιείται ώστε οι πράκτορες να επιλέξουν την καλύτερη στρατηγική από τον χώρο όλων των πιθανών στρατηγικών, αναλογιζόμενοι όλες τις πιθανές αλληλεπιδράσεις. Σημαντική δουλειά στο χώρο αυτό έκαναν οι Rosenschein και Zlotkin (1994). Αυτοί χρησιμοποίησαν ένα μονότονο πρωτόκολλο συμβιβασμού με στρατηγική Zeuthen (μοντέλο λήψης απόφασης). Σύμφωνα με αυτό η διαπραγμάτευση προχωρά με γύρους διαπραγματεύσεων. Σε κάθε γύρο κάθε εμπλεκόμενος πράκτορας κάνει μια προσφορά. Αν οι προσφορές είναι αλληλοκαλυπτόμενες η διαπραγμάτευση σταματά. Αν όχι ξεκινά νέος γύρος διαπραγματεύσεων, στον οποίο οι πράκτορες μπορούν είτε να κάνουν μια παραχώρηση (concession) είτε να προτείνουν την ίδια πρόταση που είχαν πριν. Αν κανείς δεν κάνει παραχώρηση τότε η διαπραγμάτευση σταματά χωρίς αποτέλεσμα. Σύμφωνα με την στρατηγική Zeuthen ο πράκτορας που θα κάνει παραχώρηση είναι αυτός που έχει να χάσει τα περισσότερα σε περίπτωση μη κατάληξης σε συμφωνία αλλά η παραχώρηση πρέπει να είναι η ελάχιστη που απαιτείται ώστε κάποιος άλλος να έχει πιο πολλά να χάσει σε επόμενο γύρο διαπραγμάτευσης.

- *Ευριστική προσέγγιση* (Heuristic approach). Εδώ, ευριστικά πρωτόκολλα και μηχανισμοί λήψης απόφασης στοχεύουν στο να βρεθεί σε βέλτιστο χρόνο μια καλή (όχι βέλτιστη) απόφαση συμβιβασμού.

- *Προσέγγιση βασισμένη σε επιχειρήματα* (argumentation-based approach). Σε αντίθεση με τους προηγούμενους τρόπους διαπραγμάτευσης εδώ δίνεται έμφαση στην άσκηση κριτικής σε μία προσφορά αντί για να κατατεθεί μια αντι-προσφορά. Ακόμα, μπορεί να περιλαμβάνονται απειλές, μπόνους, κλπ σύμφωνα με την ανθρώπινη επιχειρηματολογία. Οι Parsons et al (1998) δείχνουν πως ένα μοντέλο επιχειρηματολογίας μπορεί να λειτουργήσει σε BDI πράκτορες (μια έρευνα για BDI αρχιτεκτονικές έκαναν οι Haddadi και Sundermeyer το 1996) επηρεάζοντας, με επιχειρηματολογία, ο ένας τις πεποιθήσεις του άλλου.

Όλες οι μέθοδοι αυτές έχουν πλεονεκτήματα και μειονεκτήματα. Οι πλειστηριασμοί περιλαμβάνουν μεγάλο όγκο ανταλλαγής μηνυμάτων και απευθύνονται σε ΣΠΠ στα οποία υπάρχουν πολλοί πράκτορες. Ακόμα ο πράκτορας που κάνει προσφορές δεν μπορεί να ξέρει γιατί δεν γίνονται αποδεκτές (η μόνη πληροφορία που δέχεται από τους άλλους πράκτορες είναι η αποδοχή ή όχι της προσφοράς του). Η θεωρία παιγνίων είναι τεκμηριωμένη αλλά είναι ιδιαίτερα απαιτητική σε υπολογιστικό χρόνο και αφορά διαπραγματεύσεις σε συγκεκριμένα θέματα (ένα μοντέλο δεν μπορεί να γενικευτεί) και επίσης να γνωρίζουν οι πράκτορες όλο τον χώρο των λύσεων. Οι ευρετικοί μέθοδοι επειδή δεν αποδεικνύουν την επίτευξη ή όχι συμφωνίας θέλουν συνήθως εξαντλητικό έλεγχο για τη σωστή τους λειτουργία. Τέλος, η επιχειρηματολογία απευθύνεται σε πολύπλοκες αρχιτεκτονικές πρακτόρων και είναι ακόμα νέα ως κλάδος της διαπραγμάτευσης.

Οι παραπάνω περιπτώσεις αφορούν κυρίως ΣΠΠ όπου οι πράκτορες είναι πρόθυμοι να συνεργαστούν. Σε αντίθετη περίπτωση η διαπραγμάτευση πολύ συχνά φτάνει σε αδιέξοδο. Η Sycara (1990) πρότεινε την ύπαρξη τρίτων πρακτόρων οι οποίοι θα μεσολαβούν σε περίπτωση σύγκρουσης απόψεων. Η συγκεκριμένη εφαρμογή (PERSUADER) προέβλεπε ΕΠ που εκπροσωπούσαν τον εργοδότη και το σωματείο των εργαζομένων οι οποίοι πολύ συχνά είχαν αντίθετα συμφέροντα. Σε αυτή την περίπτωση ένας τρίτος πράκτορας – ο

διαμεσολαβητής – αναλάμβανε να κάνει νέες προτάσεις για να μικρύνει τις διαφορές στις προτάσεις των προηγούμενων δύο.

2.2.5. Επικοινωνία και Αλληλεπιδράσεις των ΕΠ

Σε αυτή την παράγραφο εξετάζονται τα είδη επικοινωνίας μεταξύ πρακτόρων. Η επικοινωνία επιτρέπει στους πράκτορες να ανταλλάξουν πληροφορίες, να συντονίσουν τη δράση τους. Δύο είδη επικοινωνίας κυριαρχούν μεταξύ των πρακτόρων και αυτά είναι η κατευθείαν ανταλλαγή μηνυμάτων και η χρησιμοποίηση ενός κοινού σημείου συγκέντρωσης δεδομένων (μαυροπίνακας).

Υπάρχουν και άλλες μέθοδοι συντονισμού οι οποίες έχουν χρησιμοποιηθεί και προβλέπουν ελάχιστη ή καθόλου επικοινωνία μεταξύ των ΕΠ. Έτσι, αν οι ΕΠ δεν χρειάζονται να συνεργαστούν για να πετύχουν τους στόχους τους αλλά ούτε συγκρούονται τα συμφέροντά τους τότε η επικοινωνία μεταξύ τους δεν είναι απαραίτητη για την εύρυθμη λειτουργία τους. Ακόμα υπάρχουν περιπτώσεις όπου ελάχιστη επικοινωνία με τη μορφή ενός πεπερασμένου συνόλων σημάτων αρκεί για τον συντονισμό των πρακτόρων (Georgeff, 1983). Τέλος, οι πράκτορες σε κάποια άλλα μοντέλα μπορούν να συντονιστούν παρατηρώντας τις αλλαγές που επιφέρουν άλλοι πράκτορες στο περιβάλλον (παρατηρούν τις αλλαγές γύρω τους). Σε αυτή την περίπτωση οι πράκτορες ενδέχεται να προσπαθήσουν να «μαντέψουν» ποιοι κινήθηκαν και τι κίνηση έκαναν παρατηρώντας τις επιδράσεις τους στο περιβάλλον (Rosenscheim και Breese, 1989). Τέτοιες μεθόδους επικοινωνίας-συντονισμού χρησιμοποιούν συνήθως οι *πράκτορες που αντιδρούν σε ερεθίσματα*.

Ο μαυροπίνακας είναι μια μορφή κοινής μνήμης (Nii, 1986) όπου όλοι οι πράκτορες μπορούν να γράψουν κάτι ή να τον διαβάσουν. Συνήθως ο μαυροπίνακας χωρίζεται σε περιοχές και οι πράκτορες αποκτούν προνόμια με την ικανότητα να διαβάζουν περιοχές. Έτσι, άλλοι μπορούν να διαβάσουν και να γράψουν σε περισσότερες περιοχές από κάποιους άλλους.

Ο πιο διαδεδομένος τρόπος επικοινωνίας στα σύγχρονα συστήματα πολλαπλών πρακτόρων είναι η ανταλλαγή μηνυμάτων. Το περιεχόμενο του μηνύματος διαφοροποιείται σημαντικά από σύστημα σε σύστημα. Αυτό που χρειάζεται στην περίπτωση των μηνυμάτων είναι η διαμόρφωση ενός πρωτοκόλλου σύνταξης και μεταφοράς του μηνύματος όπως και ο προσδιορισμός των διαφορετικών τύπων μηνυμάτων (π.χ. το πρωτόκολλο contract-net των Smith και Davis, 1981). Κάποιοι μελετητές πρότειναν την ανταλλαγή των σχεδίων δράσης των πρακτόρων μέσω των μηνυμάτων για τον καλύτερο συντονισμό τους. Ο Rosenscheim το 1986 περιέγραψε τους κινδύνους του να μεταφέρεται μέσω μηνυμάτων το σχέδιο δράσης ενός πράκτορα (μεγάλα μηνύματα, δυναμικό περιβάλλον, αδυναμία άλλων πρακτόρων να υιοθετήσουν το σχέδιο, κλπ).

Οι δυνατότητας επικοινωνίας μεταξύ πρακτόρων μπορούν να επεκταθούν και πέραν της ανταλλαγής ενός απλού μηνύματος εισάγοντας σε ένα μήνυμα τη δυνατότητα να επηρεάζει τις γνώσεις και πεποιθήσεις του πράκτορα που λαμβάνει το μήνυμα όπως και να περιέχει στοιχεία της θεωρίας λόγου-δράσης (speech-act, προτάθηκε από τον Searle το 1969). Έτσι όταν οι πράκτορες ακολουθούν την BDI λογική οι πεποιθήσεις τους μπορεί να αλλάζουν αντίστοιχα με αλλαγές στο περιβάλλον ή στη συμπεριφορά άλλων πρακτόρων. Με χρήση σύγχρονων γλωσσών όπως η γλώσσα χειρισμού ερωταπαντήσεων γνώσης (Knowledge Query Manipulation Language, KQML) ή η γλώσσα επικοινωνίας πρακτόρων (Agent Communication Language, συντομογραφία: ACL) είναι δυνατή αυτή η λειτουργία όπως και η δυνατότητα speech-act δηλαδή δράσης μέσω του λόγου (κάποιος πράκτορας μπορεί να ζητήσει κάτι λέγοντας «παρακαλώ θέλω…» ενώ κάποιος άλλος-πιθανόν σε ιεραρχικές δομές- να το ζητήσει λέγοντας «θέλω αμέσως…»). Αυτές οι δυνατότητες επικοινωνίας συνήθως αξιοποιούνται από *κοινωνικούς πράκτορες*.

2.3. Υπάρχουσες Πλατφόρμες Ανάπτυξης ΣΠΠ

Τα ΣΠΠ τα οποία μελετήθηκαν είναι τα RETSINA (Sycara et al., 1996), ARCHON (Jennings et al., 1996b), Open Agent Architecture (Artificial Intelligence Center, 2000), AgentBuilder (Reticular Systems Inc, 1999), GrassHopper (IKV++ GmbH, 1999a) και ZEUS (Collis και Ndumu, British Telecommunications, 1999a). Ο στόχος της μελέτης των πλατφορμών αυτών ήταν η ανάλυση της υφιστάμενης κατάστασης στον τομέα αυτό με σκοπό την διερεύνηση της δουλειάς που έχει γίνει στον τομέα αυτό, την αξιολόγηση τους και την εξεύρεση κοινών χαρακτηριστικών ή στάνταρντς. Εκτός από δύο οι οποίες παρουσιάζονται ως κλασσικές (του 1996) πλατφόρμες οι υπόλοιπες είναι όλες σύγχρονες, οι ίδιες ακόμα στη φάση βελτίωσης.

Δυστυχώς η μελέτη που έγινε δεν έδειξε κάποια πρότυπα τα οποία να υιοθετούνται από όλες τις υπάρχουσες πλατφόρμες, πράγμα το οποίο θα έκανε δυνατή την επικοινωνία όλων των συστημάτων.

2.3.1. RETSINA

Οι Sycara et al., το 1996 παρουσίασαν την αρχιτεκτονική και οργάνωση ενός συστήματος πρακτόρων. Η δουλειά αυτή (επονομαζόμενη ως Retsina) είχε σαν στόχο την κάλυψη των παρακάτω κατευθύνσεων μέσω της χρησιμοποίησης ενός ΣΠΠ:

- την πρόσβαση σε διασκορπισμένες στο χώρο του διαδικτύου πληροφορίες,
- τη διείσδυση σε διάφορα θέματα (άσχετα μεταξύ τους),
- την *απόκρυψη της πολυπλοκότητας*. Οι πράκτορες θα πρέπει να δίνουν στον χρήστη σαν αποτέλεσμα κάτι απλό (τη λύση στο πρόβλημά του ή την πληροφορία που χρειάζεται) χωρίς να γίνεται αναφορά στην πολυπλοκότητα της μεθόδου που ακολουθήθηκε,

- οι πράκτορες θα πρέπει να είναι φτιαγμένοι κατά τέτοιο τρόπο ώστε να μπορούν να *ξαναχρησιμοποιηθούν* (reusable) ή έστω ένα κομμάτι αυτών να χρησιμοποιηθεί για την κατασκευή ενός άλλου πράκτορα,
- θα πρέπει να είναι *ευέλικτοι* (flexible). Να μπορούν δηλαδή να δράσουν και με νέα δεδομένα του χρήστη (να προσαρμοστούν),
- την *επιμονή* (persistence). Όταν οι πληροφορίες είναι διασκορπισμένες στο Internet είναι δύσκολο να συλλεχθούν και το σύστημα στο οποίο δουλεύουν οι πράκτορες μπορεί να παρουσιάσει προβλήματα,
- την ποιότητα των πληροφοριών. Οι πληροφορίες υπάρχουν παντού και σε διάφορες μορφές. Αυτό που πρέπει να κάνουν οι πράκτορες είναι να ελέγχουν τις πληροφορίες και να βρίσκουν τις πιο ορθές(για το χρήστη),
- τη βελτίωση. Οι πράκτορες θα πρέπει να έχουν τη δυνατότητα βελτίωσης, καθώς συνεχώς εφευρίσκονται καινούργιες μέθοδοι για την πληροφορία.

Το θέμα είναι πώς αυτό το σύστημα που θα καλύπτει όλες τις παραπάνω κατευθύνσεις μπορεί να φτιαχτεί. Η κύρια δουλεία των προγραμματιστών του Retsina ήταν να φτιάξουν ένα γενικό λογισμικό πρακτόρων, έτσι ώστε μετά για οτιδήποτε άλλη εργασία χρειαζόταν ένας πράκτορας, να ήταν εύκολο να φτιαχτεί στηριζόμενος στο γενικό σύστημα.

Στο Retsina ο κάθε χρήστης έχει στη διάθεσή του μερικούς πράκτορες, οι οποίοι εκτελούνται σε διαφορετικές μηχανές, και δρουν εκ μέρους του. Βασίζονται όμως στην επεξεργασία και στον έλεγχο από το χρήστη για το κάθε τρέχον πρόβλημα. Οι πράκτορες αυτοί έχουν πρόσβαση σε διάφορα μοντέλα του χρήστη και σε άλλους πράκτορες. Βασιζόμενοι στις γνώσεις του, ο κάθε πράκτορας αποφασίζει πώς θα δράσει, τι πληροφορίες χρειάζεται για κάθε εργασία, πότε θα χρειαστεί βοήθεια άλλων πρακτόρων και γενικά την εν γένει συμπεριφορά του, προκειμένου να φέρει σε πέρας την αποστολή του. Ο χρήστης μπορεί από μόνος του να ελέγχει την αυτονομία ενός πράκτορα. Όσο ο πράκτορας κερδίζει την εμπιστοσύνη του χρήστη, τόσο μεγαλύτερη αυτονομία του δίνει αυτός.

Στο Retsina υπάρχουν τρεις τύποι πρακτόρων:

1) Πράκτορες διεπαφής (interface agents): παίρνουν πληροφορίες από το χρήστη και σύμφωνα με αυτές προσπαθούν να φέρουν εις πέρας την αποστολή τους.

2) Πράκτορες έργου (task agents): παίρνουν αποφάσεις χρησιμοποιώντας τις γνώσεις τους και τις γνώσεις άλλων πρακτόρων(interface agents,information agents),

3) Πράκτορες πληροφοριών (information agents): συλλέγουν πληροφορίες κάποιας συγκεκριμένης μορφής π.χ. η τιμή του δολαρίου κάθε χρονική στιγμή.

Θα ήταν χρήσιμο να δούμε περίπου τον τρόπο λειτουργίας του Retsina. Σε μια τυπική εκτέλεση, ο πράκτορας διεπαφής παίρνει μια αποστολή από τον χρήστη και τη μεταφέρει στον πράκτορα έργου (στον πράκτορα έργου μπορεί να ανατεθεί αποστολή και από άλλο πράκτορα έργου). Ο πράκτορας έργου χρησιμοποιώντας τις γνώσεις του προσπαθεί να φέρει εις πέρας την αποστολή. Εάν αυτές δεν είναι επαρκείς, μπορεί να χρησιμοποιήσει τους πράκτορες πληροφοριών προκειμένου να του συλλέξουν τις απαραίτητες πληροφορίες που χρειάζεται κάθε φορά. Οι πράκτορες πληροφοριών ερευνούν, μαζεύουν, φιλτράρουν και δίνουν τις πληροφορίες στον πράκτορα έργου. Εάν οι πληροφορίες που μεταφέρει ο πράκτορας πληροφοριών στον πράκτορα έργου δεν είναι χρήσιμες ο πράκτορας πληροφοριών θα αρχίσει από την αρχή τη συλλογή πληροφοριών. Επίσης, ο πράκτορας έργου μπορεί να συνεργαστεί και με άλλους πράκτορες έργου και να χρησιμοποιήσει και τις δικές τους γνώσεις. Όλες οι παραπάνω διαδικασίες γίνονται μέχρι η αποστολή που έχει ανατεθεί από τον χρήστη να ολοκληρωθεί.

Οι πράκτορες βρίσκουν άλλους πράκτορες μέσω ενός πράκτορα πληροφοριών που βρίσκει ποιοι ταιριάζουν στις απαιτήσεις κάποιου τρίτου και λέγεται ταιριαστής (matchmaker).

Από αρχιτεκτονική άποψη, ο κάθε πράκτορας αποτελείται από τέσσερα τμήματα:

1) Τμήμα Σχεδιασμού. Το τμήμα αυτό παίρνει στην είσοδό του ένα σετ από στόχους και παράγει ένα σχέδιο για να τους κατακτήσει. Η διαδικασία αυτή στηρίζεται σε ένα ιεραρχικό δίκτυο εργασιών (hierarchical task network, HTN). Το τμήμα αυτό έχει πρόσβαση σε έτοιμα σχέδια δράσης για την επίτευξη κάποιων στόχων όπως και σε μια βάση πεποιθήσεων και αρχών (knowledge and facts base)

2) Τμήμα Επικοινωνίας και Συντονισμού. Το τμήμα αυτό συλλέγει τα μηνύματα των άλλων πρακτόρων. Αν αυτά περιέχουν αιτήσεις για εξυπηρέτηση αυτές γίνονται στόχοι του πράκτορα. Τεχνικά η επικοινωνία γίνεται με ανταλλαγή email. Στο Retsina χρησιμοποιείται σαν γλώσσα επικοινωνίας, η γλώσσα KQML.

3) Τμήμα Δρομολόγησης. Το τμήμα αυτό δρομολογεί προς εκτέλεση τις εργασίες που έχουν αποφασιστεί από το τμήμα σχεδιασμού.

4) Τμήμα Παρακολούθησης Εκτέλεσης. Το τμήμα αυτό παρακολουθεί την εκτέλεση των εργασιών. Αν αυτές δεν ολοκληρωθούν ομαλά τότε το τμήμα σχεδιασμού καλείται να ξανασχεδιάσει την δράση.

2.3.2. ARCHON

Στο άρθρο των Jennings et al, 1996b περιγράφεται η αρχιτεκτονική του ΣΠΠ Archon. Το ΣΠΠ αυτό έχει χρησιμοποιηθεί για την επίλυση πραγματικών προβλημάτων, δύο από τα οποία (CERN, Iberdrola) περιγράφονται στην ίδια έκδοση η οποία φιλοξενούσε και το προηγούμενο άρθρο, ένα από τα οποία θα περιγραφεί σύντομα μετά την παρουσίαση της αρχιτεκτονικής.

Η αρχιτεκτονική Archon μπορεί να χρησιμοποιηθεί για την υλοποίηση πραγματικών εφαρμογών κατανεμημένης τεχνητής νοημοσύνης (distributed artificial intelligence, DAI). Στόχος του Archon ήταν να είναι μια αρχιτεκτονική όσο το δυνατόν πιο ανοιχτή ώστε να μπορούν υπάρχοντα

έμπειρα συστήματα ή τμήματα λογισμικού να ενσωματωθούν στο σύστημα υπό υλοποίηση με την μορφή ΕΠ. Έτσι βάρος δόθηκε στην δημιουργία ενός πλαισίου το οποίο θα υποστήριζε την αλληλεπίδραση μεταξύ τμημάτων λογισμικού και μιας μεθοδολογίας για την δόμηση των αλληλεπιδράσεων αυτών.

Για την επίτευξη του παραπάνω στόχου κάθε πράκτορας αποτελείται από δύο επίπεδα. Ένα επίπεδο Archon (Archon layer, AL) και ένα επίπεδο ευφυούς συστήματος (intelligent system, IS). Το πρώτο επίπεδο περιλαμβάνει το κοινωνικό know-how του ΕΠ ενώ το δεύτερο τον τομέα στον οποίο λύνει προβλήματα ο πράκτορας.

Το τμήμα του πράκτορα το οποίο ανήκει στην αρχιτεκτονική Archon είναι το AL. Αυτό αποτελείται από τέσσερα διαφορετικά τμήματα:

- Τμήμα επιτήρησης (monitor). Το τμήμα αυτό ελέγχει το IS του ΕΠ χρησιμοποιώντας τρία επίπεδα αναπαράστασης:
  - Μονάδες επιτήρησης (monitoring units, MUs). Κάθε τέτοια μονάδα αντιστοιχεί σε μία λειτουργία του IS. Μέσω των MUs εκτελούνται οι λειτουργίες του IS. Ουσιαστικά καλούν ρουτίνες οι οποίες μπορεί να είναι προγραμματισμένες σε διαφορετικές γλώσσες προγραμματισμού και επιστρέφουν τα αποτελέσματα. Με τον προγραμματισμό των MUs ο μηχανικός ορίζει πως θα μεταφράζονται οι ανάγκες σε λειτουργίες σε κλήσεις ρουτινών του IS.
  - Σχέδια. Τα σχέδια αφορούν δέντρα εκτέλεσης ρουτινών του IS για την επίτευξη όλων των στόχων που μπορεί να επιλύσει ο πράκτορας. Είναι άκυκλοι OR γράφοι των οποίων κόμβοι είναι MUs και ακμές οι συνθήκες που πρέπει να ισχύουν για τη μετάβαση από ένα κόμβο σε κάποιον άλλο. Ο μηχανισμός των σχεδίων έχει ενσωματωμένη μια μονάδα οπισθοδρόμησης (backtracking) ώστε αν σε έναν κόμβο δεν ικανοποιείται καμιά συνθήκη μετάβασης να οπισθοχωρεί στον τελευταίο κόμβο που είχε εναλλακτικό μονοπάτι εκτέλεσης και να

προχωρήσει την εκτέλεση του σχεδίου. Τα μονοπάτια επιλέγονται από αριστερά προς τα δεξιά.

- Συμπεριφορές. Οι συμπεριφορές αποτελούν το υψηλότερο επίπεδο αναπαράστασης των δραστηριοτήτων του IS. Περιέχουν τα σχέδια μαζί με συνθήκες επιλογής, περιγραφή των εισόδων που χρειάζεται και αναμενόμενες εξόδους για κάθε συγκεκριμένο σχέδιο.

- Τμήμα διαχείρισης πληροφοριών πρακτόρων (agent information management module, AIM). Το AIM είναι ένα σύστημα διαχείρισης κατανεμημένων αντικειμένων το οποίο σχεδιάστηκε για να παρέχει υπηρεσίες διαχείρισης πληροφορίας σε συνεργαζόμενους πράκτορες. Περιέχει τις δομές:
  - Μοντέλα πρακτόρων. Τα μοντέλα αυτά αφορούν το προσωπικό μοντέλο του πράκτορα και τα μοντέλα των γνωριμιών του. Τα μοντέλα περιέχουν πληροφορίες σχετικές με τις ικανότητες των πρακτόρων, τα ενδιαφέροντά τους, τον φόρτο εργασίας τους κλπ.
  - Δεδομένα επιπέδου τομέα (domain-level data). Οι τύποι των δεδομένων που ανταλλάσσουν οι ΕΠ.

- Τμήμα σχεδιασμού και συντονισμού (planning and coordination module, PCM). Το τμήμα αυτό κρίνει το ρόλο του ΕΠ στην κοινότητα. Εκτιμά την κατάσταση του ΕΠ και αποφασίζει ποιες δράσεις θα πρέπει να αναλάβει για να αξιοποιήσει αλληλεπιδράσεις με τους άλλους ΕΠ εξασφαλίζοντας τη θετική συνεισφορά του ΕΠ στο ΣΠΠ. Ακόμα αποφασίζει ποιες εργασίες πρέπει να ανατεθούν σε άλλους πράκτορες. Στην εργασία του αυτό το τμήμα χρησιμοποιεί τα μοντέλα του AIM.

- Τμήμα επικοινωνίας υψηλού επιπέδου. Το τμήμα αυτό αναλαμβάνει την επικοινωνία μεταξύ πρακτόρων το οποίο βασίζεται σε ανοιχτά εικονικά κανάλια μεταξύ πρακτόρων.

Το επιχειρησιακό σχέδιο της Iberdrola, για παράδειγμα (Corera et al, 1996), προέβλεπε τη χρήση του Archon για την επίλυση του προβλήματος της

ασφάλειας του δικτύου της και της επιτυχούς λειτουργίας της μέσα σε κάποιους οικονομικούς περιορισμούς. Συγκεκριμένα, το δίκτυο της εταιρίας περιείχε 123 σημεία συγκέντρωσης και αποστολής πληροφοριών από 25000 παραγωγούς δεδομένων σχετικών με πιθανές βλάβες. Το κέντρο ελέγχου του Μπιλμπάο έπαιρνε όλα αυτά τα δεδομένα και έμπειροι μηχανικοί τα επεξεργάζονταν και ανίχνευαν πιθανές βλάβες. Η εταιρία στην προσπάθειά της να κάνει αποτελεσματικό το έργο των μηχανικών χρηματοδότησε την ανάπτυξη διαφόρων έμπειρων συστημάτων υποστήριξης αποφάσεων. Το Archon χρησιμοποιήθηκε επιτυχώς για το δέσιμο των συστημάτων αυτών σε ένα συμπαγές σύστημα υποστήριξης αποφάσεων με πολλούς λυτές προβλημάτων (ΕΠ).

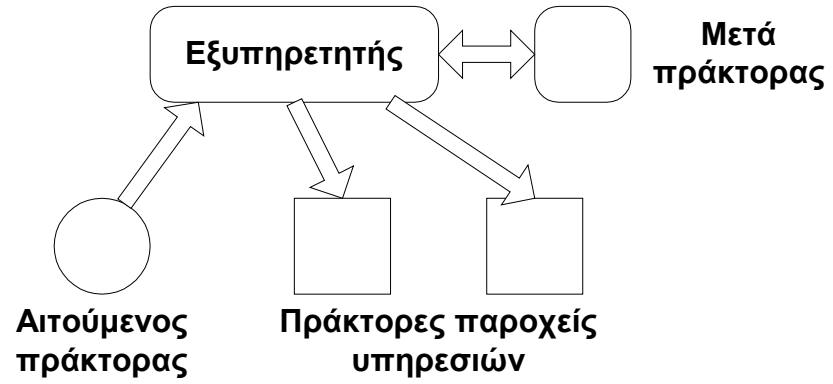
2.3.3. Open Agent Architecture

Η ολοκληρωμένη ονομασία της αρχιτεκτονικής αυτής, η οποία καλείται και OAA χάριν συντομίας, είναι «μοντέλο ανάθεσης υπολογισμών ανοιχτής αρχιτεκτονικής πρακτόρων». Η SRI International που την ανέπτυξε (στο τμήμα Artificial Intelligence Center) είναι εταιρία R&D με πελάτες ανά την υφήλιο.

Η αρχιτεκτονική αυτή έχει χρησιμοποιηθεί σε εφαρμογές αυτοματισμού γραφείου, ελέγχου πολλών ρομπότ, αρχειοθέτησης και ανάκτησης βίντεο κ.α. Ενσωματώνει μηχανισμό αναγνώρισης ερωτημάτων σε φυσική αγγλική γλώσσα των τύπων «τι», «ποιος», «πως». Ενδεικτικές ερωτήσεις που περιλαμβάνονται στο demo του εργαλείου είναι «Τι καιρό θα έχει αύριο το Λονδίνο;» ή «Ποιος είναι ο διευθυντής του τμήματος πωλήσεων της εταιρίας;» ή «Όταν φτάσει ένα μήνυμα για μένα, ειδοποίησέ με αμέσως».

Τεχνικά (Artificial Intelligence Center, 2000), η OAA λύνει τα παραπάνω προβλήματα χρησιμοποιώντας γνώση κατανεμημένη σε τέσσερις τύπους πρακτόρων (η Εικόνα 2-5 σχηματοποιεί επιτυχώς την λογική της αρχιτεκτονικής):

- Αιτούμενους πράκτορες, οι οποίοι ορίζουν ένα στόχο στον εξυπηρετητή και δίνουν συμβουλές ως προς την επίτευξη του στόχου αυτού.
- πράκτορες παροχής υπηρεσιών, οι οποίοι καταχωρούν τον τύπο υπηρεσιών που παρέχουν στον εξυπηρετητή, ενώ η γνώση τους αφορά τις δυνατότητες και τους περιορισμούς τους στη λύση προβλημάτων.
- Ο Εξυπηρετητής, ο οποίος διατηρεί μια λίστα με διαθέσιμους πράκτορες και ένα σύνολο γενικών στρατηγικών για τη λύση προβλημάτων.
- Μετά-πράκτορες, οι οποίοι περιέχουν γνώση σχετική με στόχους ή τομείς δραστηριότητας και στρατηγικές που χρησιμοποιούνται από τον εξυπηρετητή.



Εικόνα 2-5: Η ΟΑΑ σχηματικά

Η αρχιτεκτονική αυτή προβλέπει έναν πράκτορα με ρόλο κλειδί στη λειτουργία του συστήματος τον «Εξυπηρετητή». Όταν κάποιος πράκτορας κάνει μια αίτηση για εξυπηρέτηση ο εξυπηρετητής ταιριάζει την αίτηση αυτή στη δουλειά ενός ή περισσότερων πρακτόρων, αναθέτει εργασίες, συντονίζει τη δραστηριότητα των πρακτόρων και παραδίδει τα αποτελέσματα στον αιτούμενο. Η ΟΑΑ παρέχει ακόμα τη δυνατότητα απευθείας κλήσης κάποιου πράκτορα από τον αιτούμενο όπως και την εκπομπή αιτήσεων σε όλη την κοινότητα.

Η ύπαρξη του εξυπηρετητή αποτελεί και το μεγάλο μειονέκτημα της αρχιτεκτονικής αυτής καθώς αν αυτός δυσλειτουργήσει όλο το οικοδόμημα καταρρέει.

2.3.4. AgentBuilder

Η αρχιτεκτονική ΣΠΠ AgentBuilder είναι της εταιρίας Reticular Systems Inc (Reticular Systems Inc., 1999). Αποτελεί μια ολοκληρωμένη πλατφόρμα ανάπτυξης συστημάτων πολλαπλών πρακτόρων με γραφικό περιβάλλον σχεδίασης των πρακτόρων και παρακολούθησης της εκτέλεσής των.

Γενική παραδοχή του AgentBuilder είναι ότι τα ΣΠΠ χρησιμοποιούνται για να επιλύσουν ένα πρόβλημα. Η ανάπτυξη λοιπόν των πρακτόρων γίνεται κάτω από κάποιο project. Ακόμα, κάθε πράκτορας ανήκει σε ένα ή περισσότερα πρακτορεία (agency). Συνοψίζοντας, ένα project αποτελείται από ένα ή περισσότερα πρακτορεία τα οποία αποτελούνται από έναν ή περισσότερους πράκτορες οι οποίοι μπορεί να ανήκουν σε περισσότερα από ένα πρακτορεία.

Σύμφωνα με τον ορισμό της AIC πράκτορας είναι λογισμικό με τις εξής έμφυτες ικανότητες:

- Επικοινωνίας (οι πράκτορες από τη δημιουργία τους ξέρουν πώς να επικοινωνήσουν με άλλες οντότητες και άλλους πράκτορες)
- Ελέγχου (οι ίδιοι αποφασίζουν τι θα κάνουν και με ποια σειρά)
- Λήψης απόφασης (οι πράκτορες μπορούν να εξαγάγουν συμπεράσματα και να πάρουν αποφάσεις)
- Αυτόνομης λειτουργίας (οι πράκτορες μπορούν να λειτουργήσουν χωρίς εξωτερική βοήθεια)
- Διατήρησης (οι πράκτορες διατηρούν πληροφορία σχετική με την κατάστασή τους)

Από τις ικανότητες αυτές προκύπτουν οι ιδιότητες:

- Κελυφοποίησης: Οι πράκτορες κρύβουν εντελώς τους εσωτερικούς μηχανισμούς τους από εξωτερικές οντότητες οι οποίες μπορούν να επηρεάσουν τη συμπεριφορά τους μόνο στέλνοντάς τους καλά ορισμένα μηνύματα.
- Τμηματικότητας και επαναχρησιμοποίησης: Λόγω της αυτόνομης λειτουργίας τους είναι εύκολη η μετακίνηση και η επαναχρησιμοποίηση τους.
- Παράλληλης εκτέλεσης: Εφόσον κάθε πράκτορας λειτουργεί αυτόνομα, παίρνει τις δικές του αποφάσεις και διατηρεί πληροφορία για την κατάστασή του είναι εύκολη η υλοποίηση παράλληλων διεργασιών με χρήση πολλαπλών πρακτόρων.
- Κατανεμημένης λειτουργίας: Οι πράκτορες είναι ένας ιδανικός μηχανισμός για την υλοποίηση κατανεμημένων πληροφοριακών συστημάτων (κάθε πράκτορας λειτουργεί αυτόνομα και εκτελεί τις λειτουργίες που ξέρει, ενώ ταυτόχρονα επικοινωνεί με άλλους πράκτορες που συμμετέχουν στη διαδικασία λύσης ενός προβλήματος)

Οι πράκτορες προγραμματίζονται στην Γλώσσα Ορισμού Πρακτόρων της Reticular (Reticular Agent Definition Language, RADL), η οποία είναι επέκταση των εργασιών των Shoham, 1991 και Thomas, 1994 οι οποίοι ανέπτυξαν αντίστοιχα τις γλώσσες Agent-0 και PLACA (PLAnning Communicating Agents). Σύμφωνα με αυτή τη γλώσσα για να οριστεί ένας πράκτορας πρέπει να οριστεί το νοητικό μοντέλο του (mental model) το οποίο φαίνεται στην Εικόνα 2-6, είναι παρόμοιο με το μοντέλο του Shoham (1993, δεν περιέχει τους κανόνες συμπεριφοράς) και αποτελείται από:

- Πεποιθήσεις: Οι πεποιθήσεις είναι πρωταρχικό κομμάτι του πράκτορα. Αναπαριστούν την τρέχουσα κατάσταση του πράκτορα όπως και του περιβάλλοντός του και μεταβάλλονται καθώς λαμβάνονται νέες πληροφορίες για το περιβάλλον του πράκτορα. Πιο συγκεκριμένα, ένας πράκτορας έχει πεποιθήσεις σχετικές με το περιβάλλον του, σχετικές με τις

πεποιθήσεις κάποιου άλλου πράκτορα, σχετικές με τις αλληλεπιδράσεις του με άλλους πράκτορες και σχετικές με τις δικές του πεποιθήσεις.

- Ικανότητες: Είναι μια δομή που χρησιμοποιεί ο πράκτορας για να συσχετίσει μια δράση με τις προϋποθέσεις εκτέλεσής της. Η λίστα των ικανοτήτων ενός πράκτορα λοιπόν περιγράφει τις δράσεις που μπορεί να εκτελέσει μαζί με τις προϋποθέσεις εκτέλεσης κάθε μίας από αυτές. Οι δράσεις που μπορεί να αναλάβει ο πράκτορας είναι δύο τύπων:
    - Ιδιωτικές δράσεις: δράσεις εκτελέσιμες από τον ίδιο τον πράκτορα και επιδρούν ή αλληλεπιδρούν με το περιβάλλον του χωρίς να προϋποθέτουν αλληλεπίδραση με άλλους πράκτορες.
    - Επικοινωνιακές δράσεις: δράσεις αποστολής μηνύματος σε άλλους πράκτορες.
- Δεσμεύσεις: Μια δέσμευση είναι μια συμφωνία (στην οποία συνήθως είναι κοινωνοί δύο πράκτορες) για την εκτέλεση μιας συγκεκριμένης δράσης σε ένα συγκεκριμένο χρόνο.
- Κανόνες συμπεριφοράς: Οι κανόνες δέσμευσης του Shoham έχουν επισκιαστεί από τους κανόνες συμπεριφοράς της Reticular. Οι κανόνες αυτοί αποφαίνονται για την δράση του πράκτορα κατά τη διάρκεια της εκτέλεσής του. Οι κανόνες ταιριάζουν ένα σύνολο πιθανών αποκρίσεων σε ερεθίσματα που προέρχονται από το περιβάλλον του (μηνύματα, πέρας κάποιου χρονικού διαστήματος κλπ) όπως τα τελευταία περιγράφονται στις πεποιθήσεις του. Τεχνικά οι κανόνες είναι WHEN-IF-THEN δηλώσεις. *Όταν*, δηλαδή, συμβεί ένα γεγονός, *αν* επαληθεύονται κάποιες προϋποθέσεις, *τότε* ο πράκτορας αναλαμβάνει κάποιες ιδιωτικές ή επικοινωνιακές δράσεις.
- Προθέσεις: Οι προθέσεις είναι δεσμεύσεις αλλά όχι για την ανάληψη κάποιας συγκεκριμένης δράσης αλλά για την επίτευξη μιας κατάστασης του περιβάλλοντος του πράκτορα η οποία μπορεί να περιλαμβάνει πολλαπλή ανάληψη δράσεων.

Εικόνα 2-6: Νοητικό μοντέλο πράκτορα της Reticular Inc.

Κάθε πράκτορας εκτελεί ένα συγκεκριμένο κύκλο εκτέλεσης ο οποίος αποτελείται από πέντε βήματα:

1. Επεξεργασία νέων μηνυμάτων
2. Προσδιορισμός των κανόνων που εφαρμόζονται στην τρέχουσα κατάσταση
3. Εκτέλεση των δράσεων που προβλέπονται από αυτούς τους κανόνες
4. Ενημέρωση του νοητικού μοντέλου σύμφωνα με αυτούς τους κανόνες
5. Σχεδιασμός

Η επεξεργασία ενός νέου μηνύματος προϋποθέτει την αναγνώριση του αποστολέα. Αφού πιστοποιηθεί ο αποστολέας το μήνυμα γίνεται μέρος του νοητικού μοντέλου του πράκτορα. Ακολούθως, μέσω ταιριάσματος προτύπων (pattern matching) συγκρίνονται στοιχεία του νοητικού μοντέλου με τις προϋποθέσεις των κανόνων συμπεριφοράς με στόχο την αναγνώριση των κανόνων των οποίων οι προϋποθέσεις ικανοποιούνται. Τέτοιοι κανόνες εισάγονται στην ατζέντα του πράκτορα για εκτέλεση. Κάθε εκτέλεση κανόνα συμπεριλαμβάνει την εκτέλεση ιδιωτικών και επικοινωνιακών δράσεων όπως και την ενημέρωση του νοητικού μοντέλου του πράκτορα σχετικά με τις αλλαγές που επιφέρει στο περιβάλλον και στον ίδιο τον πράκτορα η εκτέλεση των δράσεων αυτών. Οι δράσεις εκτελούνται σειριακά και στο τέλος

ενημερώνεται το νοητικό μοντέλο του πράκτορα. Τέλος το τμήμα σχεδιασμού του πράκτορα δημιουργεί σχέδια για την εκπλήρωση των προθέσεών του.

Η γλώσσα επικοινωνίας των πρακτόρων είναι η γλώσσα χειρισμού ερωταπαντήσεων σε βάσεις γνώσης (Knowledge Query Manipulation Language, KQML) η οποία έχει προτυποποιηθεί.

Τα πλεονεκτήματα της αρχιτεκτονικής αυτής είναι η ευκολία διαχείρισης του ΣΠΠ.

Μειονεκτήματα αποτελούν η διπλή μεταγλώττιση, η βραδύτητα (κάθε πράκτορας τρέχει στη δική του μηχανή εκτέλεσης πράκτορα) και η μη καλά ορισμένη λειτουργία του τμήματος σχεδιασμού.

2.3.5. GrassHopper

Η IKV ιδρύθηκε το 1995 στην Γερμανία, σαν θυγατρική της GMD FOKUS. Το GrassHopper είναι μία πλατφόρμα ανάπτυξης πρακτόρων. Πρωτοεκδόθηκε το 1998 βάση της τυποποιημένης μορφής δημιουργίας πρακτόρων OMG MASIF. Από τότε η IKV συνεχώς βελτιώνει το προϊόν. Το Grasshopper είναι παγκοσμίως αναγνωρισμένο σαν μία δυναμική πλατφόρμα πρακτόρων. Δίνει την δυνατότητα στον χρήστη να δημιουργήσει μία πληθώρα εφαρμογών βασισμένη την τεχνολογία αυτή.

Οι τομείς στους οποίους κυρίως χρησιμοποιείται είναι στο ηλεκτρονικό εμπόριο, στην εύρεση δυναμικών πληροφοριών, στις ανεπτυγμένες τηλεπικοινωνιακές υπηρεσίες. Το Grasshopper επιτρέπει στους πράκτορες να κινούνται ανάμεσα σε διαφορετικά συστήματα και να εκτελούν διάφορες εντολές.

Η πλατφόρμα αυτή ουσιαστικά υποστηρίζει κινητούς πράκτορες σύμφωνα με το μοντέλο της FIPA υποστηρίζοντας το πρότυπο MASIF. Αυτό που ενδιαφέρει αυτή την εργασία είναι η συγκρότηση του ΣΠΠ που προτείνουν. Το ΣΠΠ έχει ως δομικές μονάδες τα πρακτορεία (agencies) και τους πράκτορες (IKV++ GmbH, 1999b).

Ένα πρακτορείο είναι μια διαδικασία της Java, που διευκολύνει και ελέγχει την εκτέλεση, διεύθυνση, μεταφορά, επικοινωνία κ.α. των πρακτόρων Κάθε πρακτορείο καλύπτει τις ακόλουθες υπηρεσίες:

- Ασφάλεια. Απ' τη στιγμή που ένας κινητός πράκτορας πρέπει να ληφθεί υπόψιν σαν εξωγενές στοιχείο σε επισκεπτόμενες περιοχές, ο βασικός σκοπός για ένα πρακτορείο είναι να προστατέψει τις περιοχές από μη εξουσιοδοτημένη πρόσβαση πρακτόρων. Από τη μια πλευρά το Grasshopper παρέχει εξωτερική προστασία μηχανημάτων που επιτρέπουν την κρυπτογράφηση (code and state) κατά τη διάρκεια της αποδημίας πρακτόρων, χρησιμοποιώντας το πρωτόκολλο Secure Socket Layer (SSL). Αν απαιτούνται επίσης απομακρυσμένες επιδράσεις μεταξύ Grasshopper οντοτήτων (πράκτορες, πρακτορεία, όχι βασιζόμενα σε πράκτορες συστατικά στοιχεία), αυτές μπορούν επίσης να προστατευθούν διαμέσου SSL. Με πιστοποιητικά μέσα, ένα πρακτορείο είναι σε θέση να αναγνωρίσει και να βεβαιώσει τη γνησιότητα του πράκτορα. Από την άλλη πλευρά, εσωτερικοί μηχανικοί ασφαλείας επιτρέπουν την πρόσβαση για έλεγχο μέσα στο πρακτορείο.
- Εγγραφή. Κάθε πράκτορας εγγράφεται σε όλα τα τοπικά λειτουργικά πρακτορεία τα οποία διατηρούν πληροφορίες σχετικές με την διεύθυνσή του και τις ικανότητές του για την επικοινωνία του με άλλους πράκτορες.
- Επιμονή. Για να σώσει τα σχετικά με τον πράκτορα δεδομένα σε περίπτωση που ένα σύστημα διαλυθεί, η υπηρεσία της επιμονής του πρακτορείου μπορεί να χρησιμοποιηθεί περιοδικά σώζοντας την εσωτερική κατάσταση από όλους τους τοπικούς λειτουργικά πράκτορες. Μ' αυτόν τον τρόπο οι πράκτορες μπορούν να συνεχίσουν τις λειτουργίες τους μετά την επαναλειτουργία του πρακτορείου.
- Διοίκηση. Ανάμεσα σε άλλα η υπηρεσία της διοίκησης είναι υπεύθυνη για δημιουργία, μετακίνηση, αναστολή/ επανάληψη και αντιγραφή πρακτόρων. Δια μέσου ενός γραφικού interface, οι διαχειριστές είναι πάντα ικανοί να παρέμβουν στις εκτελέσεις των πρακτόρων.

- Μεταφορά. Οι υπηρεσίες της μεταφοράς διευκολύνουν την τοποθέτηση σε σειρά από μια κατάσταση πρακτόρων, οι μεταφορείς του πράκτορα σε κάθε προορισμό πρακτορείου (με πρόσβαση στην υπηρεσία επικοινωνίας των πρακτορείων) και η επαναλειτουργία του πράκτορα σε κάθε προορισμό.

- Επικοινωνία. Από την άλλη πλευρά η υπηρεσία της επικοινωνίας διευκολύνει τον μεταφορέα ενός πράκτορα μεταξύ διαφορετικών πρακτορείων, με την υπηρεσία μεταφοράς. Από την άλλη πλευρά η υπηρεσία επικοινωνίας είναι υπεύθυνη για την αλληλεπίδραση μεταξύ απομακρυσμένων πρακτόρων. Σ' αυτών τον τρόπο ως εξήγηση των παραπάνω το Grasshopper πετυχαίνει μια ακεραιότητα για την παραδοσιακή προσέγγιση του πελάτη/ εξυπηρετητή και τεχνικού κινητού πράκτορα (χρησιμοποιώντας την αποδημία των πρακτόρων)

Προαιρετικά, ένα πρακτορείο μπορεί να καταγραφεί μόνο του σε μια εγγραφή συστατικού μέρους, που καλείται περιοχή γραφείου μητρώου. Η περιοχή μητρώου είναι μια υπηρεσία κεντρικών πληροφοριών που διατηρεί πληροφορίες για όλα τα πρακτορεία εγγράφων τόσο καλά όσο λειτουργούν όλοι οι πράκτορες μέσα στα πρακτορεία. Η ρύθμιση από όλα τα πρακτορεία που εγγράφονται σε μια μοναδική περιοχή γραφείου μητρώων φτιάχνουν μια περιοχή. Η εγγραφή από έναν πράκτορα είναι αυτόματα εκτελεσμένη από τον φιλοξενούμενο πράκτορα και είναι απολύτως φανερή για τον πράκτορα. Σε μια αποδημία πράκτορα κάθε τοποθεσία πληροφοριών είναι αυτόματα ενημερωμένη μέσα στο γραφείο μητρώου. Μ' αυτόν τον τρόπο ένας διαχειριστής ή πράκτορας έχει πάντα τη δυνατότητα να βρει πληροφορίες για την περιοχή ή να εντοπίσει ειδικούς πράκτορες. Η περιοχή γραφείων μητρώου προβάλλει τις ακόλουθες υπηρεσίες/ συστατικά μέρη:

- Διοίκηση. Η υπηρεσία της διοίκησης είναι υπεύθυνη για τον εντοπισμό πρακτόρων μέσα σε μια περιοχή.

- Επικοινωνία. Ισοδύναμο στις ικανότητες επικοινωνίας ενός πρακτορείου αυτή η υπηρεσία διευκολύνει τις αλληλεπιδράσεις ανάμεσα στην περιοχή

γραφείων μητρώου και υπάρξεων. Οι διαχειριστές μπορούν να χειρίζονται και να ελέγχουν τη διαδικασία του γραφείου μητρώου.

Υπάρχουν δύο είδη πρακτόρων: οι κινητοί πράκτορες και οι στατικοί ή ακίνητοι πράκτορες. Το κοινό τους στοιχείο είναι η αυτονομία δηλαδή είναι και οι δύο προγράμματα τα οποία λειτουργούν αυτόνομα εκ μέρους ενός ατόμου ή ενός οργανισμού. Κάθε πράκτορας είναι μια νηματοποιημένη (threaded) κλάση που υποστηρίζει διάφορες μεθόδους.

Οι κινητοί πράκτορες μπορούν να αποδημούν σε διάφορες τοποθεσίες του δικτύου προκειμένου να πάρουν πληροφορίες και η διαφορά τους από το κλασικό παράδειγμα client/server είναι ότι οι κινητοί πράκτορες μπορούν να αλλάξουν τοποθεσία κατά τη διάρκεια της λειτουργίας τους σε αντίθεση με τον παραδοσιακό κώδικα ο οποίος στέλνει το πρόγραμμα στη περιοχή πριν καν αρχίσει η λειτουργία του και παραμένει εκεί μόνιμα.

Κατά τη διάρκεια της ζωής τους οι πράκτορες μπορούν να βρίσκονται σε μία από τις καταστάσεις: α) ενεργοί, εκτελούν το έργο τους, β) σε διαθεσιμότητα, όταν διακόπτεται προσωρινά η λειτουργία τους και γ) εκτός λειτουργίας, οι πληροφορίες του πράκτορα αποθηκεύονται στο σκληρό δίσκο. Έτσι ο πράκτορας μπορεί αργότερα να ενεργοποιηθεί ξανά.

Οι πράκτορες σύμφωνα με την πλατφόρμα αυτή είναι κλάσεις Java. Όλη η εκτέλεση του πράκτορα βασίζεται σε ένα μόνο νήμα πράγμα που είναι μειονέκτημα καθώς ο πράκτορας έτσι είναι μονοδιάστατος, η υλοποίησή του σε ένα μόνο επίπεδο πολύπλοκη.

2.3.6. ZEUS

Το ZEUS είναι μια μοντέρνα αρχιτεκτονική ΣΠΠ η οποία αναπτύχθηκε στα εργαστήρια του Οργανισμού Τηλεπικοινωνιών της Βρετανίας (BT: British Telecommunication). Αποτελεί. όπως το AgentBuilder μια ολοκληρωμένη πλατφόρμα ανάπτυξης συστημάτων πολλαπλών πρακτόρων με γραφικό περιβάλλον σχεδίασης των πρακτόρων και παρακολούθησης της εκτέλεσής των.

Η λογική της ομάδας ανάπτυξης της πλατφόρμας και αρχιτεκτονικής αυτής ήταν να χρησιμοποιηθούν διαθέσιμες τεχνολογίες για την επικοινωνία, συνεργασία και διαπραγμάτευση μεταξύ των ΕΠ και το ερευνητικό έργο να είναι πάνω στην αρχιτεκτονική και τους ρόλους των ΕΠ.

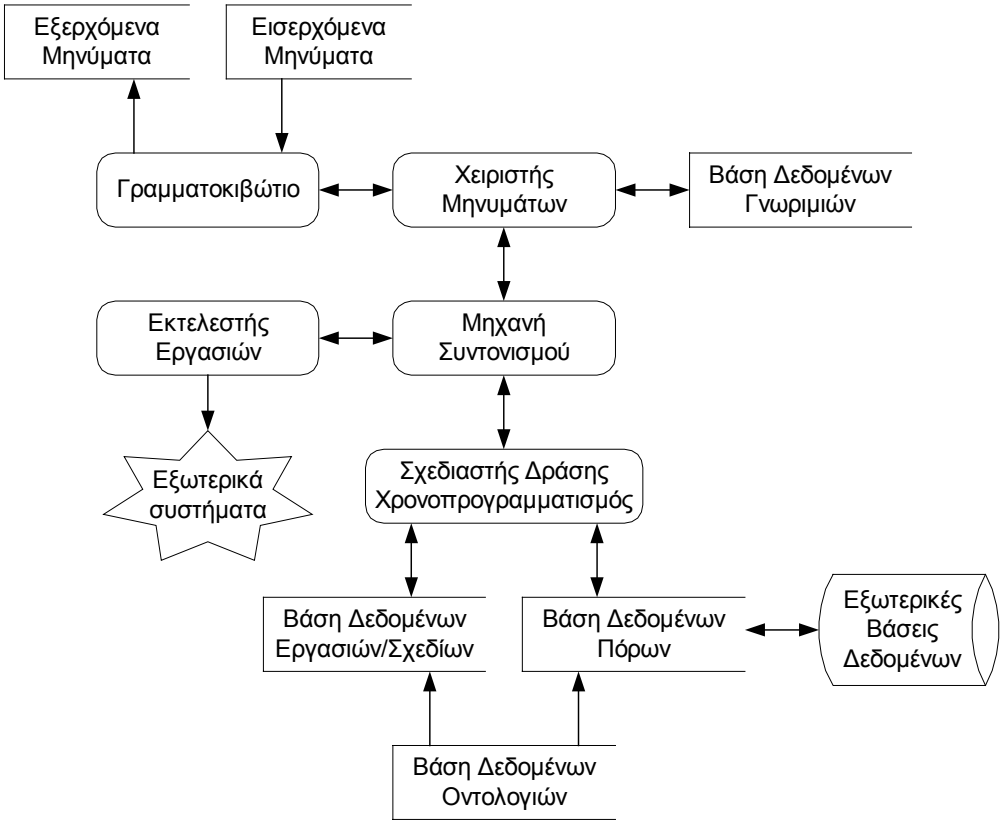
Σύμφωνα με τους Collis και Ndumu (1999b) οι ΕΠ για το Zeus ΣΠΠ έχουν τα εξής χαρακτηριστικά:

- Συνεργάζονται με άλλους, οδηγούνται από τους σκοπούς τους και λειτουργούν με λογική.
- Είναι ειλικρινείς στις συνεννοήσεις τους με άλλους πράκτορες.
- Πολυπράγμονες - ευέλικτοι, μπορούν να έχουν πολλούς σκοπούς και να εκτελούν μια ποικιλία εργασιών και
- προσωρινά μόνιμοι.

Το Zeus παρέχει κάποια έτοιμα εργαλεία και μεθόδους για την ανάπτυξη πρακτόρων. Αυτά είναι:

- Η KQML γλώσσα επικοινωνίας
- Ένα ασύγχρονο βασισμένο σε θύρες TCP/IP σύστημα ανταλλαγής μηνυμάτων
- Ένας επεξεργαστής οντολογιών
- Μια γλώσσα βασισμένη σε πλαίσια για την αναπαράσταση γνώση σε κάποιο τομέα
- Ένα γενικού σκοπού σύστημα σχεδιασμού και χρονοπρογραμματισμού κατάλληλο για τυπικούς προσανατολισμένους στο στόχο τομείς εφαρμογών
- Μια μηχανή συντονισμού η οποία ελέγχει την κοινωνική συμπεριφορά του ΕΠ (πότε και πως αλληλεπιδρά με άλλους ΕΠ, τι τύπους συμβολαίων μπορεί να επισυνάψει με τον καθένα)

- Μια βιβλιοθήκη προκαθορισμένων πρωτοκόλλων συντονισμού (το contract net και διάφορα άλλα πρωτόκολλα δημοπρασίας)
- Έναν αριθμό προκαθορισμένων σχέσεων οργάνωσης (π.χ. *ανώτερος, υφιστάμενος, συνεργαζόμενος, συνάδελφος*)
- Μηχανισμούς αναπαράστασης γνώσης και βάσεις δεδομένων για την περιγραφή και αποθήκευση των ικανοτήτων κάθε πράκτορα.



Εικόνα 2-7: Αρχιτεκτονική του στοιχειώδους πράκτορα του ZEUS

Η δουλειά που έγινε στον ΒΤ είναι η αρχιτεκτονική των ΕΠ η οποία παρουσιάζεται σχηματικά στην Εικόνα 2-7, η οποία αποτυπώνει ένα ιδιότυπο διάγραμμα ροής δεδομένων. Πέντε βασικές διεργασίες εκ των οποίων η κάθε μία τρέχει στο δικό της νήμα (thread) ολοκληρώνουν έναν ΕΠ Ο πράκτορας στην πραγματικότητα τρέχει περισσότερα νήματα από αυτές τις διεργασίες (π.χ. τα εισερχόμενα μηνύματα είναι ένας εξυπηρετητής ο οποίος τρέχει στο δικό του νήμα).

Αναλυτικά ο πράκτορας του ZEUS αποτελείται από:

- Το *γραμματοκιβώτιο* χειρίζεται την επικοινωνία μεταξύ των πρακτόρων. Δύο μόνιμα νήματα εκτέλεσης είναι υπεύθυνα για την συλλογή (εξυπηρετητής) και αποστολή μηνυμάτων.
- Ο *χειριστής μηνυμάτων* επεξεργάζεται τα εισερχόμενα μηνύματα και τα προωθεί προς το κατάλληλο τμήμα του πράκτορα. Τα μηνύματα είναι κάποιου τύπου από τους 20 διαθέσιμους και η σύνταξή τους ακολουθεί τη σύσταση της FIPA για ACL του 1997.
- Η *μηχανή συντονισμού* η οποία αποφασίζει σχετικά με τους σκοπούς-στόχους του πράκτορα. Επίσης είναι υπεύθυνη για το συντονισμό της δράσης του με αυτή των άλλων πρακτόρων χρησιμοποιώντας διαθέσιμα πρωτόκολλα και στρατηγικές. Ο σχεδιασμός της δράσης του πράκτορα γίνεται με τη χρήση γράφων αντί κανόνων. Αυτή η επιλογή έγινε κυρίως επειδή καθώς καταφτάνουν συμβόλαια ή απαντήσεις πρακτόρων το σχέδιο δράσης προχωρά, αν είχε γίνει η δουλειά με κανόνες θα ήταν δύσκολο να αλλάζουν συνεχώς τα δεδομένα εισόδου αλλά και αδύνατο το πισωπάτημα (backtracking).
- Η *βάση δεδομένων γνωριμιών* η οποία περιγράφει τη σχέση του πράκτορα με τους άλλους στην κοινότητα όπως και τις πεποιθήσεις του σχετικά με τις ικανότητές τους.
- Ένα *σχεδιαστή δράσης – χρονοπρογραμματιστή*, ο οποίος αναλαμβάνει να κατανείμει πόρους και χρόνο σε εργασίες που αποφασίστηκε από τη *μηχανή συντονισμού* να έρθουν σε πέρας, χρησιμοποιώντας διαθέσιμα πλάνα. Ουσιαστικά ο σχεδιαστής διατηρεί ένα ημερολόγιο όπου υπολογίζει ποιος πράκτορας θα κάνει ποια δουλειά, τι πόρο θα χρησιμοποιήσει ο κάθε πράκτορας και για πόση ώρα. Έτσι αν κάποιοι πράκτορες που συμμετέχουν σε μια ομάδα υπολογίζουν ότι αυτός που τους στέλνει τα συμβόλαια εξασφαλίζει και τη διαθεσιμότητα των πόρων.

- Μια *βάση δεδομένων πόρων* η οποία αφενός διατηρεί μια λίστα πόρων διαθέσιμων στον πράκτορα και αφετέρου υποστηρίζει ένα μηχανισμό διεπαφής με εξωτερικές βάσεις δεδομένων ο οποίος της επιτρέπει δυναμικά να συνδέεται με αυτές και να τις χρησιμοποιεί.
- Μια *βάση οντολογιών* η οποία αποθηκεύει τη λογική ερμηνεία κάθε πεποίθησης του πράκτορα (περιορισμούς τιμών, σχέσεις χαρακτηριστικών μιας πεποίθησης και άλλων πεποιθήσεων, κλπ)
- Μια *βάση εργασιών-σχεδίων* η οποία παρέχει λογικές περιγραφές των εργασιών που ξέρει να φέρει σε πέρας ο πράκτορας
- Ένα *εκτελεστή εργασιών* ο οποίος διατηρεί το εσωτερικό ρολόι του πράκτορα, αρχικοποιεί, σταματά και παρακολουθεί εργασίες που δρομολογήθηκαν από τον *σχεδιαστή δράσης – χρονοπρογραμματιστή*. Ακόμα, ενημερώνει τον *σχεδιαστή δράσης – χρονοπρογραμματιστή* για την επιτυχή ή όχι πραγματοποίση των δρομολογημένων εργασιών. Ο εκτελεστής εργασιών μπορεί να καλέσει και εξωτερικές συναρτήσεις ή συστήματα.

Με λίγα λόγια η λειτουργία ενός πράκτορα περιγράφεται ως εξής: Ένα εισερχόμενο μήνυμα λαμβάνεται από το γραμματοκιβώτιο και προωθείται στο *χειριστή μηνυμάτων* για επεξεργασία. Αν αυτό το μήνυμα αρχικοποιεί μια εργασία τότε ο χειριστής το στέλνει στη μηχανή συντονισμού η οποία θα αποφασίσει αν θα γίνει στόχος του πράκτορα η πραγματοποίηση αυτής της εργασίας και αν ναι να προωθηθεί στον *σχεδιαστή* ο οποίος θα δημιουργήσει ένα σχέδιο δράσης.

Αν η εργασία γίνει στόχος του πράκτορα και δεν βρεθεί ένα σχέδιο δράσης το οποίο να χρειάζεται πόρους και εργασία διαθέσιμους στον πράκτορα τότε καλείται η *μηχανή συντονισμού* να βρει ένα ή περισσότερους πράκτορες με τη βοήθεια των οποίων θα λυθεί το πρόβλημα. Με τη βοήθεια συμβολαίων η μηχανή συντονισμού (με τη χρήση των *γραμματοκιβώτιο* και *χειριστή μηνυμάτων*) προσπαθεί να φτιάξει την κατάλληλη ομάδα επιλέγοντας

πράκτορες από την βάση γνωριμιών ή με ερώτημα σε έναν πράκτορα υποστήριξης. Αν η ομάδα δημιουργηθεί μέσα στα χρονικά περιθώρια της εργασίας τότε αρχίζει η εκτέλεση. Σε όλη αυτή τη διαδικασία ο *σχεδιαστής* κρίνει τα συμβόλαια που επιστρέφονται ως προς το κόστος τους, τον χρόνο εκτέλεσης και τα απορρίπτει ή τα δέχεται.

Οι πράκτορες υποστήριξης είναι δύο τύπων και είναι απαραίτητοι για την λειτουργία του ΣΠΠ. Ο πρώτος είναι ο εξυπηρετητής ονομάτων (nameserver) και ξέρει τις διευθύνσεις όλων των πρακτόρων. Ο δεύτερος είναι ο παροχέας πληροφοριών πρακτόρων (facilitator) και ξέρει τι ικανότητες έχουν όλοι οι πράκτορες. Για να κάνει σωστά τη δουλειά του κάθε τόσο «ρωτάει» όλους τους πράκτορες για την κατάστασή τους.

Η πλατφόρμα αυτή χαρακτηρίζεται γενικά από την ποικιλία διαθέσιμων τεχνολογιών αλλά έχει έναν πολύ καλά ορισμένο τρόπο συντονισμού των πρακτόρων ο οποίος δεν περιλαμβάνει multicasting κάνοντας έτσι χρονοβόρο τον συντονισμό τους, όπως και δεν εγγυάται την επιτυχία στις διαπραγματεύσεις.

2.4. Μεθοδολογίες Ανάπτυξης ΣΠΠ

Πολλές μεθοδολογίες για την ανάπτυξη ενός ΣΠΠ έχουν προταθεί στη διεθνή βιβλιογραφία. Λίγες όμως από αυτές καλύπτουν όλη τη διαδικασία ανάπτυξης ενός ΣΠΠ, δηλαδή από την διατύπωση των απαιτήσεων του συστήματος μέχρι την οριστική λειτουργία του και συντήρησή του. Οι πιο πολλοί κατασκευαστές πλατφορμών ανάπτυξης ΣΠΠ προτείνουν και μια μεθοδολογία ανάπτυξης του ΣΠΠ. Συνήθως όμως αυτή περιορίζεται στην περιγραφή της λειτουργίας των εργαλείων που παρέχουν στο χρήστη-μηχανικό. Μόνο ο οργανισμός που παράγει το agentTool εργαλείο (AFIT, ινστιτούτο τεχνολογίας της Πολεμικής Αεροπορίας των ΗΠΑ), το οποίο δεν αναφέρεται στην περιγραφή διαφόρων πλατφορμών λόγω της ομοιότητάς του με το AgentBuilder και λόγω της ελλιπούς τεκμηρίωσής του, αναφέρει μία πειστική μεθοδολογία για την ανάπτυξη ενός ΣΠΠ την MaSE (Wood και DeLoach, 2000). Μια άλλη

σύγχρονη μεθοδολογία, η οποία θα παρουσιαστεί παρακάτω σε λεπτομέρεια, είναι η Gaia (Wooldridge και άλλοι, 2000). Τέλος, οι Collis και Ndumu (1999c) έχουν κάνει μια καλή δουλειά στην μοντελοποίηση ρόλων. Οι ρόλοι είναι μια ιδέα η οποία υπάρχει σε όλες τις σύγχρονες μεθοδολογίες ανάπτυξης ΣΠΠ (Kendall, 1998).

2.4.1. Ρόλοι Πρακτόρων

Ο φορμαλισμός των ρόλων των πρακτόρων επιτρέπει την εύκολη μοντελοποίηση σχεδιασμό και υλοποίησή τους. Το μοντέλο ρόλων προβλέπεται στον αντικειμενοστραφή τρόπο σχεδίασης και μοντελοποίησης συστημάτων (Kristensen, 1996). Οι ρόλοι πρέπει να (Collis και Ndumu 1999c):

- Είναι *τμηματοποιημένοι*: ένας ρόλος ορίζει ένα σύνολο οντοτήτων όλοι εκ των οποίων μπορούν να πάρουν την ίδια θέση σε μια επαναλαμβανόμενη δομή. Έτσι όλοι οι πράκτορες που μπορούν να αναλάβουν ένα συγκεκριμένο ρόλο θα μπορούσαν να εναλλαχθούν σε πολλαπλές λειτουργίες αυτού του ρόλου.
- Έχουν *υψηλή συνοχή*: για να προαγάγει την τμηματικότητα ένας ρόλος πρέπει να περιλαμβάνει ένα καλά ορισμένο σύνολο από αρμοδιότητες και λειτουργίες.
- Είναι *συγκεκριμένοι*: ένας ρόλος δεν μπορεί να έχει αυξημένες-πρόσθετες αρμοδιότητες.
- Είναι *ολοκληρωμένοι*: Ένας ρόλος δεν πρέπει να είναι τετριμμένος. Οι τετριμμένοι ρόλοι πρέπει να συγχωνεύονται με άλλους ρόλους.
- *Συνδέονται ελάχιστα*: Οι εξαρτήσεις μεταξύ ρόλων πρέπει να ελαχιστοποιούνται.

Στην περίπτωση κοινωνικών πρακτόρων κάποιοι πράκτορες μπορεί να αναλαμβάνουν περισσότερους από έναν ρόλους. Για παράδειγμα ο πράκτορας

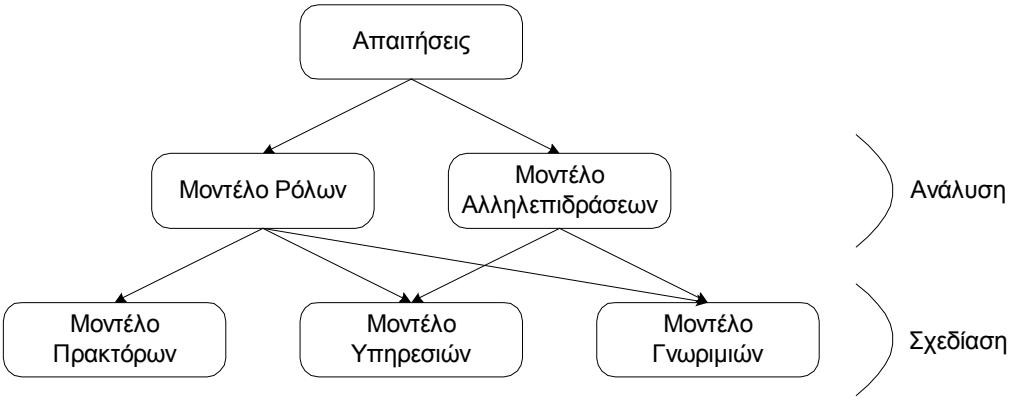
που αναλαμβάνει τον ρόλο «οδηγός φορτηγού» μπορεί να παίζει και τον ρόλο «φορτοεκφορτωτής».

2.4.2. Gaia

Η μεθοδολογία Gaia μπορεί να χρησιμοποιηθεί για την ανάπτυξη ΣΠΠ με τα εξής χαρακτηριστικά-περιορισμούς:

- οι πράκτορες χρησιμοποιούν υπολογιστικούς πόρους,
- το ΣΠΠ δεν θα έχει συγκρούσεις συμφερόντων των ΕΠ,
- είναι ετερογενείς (οι πράκτορες μπορεί να προγραμματιστούν με διαφορετικές γλώσσες προγραμματισμού, τεχνικές ή αρχιτεκτονικές),
- η οργανωσιακή δομή του ΣΠΠ είναι στατική (οι σχέσεις των πρακτόρων δεν αλλάζουν στο χρόνο εκτέλεσής τους),
- οι ικανότητες των πρακτόρων και οι υπηρεσίες που παρέχουν είναι στατικές και
- το ΣΠΠ περιέχει έναν σχετικά μικρό αριθμό διαφορετικών τύπων πρακτόρων.

Σύμφωνα με τη Gaia από τη στιγμή που διατυπωθούν οι απαιτήσεις ενός ΣΠΠ η φάση του σχεδιασμού και μοντελοποίησης του συστήματος χωρίζεται σε δύο βήματα: α) την ανάλυση (όπου δημιουργείται μια εικόνα του συστήματος και της οργανωτικής δομής των πρακτόρων) και β) τη σχεδίαση (ελαχιστοποίηση του επίπεδου αφαίρεσης έτσι ώστε με αντικειμενοστραφή σχεδιασμό να είναι δυνατή η υλοποίηση του συστήματος). Στην διάρκεια των βημάτων αυτών δημιουργούνται δύο (στο βήμα της ανάλυσης) και τρία (στο βήμα της υλοποίησης) μοντέλα αντίστοιχα με τις σχέσεις που φαίνονται στην Εικόνα 2-8:



Εικόνα 2-8: Σχέσεις μεταξύ των μοντέλων της μεθοδολογίας Gaia

- *Μοντέλο Ρόλων*: Αναγνώριση των ρόλων-κλειδιών του ΣΠΠ. Εδώ αναγνωρίζονται ρητώς οι πόροι και οι λειτουργίες που σχετίζονται με κάθε ρόλο. Συγκεκριμένα, ορίζονται οι ρόλοι, οι λειτουργίες-μεθόδους που θα μπορεί να εκτελέσει κάθε ρόλος, οι πόροι που θα του είναι διαθέσιμοι ή τους οποίους θα αξιοποιεί και τέλος οι ρόλοι με τους οποίους συνεργάζεται ο κάθε ρόλος.
- *Μοντέλο Αλληλεπιδράσεων*: Εδώ αναγνωρίζονται οι σχέσεις μεταξύ των ρόλων. Το μοντέλο ορίζει πρωτόκολλα αλληλεπιδράσεων μεταξύ των ρόλων. Έτσι, οι ρόλοι με τους οποίους συνεργάζεται κάθε ρόλος σχετίζονται και με το πρωτόκολλο αλληλεπίδρασης μεταξύ τους.
- *Μοντέλο Πρακτόρων*: Εδώ αναγνωρίζονται οι τύποι των πρακτόρων του ΣΠΠ. Ουσιαστικά κάθε τύπος πράκτορα θα αναλάβει έναν ή παραπάνω από τους προηγουμένως ορισμένους ρόλους.
- *Μοντέλο Υπηρεσιών*: Εδώ ορίζονται οι λειτουργίες που θα εκτελεί κάθε ρόλος. Συγκεκριμένα αναφέρονται η λειτουργία, ποιος ρόλος την υλοποιεί και ποιες είναι οι συνθήκες κάτω από τις οποίες είναι εφικτή η εκτέλεση της λειτουργίας αυτής.
- *Μοντέλο Γνωριμιών*: Εδώ ορίζονται οι δίοδοι επικοινωνίας μεταξύ τύπων πρακτόρων, δηλαδή ποιοι από τους τύπους των πρακτόρων του

συστήματος αλληλεπιδρούν και με ποιους. Δεν ενδιαφέρει το είδος ή το περιεχόμενο των αλληλεπιδράσεων μόνο το ποιος αλληλεπιδρά με ποιον.

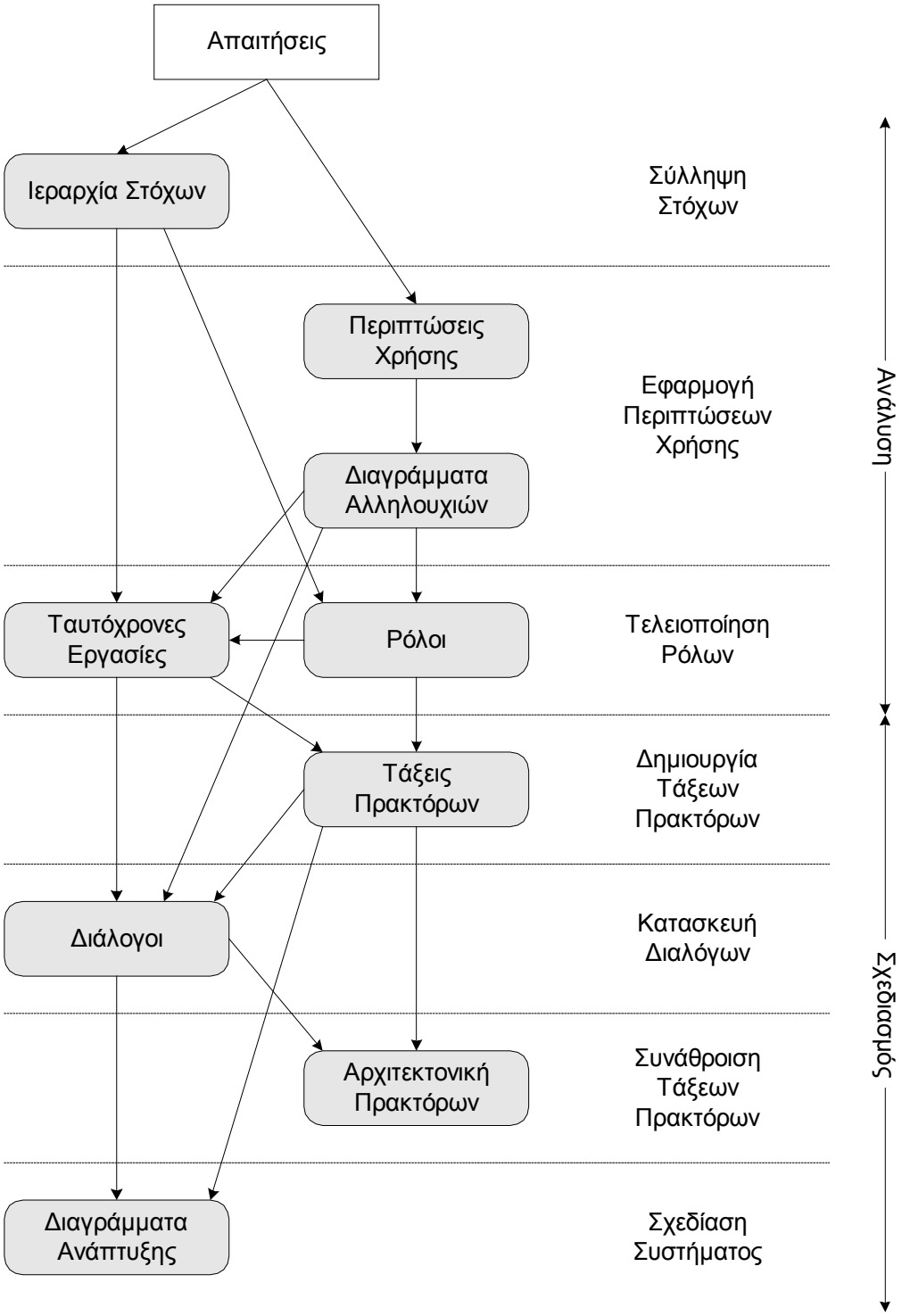
Γενικά η Gaia προσπαθεί κυρίως να ορίσει σε ένα επίπεδο αφαιρετικό τη λειτουργία ενός ΣΠΠ. Δεν την ενδιαφέρουν λεπτομέρειες και τρόποι υλοποίησης αλλά ορίζει ξεκάθαρα το ποιος κάνει τι και με ποιον σε ένα ΣΠΠ. Είναι κατά τη γνώμη του γράφοντα ένας τρόπος μοντελοποίησης ενός ΣΠΠ αλλά απέχει λίγο από την πλήρη σχεδίαση ενός ολοκληρωμένου συστήματος.

2.4.3. MaSE

Η μεθοδολογία MaSE μπορεί να χρησιμοποιηθεί για την ανάπτυξη ΣΠΠ με τα εξής χαρακτηριστικά-περιορισμούς:

- το περιβάλλον λειτουργίας των ΕΠ είναι κλειστό και όλες οι εξωτερικές διεπαφές κελυφοποιούνται με τη χρήση ενός ΕΠ,
- η μεθοδολογία δεν αφορά συστήματα όπου οι πράκτορες μπορούν να δημιουργηθούν, καταστραφούν ή μετακινηθούν κατά τη διάρκεια της εκτέλεσής τους
- οι αλληλεπιδράσεις μεταξύ των πρακτόρων είναι ένας προς ένας – αποκλείεται η πολλαπλή εκπομπή μηνυμάτων (ένα μήνυμα δεν έχει περισσότερους του ενός αποδέκτες)
- το ΣΠΠ δεν πρέπει να περιέχει πάνω από 10 τάξεις (τύπους) ΕΠ

Η μεθοδολογία εφαρμόζεται αντίστοιχα με τη Gaia αφού τεθούν οι απαιτήσεις του συστήματος. Σύμφωνα με τη μεθοδολογία αυτή παράγονται τυποποιημένα έγγραφα σχεδίασης του ΣΠΠ βασισμένα σε γραφικά. Η MaSE είναι ανεξάρτητη αρχιτεκτονικής ΣΠΠ, αρχιτεκτονικής πρακτόρων, συστήματος μεταβίβασης μηνυμάτων και γλώσσας προγραμματισμού. Η μεθοδολογία ακολουθεί τα βήματα που φαίνονται στην Εικόνα 2-9. Και πάλι, αντίστοιχα με τη Gaia, τα βήματα είναι χωρισμένα στις φάσεις ανάλυσης και σχεδίασης όπως φαίνεται στην εικόνα. Τα βήματα αυτά είναι:



Εικόνα 2-9: Η μεθοδολογία MaSE

- *Ιεραρχία Στόχων*: Αυτά τα διαγράμματα αποσαφηνίζουν τους στόχους του ΣΠΠ (απαιτήσεις) και στόχος του βήματος αυτού είναι τόσο η αναγνώριση

όσο και η δόμηση των στόχων αυτών. Η δόμηση αυτή είναι ουσιαστικά το σπάσιμο των στόχων σε υποστόχους και η κατάληξη σε στοιχειώδεις στόχους (οι οποίοι πιθανόν να είναι και οι εργασίες που θα επιτελούν οι τύποι των πρακτόρων του ΣΠΠ).

- *Περιπτώσεις Χρήσης*: Οι περιπτώσεις χρήσης του συστήματος.
- *Διαγράμματα Αλληλουχιών*: Με τα UML διαγράμματα αυτά (sequence diagrams) δημιουργούνται σενάρια λειτουργίας του συστήματος. Τα σενάρια αυτά συνήθως αφορούν τα μηνύματα που θα ανταλλάσσουν οι διάφοροι ρόλοι-πράκτορες για κάθε περίπτωση χρήσης του ΣΠΠ.
- *Ταυτόχρονες Εργασίες*: Κάθε ρόλος περιλαμβάνει την εκτέλεση μιας ή περισσοτέρων εργασιών οι οποίες μπορεί να πρέπει να εκτελούνται παράλληλα. Με τη χρήση των διαγραμμάτων αυτών αποσαφηνίζονται αυτά τα θέματα
- *Ρόλοι*: Σχεδόν κάθε στόχος ορίζει και ένα ρόλο. Εδώ ορίζονται επίσης και οι αλληλεπιδράσεις μεταξύ των ρόλων.
- *Τάξεις Πρακτόρων*: Οι τάξεις των πρακτόρων μπορεί να περιλαμβάνουν έναν ή περισσότερους ρόλους
- *Διάλογοι*: Οι διάλογοι ουσιαστικά αφορούν τα πρωτόκολλα επικοινωνίας μεταξύ των πρακτόρων.
- *Αρχιτεκτονική Πρακτόρων*: Ορισμός της αρχιτεκτονικής των ΕΠ που θα απαρτίσουν το ΣΠΠ.
- *Διαγράμματα Ανάπτυξης*: Ποιοι πράκτορες θα τρέχουν που. Τα UML διαγράμματα αυτά (deployment diagrams) παρουσιάζουν πως θα λειτουργεί το ΣΠΠ.

Η MaSE μεθοδολογία προσπαθεί να καλύψει πλήρως την διαδικασία ανάλυσης και σχεδιασμού ενός ΣΠΠ. Η βασική της διαφορά με την Gaia είναι η πρόβλεψη για σενάρια και περιπτώσεις χρήσης (use-cases) των σχεδιαζόμενων ΣΠΠ. Ακόμα, οι δημιουργοί της μεθοδολογίας αυτής δείχνουν

ότι έχουν ένα πολύ καλό υπόβαθρο στη σχεδίαση συστημάτων, και ιδιαίτερα στη σχεδίαση αντικειμενοστραφών συστημάτων και στη χρήση της γλώσσας μοντελοποίησης UML, καθώς η μεθοδολογία καλύπτει σχεδόν εξολοκλήρου τη διαδικασία σχεδίασης συστημάτων πολλαπλών πρακτόρων με μοντέρνες τεχνικές και διαγράμματα.

Παρόλα αυτά, τα μειονεκτήματά της είναι ότι θέτει μεγαλύτερο περιορισμό στον αριθμό των τύπων πρακτόρων που θα συμμετέχουν στο ΣΠΠ και επίσης απαγορεύει τη δυναμική δημιουργία-καταστροφή πρακτόρων κατά τη διάρκεια λειτουργίας του ΣΠΠ.

3ο ΚΕΦΑΛΑΙΟ

ΑΝΑΛΥΣΗ ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ ΤΟΥ ΣΠΠ

Η μεθοδολογία που χρησιμοποιήθηκε για την ανάπτυξη του προτεινόμενου ΣΠΠ δεν ήταν καμία από τις προτεινόμενες στη βιβλιογραφία λόγω των περιορισμών που αυτές θέτουν. Το προτεινόμενο ΣΠΠ για παράδειγμα προβλέπει (όπως θα φανεί παρακάτω) τη δυναμική λειτουργία της κοινότητας των πρακτόρων (νέοι πράκτορες μπορούν να εμφανιστούν ενώ μη ανταγωνιστικοί μπορεί να εξαφανιστούν). Όμως όλα τα καλά χαρακτηριστικά των μελετηθέντων μεθοδολογιών έχουν χρησιμοποιηθεί.

Ακολούθως, λειτούργησαν δύο ταυτόχρονα επίπεδα σχεδιασμού. Το πρώτο αφορούσε την υλοποίηση της μεθοδολογίας στρατηγικής διείσδυσης ενός προϊόντος στην αγορά με το προτεινόμενο ΣΠΠ, ενώ το δεύτερο αφορούσε την αρχιτεκτονική των πρακτόρων και του συστήματος. Έτσι, οι επόμενοι δύο παράγραφοι του κεφαλαίου περιγράφουν τη δουλειά που έγινε σε κάθε επίπεδο.

3.1. Μεθοδολογία

Η προτεινόμενη μεθοδολογία περιλαμβάνει τα εξής βήματα:

1. Δημιουργία της ιεραρχίας στόχων όπως αυτή περιγράφεται στην μεθοδολογία MaSE για την αναγνώριση των πρακτόρων που θα συμμετέχουν στο ΣΠΠ.

2. Περιγραφή των ρόλων των πρακτόρων και τις αλληλεπιδράσεις μεταξύ ρόλων (περιλαμβάνεται αυτό το βήμα σχεδόν σε όλες τις μεθοδολογίες της βιβλιογραφίας)
3. Περιγραφή της αρχιτεκτονικής του συστήματος με διαγράμματα τάξεων, ακολουθιών και καταστάσεων (σύμφωνα με την αντικειμενοστραφή γλώσσα μοντελοποίησης UML)
4. Σενάρια λειτουργίας του συστήματος (αντίστοιχα με τη μεθοδολογία MaSE)
5. Κωδικοποίηση
6. Συντήρηση

Τα βήματα αυτά ακολουθούν το ένα το άλλο και μετά το τελευταίο επιστρέφουμε στο πρώτο σύμφωνα με το σπειροειδές μοντέλο ανάπτυξης πληροφοριακών συστημάτων (Χριστοδουλάκης και Ξένος, 1995). Ο κύκλος συνεχίζεται μέχρις ότου το σύστημα γίνει ικανοποιητικό-λειτουργήσει σύμφωνα με τις προδιαγραφές του. Τα 1 και 2 βήματα αναλύονται στην επόμενη παράγραφο, ενώ τα 3 και 4 παρουσιάζονται το καθένα σε ξεχωριστή παράγραφο. Η κωδικοποίηση και πληροφορίες για μελλοντική συντήρηση παρουσιάζονται στο επόμενο κεφάλαιο ($4^{ο}$).

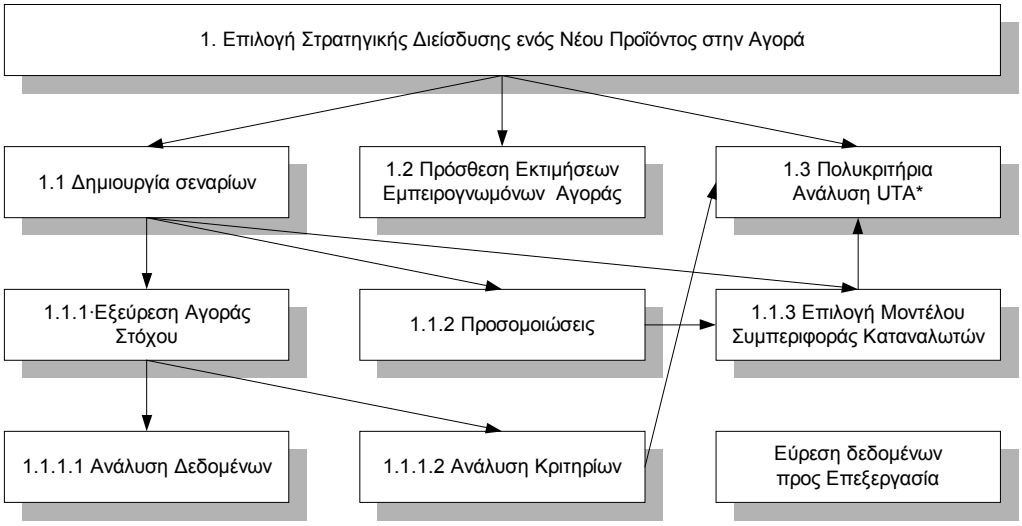
Η προτεινόμενη μεθοδολογία αυτή είναι πρωτότυπη και αφορά όλη τη διαδικασία ανάπτυξης ενός ΣΠΠ, από την ανάλυση και σχεδίαση μέχρι την υλοποίηση και συντήρηση. Συνδυάζει χαρακτηριστικά μεθοδολογιών ανάπτυξης ΣΠΠ (που παρουσιάστηκαν στο προηγούμενο κεφάλαιο) και την γλώσσα UML (Eriksson και Penker, 1998).

3.2. Σχεδιασμός του ΣΠΠ

Σε αυτή την παράγραφο παρουσιάζονται τα βήματα σχεδιασμού του προτεινόμενου ΣΠΠ.

3.2.1. Ιεραρχία Στόχων

Σύμφωνα με τη φάση αυτή της μεθοδολογίας έπρεπε να αναλυθούν οι εργασίες-στόχοι του συστήματος σε υποεργασίες-στόχους μέχρι να φτάσουμε σε στοιχειώδεις εργασίες-στόχους οι οποίοι θα αναληφθούν από ρόλους πρακτόρων. Στην Εικόνα 3-1 φαίνεται σχηματικά η δουλειά που έγινε.



Εικόνα 3-1: Διάγραμμα Ιεραρχίας Στόχων

Ο στόχος του συστήματος είναι η επιλογή στρατηγικής διείσδυσης ενός νέου προϊόντος στην αγορά (1). Άμεσοι υποστόχοι του συστήματος είναι οι:

1. Επιλογή Στρατηγικής Διείσδυσης
   - 1.1. Δημιουργία σεναρίων
     - 1.1.1. Εξεύρεση αγοράς στόχου
       - 1.1.1.1. Ανάλυση Δεδομένων
       - 1.1.1.2. Ανάλυση Κριτηρίων
     - 1.1.2. Προσομοιώσεις
     - 1.1.3. Επιλογή μοντέλου συμπεριφοράς καταναλωτών
   - 1.2. Πρόσθεση εκτιμήσεων εμπειρογνωμόνων της αγοράς

1.3. Πολυκριτήρια ανάλυση UTA*

Ακόμα προέκυψαν οι ενδιάμεσοι υποστόχοι οι οποίοι λειτουργούν επικουρικά στους προηγούμενους:

- Εύρεση δεδομένων προς επεξεργασία
    - Επιλογή βάσης δεδομένων
    - Φιλτράρισμα βάσης δεδομένων
- Αλληλεπίδραση με το χρήστη

Όλοι αυτοί οι υποστόχοι του συστήματος είναι υποψήφιοι ρόλοι πρακτόρων. Η επιλογή τους θα γίνει στην επόμενη παράγραφο όπου θα οριστούν οι ρόλοι.

3.2.2. Ρόλοι και Αλληλεπιδράσεις

Για την δημιουργία των ρόλων μπορούν να χρησιμοποιηθούν στοιχειώδεις ή αφαιρετικοί ρόλοι οι οποίοι μετά θα επεκτείνονται για να δημιουργήσουν τους ξεχωριστούς ρόλους των πρακτόρων του προτεινόμενου ΣΠΠ.

Αναγνωρίστηκαν τρεις στοιχειώδεις ρόλοι πρακτόρων οι οποίοι δημιούργησαν κατηγορίες-γενικούς τύπους πρακτόρων και αυτοί ήταν οι:

- Ρόλος πράκτορα έργου. Επιτελεί εργασίες-στοιχειώδεις στόχους του συστήματος
- Ρόλος πράκτορα διεπαφής. Αναλαμβάνει την αλληλεπίδραση του ΣΠΠ με τους ανθρώπινους πράκτορες
- Ρόλος πράκτορα πληροφοριοδότη. Αναλαμβάνει να βρει κατάλληλα δεδομένα προς επεξεργασία

Ο ρόλος πρακτόρων έργου χωρίζεται σε δύο κατηγορίες: ρόλος πράκτορα που επιτελεί ένα στοιχειώδη στόχο και ρόλος πράκτορα που συντονίζει κάποιους που επιτελούν στοιχειώδεις στόχους για να επιτευχθεί ένας σφαιρικός στόχος. Ο πρώτος ρόλος θα είναι «εργολήπτης» ενώ ο δεύτερος θα είναι ο «πολύπλοκος εργολήπτης». Αυτοί οι ρόλοι είναι απαραίτητο να διαχωριστούν

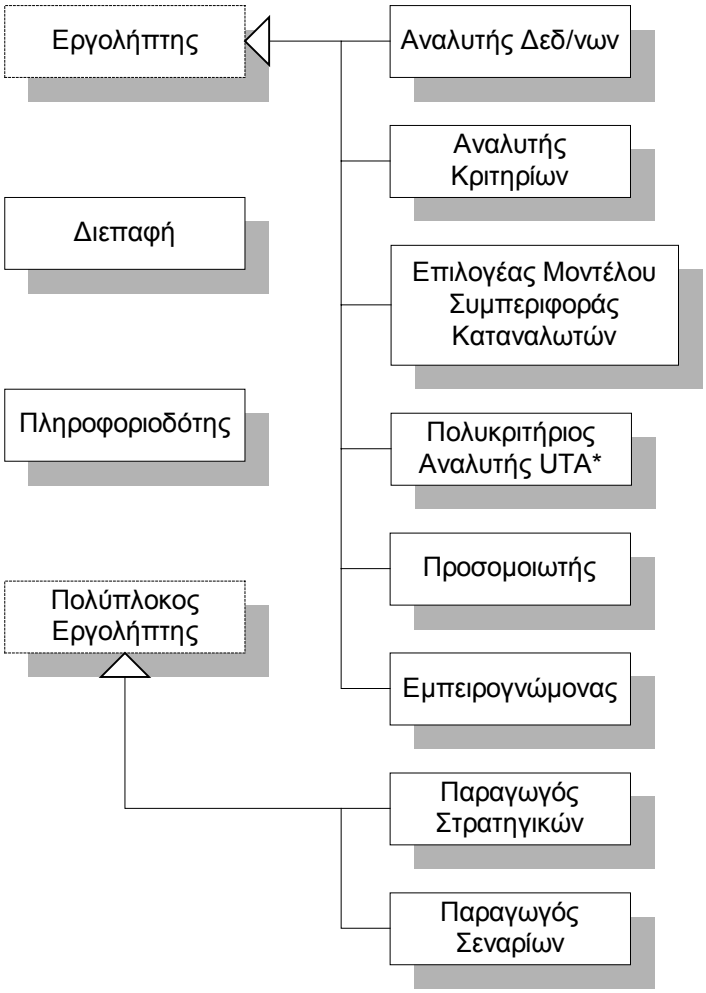
γιατί αλλάζει ο τρόπος λειτουργίας του ρόλου. Ο πρώτος επιτελεί κάποια στοιχειώδη εργασία, ενώ ο δεύτερος οργανώνει τους προηγούμενους.

Ουσιαστικά, για το προτεινόμενο σύστημα μόνο ο πρώτος ρόλος (με τις δύο του κατηγορίες) θα είναι αφαιρετικός και θα επεκτείνεται για να εξυπηρετήσει εξειδικευμένους ρόλους, οι οποίοι προκύπτουν από το διάγραμμα ιεραρχίας στόχων της προηγούμενης παραγράφου (ο μόνος υποστόχος που παραλείπεται είναι ο «εξεύρεση αγοράς στόχου» επειδή είναι τετριμμένος, ουσιαστικά είναι τίτλος για τους υποστόχους του), δημιουργώντας τους ρόλους (σε παρένθεση παρατίθεται το διεθνές όνομά του και η συντομογραφία του):

- Ρόλος αναλυτή δεδομένων (*Data Analysis, DA*)
- Ρόλος αναλυτή κριτηρίων (*Criteria Analysis, CA*)
- Ρόλος πολυκριτήριου αναλυτή (*UTASTAR, UTS*)
- Ρόλος επιλογέα μοντέλου συμπεριφοράς καταναλωτή (*Brand Choice, BC*)
- Ρόλος προσομοιωτή (*Simulator, SIM*)
- Ρόλος εμπειρογνώμονα αγοράς (*Market Expert, ME*)
- Ρόλος παραγωγού σεναρίων (*Scenario Generation, SG*)
- Ρόλος παραγωγού στρατηγικών (*Strategy Selection, StS*)

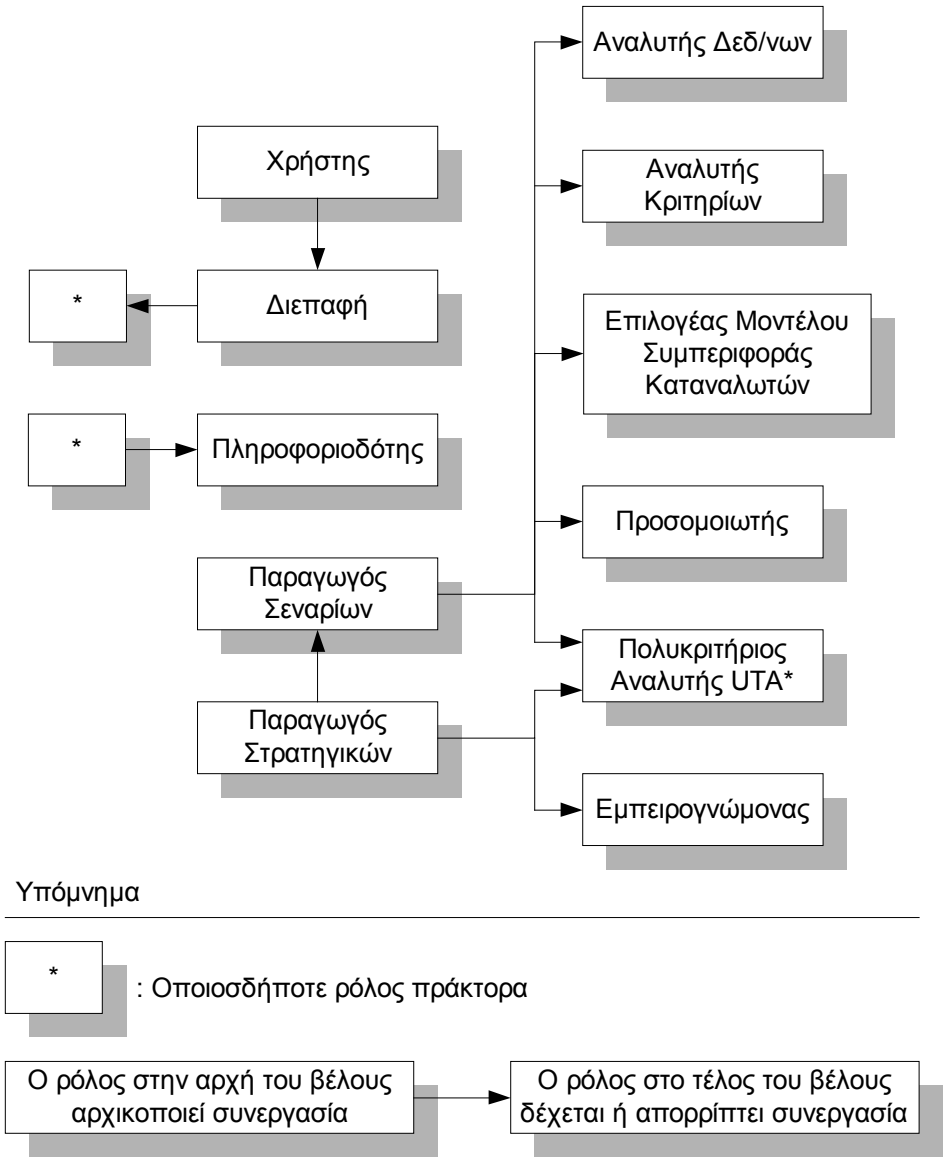
Εδώ πρέπει να παρατηρηθεί ότι οι ρόλοι δεν είναι μόνο οι στοιχειώδεις υποστόχοι του συστήματος. Ρόλοι πρακτόρων είναι όλες οι πιθανές λειτουργίες του συστήματος. Αυτό έχει γίνει σε συνάφεια με την προτεινόμενη αρχιτεκτονική για το σύστημα το οποίο θα περιέχει πολύπλοκους πράκτορες έργου οι οποίοι θα σχηματίζουν ομάδες πρακτόρων που επιτελούν στοιχειώδεις στόχους για να πετύχουν στόχους που περιέχουν υποστόχους. Έτσι αν κάποιος χρήστης θέλει να κάνει μια ανάλυση δεδομένων, αυτός μπορεί απλά να χρησιμοποιήσει τον πράκτορα με το ρόλο «αναλυτής δεδομένων». Αν θέλει να εφαρμόσει κάποια σενάρια τότε θέλει τον πολύπλοκο ρόλο πράκτορα «δημιουργό σεναρίων», ο οποίος θα αναλάβει να συντονίσει την ομάδα του με τους πράκτορες που αντιστοιχούν στους ρόλους

των στοιχειωδών στόχων (αναλυτής δεδομένων, πολυκριτήριος αναλυτής, αναλυτής κριτηρίων, επιλογέας μοντέλου συμπεριφοράς καταναλωτή, προσομοιωτής).



Εικόνα 3-2: Οι ρόλοι των πρακτόρων

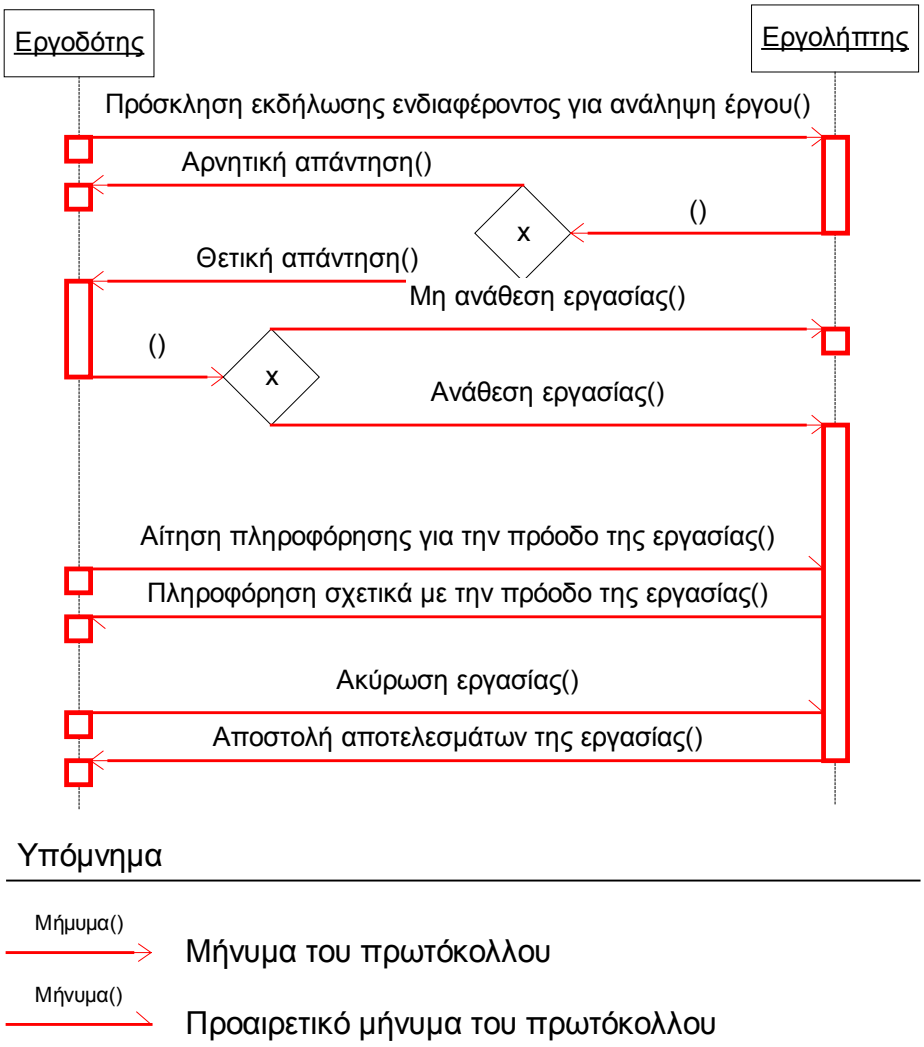
Οι ρόλοι των πρακτόρων τους συστήματος φαίνονται στην Εικόνα 3-2 (χρησιμοποιείται και το σύμβολο της κληρονομικότητας των διαγραμμάτων τάξεων της γλώσσας μοντελοποίησης UML για να συμβολίσει ρόλους που κληρονομούν χαρακτηριστικά άλλων, με διακεκομμένες γραμμές φαίνονται οι αφαιρετικοί ρόλοι).



Εικόνα 3-3: Αλληλεπιδράσεις μεταξύ των ρόλων

Στην Εικόνα 3-3 φαίνονται οι ρόλοι με τις αλληλεπιδράσεις τους (ποιοι επικοινωνούν με ποιους). Οι ρόλοι αλληλεπιδρούν αιτούμενοι ο ένας παραγωγή έργου από τον άλλο αρχικά. Στην περίπτωση που συμφωνήσουν συνεχίζεται η αλληλεπίδρασή τους, κατά την οποία θα ανταλλάξουν πληροφορίες. Άρα, μπορεί να αναφερθεί με σαφήνεια ότι μόνο στην περίπτωση που κάποιος ρόλος αιτείται κάτι από κάποιον άλλο θα υπάρχει αλληλεπίδραση μεταξύ ρόλων. Ακόμα, δεν αλληλεπιδρούν όλοι οι ρόλοι με όλους τους άλλους. Μόνο οι ρόλοι «διεπαφή» και «πληροφοριοδότης»

αλληλεπιδρούν με όλους τους άλλους ρόλους, η μεν διεπαφή για να ξεκινήσει κάποια εργασία που ζήτησε κάποιος χρήστης (η οποία μπορεί να είναι περίπλοκη ή στοιχειώδης, οπότε δικαιολογείται η δυνατότητα αλληλεπίδρασής του με όλους τους άλλους ρόλους), ο δε πληροφοριοδότης για να παρέχει πληροφορίες-δεδομένα σε οποιονδήποτε έχει ανάγκη.



Εικόνα 3-4: Πρωτόκολλο αλληλεπίδρασης των ρόλων

Οι αλληλεπιδράσεις των πρακτόρων θα ακολουθούν ένα καλά ορισμένο πρωτόκολλο ανάληψης-ολοκλήρωσης-παράδοσης έργου. Το πρωτόκολλο αυτό προβλέπει δύο συνομιλητές τον *εργοδότη* και τον *εργολήπτη*. Ένας εργοδότης

θα μπορεί να συναλλάσσεται ταυτόχρονα με περισσότερους του ενός εργολήπτες, όπως και ένας εργολήπτης θα μπορεί να έχει ταυτόχρονα περισσότερους του ενός εργοδότες.

Δύο νέοι αφαιρετικοί ρόλοι λοιπόν εμφανίζονται, αυτός του εργοδότη και αυτός του εργολήπτη. Οι ρόλοι που θα επεκτείνουν τον ρόλο του εργοδότη είναι αυτοί από τους οποίους ξεκινούν βέλη στην Εικόνα 3-3, ενώ οι ρόλοι που θα επεκτείνουν τον ρόλο του εργολήπτη είναι αυτοί στους οποίους καταλήγουν βέλη. Έτσι ο ρόλος του πληροφοριοδότη θα επεκτείνει μόνο το ρόλο του εργολήπτη, ενώ όλοι οι άλλοι θα επεκτείνουν και τους δύο ρόλους αυτούς (μπορούν να είναι εργοδότες ή/και εργολήπτες ανάλογα με το σημείο εκτέλεσης της εφαρμογής).

Η παρουσίαση του πρωτοκόλλου αλληλεπίδρασης των ρόλων εργοδότη-εργολήπτη γίνεται στην Εικόνα 3-4 σύμφωνα με το πρότυπο των Odell et al. (2000). Μια παρατήρηση που πρέπει να γίνει είναι ότι η εργασία μπορεί να ακυρωθεί και αφού έχει ανατεθεί, όμως και σε αυτή την περίπτωση επιστρέφονται όποια αποτελέσματα έχουν παραχθεί (για την περίπτωση που υπάρχουν όρια χρόνου στην εκτέλεση εργασιών, οπότε θα επιστρέφεται ότι πρόλαβε να παραχθεί).

3.3. Αρχιτεκτονική ΣΠΠ,

Μετά από την παράγραφο «Σχεδιασμός του ΣΠΠ» είναι πλέον αποσαφηνισμένοι οι ρόλοι των πρακτόρων και οι αλληλεπιδράσεις μεταξύ των τύπων. Σε αυτή την παράγραφο θα αναλυθούν περαιτέρω οι ρόλοι σε τάξεις και οι αλληλεπιδράσεις τους σε πρωτόκολλα. Πριν από αυτά όμως θα μελετηθεί η δομή και ο τρόπος οργάνωσης του ΣΠΠ.

3.3.1. Δομή και Οργάνωση του ΣΠΠ

Οι πράκτορες στο σύστημα αλληλεπιδρούν μέσω διαπρακτορικών μηνυμάτων (inter-agent messages). Το διαπρακτορικό μήνυμα είναι δομημένο με τέτοιο

τρόπο ώστε να επιτρέπει την αποτελεσματική μεταφορά σημαντικού και πληροφοριακού υλικού από ένα πράκτορα σε κάποιον άλλο με τελικό στόχο την επιτυχή συνεργασία των πρακτόρων για την επίτευξη του στόχου τους. Η αλληλεπίδραση των πρακτόρων που ανήκουν στην ομάδα ενός σύνθετου πράκτορα αναλύεται στην παράγραφο που αναφέρεται στην αρχιτεκτονική των σύνθετων πρακτόρων. Το μέσο επικοινωνίας, λοιπόν, που χρησιμοποιείται για την οργάνωση των ΕΠ στο παρόν σύστημα είναι η ανταλλαγή μηνυμάτων. Η οργάνωση της κοινωνίας γίνεται μέσω των παρακάτω θεμελιακών δομών αλληλεπίδρασης:

- Νέοι πράκτορες παρουσιάζονται στην κοινότητα ενώ άλλοι «πεθαίνουν» (τερματίζονται). Ένας πράκτορας ενημερώνεται πότε κάποιος πράκτορας παρουσιάζεται στην κοινότητα και πότε κάποιος αποχωρεί. Ένας καινούργιος πράκτορας παρουσιάζεται στην κοινότητα εκπέμποντας σε κάθε ενδιαφερόμενο πράκτορα ένα μήνυμα με την ταυτότητά του (ηλεκτρονική διεύθυνση, τύπο, ιδιαιτερότητες, κλπ). Έτσι όσον αφορά την άποψη ενός μηχανικού λογισμικού επιτυγχάνεται η τμηματικότητα (με χρησιμοποίηση πολλών πρακτόρων διαφορετικών ή ίδιων τύπων), επαναχρησιμοποίηση (ο πράκτορας δεν τερματίζεται μετά τη λειτουργία του όπως ένα σύνηθες αντικείμενο, συνεχίζει να παρέχει υπηρεσίες και ξέρουν οι άλλοι που θα τον βρούνε) και ευελιξία (αν μια υπηρεσία παρουσιάζει μεγάλη ζήτηση δημιουργούνται παραπάνω πράκτορες, ενώ στην αντίθετη περίπτωση τερματίζονται) του συστήματος.
- Κατά τη διάρκεια της διαδικασίας λύσης ενός προβλήματος η ενεργοποίηση των καταλλήλων πρακτόρων δημιουργεί δυναμικά μια οργανωτική δομή που ταιριάζει με τον τρέχοντα στόχο (η αποπεράτωση ενός έργου, η απόκτηση κάποιας πληροφορίας, κλπ). Στο σύστημά μας οι πράκτορες διεπαφών ενεργοποιούν πράκτορες έργου σύμφωνα με κατάλληλα κριτήρια (ικανότητα, διαθεσιμότητα, απόδοση σε έργα που ανατέθηκαν στο παρελθόν, κλπ).

- Οι πράκτορες πληροφοριών δραστηριοποιούνται είτε από πάνω προς τα κάτω από κάποιο χρήστη ή πράκτορα έργου μέσω ερωταπαντήσεων, είτε από κάτω προς τα πάνω στην περίπτωση που παρακολουθούν πηγές πληροφόρησης περιμένοντας να εμφανιστεί μια συγκεκριμένη πληροφορία. Μόλις η πληροφορία γίνει διαθέσιμη ο πράκτορας πληροφοριών ενημερώνει-ειδοποιεί τον ενδιαφερόμενο πράκτορα για τα αποτελέσματα της έρευνάς του.
- Οι πράκτορες διεπαφών δέχονται μηνύματα από χρήστες μέσω εφαρμογών που εκτελούνται σε browsers. Μόλις ένας πράκτορας διεπαφής συμπεράνει ποιες είναι οι ανάγκες του χρήστη (αφού επεξεργαστεί το αίτημά του) ξεκινά ένα καινούργιο έργο δίνοντάς του μια κωδική ονομασία (task id). Χρησιμοποιεί την βάση γνώσης του ή/ και πληροφορίες που μπορεί να συγκεντρώσει για να αποφασίσει σε ποιον πράκτορα έργου θα το αναθέσει αρχικά. Ενώ οι πράκτορες έργου προχωρούν προς την ολοκλήρωση του έργου οι πράκτορες διεπαφών παρακολουθούν την πρόοδό τους και ενημερώνουν τους χρήστες κατάλληλα.

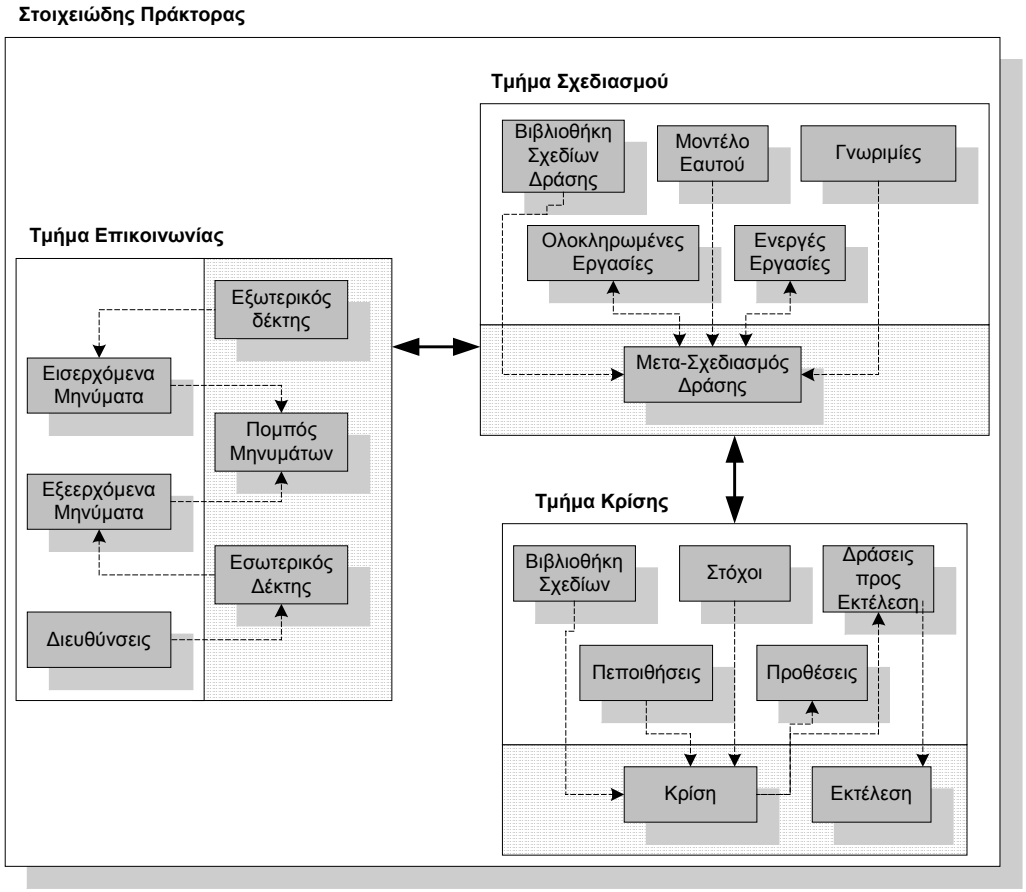
Μετά από τα παραπάνω είναι φανερή η ύπαρξη του χαρακτηριστικού της οργανωτικής αλλαγής που περιέγραψε ο Gasser (1986). Έτσι οι πράκτορες οργανώνονται κάθε φορά εξαρχής σε ομάδες στις οποίες μπορεί να συμμετέχει οποιοσδήποτε διαθέσιμος-πρόθυμος πράκτορας.

3.3.2. Εσωτερική Αρχιτεκτονική Πράκτορα

Η αρχιτεκτονική των απλών ή στοιχειωδών πρακτόρων προήλθε από την κατακράτηση των πιο καλών χαρακτηριστικών από εργασίες που παρουσιάστηκαν στην βιβλιογραφία, αλλά κυρίως από τις αρχιτεκτονικές των Brazier et al. (1997), Sycara και Zeng (1996) και Witting (1992).

Όλες οι λειτουργικότητες πρακτόρων υποστηρίζονται από μια κοινή πλατφόρμα-αρχιτεκτονική. Οι διαφορές μεταξύ των πρακτόρων φαίνονται στην υλοποίηση συγκεκριμένων τμημάτων του πράκτορα (τμήμα σχεδιασμού δράσης και τμήμα κρίσης). Αυτό προέκυψε από την επιλογή να

τμηματοποιηθεί ο πράκτορας, δηλαδή να χωριστούν οι λειτουργίες που επιτελεί σε τμήματα τα οποία είναι ανεξάρτητα μεταξύ τους. Έτσι έγινε δυνατή η χρησιμοποίηση κοινής υλοποίησης στις λειτουργίες τις οποίες επιτελούν όλοι οι πράκτορες και υλοποιήθηκαν ξεχωριστά μόνο τα κομμάτια αυτά που διαφοροποιούσαν τα λειτουργικά είδη των πρακτόρων.



Εικόνα 3-5: Αρχιτεκτονική απλού πράκτορα

Η προτεινόμενη αρχιτεκτονική στηρίζεται σε ένα μοντέλο πράκτορα με τρία τμήματα (σε παρενθέσεις δίνεται ο ξένος όρος για την εύκολη αναζήτησή τους στην βιβλιογραφία όπως και μια σύντομη περιγραφή):

- Τμήμα Επικοινωνίας (communication module, το τμήμα αυτό είναι υπεύθυνο για την ανταλλαγή μηνυμάτων μεταξύ των πρακτόρων),

- Τμήμα Σχεδιασμού (planning module, το τμήμα αυτό είναι υπεύθυνο για την συνεργασία του πράκτορα με άλλους),
- Τμήμα Κρίσης (reasoning module, το τμήμα αυτό αποφαίνεται για την ικανότητα του πράκτορα να φέρει σε πέρας εργασίες, την χρονοδρομολόγηση αυτών, την εκτέλεσή τους και την παρακολούθηση της πορείας εκτέλεσης κάθε εργασίας).

Τα τμήματα αυτά (Εικόνα 3-5) επικοινωνούν μεταξύ τους με τη χρήση ενδοπρακτορικών μηνυμάτων (intra-agent messages).

Τα τμήματα αυτά εκτελούνται παράλληλα (κάθε τμήμα τρέχει στο δικό του νήμα εκτέλεσης, executing thread). Ο πράκτορας δεν κάνει τίποτα μέχρι να έρθει ένα διαπρακτορικό μήνυμα στο τμήμα επικοινωνίας του. Σε αυτή την περίπτωση το τμήμα επικοινωνίας αποφαίνεται για την σημαντικότητα του μηνύματος, το μετατρέπει στο κατάλληλο ενδοπρακτορικό μήνυμα και το προωθεί στο τμήμα σχεδιασμού για επεξεργασία, με τη χρήση μιας ουράς η οποία υλοποιείται σε επίπεδο πράκτορα. Όλα τα τμήματα υιοθετούν αυτό τον τρόπο λειτουργίας και παραμένουν αδρανή όσο δεν έρχεται προς αυτά κάποιο ενδοπρακτορικό μήνυμα (όπως και το τμήμα επικοινωνίας) ή δεν έχουν κάποια εργασία να φέρουν σε πέρας.

Ο ίδιος μηχανισμός διαχείρισης ενδοπρακτορικών μηνυμάτων εξυπηρετεί όλα τα τμήματα του πράκτορα. Αυτός υλοποιείται σε επίπεδο τάξης πράκτορα (Agent) και τμήματος (Module), δηλαδή είναι ανεξάρτητος τμήματος. Ουσιαστικά ο μηχανισμός αυτός είναι δύο ουρές αναμονής FIFO με προτεραιότητα, μία για την επικοινωνία μεταξύ των τμημάτων επικοινωνίας και σχεδιασμού και μία μεταξύ των τμημάτων σχεδιασμού και κρίσης (όπως είναι προφανές το τμήμα σχεδιασμού δεν επικοινωνεί άμεσα με το τμήμα επικοινωνίας). Σε αυτή την εργασία θα περιγραφεί μόνο το τμήμα επικοινωνίας (παράγραφος 3.3.3) καθώς η υλοποίηση των τμημάτων σχεδιασμού και κρίσης είναι θέματα άλλης εργασίας.

Στην Εικόνα 3-6 φαίνεται η τάξη του πράκτορα (*Agent*) η οποία αθροίζει τρία αντικείμενα της τάξης *Module* (τα τρία διαφορετικά τμήματα του πράκτορα) και δύο αντικείμενα της τάξης *FIFO_Priority_Queue* (οι δύο ουρές εξυπηρέτησης ενδοπρακτορικών μηνυμάτων).

Οι μέθοδοι της τάξης *Agent* αφορούν τη διαχείριση των δύο ουρών (καλούνται από μεθόδους των *Module* αντικειμένων τα οποία μπορεί να θέλουν να διαβάσουν ή να γράψουν κάποιο ενδοπρακτορικό μήνυμα στις ουρές). Έτσι, οι μέθοδοι *updateCommPlanQueue* και *readCommPlanQueue* εξυπηρετούν τα τμήματα επικοινωνίας και σχεδιασμού για την υποβολή και ανάγνωση ενδοπρακτορικού μηνύματος αντίστοιχα, ενώ οι *updatePlanReasQueue* και *readPlanReasQueue* εξυπηρετούν με τον ίδιο τρόπο τα τμήματα σχεδιασμού και κρίσης.

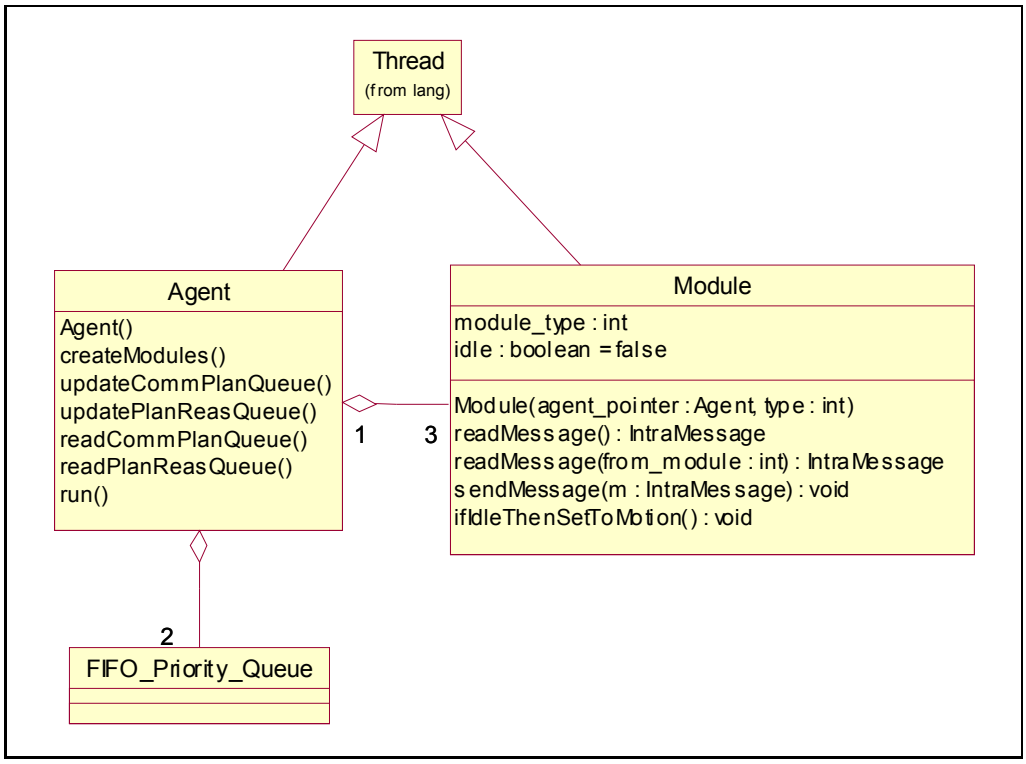
Η τάξη Module είναι υπερ-τάξη (super-class) όλων των τμημάτων του πράκτορα. Οι μέθοδοι της τάξης *Module* αφορούν το διάβασμα ή το γράψιμο ενδοπρακτορικών μηνυμάτων (*readMessage* και *sendMessage*), όπως και την εκκίνηση του τμήματος σε περίπτωση που αυτό είναι αδρανές ενώ υπάρχει εισερχόμενο ενδοπρακτορικό μήνυμα (μέθοδος *ifIdleThenSetToMotion*). Τα χαρακτηριστικά του Module αφορούν τον τύπο του τμήματος (ο ακέραιος module_type) και την κατάσταση του *Module* (σημαία idle). Κάθε στιγμιότυπο (instance) της τάξης Module μπορεί να βρίσκεται σε μία από δύο καταστάσεις:

- *Ενεργό* (εκτελεί κάποια εργασία)
- *Αδρανές* (δεν εκτελεί καμία εργασία)

Με τη βοήθεια του χαρακτηριστικού *idle* λειτουργεί η μέθοδος *ifIdleThenSetToMotion*, ενώ με τη βοήθεια του χαρακτηριστικού module_type λειτουργούν οι *readMessage* και *sendMessage* (με τη βοήθειά του ξέρουν ποιες μεθόδους διαχείρισης ουρών της τάξης Agent θα εκτελέσουν).

Η υπερφορτωμένη μέθοδος (overloaded method) readMessage μπορεί να εκτελεστεί με παράμετρο έτσι ώστε κάποιο τμήμα να μπορεί να διαβάσει από

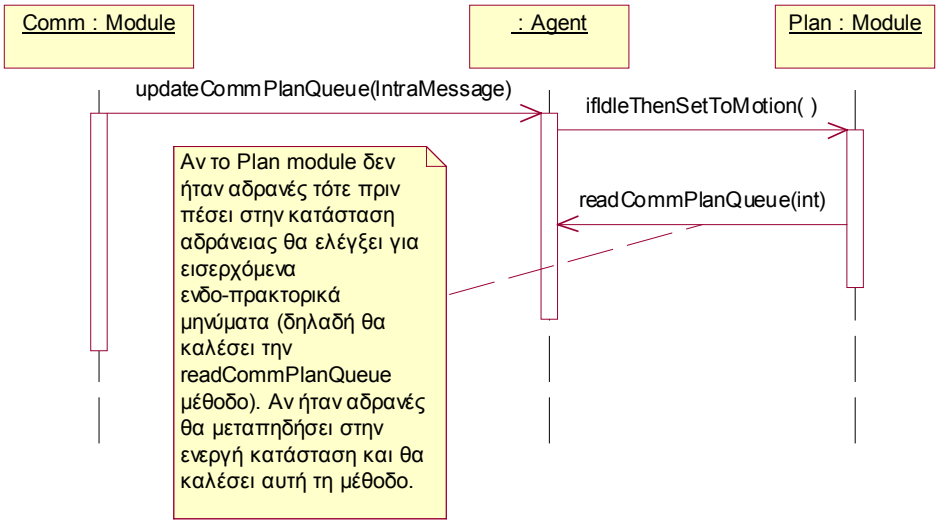
όποια ουρά θέλει (αν διαβάζει μηνύματα από περισσότερες της μίας ουρές, η περίπτωση του τμήματος σχεδιασμού).



Εικόνα 3-6: Διάγραμμα τάξεων απλού πράκτορα

Για να γίνει περισσότερο κατανοητή η διαδικασία ανταλλαγής μηνυμάτων παρουσιάζεται ένα σενάριο στην Εικόνα 3-7 όπου το τμήμα επικοινωνίας στέλνει ένα ενδοπρακτορικό μήνυμα στο τμήμα σχεδιασμού. Το σενάριο παρουσιάζεται σαν ένα διάγραμμα ακολουθιών της γλώσσας μοντελοποίησης UML. Η ακολουθία αρχικοποιείται όταν το τμήμα επικοινωνίας, το οποίο είναι ένα στιγμιότυπο μιας υποτάξης (subclass) της τάξης *Module*, καλέσει την προστατευμένη (protected, σημαίνει ότι μόνο το ίδιο το στιγμιότυπο μπορεί να την καλέσει) μέθοδό του *sendMessage*. Η *updateCommPlanQueue* μέθοδος της τάξης Agent καλείται και ένα στιγμιότυπο της τάξης *IntraMessage* (η δομή για την αναπαράσταση των ενδοπρακτορικών μηνυμάτων) μπαίνει στην ουρά που εξυπηρετεί τα δύο αυτά τμήματα. Η μέθοδος *updateCommPlanQueue* αφού βάλει το μήνυμα στην ουρά καλεί την μέθοδο *ifIdleThenSetToMotion* του τμήματος σχεδιασμού. Αν το τμήμα ήταν *αδρανές* τότε θα ξαναρχίσει τον

κύκλο εκτέλεσής του και αργά ή γρήγορα (αλλά σίγουρα πριν αδρανοποιηθεί ξανά) θα καλέσει την δική του μέθοδο *readMessage* η οποία θα διαβάσει το εισερχόμενο μήνυμα με την κλήση της *readCommPlanQueue* μεθόδου του αντικειμένου Agent. Στην περίπτωση που το τμήμα ήταν *ενεργό* δεν γίνεται τίποτα με την κλήση της μεθόδου *ifIdleThenSetToMotion* καθώς πριν πάει στην κατάσταση αδρανές το τμήμα εξετάζει τις ουρές του για ενδεχόμενα εισερχόμενα μηνύματα προς επεξεργασία.



Εικόνα 3-7: Ανταλλαγή ενδοπρακτορικού μηνύματος

3.3.3. Τμήμα Επικοινωνίας

Το τμήμα επικοινωνίας είναι υπεύθυνο για την επικοινωνία του πράκτορα με τους άλλους πράκτορες. Στέλνει και δέχεται διαπρακτορικά μηνύματα, ενώ εσωτερικά αλληλεπιδρά με το τμήμα σχεδιασμού. Η λειτουργία του είναι αρκετά τετριμμένη: Ένας εσωτερικός επεξεργαστής μηνυμάτων μετατρέπει κάθε εισερχόμενο ενδοπρακτορικό μήνυμα σε διαπρακτορικό, του προσθέτει τη διεύθυνση του παραλήπτη και το αποθέτει στην ουρά εξερχόμενων μηνυμάτων από τη μια και από την άλλη παρακολουθεί τις ουρές και στέλνει τα μηνύματα που περιμένουν σε αυτές είτε προς το τμήμα σχεδιασμού του πράκτορα είτε προς κάποιον άλλο πράκτορα. Ένας εξωτερικός δέκτης

(διεργασία με δικό της νήμα εκτέλεσης, είναι ουσιαστικά ένας εξυπηρετητής, server) δέχεται διαπρακτορικά μηνύματα τα οποία μετατρέπει σε ενδοπρακτορικά και τα αποθέτει στην ουρά εισερχομένων μηνυμάτων. Η ευφυΐα του τμήματος αυτού περιορίζεται στο να διαμορφωθεί κατάλληλα για να αναλάβει τις αλληλεπιδράσεις ενός απλού πράκτορα με τον πολύπλοκο πράκτορα στην ομάδα του οποίου γίνεται μέλος.

Τα διαπρακτορικά και ενδοπρακτορικά μηνύματα υλοποιούνται από τις κελυφοποιημένες (encapsulated) δομές δεδομένων InterMessage και IntraMessage αντίστοιχα, οι οποίες φαίνονται σε διάγραμμα τάξεων στην Εικόνα 3-8. Οι τάξεις *InterMessage* και *IntraMessage* κληρονομούν την τάξη *Message*, η οποία έχει τα παρακάτω προστατευμένα (protected) χαρακτηριστικά και υποστηρίζει μεθόδους για την ενημέρωση και ανάγνωση αυτών (κελυφοποιημένα):

- *message_id* (ένας ακέραιος ο οποίος ανατίθεται από τον αποστολέα έτσι ώστε να αναγνωρίσει-πιστοποιήσει την απάντηση στο μήνυμα αυτό, η οποία θα πρέπει να έχει τον ίδιο ακέραιο στο αντίστοιχο πεδίο)
- *message_type* (ο τύπος του μηνύματος, ακέραιος), οι διαθέσιμοι τύποι διαπρακτορικών και ενδοπρακτορικών μηνυμάτων είναι οι (αντιστοιχούν στους τύπους αλληλεπίδρασης πρακτόρων):
  - *Γέννηση* (ανακοίνωση της άφιξης ενός νέου πράκτορα στην κοινότητα)
  - *Θάνατος* ανακοίνωση της διακοπής λειτουργίας ενός πράκτορα)
  - *Task* (μήνυμα σχετικό με μια συγκεκριμένη εργασία)
  - *Info* (μήνυμα πληροφόρησης)
  - *Task_Init* (μηνύματα σχετικά με την αρχικοποίηση μιας εργασίας, πράκτορες προτείνουν και πράκτορες αποδέχονται ή όχι μια εργασία)
  - *Team* (δημιουργία ομάδας εργασίας σε έναν πολύπλοκο πράκτορα, είναι τύπος μόνο ενδοπρακτορικού μηνύματος)

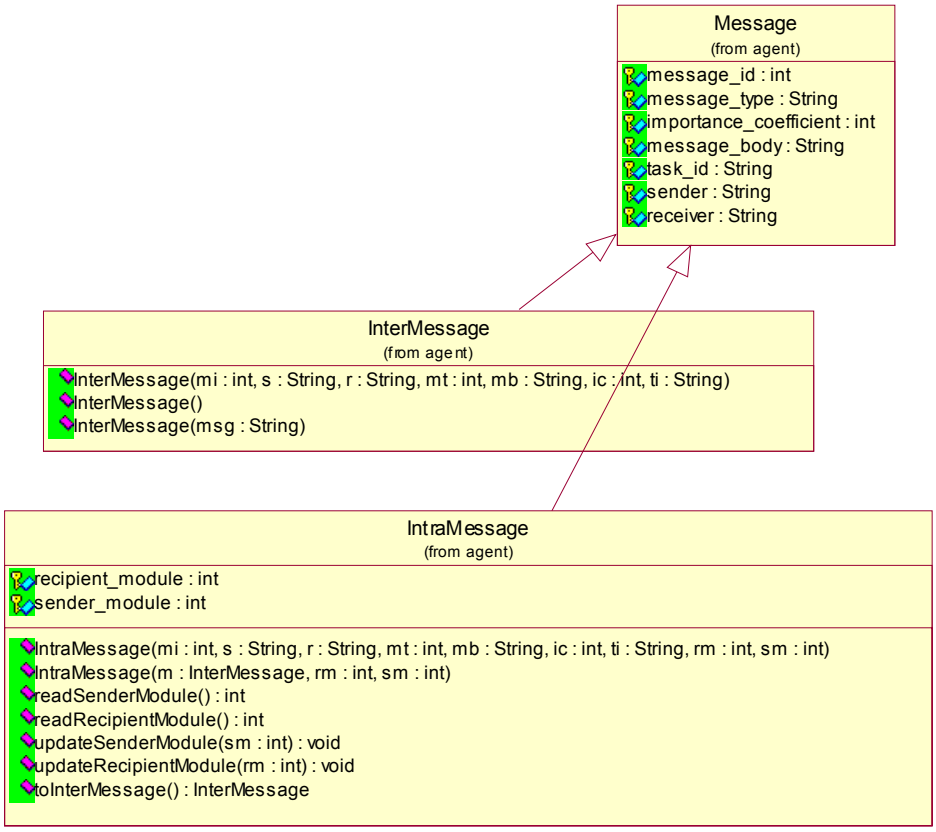
- *importance_coefficient* (ένας ακέραιος που δηλώνει την σημαντικότητα ενός μηνύματος)
- *message_body* (η μεταδιδόμενη πληροφορία αυτή καθεαυτή, το σώμα του μηνύματος)
- *task_id* (η ταυτότητα της εργασίας στα πλαίσια της οποίας αποστέλλεται αυτό το μήνυμα)
- *sender* (ο πράκτορας αποστολέας),
- *receiver* (ο πράκτορας παραλήπτης).

Η τάξη *InterMessage* είναι υπο-τάξη της τάξης *Message* και κελυφοποιεί μεθόδους σειριακοποίησης (serialization) των χαρακτηριστικών της ώστε να γίνεται εύκολη η αποστολή τους μέσω TCP θυρών (sockets). Οι μέθοδος υπεύθυνη για τη σειριακοποίηση των δεδομένων είναι η *toString*, η οποία δημιουργεί ένα αλφαριθμητικό (string) με τα χαρακτηριστικά της τάξης, ενώ υπάρχει και κατασκευαστής (constructor) για την τάξη αυτή κατάλληλος ώστε να δημιουργείται στιγμιότυπο της τάξης με παράμετρο ένα αλφαριθμητικό.

Η τάξη *IntraMessage* είναι υπο-τάξη της τάξης *Message* και προσθέτει δύο νέα χαρακτηριστικά (και τις αντίστοιχες μεθόδους για να τα τροποποιήσουμε ή διαβάσουμε):

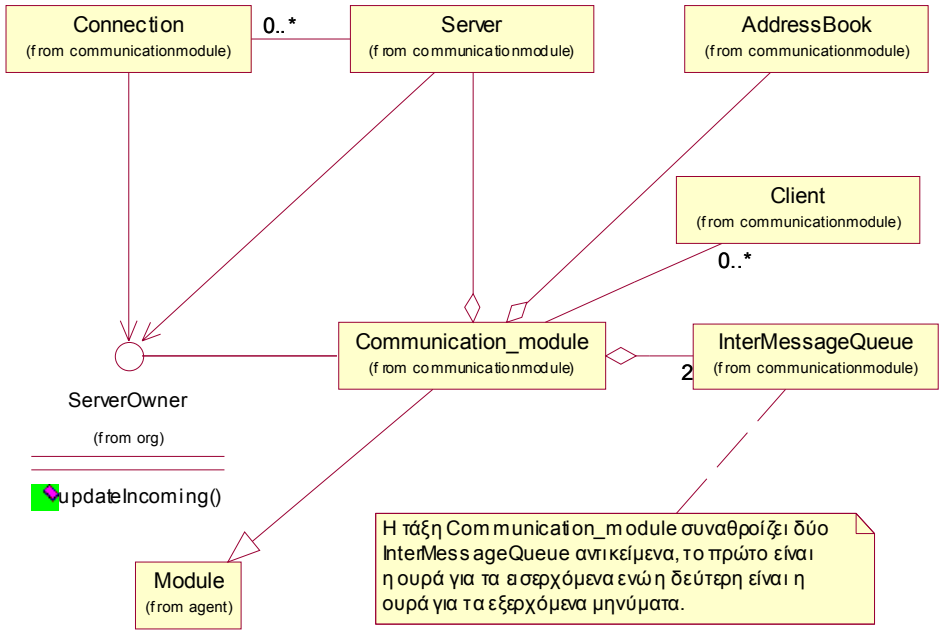
- *sender_module* (ακέραιος που αντιστοιχεί στο τμήμα πράκτορα που έστειλε το μήνυμα)
- *recipient_module* (ακέραιος που αντιστοιχεί στο τμήμα πράκτορα που πρέπει να παραλάβει το μήνυμα)

Ακόμα, επειδή το ενδοπρακτορικό μήνυμα μετατρέπεται από το τμήμα επικοινωνίας σε διαπρακτορικό και αντίστροφα η τάξη *IntraMessage* κελυφοποιεί μεθόδους οι οποίες κάνουν αυτές τις εργασίες: η μέθοδος *toInterMessage* και ο κατασκευαστής, οποίος παίρνει σαν παράμετρο ένα αντικείμενο της τάξης *InterMessage* και δύο ακεραίους οι οποίοι αντιστοιχούν στα τμήματα αποστολέα και παραλήπτη, αντίστοιχα.

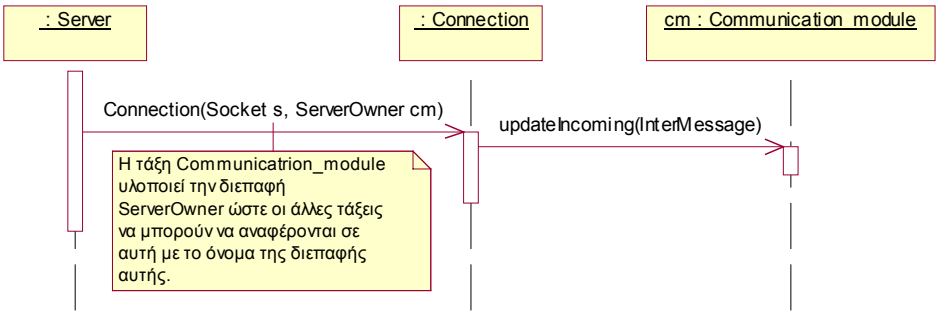


Εικόνα 3-8: Τάξεις μηνυμάτων

Η τάξη *Communication_module* παρουσιάζεται στην Εικόνα 3-9. Συναθροίζει (aggregates) δύο ουρές (για την αποθήκευση εισερχομένων και εξερχόμενων διαπρακτορικών μηνυμάτων), μία τάξη *Server* και μια τάξη *AddressBook* (χρησιμεύει για την αποθήκευση των TCP/IP διευθύνσεων των *Server* άλλων πρακτόρων). Η τάξη *Server* υλοποιεί έναν TCP socket server στον οποίο άλλα τμήματα επικοινωνίας πρακτόρων μπορούν να στείλουν μηνύματα και ο οποίος δημιουργεί στιγμιότυπα της τάξης *Connection* τα οποία δημιουργούν αντικείμενα *InterMessage* και τα αποθηκεύουν στην ουρά εισερχομένων διαπρακτορικών μηνυμάτων. Όποτε η τάξη *Communication_module* θελήσει να αποστείλει ένα διαπρακτορικό μήνυμα σε έναν άλλο πράκτορα δημιουργεί ένα στιγμιότυπο της τάξης *Client* η οποία αναλαμβάνει να αποστείλει το μήνυμα στον εξυπηρετητή μιας άλλης τάξης *Communication_module*.



Εικόνα 3-9: Διάγραμμα τάξεων για το τμήμα επικοινωνίας



Εικόνα 3-10: Σενάριο άφιξης διαπρακτορικού μηνύματος

Ένα σενάριο της άφιξης ενός διαπρακτορικού μηνύματος παρουσιάζεται στην Εικόνα 3-10. Σε κάθε αίτηση για σύνδεση ο *Server* δημιουργεί ένα στιγμιότυπο της τάξης *Connection* στο οποίο αναθέτει την παράληψη του μηνύματος. Αφού παραλάβει το μήνυμα, και αν το τελευταίο δημιουργεί ένα έγκυρο στιγμιότυπο της τάξης *InterMessage*, το αντικείμενο *Connection* καλεί την *updateIncoming* μέθοδο του αντικειμένου *Communication_module* η οποία αποθηκεύει το διαπρακτορικό μήνυμα στην ουρά εισερχομένων. Ας

παρατηρηθεί το γεγονός ότι το αντικείμενο *Server* βλέπει το *Communication_module* αντικείμενο σαν ένα *ServerOwner* αντικείμενο. Η τάξη *ServerOwner* είναι μια διεπαφή (interface) την οποία υλοποιεί (implements) η τάξη *Communication_module*.

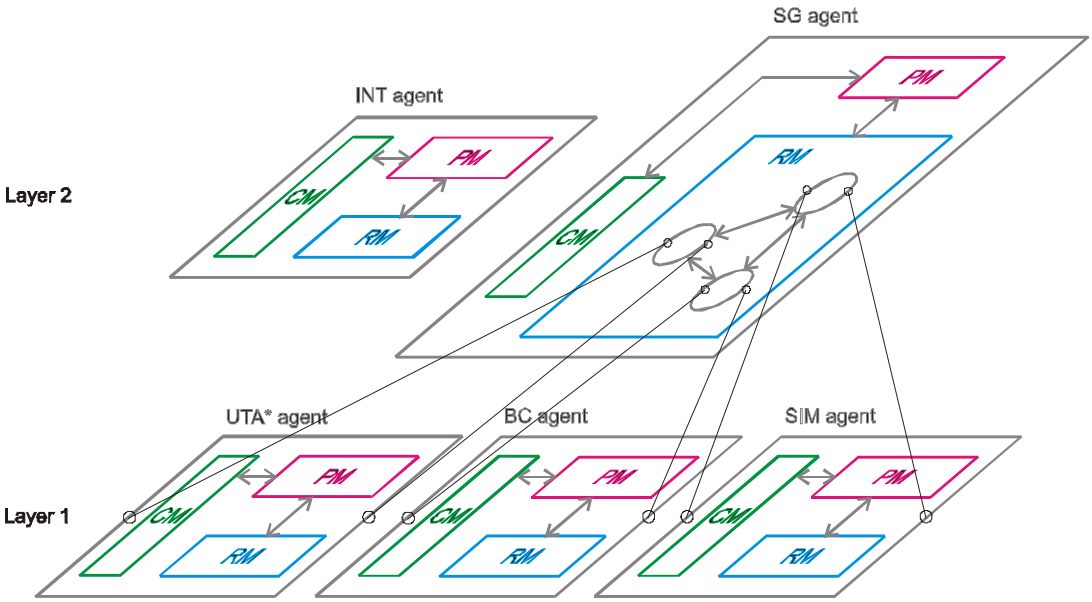
3.3.4. Αρχιτεκτονική Πολύπλοκων Πρακτόρων

Οι πολύπλοκοι πράκτορες θα μπορούσαν να ανήκουν και στα τρία είδη λειτουργίας των πρακτόρων (εργασίας, πληροφοριοδότες και διεπαφές). Η αρχιτεκτονική του πολύπλοκου πράκτορα είναι παρόμοια με αυτή του απλού πράκτορα. Έτσι και ο πολύπλοκος αποτελείται από τρία τμήματα (επικοινωνίας, σχεδιασμού και κρίσης) τα οποία επικοινωνούν μέσω του μηχανισμού ενδοπρακτορικών μηνυμάτων.

Η διαφορά βρίσκεται στη δομή του τμήματος κρίσης του πολύπλοκου πράκτορα. Το ρόλο του τμήματος αυτού αναλαμβάνει μια ομάδα πρακτόρων (απλών ή σύνθετων ή συνδυασμό αυτών), οι οποίοι διαισθητικά βρίσκονται σε ένα χαμηλότερο επίπεδο αφαίρεσης ενός προβλήματος. Η επίτευξη ενός στόχου ενός πράκτορα-πατέρα σε ένα ν-οστό επίπεδο λοιπόν βασίζεται στην συνεργασία των πρακτόρων του χαμηλότερου (ν-1) επιπέδου τους οποίους συντονίζει ο πράκτορας του ν-οστού επιπέδου. Το τμήμα κρίσης λοιπόν ενός πολύπλοκου πράκτορα μπορούμε να το δούμε ως μια οργάνωση πρακτόρων (agent organization).

Η αλληλεπίδραση μεταξύ των τμημάτων κρίσης και σχεδιασμού (όπως και στον απλό πράκτορα) ενός πολύπλοκου πράκτορα στο ν+1 επίπεδο επιτυγχάνεται μέσω των πρακτόρων του ν-οστού επιπέδου, οι οποίοι είναι τα συστατικά του τμήματος κρίσης του πράκτορα στο ν+1 επίπεδο (για παράδειγμα ο πράκτορας επιλογέας μοντέλου συμπεριφοράς καταναλωτή στέλνει τα αποτελέσματα της εργασίας του στον παραγωγό σεναρίων). Διαισθητικά, αυτό που συμβαίνει το βλέπουμε στην Εικόνα 3-11 από ένα μετα-επίπεδο. Τεχνικά, ένα διαπρακτορικό μήνυμα που αποστέλλεται από έναν πράκτορα ν-οστού επιπέδου σε έναν πράκτορα ν+1 επιπέδου

μετατρέπεται σε ενδοπρακτορικό μήνυμα του πράκτορα στο ν+1 επίπεδο. Η οργάνωση πρακτόρων για το τμήμα κρίσης πολύπλοκων πρακτόρων δημιουργείται δυναμικά κάθε φορά που ο πολύπλοκος πράκτορας αναλαμβάνει μια εργασία, σύμφωνα με την παράγραφο 3.3.1. Ο ρόλος της οργάνωσης αυτής είναι να αναλάβει όλες τις εργασίες που θα της ανατεθούν από το τμήμα σχεδιασμού του πολύπλοκου πράκτορα.



Εικόνα 3-11: Επίπεδα οργάνωσης

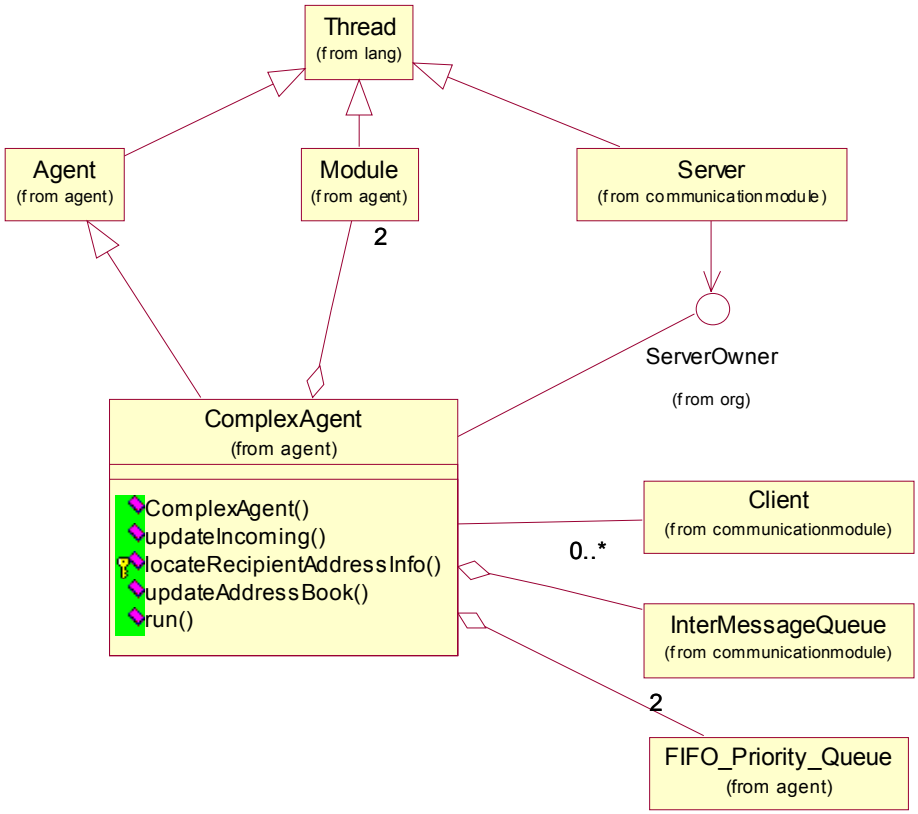
Η διεργασία δημιουργίας της οργάνωσης πρακτόρων του τμήματος κρίσης του πολύπλοκου πράκτορα αρχικοποιείται με την επιλογή κατάλληλων εργοληπτών (την οποία κάνει το τμήμα σχεδιασμού του πράκτορα). Ένα διάγραμμα ακολουθιών που περιγράφει αυτή την διαδικασία θα παρουσιαστεί παρακάτω. Οι πράκτορες που συμμετέχουν στην οργάνωση αυτή μπορεί να είναι ετερογενείς ή απομακρυσμένοι αλλά αντιλαμβάνονται ότι πλέον ανήκουν σε κάποια οργανωτική ομάδα την οποία συντονίζει ο πράκτορας του ανώτερου επιπέδου. Στην Εικόνα 3-11 τρεις πράκτορες, ένας πολυκριτήριος αναλυτής, ένας επιλογέας μοντέλου συμπεριφοράς καταναλωτή και ένας προσομοιωτής απαρτίζουν την οργάνωση που αποτελεί το τμήμα κρίσης του πολύπλοκου πράκτορα παραγωγού σεναρίων. Ο πολυκριτήριος αναλυτής παρέχει την

υπηρεσία πολυκριτήριας ανάλυσης (χρησιμοποιώντας τη γνώση του στην ανάλυση ερευνών αγοράς και επιλογής πολυκριτήριων ερωτήσεων), ο επιλογέας μοντέλου συμπεριφοράς καταναλωτή επιλέγει ένα μοντέλο το οποίο μοντελοποιεί ικανοποιητικά-καλύτερα την αγορά (χρησιμοποιώντας ευρετικό αλγόριθμο για την επιλογή του από μια εκτενή βάση μοντέλων που χρησιμοποιεί), ενώ ο προσομοιωτής δημιουργεί σενάρια για τον χρήστη (χρησιμοποιώντας δεδομένα εξόδου των δύο προηγουμένων και εντολές του χρήστη, οι οποίες δίνονται μέσω του πράκτορα διεπαφής στον παραγωγό σεναρίων).

Τα τμήματα επικοινωνίας και σχεδιασμού έχουν ίδια δομή με αυτά των απλών πρακτόρων. Η διαφορά βρίσκεται στην τάξη *ComplexAgent* όπως αυτή παρουσιάζεται στην Εικόνα 3-12. Αντί για ένα τρίτο αντικείμενο *Module* (το οποίο θα αντιστοιχούσε στο τμήμα επικοινωνίας του) ο πράκτορας χρησιμοποιεί ένα *Server* (η γνωστή τάξη από την παράγραφο 3.3.3), ένα *InterMessageQueue* (ουρά εισερχομένων διαπρακτορικών μηνυμάτων) και στιγμιότυπα της τάξης *Client* (επίσης γνωστή από την παράγραφο 3.3.3).

Εδώ πρέπει να τονιστεί ότι τα συστατικά που χρησιμοποιούνται έχουν χρησιμοποιηθεί και από τον απλό πράκτορα. Τα συστατικά αυτά, τα οποία θα τα δούμε να χρησιμοποιούνται και από τα πρακτορεία, δημιουργήθηκαν με πλήρη αντικειμενοστραφή λογική ώστε να κελυφοποιούν σωστά τις διαδικασίες τους και έτσι να είναι πλήρως επαναχρησιμοποιήσιμα.

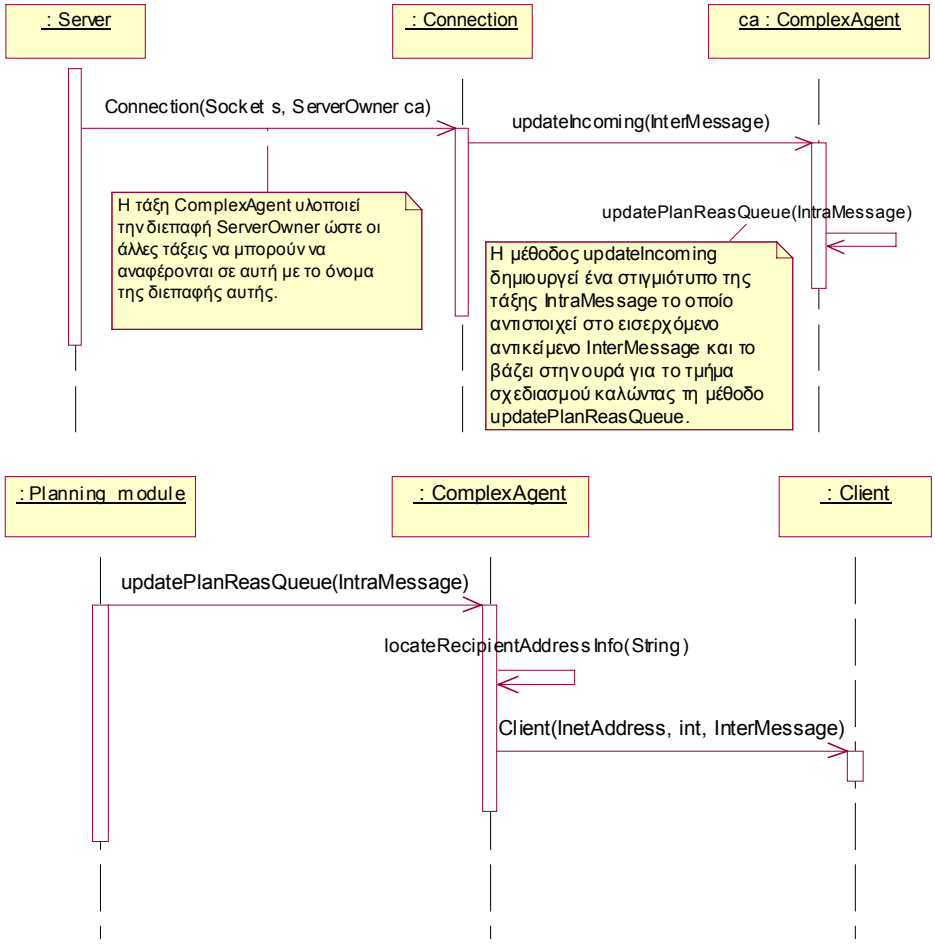
Η τάξη ComplexAgent, όπως φαίνεται και στην Εικόνα 3-12, εκτελείται στο δικό της νήμα εκτέλεσης αφού είναι υποτάξη της τάξης Agent. Από την τάξη Agent κληρονομεί και το μηχανισμό ανταλλαγής ενδοπρακτορικών μηνυμάτων. Τέλος, να σημειωθεί ότι τα τμήματα του πράκτορα διατηρούν ένα δείκτη προς τον πράκτορα ο οποίος δείχνει σε μία τάξη Agent. Άρα, τα τμήματα τελικά δεν μπορούν να καταλάβουν αν είναι τμήματα πολύπλοκου ή στοιχειώδους πράκτορα.



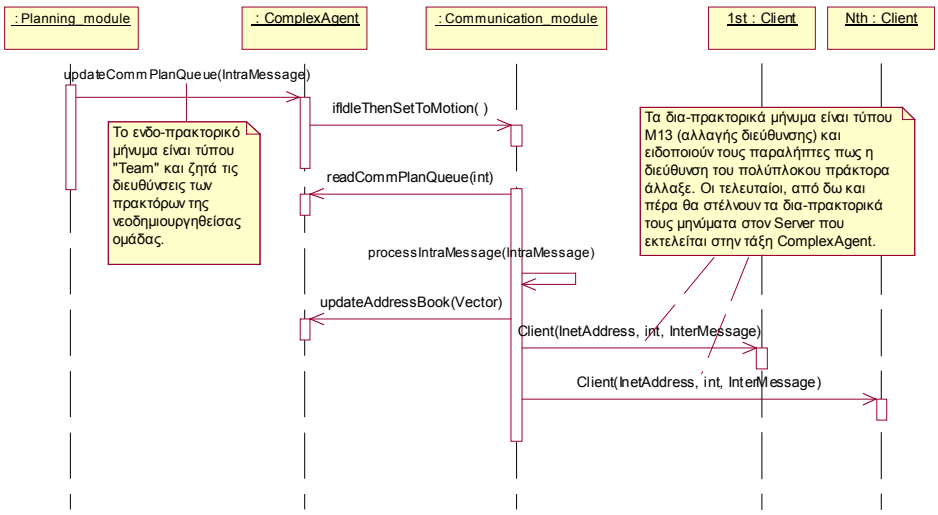
Εικόνα 3-12: Διάγραμμα τάξεων πολύπλοκου πράκτορα

Κάθε φορά που ο πολύπλοκος πράκτορας «προσλαμβάνει» έναν πράκτορα σε χαμηλότερο επίπεδο. Στέλνει στον τελευταίο ένα μήνυμα αλλαγής της TCP/IP διεύθυνσής του. Έτσι ο πράκτορας του χαμηλότερου επιπέδου στέλνει πλέον διαπρακτορικά μηνύματα στον *Server* του πολύπλοκου πράκτορα (όχι του τμήματος επικοινωνίας του, το οποίο παρόλα αυτά συνεχίζει να στέλνει και να δέχεται διαπρακτορικά μηνύματα προς και από πράκτορες οι οποίοι δεν ανήκουν σε αυτούς που απαρτίζουν το τμήμα κρίσης του πολύπλοκου πράκτορα). Αυτή η TCP/IP διεύθυνση λοιπόν ισχύει μόνο για τους πράκτορες που δημιουργούν το τμήμα κρίσης του πολύπλοκου πράκτορα. Τα διαπρακτορικά μηνύματα που φτάνουν στον *Server* του πολύπλοκου πράκτορα μετατρέπονται σε ενδοπρακτορικά μηνύματα με ένα διάφανο τρόπο και τοποθετούνται στην ουρά ενδοπρακτορικών μηνυμάτων μεταξύ των τμημάτων

κρίσης και σχεδιασμού του πράκτορα. Αντίστοιχα, ενδοπρακορικά μηνύματα που στέλνει το τμήμα σχεδιασμού στο τμήμα κρίσης μετατρέπονται σε διαπρακτορικά μηνύματα, τα οποία αποστέλλονται με το στιγμιότυπο ενός *Client* στον αντίστοιχο πράκτορα του χαμηλότερου επιπέδου (και οι δύο αυτοί μηχανισμοί παρουσιάζονται μέσω διαγραμμάτων ακολουθιών στην Εικόνα 3-13). Για να υποστηρίξει αυτή την διαδικασία η τάξη ComplexAgent περιέχει ένα στιγμιότυπο της τάξης *AddressBook* η οποία παίρνει δεδομένα από το τμήμα επικοινωνίας με τη χρήση ενός ειδικού τύπου ενδοπρακτορικού μηνύματος, του *Team*.



Εικόνα 3-13: Ανταλλαγή μηνυμάτων μεταξύ ενός πολύπλοκου πράκτορα και ενός πράκτορα χαμηλότερου επιπέδου



Εικόνα 3-14: Σενάριο δημιουργίας ομάδας

Στην Εικόνα 3-14 παρουσιάζεται ένα σενάριο δημιουργίας μιας νέας ομάδας η οποία θα αντιστοιχεί στο τμήμα κρίσης ενός πολύπλοκου πράκτορα. Η ακολουθία ξεκινά μόλις το τμήμα σχεδιασμού του πράκτορα αποφασίσει ποιοι πράκτορες θα χρησιμοποιηθούν για την ολοκλήρωση της εργασίας του πολύπλοκου πράκτορα, τους στείλει τα απαραίτητα διαπρακτορικά μηνύματα τύπου *Task_Init* και αυτοί έχουν απαντήσει πως δέχονται την εργασία που θα τους αναθέσει. Τότε, το τμήμα σχεδιασμού στέλνει στο τμήμα επικοινωνίας το ενδοπρακτορικό μήνυμα τύπου *Team* αναγκάζοντας το τμήμα επικοινωνίας να καλέσει την συνάρτηση *updateAddressBook* του *ComplexAgent* και να εμπλουτίσει τη δομή του *AddressBook* με τις διευθύνσεις TCP/IP των πρακτόρων της ομάδας. Στη συνέχεια το τμήμα επικοινωνίας στέλνει κατάλληλο διαπρακτορικό μήνυμα αλλαγής διεύθυνσης του πολύπλοκου πράκτορα. Από δω και πέρα όταν οι πράκτορες της ομάδας του χαμηλότερου επιπέδου στέλνουν δια=πρακτορικό μήνυμα στον πολύπλοκο πράκτορα, αυτό δεν πηγαίνει στο τμήμα επικοινωνίας του αλλά στον δικό του *Server*. Μόλις τελειώσει η συνεργασία τους ο πολύπλοκος πράκτορας θα ξαναστείλει στους πράκτορες της ομάδας του το μήνυμα αλλαγής διεύθυνσης και αυτοί θα στέλνουν πλέον μηνύματα πάλι στο τμήμα επικοινωνίας του.

Μετά την τεχνική περιγραφή της δομής και λειτουργίας του πολύπλοκου πράκτορα μπορούμε να διατυπώσουμε τις δύο φάσεις επίτευξης πολύπλοκων εργασιών του συστήματος. Η πρώτη είναι η από πάνω προς τα κάτω διάσπαση του έργου σε υποέργα με την ταυτόχρονη δημιουργία επιπέδων οργάνωσης πρακτόρων, ενώ η δεύτερη είναι η από κάτω προς τα πάνω σύνθεση της λύσης-ολοκλήρωση του έργου με την ταυτόχρονη διάλυση των ομάδων των πολύπλοκων πρακτόρων που συμμετείχαν στη λύση.

Είναι δυνατό με τη δημιουργία κατάλληλων τμημάτων σχεδιασμού να έχουμε πολύπλοκους πράκτορες πληροφοριοδότες οι οποίοι θα ελέγχουν μέσω των πρακτόρων της ομάδας τους μεγάλο όγκο ετερογενούς πληροφορίας. Αυτό προϋποθέτει την ύπαρξη εξειδικευμένων πρακτόρων πληροφοριοδοτών. Ακόμα θα μπορούσαν να υπάρχουν πολύπλοκοι πράκτορες διεπαφής ικανοί να συντονίσουν πολλούς πράκτορες διεπαφής οι οποίοι θα εκπροσωπούν διαφορετικούς χρήστες (π.χ. μέλη διοικητικού συμβουλίου) ώστε να παρθεί μια διοικητική απόφαση η οποία θα συνθέτει απόψεις.

3.3.5. Αρχιτεκτονική και Λειτουργία Πρακτορείου

Το πρακτορείο είναι μια μορφή "δημαρχείου" για την κοινωνία των πρακτόρων, αντίστοιχα με την δουλειά της IKV++ (1999b). Όμως, βασικός του ρόλος είναι η διατήρηση πληροφορίας σχετικά με τον αριθμό, τον τύπο και την ταυτότητα των πρακτόρων που λειτουργούν στον τοπικό υπολογιστή και όχι η οργάνωσή τους σε ομάδες δουλειάς όπως προτείνει η IKV++. Την οργάνωσή τους θα την αναλαμβάνουν από μόνοι τους οι πράκτορες.

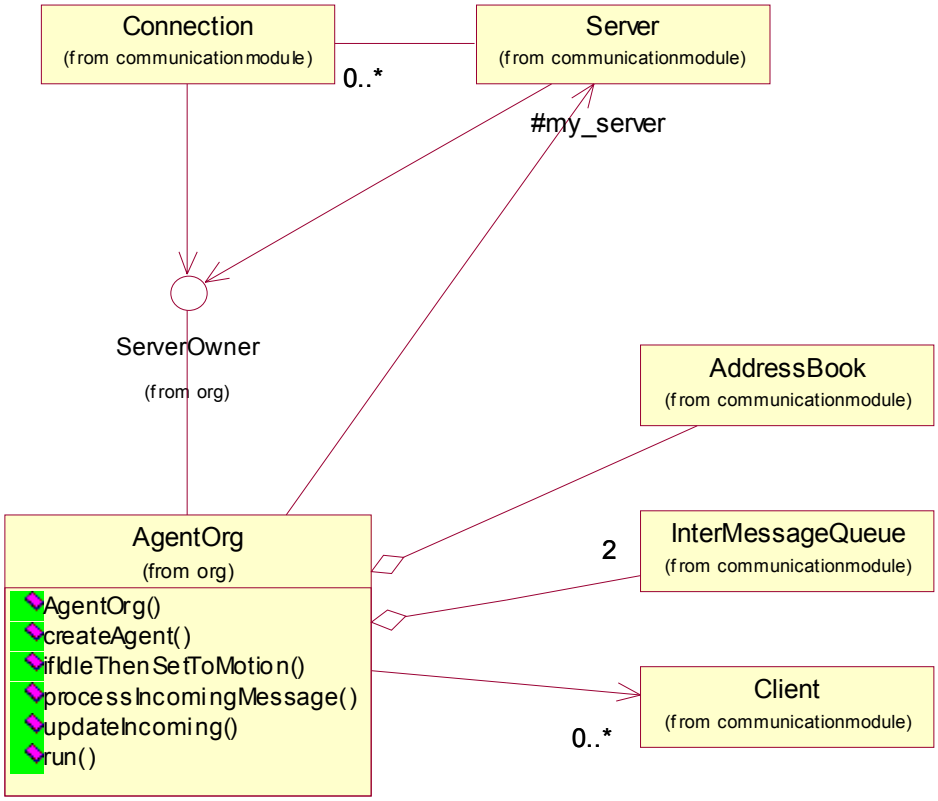
Όλοι οι πράκτορες θα μπορούν να αιτηθούν εξυπηρέτηση από το πρακτορείο το οποίο θα βλέπουν σαν έναν άλλο πράκτορα. Στην εργασία των Collis και Ndumu (1999b) χρησιμοποιούνται δύο τύποι πρακτόρων για να κάνουν τη δουλειά του πρακτορείου. Ο nameserver και ο facilitator γνωρίζουν ο ένας τις διευθύνσεις των πρακτόρων και ο άλλος τις ικανότητές τους. Το σύστημα αυτό όμως καταντά αρκετά γραφειοκρατικό και απαιτεί μεγάλη κατανάλωση πόρων για ανάγκες επικοινωνίας. Στο προτεινόμενο ΣΠΠ οι αιτήσεις των πρακτόρων

προς το πρακτορείο τους ή άλλο πρακτορείο γίνονται με κατάλληλα διαπρακτορικά μηνύματα. Ομοίως, το πρακτορείο απαντά με διαπρακτορικά μηνύματα.

Τέτοια πρακτορεία θα υπάρχουν σε κάθε υπολογιστή ο οποίος θα υποστηρίζει τη λειτουργία πρακτόρων. Οι λειτουργίες τους θα είναι οι εξής:

- *Δημιουργία Πράκτορα*: Η δημιουργία ενός συνόλου πρακτόρων θα ρυθμίζεται να γίνει με τη δημιουργία του πρακτορείου. Έτσι, για παράδειγμα, θα πρέπει να δημιουργείται τουλάχιστον ένας πράκτορας διεπαφή ο οποίος θα μπορεί να εξυπηρετήσει τους πιθανούς χρήστες και να αιτηθεί της δημιουργίας νέων πρακτόρων. Η δημιουργία νέων πρακτόρων γίνεται με την αποστολή κατάλληλου διαπρακτορικού μηνύματος προς το πρακτορείο.
- *Εύρεση πρακτόρων συγκεκριμένου τύπου*: Με κατάλληλο διαπρακτορικό μήνυμα προς το πρακτορείο οι πράκτορες μπορούν να βρουν τα στοιχεία πρακτόρων κάποιου συγκεκριμένου τύπου.
- *Καταγραφή νέων πρακτόρων*: Με αυτή τη λειτουργία ένας πράκτορας μπορεί να καταγραφεί στο συγκεκριμένο πρακτορείο και ας μην έχει δημιουργηθεί μέσω αυτού.
- *Ενημέρωση στοιχείων πράκτορα*: Ο τύπος και το όνομα του πράκτορα αποκλείεται να αλλάξουν, όμως η διεύθυνσή του είναι πιθανό να αλλάξει.
- *Καταστροφή πράκτορα*: Όταν ένας πράκτορας είναι έτοιμος να καλέσει τη συνάρτηση απενεργοποίησής του τότε στέλνει το μήνυμα αυτό στο πρακτορείο για να ειδοποιήσει για την επικείμενη καταστροφή του οπότε να σβηστεί από το μητρώο του πρακτορείου.

Στην παράγραφο 3.4 (σενάρια λειτουργίας του συστήματος) θα παρουσιαστούν σενάρια λειτουργίας του συστήματος. Εκεί θα φανεί και η αλληλεπίδραση των πρακτόρων με το πρακτορείο.



Εικόνα 3-15: Διάγραμμα τάξεων πρακτορείου

Στην Εικόνα 3-15 παρουσιάζεται η τάξη *AgentOrg* (agent organization) στιγμιότυπα της οποίας θα είναι τα πρακτορεία. Είναι εμφανή τα συστατικά που χρησιμοποιεί η τάξη αυτή.

- Υλοποιεί την διεπαφή (interface) *ServerOwner* για να χρησιμοποιήσει τον *Server*.
- Συναθροίζει δύο ουρές διαπρακτορικών μηνυμάτων (εισερχόμενα και εξερχόμενα μηνύματα) στιγμιότυπα της τάξης *InterMessageQueue*.
- Συναθροίζει ένα βιβλίο διευθύνσεων (*AddressBook*) και για την διατήρηση πληροφορίας σχετικά με τους πράκτορες του πρακτορείου.

Η λειτουργία του πρακτορείου συνοψίζεται στα εξής: Ο *Server* "ακούει" σε μια TCP/IP θύρα και όταν έρθει διαπρακτορικό μήνυμα καλεί την συνάρτηση *updateIncoming* του *AgentOrg*, η οποία το βάζει στην ουρά εισερχομένων και

καλεί την *ifIdleThenSetToMotion* συνάρτηση του πρακτορείου η οποία το μεταθέτει στην κατάσταση "ενεργό" αν αυτό ήταν στην κατάσταση "αδρανές".

Το πρακτορείο λειτουργεί όταν είναι σε κατάσταση "ενεργό" απλούστερα αλλά παρόμοια με το τμήμα επικοινωνίας ενός πράκτορα. Έτσι ελέγχει αν υπάρχουν εισερχόμενα μηνύματα προς επεξεργασία στην ουρά εισερχομένων (αν υπάρχουν τα επεξεργάζεται καλώντας την μέθοδο *processIncomingMessage*), ύστερα αν υπάρχουν έτοιμα μηνύματα για αποστολή (αν υπάρχουν τα αποστέλλει με χρήση στιγμιότυπων της τάξης *Client*). Αυτό το κάνει συνεχώς μέχρι να μην υπάρχουν ούτε εισερχόμενα ούτε εξερχόμενα μηνύματα οπότε πέφτει στην κατάσταση "αδρανές".

3.3.6. Πρωτόκολλα και Τυποποίηση Μηνυμάτων

Το χρησιμοποιούμενο πρωτόκολλο επικοινωνίας μεταξύ των πρακτόρων του προτεινόμενου ΣΠΠ είναι το TCP/IP. Οι πράκτορες επικοινωνούν μεταξύ των τμημάτων επικοινωνίας τους τα οποία ανταλλάσσουν μηνύματα με χρήση απλών θυρών (TCP plain sockets).

Όπως ειπώθηκε και στην παράγραφο 3.3.3 τα διαπρακτορικά μηνύματα είναι στιγμιότυπα της τάξης InterMessage, η οποία διαθέτει κατάλληλες μεθόδους και κατασκευαστές ώστε να μπορεί να σειριακοποιηθεί (serialized) και ανακτηθεί από μορφή συμβολοσειράς αντίστοιχα. Έτσι, η αποστολή και λήψη του διαπρακτορικού μηνύματος μέσω απλών θυρών είναι απλή υπόθεση.

Κωδικοποιημένος Τύπος Μηνύματος	**Σώμα Μηνύματος**	**Αποστολέας**	**Παραλήπτης**
M11	Νέα γνωριμία (στο σώμα είναι serialized ένα αντικείμενο τύπου Agent_information, περιέχει πληροφορίες σχετικά με το όνομα, τον τύπο και την διεύθυνση του πράκτορα)	ΣΠ,ΠΠ	ΣΠ, ΠΠ, Π
M12	Ειδοποίηση θανάτου (τίποτα στο σώμα του μηνύματος)	ΣΠ, ΠΠ	ΣΠ, ΠΠ, Π

Κωδικοποιημένος Τύπος Μηνύματος	**Σώμα Μηνύματος**	**Αποστολέας**	**Παραλήπτης**
M13	Ενημέρωση πληροφοριών επικοινωνίας (αλλαγή διεύθυνσης). Το σώμα περιέχει το νέο TCP port στο οποίο ακούει ο πράκτορα.	ΣΠ, ΠΠ	ΣΠ, ΠΠ, Π
M14	Η νέα διεύθυνση ενός πολύπλοκου πράκτορα που σε βάζει στην ομάδα του, είναι ενδοπρακτορικό μήνυμα μόνο και ζητά τις διευθύνσεις για κάποιους πράκτορες οι οποίοι θα μπουν στην ομάδα του πολύπλοκου πράκτορα. Το σώμα του μηνύματος περιέχει τον αριθμό της TCP θύρας του ComplexAgent στιγμιότυπου ακολουθούμενο από τα ονόματα των πρακτόρων οι οποίοι θα μπουν στην ομάδα του.	ΠΠ (τμήμα σχεδιασμού προς τμήμα επικοινωνίας)	
M01	Αίτημα δημιουργίας νέου πράκτορα. Το σώμα του μηνύματος περιέχει τον τύπο και το όνομα του νέου πράκτορα.	ΣΠ, ΠΠ	Π
M02	Αίτημα πληροφόρησης σχετικά με τους διαθέσιμους πράκτορες κάποιου τύπου. Στο σώμα είναι ο κωδικός αριθμός που αντιστοιχεί σε ένα τύπο πρακτόρων.	ΣΠ, ΠΠ	Π
M2	Task_Init, αίτημα αρχικοποίησης συνεργασίας μεταξύ πρακτόρων. Το σώμα περιέχει δεδομένα ανάλογα με την εξέλιξη της διαπραγμάτευσης μεταξύ των πρακτόρων (Εικόνα 3-4, μέχρι και το μήνυμα *ανάθεση εργασίας*)	ΣΠ, ΠΠ	ΣΠ, ΠΠ

Κωδικοποιημένος Τύπος Μηνύματος	Σώμα Μηνύματος	Αποστολέας	Παραλήπτης
M3	Task, μήνυμα σχετικό με μια αρχικοποιημένη συνεργασία πρακτόρων. Το σώμα περιέχει δεδομένα ανάλογα με την εξέλιξη της εργασίας (Εικόνα 3-4, από το μήνυμα *αίτηση πληροφόρησης για την πρόοδο της εργασίας* και κάτω)	ΣΠ, ΠΠ	ΣΠ, ΠΠ
M4	Info. Αίτημα πληροφόρησης και αντίστοιχη απάντηση.	ΣΠ, ΠΠ	ΣΠ, ΠΠ

Πίνακας 3-1: Τύποι μηνυμάτων (ΣΠ: στοιχειώδης πράκτορας, ΠΠ: πολύπλοκος πράκτορας, Π: πρακτορείο)

Ο Πίνακας 3-1 παρουσιάζει τα ενδο και διαπρακτορικά μηνύματα που έχουν ήδη δημιουργηθεί και ενσωματωθεί στο πρωτόκολλο μηνυμάτων του προτεινόμενου ΣΠΠ.

3.4. Σενάρια Λειτουργίας του Συστήματος

Τα σενάρια λειτουργίας του συστήματος που πηγάζουν από την δουλειά που παρουσιάστηκε σε αυτή την εργασία είναι αυτά της δημιουργίας ενός νέου πράκτορα και της ένταξής του στην κοινωνία των πρακτόρων.

3.4.1. Δημιουργία και Ένταξη Νέου Πράκτορα στην Κοινότητα

Συγκεκριμένα, με τη γέννηση κάθε νέου πράκτορα ακολουθούνται τα εξής πέντε βήματα:

1. Δημιουργία πράκτορα (μήνυμα από κάποιον πράκτορα, χρήστη, κλπ, προς το πρακτορείο τύπου M01).
2. Ανακοίνωση της γέννησής του νεοδημιουργηθέντα πράκτορα στο πρακτορείο (μήνυμα από τον πράκτορα προς το πρακτορείο τύπου M11).

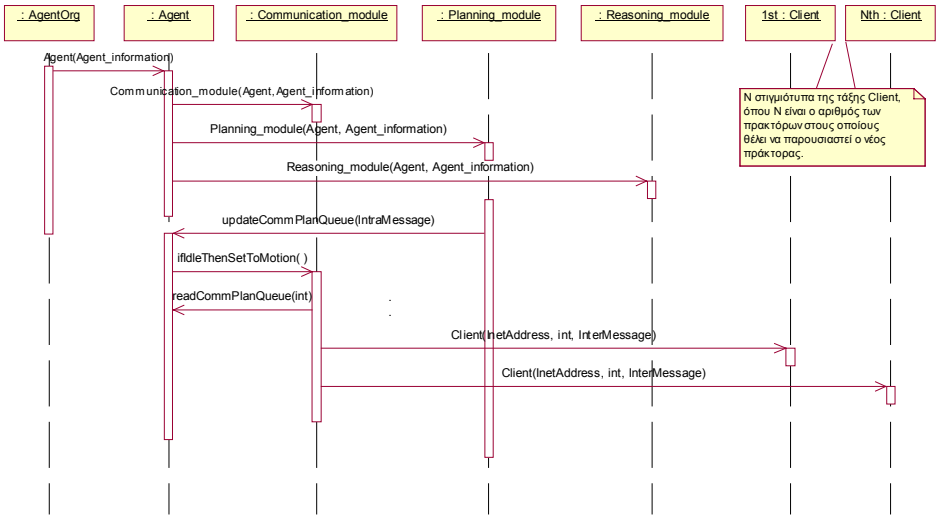
3. Αιτήσεις για πληροφόρηση σχετικά με τύπους πρακτόρων που συνεργάζονται με τον νέο πράκτορα (μηνύματα, τόσα όσοι τύποι πρακτόρων τον ενδιαφέρουν, από τον πράκτορα προς το πρακτορείο τύπου M02).
4. Απαντήσεις σχετικά με τους ζητηθέντες τύπους πρακτόρων από το πρακτορείο (μηνύματα, ένα για κάθε πράκτορα που ανήκει στους ζητηθέντες τύπους, από το πρακτορείο προς τον πράκτορα τύπου M11)
5. Ανακοίνωση της γέννησής του στους πιθανούς συνεργάτες του (μηνύματα από τον πράκτορα προς τους άλλους πράκτορες για τους οποίους πήρε πληροφορίες από το πρακτορείο τύπου M11).

Πλέον ο νέος πράκτορας είναι ενεργό μέλος της κοινότητας, γνωρίζει όσους τον ενδιαφέρουν και έχει παρουσιαστεί σε αυτούς. Οι αλληλεπιδράσεις του με τους άλλους πράκτορες ακολουθούν το μοντέλο που προτάθηκε στην παράγραφο 3.2.2.

3.4.2. Τεχνικά η Δημιουργία Νέου Πράκτορα

Στην Εικόνα 3-16 παρουσιάζεται η ακολουθία ενεργειών που συνοδεύει την δημιουργία ενός στιγμιότυπου της τάξης *Agent*. Ένα αντικείμενο της τάξης *Agent* οικοδομείται με ένα αντικείμενο της τάξης *Agent_information* σαν παράμετρο (περιέχει πληροφορίες σχετικές με τον τύπο, το όνομα, τη διεύθυνση του πράκτορα). Η μέθοδος κατασκευής στιγμιότυπων της τάξης *Agent* δημιουργεί στιγμιότυπα των τριών τμημάτων του πράκτορα (επικοινωνίας, σχεδιασμού και κρίσης) και τις ουρές του μηχανισμού διαχείρισης ενδοπρακτορικών μηνυμάτων. Τα τμήματα επικοινωνίας και κρίσης αμέσως πάνε σε κατάσταση αδράνειας.. Αντίθετα, το τμήμα σχεδιασμού αναλαμβάνει να ζητήσει από το πρακτορείο να του στείλει πληροφορίες σχετικές με τύπους πρακτόρων με τους οποίους ξέρει ότι μπορεί να συνεργαστεί. Ακολούθως ειδοποιεί αυτούς τους πράκτορες για την εμφάνισή του στην κοινωνία των πρακτόρων με κατάλληλο διαπρακτορικό μήνυμα τύπου *γέννησης*. Όλα τα παραπάνω τα πετυχαίνει βέβαια μέσω του

τμήματος επικοινωνίας. Όπως φαίνεται παραστατικά στην εικόνα το τμήμα σχεδιασμού στέλνει ένα ενδοπρακτορικό μήνυμα τύπου *γέννησης* στο τμήμα επικοινωνίας από το οποίο αποστέλλεται ως διαπρακτορικό μήνυμα σε άλλους πράκτορες. Μετά από αυτή την διαδικασία οι άλλοι πράκτορες έχουν ενημερωθεί για την ύπαρξη του νέου πράκτορα και τις ικανότητές του (αυτές στέλνονται στο σώμα του μηνύματος *γέννησης*) και θα τον χρησιμοποιήσουν σε μελλοντικές εργασίες τους.



Εικόνα 3-16: Σενάριο δημιουργίας απλού πράκτορα

4ο ΚΕΦΑΛΑΙΟ

ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΠΛΑΤΦΟΡΜΑΣ

Σε αυτό το κεφάλαιο θα παρουσιαστεί η προσπάθεια υλοποίησης της πλατφόρμας. Θα παρουσιαστούν οι επιλογές ανάπτυξης που έγιναν (γλώσσα προγραμματισμού, χρήση δικτυακών τεχνολογιών) και ακολούθως η προγραμματιστική δουλειά που έγινε.

4.1. Γλώσσα Προγραμματισμού και Περιβάλλον Υλοποίησης

Η υλοποίηση των σχεδίων που παρήχθησαν από την προτεινόμενη μεθοδολογία έγινε με τη χρήση της γλώσσας προγραμματισμού Java (Berg, 2000). Η γλώσσα αυτή κρίθηκε ως πιο κατάλληλη γιατί:

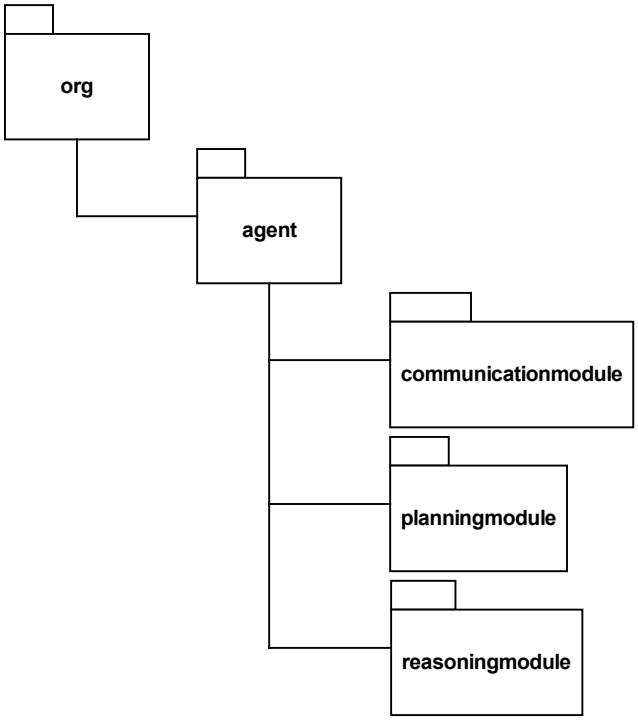
- Είναι ανεξάρτητη πλατφόρμας-λειτουργικού συστήματος. Αυτό το πλεονέκτημα ήταν σημαντικό καθώς η πλατφόρμα ανάπτυξης ΣΠΠ που προτείνουμε απευθύνεται σε όλους τους μηχανικούς ανάπτυξης λογισμικού σε όποιο λειτουργικό σύστημα και να αναπτύσσουν τις εφαρμογές τους.
- Προσφέρει πολλές έτοιμες λειτουργίες πάνω στον κώδικα που λύνουν προβλήματα συγχρονισμού (π.χ. η πρόσβαση από πολλές διεργασίες σε μια δομή δεδομένων) και προσφέρει πολύ καλή διαχείριση των νημάτων εκτέλεσης (threads). Ακόμα, προσφέρει το πακέτο net με αποτελεσματική διαχείριση των θυρών και άλλων δικτυακών υπηρεσιών του TCP/IP. Όλα

αυτά τα χαρακτηριστικά είναι σημαντικά στην υλοποίηση κατανεμημένων εφαρμογών.

- Έχει πολύ καλό ιστορικό προηγούμενο. Πολύ σημαντικές πλατφόρμες υποστήριξης ΣΠΠ είναι υλοποιημένες σε Java (Artificial Intelligence Center, 2000, Collis και Ndumu, 1999b).

4.2. Προγραμματισμός ΣΠΠ

Οι τάξεις και διεπαφές που υλοποιήθηκαν οργανώθηκαν σε πακέτα (packages) της Java. Στα πλαίσια της εργασίας αυτής δημιουργήθηκε το πακέτο *org* το οποίο περιέχει εκτός από τις δικές του τάξεις το πακέτο *agent* το οποίο περιέχει εκτός από τις δικές του τάξεις τα πακέτα *communicationmodule, planningmodule, rweasoningmodule* (Εικόνα 4-1). Οι τάξεις των πακέτων *org, agent* και *communicationmodule* αποτελούν υλοποιήσεις οι οποίες έγιναν στα πλαίσια αυτής της εργασίας.

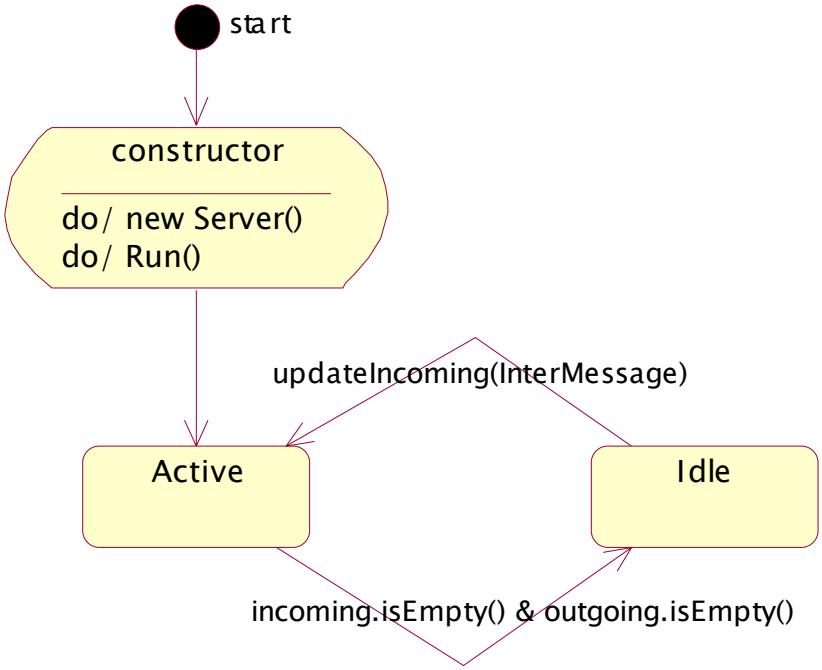


Εικόνα 4-1: Java πακέτα

Στη συνέχεια θα ακολουθήσουν παράγραφοι οι οποίοι αναφέρονται σε κάθε ένα από τα πακέτα αυτά ξεχωριστά.

4.2.1. Πακέτο org

Στο πακέτο αυτό περιλαμβάνεται η τάξη *AgentOrg* και οι διεπαφές *ServerOwner* και *Global_constants*.



Εικόνα 4-2: Διάγραμμα δραστηριότητας της τάξης AgentOrg

Στιγμιότυπα της τάξης *AgentOrg* είναι τα πρακτορεία. Τα χαρακτηριστικά της τάξης αυτής παρουσιάζονται στην Εικόνα 3-15. Η τάξη αυτή δημιουργεί ένα TCP server και η λειτουργία της στηρίζεται στην εναλλαγή μεταξύ δύο καταστάσεων: ενεργή, μη ενεργή-αδρανής. Στην Εικόνα 4-2 παρουσιάζεται η λειτουργία αυτή της τάξης. Αξιοπρόσεκτο είναι το γεγονός ότι δεν τερματίζεται ποτέ η λειτουργία του στιγμιότυπου *AgentOrg*. Σύμφωνα με την εικόνα λοιπόν ο κατασκευαστής της τάξης (constructor) δημιουργεί ένα στιγμιότυπο της τάξης *Server* και ακολούθως αρχίζει την εκτέλεση του

νήματος στο οποίο θα τρέχει το στιγμιότυπο *AgentOrg*. Το στιγμιότυπο αυτό είναι ενεργό, επεξεργάζεται εισερχόμενα μηνύματα και στέλνει τα εξερχόμενα. Όταν δεν υπάρχουν εισερχόμενα προς επεξεργασία ούτε εξερχόμενα προς αποστολή δεν έχει νόημα το νήμα αυτό να καταναλώνει πόρους του συστήματος οπότε μπαίνει σε κατάσταση αναμονής και το στιγμιότυπο *AgentOrg* μεταπηδά στην κατάσταση «Ανενεργό-Idle». Ο μόνος τρόπος για το *AgentOrg* να ξαναγυρίσει στην κατάσταση «Ενεργό-Active» είναι να έρθει κάποιο μήνυμα οπότε ο Server, του οποίου το νήμα παραμένει πάντα ενεργό, καλεί τη συνάρτηση *updateIncoming* του *AgentOrg* η οποία βάζει το μήνυμα στην ουρά εισερχομένων αλλά και ενεργοποιεί το νήμα εκτέλεσης του *AgentOrg*.

Στην Εικόνα 4-3 παρουσιάζεται σε ψευδοκώδικα ο αλγόριθμος λειτουργίας του στιγμιότυπου *AgentOrg* στην κατάσταση «Ενεργό».

```
While υπάρχουν εισερχόμενα ή εξερχόμενα μηνύματα do
      If υπάρχουν εισερχόμενα then do
            πάρε το πρώτο στην ουρά;
            If είναι τύπου M11 then κατέγραψε το νέο πράκτορα;
            If είναι τύπου M01 then δημιούργησε και κατέγραψε
το νέο
                  πράκτορα;
            If είναι τύπου M12 then κατάργησε την εγγραφή του
                  πράκτορα;
            If είναι τύπου M13 then ενημέρωσε την εγγραφή του
                  πράκτορα με τη νέα του διεύθυνση;
            If είναι τύπου M02 then do
                  for i = 1 το αριθμό πρακτόρων του τύπου που
                  ζητήθηκε do
                        βάλε στα εξερχόμενα μήνυμα τύπου M11
                        (πληροφορίες για τον πράκτορα του τύπου
που
                        ζητήθηκε) προς τον πράκτορα που ζήτησε
                        πληροφορίες;
                  End do
            End do
            Σβήσε το μήνυμα από την ουρά εισερχομένων;
      End if
      If υπάρχουν εξερχόμενα then πάρε το πρώτο στην ουρά
            Στείλε το στον κατάλληλο πράκτορα;
            If απεστάλη επιτυχώς then σβήσε το από την ουρά
                  εξερχόμενων;
      End if
End while
Πήγαινε σε κατάσταση «Idle»;
```

Εικόνα 4-3: Αλγόριθμος ενεργής κατάστασης στιγμιότυπου AgentOrg

Η διεπαφή *ServerOwner* (Εικόνα 4-4) περιέχει τη δήλωση της συνάρτησης *updateIncoming* την οποία πρέπει να ορίζουν όλες οι τάξεις που χρησιμοποιούν την τάξη *Server*. Υλοποιείται (implemented) από τις τάξεις *Communication_module*, *ComplexAgent* και *AgentOrg* (γενικά, από οποιαδήποτε τάξη περιέχει ένα *Server*).

package org;

```
import org.agent.InterMessage;

public interface ServerOwner {
      public void updateIncoming (InterMessage in);
}
```

Εικόνα 4-4: Η διεπαφή ServerOwner σε Java

```
package org;

public interface Global_constants {
        // module types
        public static final int COMM_MODULE = 1;
        public static final int PLAN_MODULE = 2;
        public static final int REAS_MODULE = 3;
        // agent types
        public static final int DAM = 1;
        public static final int DAS = 2;
        public static final int DARC = 3;
        public static final int BCM = 4;
        public static final int BCS = 5;
        public static final int UTS = 6;
        public static final int DAA = 7;
        public static final int BCA = 8;
        public static final int SG = 9;
        public static final int STS = 10;
        // MESSAGE TYPES
        public static final int M11 = 1; // νέα γνωριμία (στο body είναι
serialized ένα
         αντικείμενο τύπου Agent_information)
        public static final int M12 = 2; // ειδοποίηση θανάτου
        public static final int M14 = 7; // η νέα διεύθυνση ενός complex
πράκτορα που
         σε βάζει στην ομάδα του είναι intra-message μόνο και ζητά τις
διευθύνσεις για
         κάποιους πράκτορες οι οποίοι θα μπουν στην ομάδα του complex agent
        public static final int M13 = 3; // update contact info (change of
socket)
        public static final int M01 = 6; // new agent creation request(type,
name)
        public static final int M02 = 8; // get agent type contact info (στο
body είναι
         ο int που αντιστοιχεί σε ένα τύπο πρακτόρων)


        public static final int DEFAULT_PORT = 1001;
        // here possible reserved ports can be announced as a vector in the
future
}
```

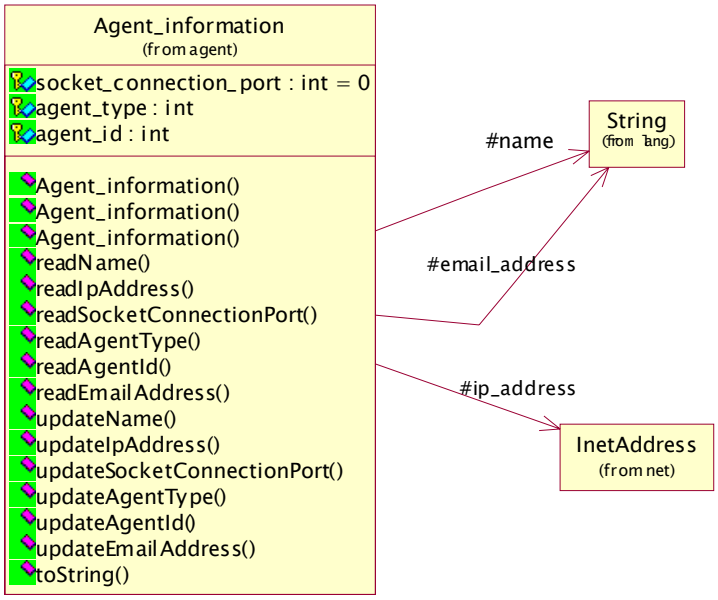
Εικόνα 4-5: Η διεπαφή Global_constants σε Java

Η διεπαφή Global_constants (Εικόνα 4-5) έχει χρησιμοποιηθεί για τον ορισμό απόλυτων μεγεθών και την διευκόλυνση της προγραμματιστικής προσπάθειας. Έτσι ενώ ο κωδικός ενός τύπου μηνύματος είναι ακέραιος και έχει πάντα την τιμή 1 αντιστοιχεί στον τύπο μηνύματος M11. Με την υλοποίηση αυτής της διεπαφής είναι δυνατό να χρησιμοποιείται ο κωδικός M11 και μέσα στον κώδικα.

4.2.2. Πακέτο org.agent

Στο πακέτο *org.agent* συμπεριλαμβάνονται οι τάξεις *Message*, *InterMessage* και *IntraMessage* οι οποίες παρουσιάζονται εξαντλητικά στην παράγραφο 3.3.3, φαίνονται στην Εικόνα 3-8. Ακόμα, συμπεριλαμβάνονται οι τάξεις *Agent*, *Module* (παράγραφος 3.3.2, Εικόνα 3-6), *ComplexAgent* (παράγραφος 3.3.4, Εικόνα 3-12) και *Agent_information*.

Η μόνη τάξη που δεν έχει περιγραφεί ήδη από τις τάξεις του πακέτου *org.agent* είναι η *Agent_information*. Το διάγραμμα τάξεων που περιέχει το UML ορισμό για την τάξη αυτή παρουσιάζεται στην Εικόνα 4-6. Οι πληροφορίες λοιπόν που αυτή τη στιγμή παρουσιάζουν ένα πράκτορα είναι το όνομά του (*String name*), η IP διεύθυνσή του (*InetAddress ip_address*), η TCP θύρα στην οποία ακούει (*int socket_connection_port*), η email διεύθυνσή του (*String email_address*), ο τύπος του (*int agent_type*) και το id του (*int agent_id*). Επειδή τα δεδομένα αυτά αποστέλλονται σε πράκτορες ως πληροφορίες για κάποιον πράκτορα η τάξη έχει το κατάλληλο δίδυμο κατασκευαστή – μεθόδου δημιουργία αντικειμένου με είσοδο *String* και μετατροπή των πληροφοριών του αντικειμένου σε *String* ώστε να είναι εύκολη η σειριακοποίησή της, η αποστολή της μέσω απλών sockets και η δημιουργία του αντικειμένου ξανά. Οι μέθοδοι που κάνουν δυνατή αυτή τη συμπεριφορά είναι οι *public Agent_information(String s)* και *public String toString()* αντίστοιχα.



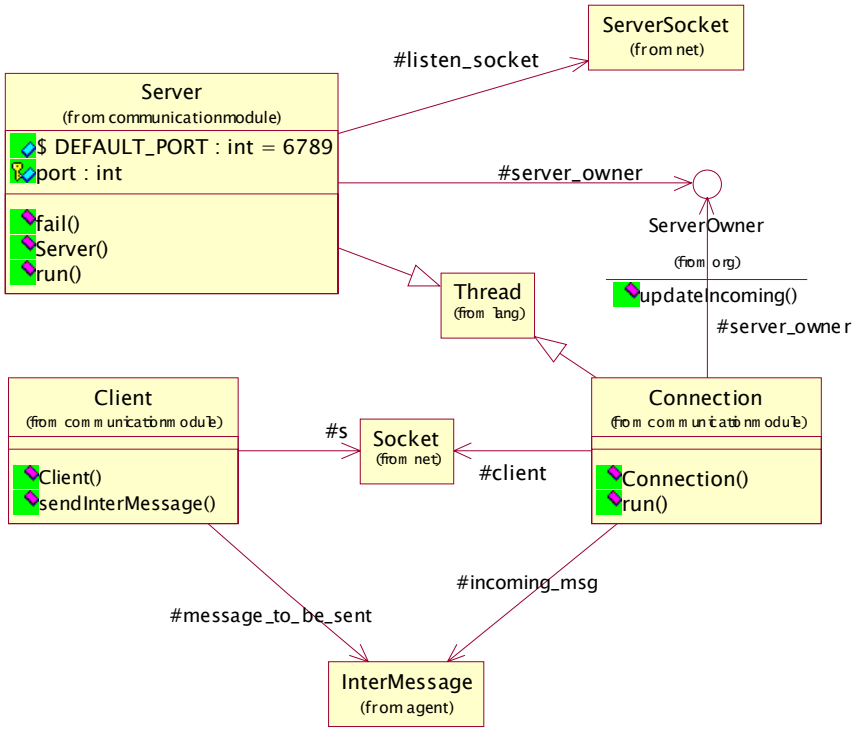
Εικόνα 4-6: Διάγραμμα τάξεων Agent_information

Εδώ αξίζει μια ιδιαίτερη αναφορά στην τάξη Module της οποίας υποτάξεις είναι όλα τα τμήματα ενός πράκτορα. Τα τμήματα του πράκτορα έχουν μια λειτουργία η οποία μοιάζει πολύ με αυτή που περιγράφηκε στην τάξη *AgentOrg* και συγκεκριμένα στην Εικόνα 4-2. Η διαφορά εδώ πέρα είναι ότι τα τμήματα πάνε από την κατάσταση «Αδρανές» στην κατάσταση «Ενεργό» όταν καλέσει η τάξη *Agent* την συνάρτησή τους *ifIdleThenSetToMotion*.

4.2.3. Πακέτο org.agent.communicationmodule

Στο πακέτο αυτό περιέχονται η τάξη *Communication_module* μαζί με την τάξη *AddressBook* (η οποία είναι μια δομή δεδομένων υποσύνολο της *Agent_information*, περιέχει μόνο πεδία που αφορούν τη διεύθυνση ενός πράκτορα, δηλαδή το όνομά του, την IP διεύθυνσή του, την TCP θύρα στην οποία ακούει και την email διεύθυνσή του) και τις τάξεις που αναλαμβάνουν την διακίνηση μηνυμάτων μέσω του sockets του TCP πρωτοκόλλου. Οι τάξεις αυτές είναι οι *Connection*, *Client* και *Server*.

Στην Εικόνα 4-7 παρουσιάζονται οι τρεις αυτές τάξεις. Η λειτουργία τους είναι η εξής.



Εικόνα 4-7: Διάγραμμα τάξεων Server, Client και Connection

- Ο *Server* αρχικοποιείται και τρέχει στο δικό του νήμα εκτέλεσης (είναι υποτάξη της τάξης *Thread*). Ακούει σε ένα *ServerSocket* για εισερχόμενα μηνύματα.

- Για την αποστολή ενός μηνύματος δημιουργείται ένα στιγμιότυπο της τάξης *Client*. Η τάξη *Client* δημιουργεί ένα στιγμιότυπο της τάξης *Socket* και μέσω αυτής στέλνει το μήνυμα σε έναν *Server*. Επειδή ο κατασκευαστής της τάξης δεν μπορεί να επιστρέφει κάποια τιμή σε αυτόν που δημιουργεί το στιγμιότυπο, η τάξη *Client* περιέχει τη μέθοδο *Boolean sendInterMessage*() η οποία επιστρέφει *true* αν το μήνυμα εστάλη επιτυχώς και *false* στην αντίθετη περίπτωση. Έτσι αυτός που θέλει να στείλει το μήνυμα (τάξη *Communication_module* ή *ComplexAgent* ή *AgentOrg*) ξέρει αν αυτό πήγε στον παραλήπτη ή όχι.

- Όταν ένας *Client* ζητήσει από ένα *Server* την παραλαβή ενός μηνύματος τότε ο τελευταίος δημιουργεί ένα στιγμιότυπο της τάξης *Connection* το

οποίο αναλαμβάνει να παραλάβει το μήνυμα ενώ ο *Server* «ακούει» για άλλες αιτήσεις παραλαβής μηνυμάτων. Η τάξη *Connection* τρέχει και αυτή σε δικό της νήμα εκτέλεσης (είναι υποτάξη της τάξης *Thread*). Λαμβάνει το εισερχόμενο *InterMessage* και καλεί τη συνάρτηση *updateIncoming* του αντικειμένου *ServerOwner* που φιλοξενεί τον *Server*.

```
If υπάρχει εισερχόμενο ενδοπρακτορικό μήνυμα Then do
      If είναι τύπου M14 then do
            Δημιούργησε διαπρακτορικά μηνύματα τύπου M13 με τη
νέα
                  θύρα στην οποία ακούει ο πολύπλοκος πράκτορας
ένα
                  για κάθε πράκτορα που θα συμμετέχει στην
ομάδα του
                  πολύπλοκου πράκτορα;
            Εισήγαγέ τα στην ουρά εξερχόμενων διαπρακτορικών
                  Μηνυμάτων;
      Else do
            Μετέτρεψέ το ενδοπρακτορικό σε διαπρακτορικό
μήνυμα;
            Εισήγαγέ το στην ουρά εξερχόμενων διαπρακτορικών
                  Μηνυμάτων;
      End if
End if
While υπάρχουν εισερχόμενα ή εξερχόμενα διαπρακτορικά μηνύματα
do
      If υπάρχουν εισερχόμενα then do
            πάρε το πρώτο στην ουρά;
            If είναι τύπου M11 then κατέγραψε τη διεύθυνση του
νέου
                  πράκτορα;
            If είναι τύπου M12 then κατάργησε την εγγραφή του
                  πράκτορα;
            If είναι τύπου M13 then ενημέρωσε την εγγραφή του
                  πράκτορα με τη νέα του διεύθυνση;
            Δημιούργησε το κατάλληλο ενδοπρακτορικό μήνυμα και
                  προώθησέ το στο τμήμα σχεδιασμού δράσης;
            Σβήσε το μήνυμα από την ουρά εισερχομένων;
      End if
      If υπάρχουν εξερχόμενα then do
            Πάρε το πρώτο στην ουρά;
            Στείλε το στον κατάλληλο πράκτορα;
            If απεστάλη επιτυχώς then σβήσε το από την ουρά
                  Εξερχόμενων else βάλε το τελευταίο στην ουρά;
      End if
End while
If υπάρχει εισερχόμενο ενδοπρακτορικό μήνυμα Then πήγαινε στην
αρχή;
Πήγαινε σε κατάσταση «Idle»;
```

Εικόνα 4-8: Αλγόριθμος ενεργής κατάστασης στιγμιότυπου Communication_module

Η τάξη *Communication_module* έχει μια ιδιαιτερότητα σε σχέση με τα άλλα τμήματα του πράκτορα. Μπορεί να μεταπέσει από την κατάσταση «Αδρανές» στην κατάσταση «Ενεργό» (Εικόνα 4-2) και όταν έρθει διαπρακτορικό μήνυμα (κλήση συνάρτησης *updateIncoming* από το αντικείμενο *Connection*) και όταν έρθει ενδοπρακτορικό μήνυμα (κλήση συνάρτησης *ifIdleThenSetToMotion* από το αντικείμενο *Agent* ή *ComplexAgent*). Ο αλγόριθμος λειτουργίας του στην ενεργή κατάσταση παρουσιάζεται στην Εικόνα 4-8 σε μορφή ψευδοκώδικα.

5ο ΚΕΦΑΛΑΙΟ

ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΟΙ ΣΤΟΧΟΙ

Η εργασία αυτή παρουσίασε μια πλατφόρμα ανάπτυξης συστημάτων πολλαπλών πρακτόρων ανεξάρτητη λειτουργικού συστήματος και εφαρμογής. Ακόμα παρουσίασε τη μεθοδολογία ανάπτυξης και πρότεινε ένα ΣΠΠ για την υποστήριξη αποφάσεων στο μάρκετινγκ.

Το προτεινόμενο σύστημα είναι ΣΠΠ γιατί (Jennings et al., 1998):

- κάθε πράκτορας έχει περιορισμένο γνωστικό πεδίο (δεν μπορεί μόνος του να λύσει το πρόβλημα του συστήματος),
- το σύστημα δεν ελέγχεται από κάποιο σφαιρικό ελεγκτή ή διεργασία,
- τα δεδομένα είναι αποκεντρωμένα και
- οι υπολογισμοί είναι ασύγχρονοι.

Το προτεινόμενο ΣΠΠ δημιουργείται με τη χρήση μιας γενικής επαναχρησιμοποιήσιμης αρχιτεκτονικής πρακτόρων, η οποία αναφέρεται ταυτόχρονα σε δύο επίπεδα, το λειτουργικό επίπεδο και το δομικό επίπεδο του πράκτορα. Στο λειτουργικό επίπεδο ξεχωρίζουν τρία είδη πρακτόρων, πράκτορες εργασιών, πράκτορες πληροφοριοδότες και πράκτορες διεπαφής. Στο δομικό επίπεδο διακρίνονται οι στοιχειώδεις και οι πολύπλοκοι πράκτορες. Η διαφοροποίηση των πρακτόρων στο δομικό επίπεδο επιτρέπει στο σύστημά μας να υποστηρίζει πολλές απόψεις σε πολλαπλά επίπεδα

αφαίρεσης στη διαδικασία λήψης αποφάσεων. Αυτή η ιδιαιτερότητα είναι σημαντική για το στρατηγικό μάρκετινγκ όπου η λήψη απόφασης περιλαμβάνει τη συνεργασία ειδικών σε διαφορετικά τμήματα (διαφορετική άποψη) και σε διαφορετική θέση στην εταιρία (διαφορετικό επίπεδο αφαίρεσης). Άλλο τόσο σημαντική είναι η υποστήριξη αλληλεπίδρασης μεταξύ αποφασιζόντων αλλά και ομάδας αποφασιζόντων.

Η διαφορά του προτεινόμενου ΣΠΠ με άλλες παρόμοιες εργασίες (Brazier et all, 1995, Ishida et all, 1990, Jennings et all, 1995) είναι η, όπως στην εργασία των Sycara και Zeng (1996), διαφοροποίηση μεταξύ των τριών τύπων πρακτόρων στο λειτουργικό επίπεδο, επιτρέποντας την αποτελεσματική λειτουργία του ΣΠΠ σε πραγματικές πολύπλοκες εργασίες οι οποίες χρειάζονται συντονισμό, πληροφόρηση από ετερογενείς πηγές δεδομένων και αλληλεπίδραση με το χρήστη. Σε σύγκριση με την εργασία των Sycara και Zeng (1996) η πρωτοτυπία και βελτίωση που προτείνεται είναι στη δομική διαφορά των τριών τύπων πρακτόρων (παρόλο που στην εργασία αυτή παρουσιάζονται μόνο πολύπλοκοι πράκτορες εργασίας). Χρησιμοποιώντας την έννοια του πολύπλοκου πράκτορα, δηλαδή συγκεντρώνοντας σε μια ομάδα όλους τους πράκτορες οι οποίοι εμπλέκονται στην περάτωση μιας πολύπλοκης εργασίας, η πολυπλοκότητα του συντονισμού του ΣΠΠ μειώνεται δραστικά κάνοντας πολύ εύκολη τη μοντελοποίηση εφαρμογών. Στην πραγματικότητα, ακόμα και σε μια εφαρμογή μεγάλης εμβέλειας, ο συντονισμός των εργασιών γίνεται με συνεργασία πρακτόρων (στοιχειώδεις ή πολύπλοκους) οι οποίοι ανήκουν σε σχετικά μικρές ομάδες ή μεταξύ πολύπλοκων πρακτόρων ενός ψηλότερου επιπέδου.

Όσον αφορά την εφαρμογή στο στρατηγικό μάρκετινγκ αυτή είναι η πρώτη φορά που μια πολυκριτήρια μεθοδολογία μοντελοποιείται σύμφωνα με την κατανεμημένη διάστασή της στον πραγματικό κόσμο. Τεχνικές της κατανεμημένης τεχνητής νοημοσύνης έχουν ξαναχρησιμοποιηθεί στο παρελθόν για την ανάπτυξη εφαρμογών στρατηγικού μάρκετινγκ (Pinson and Moraitis. 1996).

Η πρωτοτυπία της παρουσιαζόμενης δουλειάς έγκειται στην υλοποίηση ενός συστήματος βασισμένου σε πράκτορες οι οποίοι χρησιμοποιούν *πολυκριτήριες μεθοδολογίες* για ανάλυση αγορών και πρόταση στρατηγικής μάρκετινγκ.

Όσον αφορά την προτεινόμενη πλατφόρμα ανάπτυξης ΣΠΠ, αυτή υπερέχει σημαντικά σε σχέση με αυτές που κυκλοφορούν στο εμπόριο και που έχουν δημοσιευτεί.

Η πλατφόρμα Grasshopper έχει σημαντικά μειονεκτήματα καθώς αφενός προβλέπει μόνο μία τάξη πράκτορα η οποία προσφέρει αντίστοιχα ένα νήμα εκτέλεσης, αφετέρου συνδέει την εκάστοτε εφαρμογή με ένα πρακτορείο σχετικό με αυτήν φορτώνοντας ένα σύστημα με πολλές ομάδες πρακτόρων με πρακτορεία.

Η πλατφόρμα OAA από την άλλη έχει το σημαντικό μειονέκτημα του εξυπηρετητή πράκτορα ο οποίος ρυθμίζει όλες τις αλληλεπιδράσεις μεταξύ των πρακτόρων του συστήματος. Ουσιαστικά δεν είναι πλατφόρμα ανάπτυξης ΣΠΠ αλλά κατανεμημένων λυτών προβλημάτων με μια κεντρική διοίκηση.

Η πλατφόρμα AgentBuilder έχει το πολύ σημαντικό μειονέκτημα ότι οι πράκτορες ορίζονται σε μια νέα γλώσσα προγραμματισμού η οποία μεταγλωττίζεται κατά την εκτέλεση του προγράμματος όπως η Java και αυτή τρέχει με τη σειρά της σε Java. Έτσι το πρόγραμμα ουσιαστικά μεταγλωττίζεται δύο φορές τη στιγμή της εκτέλεσης και είναι σημαντικά αργό.

Όσον αφορά τις παλιότερες RETSINA και ARCHON το μεγάλο πλεονέκτημα της προτεινόμενης πλατφόρμας είναι η ιδέα του πολύπλοκου πράκτορα ο οποίος προσφέρει έναν έτοιμο μηχανισμό συντονισμού ομάδας πρακτόρων.

Σε σύγκριση με το ZEUS το μόνο ουσιαστικό πλεονέκτημα της προτεινόμενης πλατφόρμας είναι ο ανεξάρτητος τμήματος μηχανισμός επικοινωνίας των τμημάτων ενός πράκτορα το οποίο όμως την κάνει γενικότερη και καλύτερα προσαρμόσιμη στις ανάγκες κάθε εφαρμογής.

Η καινοτομία που παρουσιάζεται στην εργασία αυτή είναι η αρχιτεκτονικές των πρακτόρων (στοιχειώδους και πολύπλοκου) και συγκεκριμένα τα χαρακτηριστικά τους:

α) η παροχή τρόπου διαβίβασης ενδοπρακτορικών μηνυμάτων μεταξύ τμημάτων του πράκτορα ο οποίος τρόπος είναι ανεξάρτητος τμήματος και

β) η δυναμική δημιουργία του τμήματος κρίσης του πολύπλοκου πράκτορα κάθε φορά που αυτός αναλαμβάνει κάποιο έργο.

Τα πλεονεκτήματα αυτών των αρχιτεκτονικών επιλογών είναι:

- η πραγματική δημιουργία μιας πλατφόρμας ανάπτυξης πρακτόρων ανεξάρτητης θέματος-εφαρμογής με χαρακτηριστικά ελάχιστης πολυπλοκότητας,
- η δημιουργία μιας αρχιτεκτονικής η οποία αντιμετωπίζει και το θέμα της οργάνωσης ομάδων πρακτόρων πάλι ανεξαρτήτως θέματος-εφαρμογής,
- η ικανότητα της προτεινόμενης αρχιτεκτονικής πρακτόρων να προσαρμοστεί και να υλοποιήσει αρχιτεκτονικές πολλαπλών οριζόντιων ή καθέτων επιπέδων (όπως αναφέρθηκαν στην παράγραφο 2.2.2). Αυτό επιτυγχάνεται με τον επαναπρογραμματισμό των προσβάσεων των τμημάτων του πράκτορα στις ουρές ενδοπρακτορικών μηνυμάτων που υλοποιούνται σε επίπεδο πράκτορα. Έτσι αν όλα τα τμήματα γράφουν και διαβάζουν από την ίδια ουρά έχουμε μια αρχιτεκτονική πολλαπλών οριζόντιων επιπέδων ενώ σε αυτή την διατριβή παρουσιάστηκε μια αρχιτεκτονική με δύο κάθετα επίπεδα, αυτό του σχεδιασμού δράσης και αυτό της κρίσης-εκτέλεσης εργασιών. Σε αυτή την υλοποίηση, βέβαια, τα τμήματα αυτά δεν λειτουργούν σαν κάθετα επίπεδα αλλά θα ήταν πολύ εύκολη η λειτουργία τους ως τέτοια. Προσθέτοντας ουρές μηνυμάτων και τμήματα η αρχιτεκτονική μπορεί να υποστηρίξει πολλά κάθετα τμήματα-επίπεδα.

Οι μελλοντικοί στόχοι του συγγραφέα είναι η συνέχεια της δουλειάς που παρουσιάστηκε σε αυτήν τη διατριβή. Η συνέχεια αυτή μπορεί να είναι η

δημιουργία πολύπλοκων πρακτόρων διεπαφής ή πληροφοριοδότες. Θα μπορούσε να αφορά το δυναμικό σχεδιασμό δράσεων των πρακτόρων μιας ομάδας. Ακόμα, οι τομείς του δυναμικού συντονισμού ομάδας πρακτόρων ή της διαπραγμάτευσης αυτών παρουσιάζουν εξαιρετικό ερευνητικό ενδιαφέρον.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Artificial Intelligence Center. 2000. Open Agent Architecture: Technical White Paper. SRI International, http://www.ai.sri.com/~oaa/whitepaper.html.

Agent Working Group. 2000. Agent Technology Green Paper. Object Management Group: OMG Document ec/2000-08-01.

Berg, C.J.. 2000. Advanced Java 2 Development for Enterprise Applications. 2nd ed. Sun Microsystems Press. ISBN: 0130848751.

Brazier, F.M.T., Dunin-Keplicz, B.M., Jennings, N.R., Treur, J. 1995. Formal specification of multi-agent systems. First International Conference on Multi-Agent Systems (ICMAS'95). San Francisco, CA: 25-32.

Brazier, F.M.T., Dunin-Keplicz, B.M., Jennings, N.R., Treur, J.. 1997. DESIRE: Modeling Multi-Agent Systems in a Compositional Formal Framework. Huhns, M., Singh, M. (eds.): International Journal of Cooperative Information Systems. Special issue on Formal Methods in Cooperative Information Systems: Multi-Agent Systems.

Cammarata, S., McArthur, D, Steeb, R.. 1983. Strategies of cooperation in distributed problem solving. Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI-83). Karlsruhe. Federal Republic of Germany: 767-770.

Collis, J. and Ndumu, D.. 1999a. Zeus Methodology Documentation Part III: The Application Realization Guide. Intelligent Systems Research Group, BT labs. British Telecommunications.

Collis, J. and Ndumu, D.. 1999b. Zeus Technical Manual. Intelligent Systems Research Group, BT Labs. British Telecommunications.

Collis, J. and Ndumu, D.. 1999c. Zeus Methodology Documentation Part I: The Role Modelling Guide. Intelligent Systems Research Group, BT labs. British Telecommunications.

Corera, J.M., Laresgoiti, I., Jennings N.R.. 1996. Using Archon, Part 2: Electricity Transportation Management. Intelligent agents. IEEE Expert: 71-79.

Corkill, D.D. and Lesser, V.R.. 1983. The use of meta-level control for coordination in a distributed problem solving network. Proc. Int. Jt. Conf. Artif. Intell., 8th, Karlsruhe, Germany: 748-756.

Decker, K.S.. 1987. Distributed problem-solving techniques: A survey. IEEE Trans. Syst. Man Cybernet. 17(5): 729-740.

Durfee, E.H. and Lesser, V.R.. 1987. Using partial global plans to coordinate problem solvers, Proc. Int. Jt. Conf. Artif. Intell., 10th, pp. 875-883.

Durfee, E.H., Lesser, V.R., Corkill, D.D.. 1989. Trends in cooperative distributed problem solving. IEEE Trans. Knowl. Data Eng. KOE-11 (1): 63-83.

Eriksson, H.-E. and Penker, M.. 1998. UML Toolkit. Wiley. ISBN: 0471191612.

Ferguson, I.A.. 1992. Toward an architecture for adaptive, rational, mobile agents. Decentralized Artificial Intelligence 3. Eds. E. Werner and Y. Demazeau, pp. 249-261 Elsevier/North-Holland, Amsterdam.

Gasser, L.. 1986. The integration of computing and routine work. ACM Trans. Off. Inf. Syst. 4(3): 205-225.

Gasser, L.. 1991. Social conceptions of knowledge and action: DAI foundations and open systems symantics. Artificial Intelligence, 47: 107-138.

Georgeff, M.. 1983. Communication and interaction in multi-agent planning. Proc. Int. Jt. Conf. Artif. Intell.. Karlsruhe, Germany: 125-129.

Haddadi, A. and Sundermeyer, K.. 1996. Belief-Desire-Intention Agent Architectures. Foundations of Distributed Artificial Intelligence. Eds: O'Hare, G.M.P. and Jennings, N.R. Willey, ISBN 0471006750: 169-185.

Hayes-Roth, B., Hewett, M., Washington, R., Hewett, R., and Seiver, A.. 1989 Distributed intelligence within an individual. Distributed Artificial Intelligence. Eds. Gasser, L. and Huhns, M.. Vol. 2. Morgan Kaufmann, Los Altos, CA/Pitman London: 385-412.

Huhns, M. N., Makhopadhyay, U., Stephens, L. M., and Bonnell, R.D.. 1987. DAI for document retrieval: The MINDS project. Distributed Artificial Intelligence. Ed. Huhns, M.N.. Morgan Kaufmann, San Mateo, CA/Pitman, London: 249-283.

IKV++ GmbH. 1999a. Grasshopper Technical Overview. Revision 1.1. http://www.ikv.de.

IKV++ GmbH. 1999b. Grasshopper, Release 1.2, Basics and Concepts. Revision 1.1. http://www.ikv.de.

Ishida, T., Yokoo, M., Gasser, L.. 1990. An organizational approach to adaptive production systems. Proceedings of AAAI-90. Boston, Mass.

Jennings, N.R.. 1993. Coordination: Commitment and conventions, the foundation of coordination in multi-agent systems. Knowl. Eng. Rev. 8(3): 223-250.

Jennings, N.R., Corera, J.M., Laresgoiti, I.. 1995. Developing industrial multi-agent systems. First International Conference on Multi-Agent Systems (ICMAS'95). San Francisco, CA: 423-430.

Jennings, N.R., Faratin, P., Johnson, M. J., O'Brien, P., Wiegand, M.E.. 1996a. Using Intelligent Agents to Manage Business Processes. Proceedings of PAAM'96: 345-360.

Jennings, N.R., Mandami, E.H, Corera, J.M., Laresgoiti, I., Perriolat, F., Skarek, P. Vargam L.Z.. 1996b. Using Archon to Develop Real-World DAI Applications, Part 1. Intelligent agents. IEEE Expert: 64-70.

Jennings, N.R., Sycara, K., Wooldridge, M.. 1998. A Roadmap of Agent Research and Development. Int Journal of Autonomous Agents and Multi-Agent Systems 1 (1): 7-38.

Jennings, N.R., Faratin, P.A., Lomuscio, R., Parsons, S., Sierra, C., Wooldridge, M.. 2001. Automated negotiation: prospects, methods and challenges. Int. J. of Group Decision and Negotiation 10 (2): 199-215.

Kendall, E.A.. 1998. Agent Roles and Role Models: New abstractions for Multiagent System Analysis and Design. International Workshop on Intelligent Agents in Information and Process Management.

Kiss, G.. 1992. Variable coupling of agents to their environment: Combining situated and symbolic automata. Decentralized Artificial Intelligence 3. Eds. Werner, E. and Demazeau, Y.. Elsevier/North-Holland, Amsterdam: 231-248.

Kotler, P.. 1994. Marketing management: Analysis, planning, implementation and control. 8th ed. Prentice-Hall, London.

Kristensen, B.B.. 1996. Object Oriented Modelling with Roles. OOIS'95, Proceedings of the 2nd International Conference on Object Oriented Information Systems. http://www.mip.ou.dk/~bbk/research/recent_publications.html

Lesser, V.R. and Corkill, D.D.. 1983. The distributed vehicle monitoring testbed : A tool for investigating distributed problem solving networks. AI Mag. Fall: 15-33.

Levesque, H.J., Cohen, P.R., Nunes, J.H.T.. 1990. On acting together. Proceedings of the 8th International Conference on Artificial Intelligence. Boston, MA: 94-99.

Liberatore, M.J., Stylianou, A.C.. 1995. Expert support systems for new product development decision making. A modeling framework and applications. Management Science, Vol. 41. No. 8: 1296-1316.

Maines, D.. 1984. Urban Life. Special Issue on Negotiated Order Theory.

Malone, T.W.. 1990. Organizing information processing systems: Parallels between human organizations and computer systems. Cognition, Computation and Cooperation. Eds. Zachary, W.W. and Robertson, S.P.. Ablex, Norwood, NJ: 56-83.

Matsatsinis, N.F., Siskos, Y.. 1999. MARKEX: An intelligent decision support system for product development decisions. European Journal of Operational Research, Vol. 113. No. 2: 336-354.

Mesarovic, M.D., Marko, D., Takahara, Y.. 1970. Theory of Hierarchical, Multilevel, Systems. Academic Press, New York.

Moulin, B. and Chaib-Draa, B.. 1996. An Overview of Distributed Artificial Intelligence. Foundations of Distributed Artificial Intelligence. Eds: O'Hare, G.M.P. and Jennings, N.R. Willey, ISBN 0471006750: 3-55.

Nierenburg, S., and Lesser, V.R.. 1986. Providing intelligent assistance in distributed office environments Distributed Artificial Intelligence Eds A. H. Bond and L. Gasser. Morgan Kaufmann. San Mateo, CA.

Nii, H.P.. 1986. Blackboard systems: The blackboard model of problem solving and the evolution of blackboards architectures. Part I. Al Mag.: 38-53.

Odell, J.H., Parunak, V.D., Bauer, B.. 2000. Extending UML for Agents. Proc. of the Agent-Oriented Information Systems Workshop (AOIS) at the 17th National conference on Artificial Intelligence (AAAI). Eds. Wagner, G., Lesperance, Y., Yu, E.. Austin, TX: 3-17

Parsons, S., Sierra, C. Jennings, N.R.. 1998. Agents that reason and negotiate by arguing. Journal of Logic and Computation. Vol. 8, No. 3: 261-292.

Pinson, S. and Moraitis, P.. 1996. An Intelligent Distributed System for Strategic Decision Making. Journal Group Decision and Negotiation, 6, Kluwer Academic Publishers: 77-108.

Reticular Systems Inc. 1999. AgentBuilder An Integrated Toolkit for Constructing Intelligent Software Agents. Revision 1.3. http://www.agentbuilder.com.

Rosenschein, J.S.. 1986. Rational interactions: Cooperating among intelligent agents. Ph.D. Disertation, Computer Science Department, Stanford University, Stanford, CA.

Rosenschein, J.S. and Breese, J.S.. 1989. Communication-free interactions among rational agents. Distributed Artificial Intelligence 2. Eds. Gasser, L. and Huhns M.N.. Morgan Kaufmann, Los Altos, CA/Pitman, London: 99-118.

Rosenschein, J.S. and Zlotkin, G.. 1994. Rules of Encounter. MIT Press.

Searle, J.R.. 1969. Speech Acts. Cambridge Univ. Press, Cambridge, UK.

Shoham, Y.. 1991. AGENT-0: A simple agent language and its interpreter. Proceedings of the Ninth National Conference on Artificial Intelligence. Vol II. Anaheim, CA. MIT Press: 704-709.

Shoham, Y.. 1993. Agent-oriented programming. Artificial Intelligence 60(1): 51-92.

Siskos, J., Yannacopoulos, D.. 1985. UTASTAR: An ordinal regression method for building additive value functions. In: Investiçao Operational, Vol. 5. No. 1: 39-53.

Smith, R.G. and Davis, R.. 1981. Frameworks for cooperation in distributed problem solving. IEEE Trans. Syst. Man Cybernet. SMC-11: 61-70.

Strauss, A.. 1978. Negotiations: Varieties, Processes, Contexts and Social Order. Jossey-Bass, San Frncisco.

Sycara K.. 1990. Negotiation planning: An AI approach. European Journal of Operational research. Vol. 46: 216-234.

Sycara, K., Zeng, D.. 1996. Coordination of Multiple Intelligent Software Agents. International Journal of Cooperative Information Systems. World Scientific Publishing Company.

Sycara, K., Pannu, A., Williamson, M., Zeng, D., Decker, K.. 1996. Distributed Intelligent Agents. Intelligent agents. IEEE Expert: 36-46.

Thomas, S.R.. 1994. The PLACA Agent Programming Language. Lecture Notes in Artificial Intelligence. Wooldridge, M.J. & Jennings, N.R. (Eds.). Berlin: Springer-Verlag: 355-370.

Van Bruggen, G.H.. 1992. Performance effects of a marketing decision support system: A laboratory experiment. Proceedings of the 21st Annual Conference of the European Marketing Academy, AARHUS.

Wierenga, B.. 1992. Knowledge-based systems in marketing: Purpose, Performance, Perceptions and Perspectives. Working Paper 112, Rotterdam School of Management, Erasmus University.

Witting, T. (Ed.). 1992. ARCHON: An Architecture for Multi-Agent Systems. Ellis Horwood Series in AI.

Wood, M.F. and DeLoach, S.A.. 2000. An Overview of the Multiagent Systems Engineering Methodology. AOSE-2000, The First International Workshop on Agent-Oriented Software Engineering. Limerick, Ireland.

Wooldridge, M., Jennings, N.R., Kinny, D.. 2000. The Gaia Methodology for Agent-Oriented Analysis and Design. Journal of Autonomous Agents and Multi-Agent Systems Vol. 3. No. 3: 285-312.

Μωραΐτης, Π.. 2001. Μηχανική Γνώσεων και Αποφάσεων. Σημειώσεις μαθήματος Μηχανική Γνώσεων και Αποφάσεων του Τμήματος Μηχανικών Παραγωγής και Διοίκησης του Πολυτεχνείου Κρήτης. http://www.ergasya.tuc.gr/tuc/dep/ergasya/ mixanikiGnoseon.htm

Χριστοδουλάκης Δ. και Ξένος Μ.. 1995. Τεχνολογία Λογισμικού, Αρχές και Μεθοδολογίες. Σημειώσεις μαθήματος Τεχνολογία Λογισμικού του Τμήματος Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής του Πανεπιστήμιου Πατρών. Εκτυπωτικό Κέντρο Πανεπιστημίου Πατρών.

Ψωματάκης Ε.. 2001. Ανάλυση και μοντελοποίηση σχεδίασης δράσης και λογικής κρίσης πρακτόρων σε πρότυπο σύστημα πολλαπλών πρακτόρων για την υλοποίηση συστήματος υποστήριξης αποφάσεων. Ερευνητική μεταπτυχιακή διατριβή. Πολυτεχνείο Κρήτης. Βιβλιοθήκη.

ΠΑΡΑΡΤΗΜΑΤΑ

Συντομογραφίες

ΕΠ	Ευφυής Πράκτορας (Intelligent Agent)
ΣΠΠ	Σύστημα Πολλαπλών Πρακτόρων (Multi-Agent System)
ACL	Agent Communication Language (γλώσσα επικοινωνίας πρακτόρων)
AFIT	Air Force Institute of Technology (ινστιτούτο τεχνολογίας της Πολεμικής Αεροπορίας των ΗΠΑ)
BDI	Belief-Desire-Intention (αρχιτεκτονικές που βασίζονται σε δομές πεποιθήσεων, επιθυμιών και προθέσεων)
CDPS	cooperative distributed problem-solving (συνεργατικοί κατανεμημένοι λυτές προβλημάτων):
DAI	Distributed Artificial Intelligence (κατανεμημένη τεχνητή νοημοσύνη)
FIFO	First In – First Out (είναι τρόπος αναμονής σε ουρά, το πρώτο αντικείμενο που μπαίνει, βγαίνει πρώτο)
FIPA	Foundation for Intelligent Physical Agents (Θεμελίωση για ευφυείς φυσικούς πράκτορες)
KQML	Knowledge Query Manipulation Language (γλώσσα χειρισμού ερωταπαντήσεων γνώσης)
MaSE	Multi-agent Systems Engineering (μηχανική συστημάτων

	πολλαπλών πρακτόρων)
MASIF	Mobile Agent System Interoperability Facility (προτυποποίηση του τρόπου παροχής ενδοεπικοινωνίας και λειτουργικότητας ενός συστήματος κινητών πρακτόρων)
OMG	Object Management Group (ομάδα διαχείρισης αντικειμένων)
SSL	Secure Socket Layer (ασφαλής επικοινωνία σε επίπεδο socket του πρωτοκόλλου TCP)
TCP/IP	Transport Control Protocol/ Internet Protocol: τα πρωτόκολλα υπεύθυνα για την αποστολή και λήψη πακέτων δεδομένων (TCP) και διευθυνσιοδότησης (IP), αντίστοιχα, στο διαδίκτυο
UML	Unified Modelling Language (ενοποιημένη γλώσσα μοντελοποίησης): Χρησιμοποιείται για την αντικειμενοστραφή μοντελοποίηση και σχεδίαση πληροφοριακών συστημάτων

Ευρετήριο Εικόνων

Ευρετήριο Πινάκων