ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΕΦΑΡΜΟΓΕΣ ΕΥΦΥΩΝ ΠΡΑΚΤΟΡΩΝ

Όνομα φοιτητή : ΧΡΗΣΤΟΣ ΤΖΙΒΑΝΑΚΗΣ

Επιβλέπων καθηγητής : Δρ. ΔΗΜΟΣΘΕΝΗΣ ΣΤΑΜΑΤΗΣ

ΘΕΣΣΑΛΟΝΙΚΗ 2012

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ

Στα πλαίσια της παρούσας πτυχιακής εργασίας εξετάζεται το παράδειγμα των ευφυών πρακτόρων, δηλαδή συστημάτων λογισμικού που ενεργούν εκ μέρους του χρήστη. Παρουσιάζονται οι βασικότερες έννοιες των ευφυών πρακτόρων, οι μηχανισμοί λήψης αποφάσεων, σχεδιασμού, μάθησης και η αρχιτεκτονική τους όπως αυτές συναντώνται στην σχετική βιβλιογραφία (Fasli, 2007 και Wooldridge, 2009). Γίνεται αναφορά στη δυνατότητα συνύπαρξης και τα είδη αλληλεπίδρασης μεταξύ των πρακτόρων μέσω κατάλληλων γλωσσών επικοινωνίας που εξυπηρετούν στην υλοποίηση Συστημάτων Πολλαπλών Πρακτόρων. Οι δυνατότητες μοντελοποίησης που προσφέρει ένα τέτοιο σύστημα διευκολύνει σημαντικά την αντιμετώπιση σύνθετων προβλημάτων. Σαν μελέτη περίπτωσης έχει επιλεγεί ένα πιλοτικό Σύστημα Πολλαπλών Πρακτόρων που εξετάζει διαφορετικά σενάρια αποκομιδής απορριμμάτων σε μία αστική συνοικία. Το προγραμματιστικό περιβάλλον που χρησιμοποιήθηκε για την μοντελοποίηση και προσομοίωση του παραπάνω συστήματος είναι η NetLogo του Center for Connected Learning and Computer-Based Modeling (NetLogo User Manual, 2012), του οποίου οι ιδιότητες της εύκολης εξοικείωσης και λεπτομερούς μοντελοποίησης επαληθεύτηκαν.

1. ΕΙΣΑΓΩΓΗ

1.1 Η αλλαγή κατεύθυνσης

Αν και σήμερα βρίσκεται απεριόριστη πληροφορία στη διάθεση των χρηστών, αυτοί όλο και περισσότερο έχουν την ανάγκη βοήθειας, καθοδήγησης και υποστήριξης στον αχανή κόσμο του Διαδικτύου (Fasli, 2007). Ως αποτέλεσμα, οι ανάγκες των χρηστών έχουν μεταβληθεί. Το παραδοσιακό μοντέλο αλληλεπίδρασης υπολογιστή-χρήστη, το οποίο βασίζεται στην έκδοση οδηγιών από τον χρήστη στον υπολογιστή δεν είναι πλέον αρκετό. Η αλληλεπίδραση δεν περιορίζεται στην απλή εντολή που δίνεται είτε από την γραμμή εντολών είτε με ένα κλικ σε παραθυρικό περιβάλλον και προκαλεί την εκτέλεση μίας ενέργειας. Αντί ο χρήστης να *δρα* σε ένα σύστημα, σήμερα πλέον λέμε ότι *αναθέτει σε αυτό* ή *αλληλεπιδρά μαζί του* με πιο σύνθετο και διαλογικό τρόπο. Έτσι, η λειτουργία των συστημάτων έχει μετατραπεί από απλά υπολογιστική σε αλληλεπιδραστική με παράλληλη ανάληψη καθηκόντων. Απαιτούνται νέα θεωρητικά μοντέλα και μεθοδολογίες προγραμματισμού λογισμικού για την υποστήριξη του σχεδιασμού και της ανάπτυξης συστημάτων που αντικατοπτρίζουν αυτή την αλλαγή κατεύθυνσης.

Η παραπάνω τάση οδήγησε στην κατασκευή συστημάτων που θα μπορούν να ενεργούν εκ μέρους του χρήστη. Λόγω της αύξησης της πολυπλοκότητας των καθηκόντων που ανατίθενται στα συστήματα, αυτά οφείλουν να είναι εξυπνότερα ώστε να μπορούν να προβλέπουν τις ανάγκες του χρήστη, να μαθαίνουν από την αλληλεπίδραση με αυτόν, να επιδιώκουν ενεργά την επίτευξη στόχων και πάντα να λειτουργούν όσο το δυνατό πιο αυτόνομα, χωρίς την συχνή και άμεση παρέμβαση του χρήστη. Τέτοια συστήματα παρατηρούν το περιβάλλον τους και αντιδρούν σε κάθε αλλαγή που συμβαίνει και επηρεάζει την επίτευξη των στόχων τους. Έχοντας ως γνώμονα την ικανοποίηση των συμφερόντων του χρήστη ένα ευφυές πληροφοριακό σύστημα μπορεί να αλληλεπιδρά ή να ζητά την βοήθεια ή εξυπηρέτηση άλλων πληροφοριακών συστημάτων.

Η μεταφορά στον πραγματικό κόσμο ενός τέτοιου πολύπλοκου συστήματος που θα μπορεί να μας εξυπηρετεί στην καθημερινή μας ζωή με δυνατότητα να ενεργεί, να μαθαίνει και να προσαρμόζεται στο περιβάλλον του καθώς και να αλληλεπιδρά με άλλα συστήματα είναι η έννοια του *πράκτορα*. Απλούστερα διατυπωμένο, ένας πράκτορας είναι ένα πληροφοριακό σύστημα που ενεργεί για λογαριασμό του χρήστη και προσπαθεί να πετύχει τους στόχους του χρήστη λειτουργώντας αυτόνομα. Τις περισσότερες φορές ο πράκτορας δεν είναι κάποια απομονωμένη οντότητα. Βρίσκεται σε ένα περιβάλλον και συνεχώς αλληλεπιδρά με αυτό και με άλλες οντότητες, είτε ανθρώπους, είτε άλλους πράκτορες. Ένα σύστημα πολλαπλών πρακτόρων αποτελείται από έναν αριθμό πρακτόρων που επικοινωνούν και αλληλεπιδρούν μεταξύ τους για να επιλύσουν ένα σύνθετο πρόβλημα ή να εκπληρώσουν τους στόχους του χρήστη. Αυτό που διαφοροποιεί του πράκτορες από τα άλλα είδη λογισμικού είναι ότι δεν περιορίζονται μόνο σε υπολογισμούς, αλλά δέχονται αναθέσεις και αλληλεπίδραση.

Οι πράκτορες επιτρέπουν την απόκρυψη της υποκείμενης πολυπλοκότητας της χρήσης των σύγχρονων υπολογιστών, ενώ ταυτόχρονα διευκολύνουν την επίτευξη των εργασιών και των στόχων μας με την ελάχιστη δυνατή προσπάθεια. Με αυτό τους το γνώρισμα καθιστούν την Πληροφορική πιο φιλική και προσβάσιμη

ακόμη και στους αρχάριους χρήστες με το να κάνουν την ίδια την πληροφορία πιο κατανοητή και διαχειρίσιμη. Αφού φιτλτράρουν, οργανώσουν και αναδιοργανώσουν την πληροφορία, την παρουσιάζουν στον χρήστη σε μία πιο εύληπτη μορφή.

Καθώς η ψηφιακή τεχνολογία εξαπλώνεται όλο και περισσότερο στις συσκευές που χρησιμοποιούμε στην καθημερινότητά μας για κάθε πιθανή δραστηριότητα, από τις πιο μονότονες στις πιο δημιουργικές, οι πράκτορες διεισδύουν σε στατικές και φορητές συσκευές για να διευκολύνουν την ολοκλήρωση υπηρεσιών. Στο όχι και τόσο μακρινό μέλλον δεν θα μιλάμε απλά για ευφυή πληροφοριακά συστήματα, αλλά και για ευφυή αυτοκίνητα, σπίτια με εγκατεστημένους πράκτορες που θα μπορούν να μας εξυπηρετήσουν αλληλεπιδρώντας μαζί μας και μεταξύ τους.

1.2 Η καινοτομία των ευφυών πρακτόρων

Το χαρακτηριστικό των Ευφυών Πρακτόρων που τους κάνει να διαφέρουν από τις παραδοσιακές εφαρμογές λογισμικού είναι το γεγονός ότι ενώ οι τελευταίες χρειάζονται ξεκάθαρες οδηγίες για το τι θα πρέπει να φέρουν σε πέρας και τα βήματα που θα ακολουθήσουν, οι ευφυείς πράκτορες αρκούνται στην περιγραφή του στόχου. Στη συνέχεια, έχοντας νοημοσύνη θα αναζητήσουν ενεργά τρόπους για να πετύχουν τον στόχο απαιτώντας ελάχιστη παρέμβαση του χρήστη. Σε αυτή την διαδικασία θα αντιδράσουν στις αλλαγές του περιβάλλοντος μέσα στο οποίο λειτουργούν και ανάλογα με το πώς αυτές οι αλλαγές επηρεάζουν την επίτευξη του στόχου θα αναθεωρήσουν το αρχικό τους πλάνο ενεργειών (Fasli, 2007).

Η συνεισφορά μίας νέας τεχνολογίας θεωρείται σημαντική εφόσον είτε επιτρέπει την επίλυση που δεν μπορούν να αντιμετωπίσουν οι υπάρχουσες τεχνολογίες, είτε προσφέρει εναλλακτικούς τρόπους επίλυσης, οι οποίο κρίνονται ταχύτεροι, αποδοτικότεροι ή απλούστεροι. Οι ευφυείς πράκτορες και η αντίστοιχη μεθοδολογία παρουσιάζει και τα δύο παραπάνω στοιχεία.

Αρχικά, οι πράκτορες είναι σε θέση να επιλύσουν προβλήματα που ξεπερνούν τις δυνατότητες άλλων τεχνολογιών, όπως για παράδειγμα τα ανοιχτά συστήματα. Τα συστατικά αυτών των συστημάτων δεν είναι γνωστά εκ των προτέρων, η δομή και η τοπολογία τους μπορεί να μεταβληθούν, ενώ τα επιμέρους συστατικά μπορεί να είναι εξαιρετικά ετερογενή στην περίπτωση που είναι οντότητες σε διαφορετικά σημεία του Διαδικτύου. Οι πράκτορες είναι οι πλέον κατάλληλοι για πολύπλοκα συστήματα μεγάλης κλίμακας, αφού η αφαίρεση και η συναρμολογησιμότητα που τους χαρακτηρίζει αποτελεί απαραίτητη προϋπόθεση για να λειτουργήσει ένα τέτοιο σύστημα. Ένας τρόπος θεώρησης αυτών των συστημάτων είναι ως ένα χαλαρά συνδεδεμένο δίκτυο οντοτήτων, όπου κάθε μία έχει τις δικές της ικανότητες, γνώσεις και πόρους. Αυτές οι οντότητες μέσα στα πλαίσια του δικτύου μπορούν να συνεργαστούν στην επίλυση ενός προβλήματος το εύρος του οποίου ξεπερνάει τις δυνατότητες της κάθε μίας ξεχωριστά. Γίνεται κατανοητό ότι οι πράκτορες είναι κατάλληλοι για δυναμικά και περιβάλλοντα αβεβαιότητας με πλήθος πληροφοριών και διεργασιών.

Επιπλέον οι ευφυείς πράκτορες μπορούν να χρησιμοποιηθούν και στην επίλυση προβλημάτων, όπου χρησιμοποιούνται άλλες μεθοδολογίες, όμως με σημαντικά ταχύτερα αποτελέσματα, φθηνότερα και με πιο φυσικό τρόπο. Για παράδειγμα, σε προβλήματα όπου τα δεδομένα, ο έλεγχος και η ειδική γνώση ή οι

πόροι είναι κατανεμημένα σε ένα δίκτυο προτιμάται η χρήση πρακτόρων. Πράκτορες με διαφορετικούς πόρους γνώσης και υπολογιστικής ισχύος κατανεμημένοι σε ένα δίκτυο μπορούν να έρθουν σε επαφή και να συντονιστούν για την επίτευξη ενός σύνθετου στόχου. Ένας τρόπος για να αντιληφθεί κάποιος αυτή τη συνύπαρξη είναι εάν τους φανταστεί ως μία κοινωνία πρακτόρων. Σε άλλες περιπτώσεις οι πράκτορες μπορούν να συνδυαστούν με προϋπάρχοντα συστατικά ή ακόμα και ολόκληρα συστήματα, τα οποία αν και παρωχημένα είναι συχνά δύσκολο να παροπλιστούν.

1.3 Εφαρμογές Ευφυών Πρακτόρων

Η τεχνολογία των ευφυών πρακτόρων αναγνωρίστηκε από νωρίς ως αλλαγή παραδείγματος ανάπτυξης σύνθετων και ανοιχτών συστημάτων για μεγάλο εύρος πεδίου εφαρμογών (Fasli, 2007). Έξυπνα υπολογιστικά συστήματα αυτού του τύπου μπορούν να μας βοηθήσουν στην οργάνωση του ηλεκτρονικού μας ταχυδρομείου, στην αναζήτηση άρθρων της ειδησεογραφίας ή επιστημονικών άρθρων που μας ενδιαφέρουν, στον έλεγχο της διαδικασίας παραγωγής σε ένα εργοστάσιο, ακόμα και στην πλοήγηση ενός διαστημοπλοίου για την εξερεύνηση μακρινών πλανητών. Ενδεικτικά τα πεδία αυτά αναφέρονται παρακάτω.

Έλεγχος Διαδικασιών. Ο έλεγχος διαδικασιών είναι μία τυπική εφαρμογή πρακτόρων ή συστημάτων πολλαπλών πρακτόρων, αφού οι ελεγκτές διαδικασιών είναι από μόνοι τους αυτόνομα συστήματα αντίδρασης. Μία από της πιο γνωστές προσεγγίσεις είναι το ARCHON, μία πλατφόρμα λογισμικού που συνοδεύεται από μία μεθοδολογία για την ανάπτυξη συστημάτων πολλαπλών πρακτόρων. Το ARCHON έχει χρησιμοποιηθεί σε συστήματα διαχείρισης μεταφοράς ηλεκτρικής ενέργειας και ελέγχου επιταχυντών σωματιδίων.

Διαχείριση Επιχειρήσεων. Ένα σύμπλεγμα εργοστασίων μπορεί να μοντελοποιηθεί από ένα σύστημα πολλαπλών πρακτόρων, οι οποίοι θα συντονίζουν τα εισερχόμενα και εξερχόμενα. Τέτοια συστήματα έχουν αναπτυχθεί για τον προγραμματισμό της παραγωγής.

Διαχείριση Πληροφοριών. Αν και το Διαδίκτυο μας παρέχει πρόσβαση σε ανεξάντλητο όγκο πληροφοριών έχει προκαλέσει σύγχυση λόγω μεγέθους. Με λίγα λόγια υπάρχει πολλή περισσότερη πληροφορία από όση μπορεί να εξετάσει ένας άνθρωπος από μόνος του. Οι πράκτορες μπορούν να φιλτράρουν τις πληροφορίες προκειμένου να απομονώσουν αυτές που ενδιαφέρουν τον χρήστη. Μία πληροφορία μπορεί να βρίσκεται κυριολεκτικά οπουδήποτε στον Παγκόσμιο Ιστό, αλλά οι πράκτορες είναι σε θέση να συλλέξουν και να συνθέσουν με γνώμονα κριτήρια που θα έχει θέσει ο χρήστης.

Ψυχαγωγία. Οι πράκτορες παίζουν σημαντικό ρόλο στην βιομηχανία της ψυχαγωγίας, όπως σε παιχνίδια σε Η/Υ ή σε κινηματογραφικές ταινίες. Μπορούν να «ενσαρκώσουν» με μεγάλη φυσικότητα χαρακτήρες παιχνιδιών με δυνατότητα να αποκτούν γνώσεις και να προσαρμόζονται στο επίπεδο εμπειρίας του χρήστη-παίκτη προσφέροντας περισσότερο ρεαλισμό στην εξέλιξη του παιχνιδιού. Τέτοια παραδείγματα παιχνιδιών είναι το Haunt 2 που χρησιμοποιεί χαρακτήρες με τεχνητή νοημοσύνη, οι οποίοι εμφανίζουν επιθυμίες και ανάγκες, καθώς και το Creatures. Στην κινηματογραφική ταινία Lord Of The Rings έγινε εκτενής χρήση αρκετά πιστευτών πρακτόρων στις σκηνές που αναπαρίστανται μάχες με πολυάριθμους στρατούς.

Εκπαίδευση. Οι ευφυείς πράκτορες έχουν χρησιμοποιηθεί και για εκπαιδευτικούς σκοπούς. Η έννοια του έξυπνου συστήματος διδασκαλίας πλέον προεκτείνεται σε αυτήν ενός βοηθού στη μάθηση. Ένας εικονικός δάσκαλος πέρα από το να διδάσκει τον μαθητή, αποκτά την ικανότητα να προσαρμόζεται στις ανάγκες του.

Αναμφίβολα οι πράκτορες θα παίξουν ένα σημαντικό ρόλο στην πραγματοποίηση του οράματος ενός ψηφιακού κόσμου που λειτουργεί σε 24ωρη βάση χωρίς γεωγραφικά σύνορα, αρκεί να κερδίσουν την εμπιστοσύνη των χρηστών.

2. ΕΥΦΥΕΙΣ ΠΡΑΚΤΟΡΕΣ

2.1 Λήψη Αποφάσεων

2.1.1 Περιβάλλοντα

Ένας Ευφυής Πράκτορας (ΕΠ) υπάρχει μεμονωμένα ή συνυπάρχει με άλλους στα πλαίσια ενός περιβάλλοντος. Ο πράκτορας αντιλαμβάνεται το περιβάλλον και τις αλλαγές που συμβαίνουν σε αυτό μέσω των πληροφοριών που του παρέχουν οι αισθητήρες του, τους οποίους χρησιμοποιεί για να επηρεάσει το περιβάλλον με ενέργειες που προκύπτουν από την διαδικασία λήψης αποφάσεων που ακολουθεί (Fasli, 2007).

Για την κατανόηση του περιβάλλοντος ως αφηρημένη έννοια αρκεί να θεωρήσουμε ότι σε κάθε χρονική στιγμή αυτό βρίσκεται σε μία ιδιαίτερη κατάσταση (State) s. Το σύνολο αυτών των καταστάσεων ορίζεται ως $S=\{s_1, s_2, \ldots\}$. Καθώς ένας πράκτορας αντιλαμβάνεται το περιβάλλον του μέσω των αισθητήρων του, η λειτουργία του αισθητήρα see αντιστοιχίζει τις καταστάσεις του περιβάλλοντος σε αντιλήψεις (Percepts):

$$see : S \rightarrow P$$

Ένας πράκτορας μπορεί να βρίσκεται σε μια συγκεκριμένη εσωτερική κατάσταση (Internal State) και το σύνολο αυτών των εσωτερικών καταστάσεων ορίζεται ως IS. Αυτή η εσωτερική κατάσταση ή μνήμη του πράκτορα ενημερώνεται μέσω μίας συνάρτησης next που αντιστοιχίζει την εσωτερική κατάσταση και την αντίληψη σε μία νέα εσωτερική κατάσταση:

$$next : IS \times P \rightarrow IS$$

Ο πράκτορας μπορεί να θεωρηθεί ότι εκτελεί ένα σύνολο ενεργειών (actions) $A=\{\alpha_1, \alpha_2, \ldots\}$ με τις οποίες επηρεάζει το περιβάλλον. Έτσι, μία συνάρτηση από την εσωτερική κατάσταση σε ενέργεια είναι:

$$action : IS \rightarrow A$$

Η επίδραση του πράκτορα «συλλαμβάνεται» από μία συνάρτηση do που αντιστοιχίζει την εκτελεσθείσα ενέργεια σε μια συγκεκριμένη κατάσταση:

$$do : A \times S \rightarrow S$$

Συνεπώς, ένας πράκτορας αποφασίζει ποια ενέργεια θα εκτελέσει ανάλογα με την ακολουθία των καταστάσεων του περιβάλλοντος, το οποίο τελικά αναπαριστά και την αποκτηθείσα εμπειρία του μέχρι μία δεδομένη στιγμή. Χρησιμοποιώντας ψευδοκώδικα όλο αυτό θα μπορούσε να διατυπωθεί ως εξής:

```
// control loop for Simple-Agent
begin
    IS: = IS0: //initial internal state
```

```
        while true do
            p: = get-percept();
            IS: = update(IS,p);
            action: = choose-best-action(IS);
            execute(action);
        end-while
    end
```

Η εσωτερική κατάσταση IS του πράκτορα ενημερώνεται σε συνδυασμό με την νέα αντίληψη P και την προηγούμενη εσωτερική κατάσταση. Η βέλτιστη ενέργεια (action) επιλέγεται με βάση αυτή τη νέα εσωτερική κατάσταση και ακολούθως εκτελείται. Ο πράκτορας έχει μέσα στην εσωτερική του κατάσταση μία σαφή απεικόνιση του κόσμου, δηλαδή έχει ένα μοντέλο αυτού του κόσμου. Υπάρχουν όμως και κάποιοι πράκτορες που δεν έχουν μία τέτοιου είδους εσωτερική αναπαράσταση του κόσμου, άρα δεν υπάρχει ανάγκη για ενημέρωση. Στην δική τους περίπτωση μία αντίληψη αντιστοιχίζεται άμεσα σε κάποια ενέργεια.

Τα χαρακτηριστικά του περιβάλλοντος επιβάλλουν περιορισμούς στην συμπεριφορά του πράκτορα. Προδιαγράφονται ιδιότητες και χαρακτηριστικά που απαιτούνται από την πλευρά του πράκτορα ώστε αυτός να λειτουργεί με επιτυχία εντός του περιβάλλοντος. Ένα περιβάλλον, λοιπόν, μπορεί να χαρακτηριστεί:

- **Πλήρως ή μερικώς παρατηρήσιμο**. Η παρατηρησιμότητα αναφέρεται στην πρόσβαση πληροφοριών που περιγράφουν την τρέχουσα κατάσταση του περιβάλλοντος. Αν και η πλήρης παρατηρησιμότητα είναι πρακτικά αδύνατη στις περισσότερες περιπτώσεις, για να ισχύει κάτι τέτοιο, αρκεί ο πράκτορας να έχει στην διάθεση του όλες εκείνες τις πτυχές του περιβάλλοντος που τον αφορούν και σχετίζονται με την συμπεριφορά του μέσα σε αυτό. Η μερική παρατηρησιμότητα οφείλεται συχνά είτε σε θόρυβο (noise) λόγω κακής λειτουργίας των αισθητήρων, είτε σε επικάλυψη (aliasing) λόγω ανικανότητας των αισθητήρων να διακρίνουν μεταξύ διαφορετικών καταστάσεων (αντιλήψεων).
- **Ντερμενιστικό ή στοχαστικό**. Σε ένα ντετερμενιστικό περιβάλλον η επόμενη κατάσταση καθορίζεται απολύτως από την τρέχουσα κατάσταση και την ενέργεια ενός πράκτορα. Σε μια πιο ρεαλιστική θεώρηση του περιβάλλοντος ο πράκτορας έχει από μόνος του περιορισμένο έλεγχο του περιβάλλοντος, αλλά και της δράσης των υπολοίπων πρακτόρων που συνυπάρχουν. Η επίδραση των ενεργειών του δεν είναι προκαταβολικά γνωστή, αφού υπάρχουν διάφοροι παράγοντες που συμβάλλουν επίσης στις αλλαγές του περιβάλλοντος μη αποκλείοντας ακόμα και το ενδεχόμενο να αποτύχει η ενέργεια του πράκτορα. Είναι λοιπόν φανερό ότι στην περίπτωση στοχαστικού περιβάλλοντος ο σχεδιασμός των πρακτόρων γίνεται πιο σύνθετος.
- **Στατικό ή δυναμικό**. Σε ένα στατικό περιβάλλον ο κόσμος αλλάζει μόνο ως συνέπεια των ενεργειών του πράκτορα. Σε ένα δυναμικό περιβάλλον ο κόσμος αλλάζει και ανεξάρτητα της δραστηριότητας του πράκτορα και μάλιστα ακόμα και κατά την διάρκεια εκτέλεσης της ίδιας της ενέργειας. Για το λόγο αυτό ο πράκτορας θα πρέπει να συλλέγει με μεγάλη συχνότητα

πληροφορίες προκειμένου να επιλέξει την βέλτιστη ενέργεια. Προφανώς η αυτή η συχνότητα καθορίζεται από το ρυθμό αλλαγών στο περιβάλλον.

- **Επεισοδιακό ή ακολουθιακό**. Σε ένα επεισοδιακό περιβάλλον ένας κύκλος αντίληψης και κατόπιν ενέργειας θεωρείται ένα ξεχωριστό επεισόδιο. Η απόδοση του πράκτορα εξαρτάται εξ' ολοκλήρου από το συγκεκριμένο επεισόδιο και δεν εξετάζονται τα προηγούμενα ή επόμενα. Με αυτό το δεδομένο ο πράκτορας δεν οφείλει να ασχοληθεί με τις συνέπειες των ενεργειών του στους επόμενους κύκλους. Ως πρόβλημα με επεισοδιακό χαρακτήρα μπορεί να αναφερθεί η επιλογή της βέλτιστης διαδρομής από έναν οδηγό ταξί. Η επιλογή δρομολογίου του τρέχοντος πελάτη δεν επηρεάζει την απόδοση του οδηγού στον επόμενο πελάτη. Σε αντίθετη περίπτωση, ένα ακολουθιακό σενάριο είναι αυτό μίας παρτίδας σκάκι, όπου κάθε κίνηση του παίκτη έχει επιπτώσεις στην εξέλιξη του παιχνιδιού.
- **Διακριτό ή συνεχές**. Σε ένα διακριτό περιβάλλον υπάρχει πεπερασμένος αριθμός ενεργειών και αντιλήψεων, ενώ σε ένα συνεχές αυτοί οι αριθμοί είναι άπειροι. Το σκάκι και η οδήγηση στα πλαίσια μίας πόλης είναι παραδείγματα συνεχούς και διακριτού περιβάλλοντος, αντίστοιχα.
- **Ενός πράκτορα ή πολλαπλών πρακτόρων**. Προφανώς αναφέρεται στην ύπαρξη και δραστηριοποίηση ενός ή περισσότερων πρακτόρων μέσα στο περιβάλλον.

Από τα παραπάνω προκύπτει ότι η πιο σύνθετη τάξη περιβάλλοντος είναι η μερικώς παρατηρήσιμη, στοχαστική, δυναμική, ακολουθιακή, συνεχής και πολλαπλών πρακτόρων. Ένα περιβάλλον με αυτά τα χαρακτηριστικά καλείται και *ανοιχτό*.

2.1.2 Μέτρο απόδοσης και λογικότητα

Η ιδέα ενός έξυπνου πράκτορα που διεκπεραιώνει στη θέση μας διάφορες δουλειές είναι πολύ ελκυστική. Το μόνο που έχουμε να κάνουμε είναι να πούμε στον πράκτορα τι ακριβώς θέλουμε και στην συνέχεια αυτός όντας αυτόνομος, προνοητικός και με δυνατότητα αντίδρασης και επικοινωνίας με τους υπόλοιπους θα βρει τον τρόπο να πετύχει τον στόχο του. Το ζητούμενο είναι να σχεδιάσουμε και να αναπτύξουμε πράκτορες που αποδίδουν καλά στο περιβάλλον τους. Πως όμως μπορούμε να εκτιμήσουμε την απόδοση του πράκτορα και πόσο επιτυχής είναι; Μία προφανής απάντηση είναι ότι ένας επιτυχημένος πράκτορας θα κάνει πάντα τη σωστή δουλειά, κάτι που όμως με την σειρά του δημιουργεί το ερώτημα του πως ορίζουμε πότε το αποτέλεσμα είναι σωστό. Ο όρος Μέτρο Απόδοσης εισάγεται για να εκφράσει πόσο επιτυχημένος είναι ένας πράκτορας. Η έννοια αυτή αποτελείται από ένα σύνολο κριτηρίων που προσδιορίζουν την επιτυχία του και ομαδοποιούνται σε δύο μεγάλες ομάδες, το *πώς* και το *πότε*.

Ένας ιδανικός λογικός πράκτορας λοιπόν είναι ο πράκτορας που εκτελεί ενέργειες που αναμένεται να αυξήσουν το μέτρο απόδοσής του. Το τι θεωρείται λογικό κάθε στιγμή εξαρτάται από:

- το μέτρο απόδοσης που καθορίζει τον βαθμό της επιτυχίας
- όλες τις μέχρι τώρα προσλαμβάνουσες του πράκτορα
- τις προσδοκίες του πράκτορα για το τι θα αντιληφθεί και τι θα συμβεί το μέλλον

- το τι γνωρίζει ο πράκτορας για το περιβάλλον
- τις ενέργειες που μπορεί να εκτελέσει

2.2 Σχεδιασμός

Ο σχεδιασμός είναι μία κρίσιμη ικανότητα ευφυούς συμπεριφοράς, αφού αυξάνει την ευελιξία του πράκτορα δίνοντας του την δυνατότητα να δημιουργεί ακολουθίες ενεργειών που θα οδηγήσουν στην επίτευξη του στόχου του. Ο σχεδιασμός αφορά το *τι* θα γίνει αλλά και το *πότε* θα γίνει αυτό. Συχνά, προκειμένου ο πράκτορας να πετύχει τον τελικό του στόχο πρέπει να θέσει κάποιους υπο-στόχους και να κινηθεί προς την επίτευξη αρχικά αυτών (Fasli, 2007).

Υπάρχουν αρκετές προσεγγίσεις στο θέμα του προβλήματος του σχεδιασμού.

- **Λογική προσέγγιση**. Οι αλγόριθμοι αυτού του τύπου έχουν ως είσοδο περιγραφές διατυπωμένες σε συμβατική γλώσσα. Οι καταστάσεις και οι στόχοι αναπαρίστανται από ένα σύνολο προτάσεων κα οι ενέργειες από λογικές περιγραφές προαπαιτούμενων και αποτελεσμάτων.
- **Προσέγγιση περιπτώσεων**. Σε αυτή την προσέγγιση χρησιμοποιούνται αποθηκευμένα προηγούμενα σχέδια που μπορούν να επαναχρησιμοποιηθούν στην επίλυση συγγενών προβλημάτων. Ο πράκτορας εξετάζοντας κατά την αρχή επίλυσης του προβλήματος τον στόχο και την αρχική κατάσταση, κάνει μία αναζήτηση στην βιβλιοθήκη σχεδίων για να τα ταυτοποιήσει με κάποιο αποθηκευμένο.
- **Προσέγγιση τελεστών**. Εδώ κάθε ενέργεια αντιστοιχίζεται σε κάποιον τελεστή που αποτελείται από τρία συστατικά, την περιγραφή της ενέργειας, τα προαπαιτούμενα ως ένα σύνολο συνθηκών που πρέπει να είναι αληθή για να εκτελεστεί η ενέργεια και τέλος το αποτέλεσμα, δηλαδή ένα σύνολο παραμέτρων που θα είναι αληθή όταν ολοκληρωθεί η ενέργεια.
- **Σχεδιασμός ως αναζήτηση**. Ο σχεδιασμός μπορεί να θεωρηθεί και ένα είδος αναζήτησης μέσα σε ένα χώρο πιθανοτήτων

2.3 Μάθηση

Μάθηση είναι η διεργασία απόκτησης γνώσεων, δεξιοτήτων ή συμπεριφοράς μέσω της εμπειρίας, της μίμησης ή της διδασκαλίας, οι οποίες στην συνέχεια προκαλούν αλλαγή της συμπεριφοράς. Ένας πράκτορας μαθαίνει όταν συμβαίνει κάποια αλλαγή στη δομή του, στον κώδικα ή στα δεδομένα με τέτοιο τρόπο που αυτή η αλλαγή βελτιώνει την μελλοντική του απόδοση.

Συχνά, κατά το στάδιο του σχεδιασμού και της ανάπτυξης δεν είναι γνωστά όλα τα χαρακτηριστικά του περιβάλλοντος. Προκειμένου να βελτιωθεί η λειτουργικότητα και η απόδοση ενός συστήματος που βρίσκεται ήδη σε λειτουργία επιστρατεύονται τεχνικές μάθησης. Σε κάποιες περιπτώσεις, ενώ η σχετική γνώση είναι διαθέσιμη εκ των προτέρων, ο όγκος της είναι τέτοιος που κάνει αδύνατη την μεταφορά της σε κώδικα. Επιπλέον η δυναμικότητα του περιβάλλοντος μέσα στο οποίο καλείται να λειτουργήσει ένας πράκτορας υπάρχει περίπτωση να καταστήσει κάποια προϋπάρχουσα γνώση παρωχημένη στο εγγύς μέλλον. Με αυτά τα δεδομένα ένας πράκτορας οφείλει συνεχώς να προσαρμόζεται στις νέες συνθήκες που συχνά διαμορφώνονται λόγω των αλλαγών που η λειτουργία του ίδιου προκαλεί στο περιβάλλον του (Fasli, 2007).

Ανάλογα με το αν ο πράκτορας μαθαίνει από μόνος του ή μαθαίνει στα πλαίσια μιας κοινωνίας πρακτόρων που τον ενημερώνουν, τότε μιλάμε για συγκεντρωτική ή αποκεντρωμένη μάθηση, αντίστοιχα. Ο τρόπος απόκτησης γνώσης από την διαθέσιμη ανάδραση μίας οντότητας-μαθητή καθορίζει και την μέθοδο μάθησης.

- **Μάθηση μέσω απόκτησης γνώσης**. Απόκτηση και αποθήκευση γνώσης χωρίς επεξεργασία.
- **Μάθηση μέσω λήψης οδηγιών ή συμβουλών**. Η γνώση μετασχηματίζεται και υιοθετείται για να αποκτήσει αποδοτική μορφή.
- **Μάθηση από παραδείγματα και την πράξη**. Οι άνθρωποι ερχόμενοι σε επαφή με θετικά και αρνητικά παραδείγματα είναι σε θέση να κατηγοριοποιούν έννοιες και αντικείμενα του πραγματικού κόσμου χωρίς προηγουμένως να κατέχουν σαφείς κανόνες σχετικά. Αυτή η μέθοδος συνήθως υλοποιείται υπό επίβλεψη.
- **Μάθηση μέσω αναλογιών**. Ο πράκτορας χρησιμοποιεί αποθηκευμένη γνώση την οποία μεταφέρει στην βελτίωση επίλυσης ενός νέου αλλά άλυτου προβλήματος.
- **Μάθηση μέσω επίλυσης προβλημάτων**. Ένας που έχει εμπειρία από επίλυση προηγούμενων προβλημάτων μπορεί συγκρίνοντας να εντοπίσει ομοιότητες ενός νέου προβλήματος και να το λύσει με επιτυχία. Δεν απαιτείται συλλογή νέας γνώσης, αλλά αναδιοργάνωση και ανάκληση της εμπειρίας.
- **Μάθηση μέσω αναζήτησης**. Είναι ένας τρόπος ανεξάρτητης συλλογής γνώσεων και δεξιοτήτων, βασιζόμενος εξ ολοκλήρου σε ίδιες παρατηρήσεις και εξαγωγές συμπερασμάτων και θεωριών. Αποτελεί τον πιο δύσκολο τρόπο μάθησης σε επίπεδο υλοποίησης.

Ανάλογα με την έξωθεν ανάδραση που μπορεί να δέχεται ένας πράκτορας κατά την διαδικασία της μάθησης διακρίνεται η περίπτωση της μάθησης με επίβλεψη, της ενισχυμένης μάθησης (χαλαρή παρέμβαση με περιορισμένη ροή ανάδρασης) και της μάθησης χωρίς επίβλεψη.

2.4 Αρχιτεκτονικές πρακτόρων

2.4.1 Αρχιτεκτονική βασισμένη στην λογική

Η αρχιτεκτονική αυτή έχει τις ρίζες της στην παραδοσιακή συμβολική τεχνητή νοημοσύνη. Ο πράκτορας κατέχει μία συμβολική απεικόνιση του περιβάλλοντος και κανόνες που ορίζουν την συμπεριφορά του. Οι συμβολικές αναπαραστάσεις αντιστοιχούν σε λογικά συμπεράσματα ή αποδείξεις θεωρημάτων.

Η ιδέα ενός πράκτορα ως «μηχανή» παραγωγής αποδείξεων θεωρημάτων είναι αρκετά ελκυστική. Ας φανταστούμε ένα θεώρημα φ που ερμηνεύει την συμπεριφορά ενός πράκτορα, την δημιουργία στόχων και τον τρόπο επίτευξής τους από τον πράκτορα. Αυτό το θεώρημα θα μπορούσε να αποτελεί τις προδιαγραφές του πράκτορα. Στην παραδοσιακή μηχανική λογισμικού αυτές οι προδιαγραφές θα απαιτούσαν ένα βαθμό τελειοποίησης, τον σχεδιασμό που τελικά θα οδηγούσε στην υλοποίησή του. Στην αρχιτεκτονική που αναφερόμαστε εδώ όλα τα ενδιάμεσα

βήματα παραλείπονται, αφού το θεώρημα φ μπορεί να εκτελεστεί απευθείας για την παραγωγή της συμπεριφοράς του πράκτορα.

2.4.2 Αρχιτεκτονική αντίδρασης

Στον πυρήνα αυτής της αρχιτεκτονικής εγκαταλείπεται η χρήση μίας εσωτερικής νοητικής αναπαράστασης και της συνεπαγόμενης σε αυτή λήψης αποφάσεων. Εδώ, η «έξυπνη» συμπεριφορά συνδέεται άμεσα με το περιβάλλον που φιλοξενεί τον πράκτορα και μπορεί να δημιουργηθεί απλά από την ανταπόκριση στο περιβάλλον και στις αλλαγές που συμβαίνουν μέσα σ' αυτό. Σε έναν πράκτορα αντίδρασης η κάθε προσλαμβάνουσα αντιστοιχίζεται απευθείας σε συγκεκριμένη ενέργεια.

Η αρχιτεκτονική στηρίζεται στην πεποίθηση ότι η πραγματική ευφυΐα βρίσκεται στον πραγματικό κόσμο και όχι κλεισμένη σε κάποιο ξέχωρο σύστημα, συνεπώς η ευφυής συμπεριφορά μπορεί να προκύψει μόνο σαν συνέπεια αλληλεπίδρασης ενός πράκτορα με το περιβάλλον του. Μία σύντομη διατύπωση αυτής της θέσης είναι η εξής: Η ευφυΐα είναι μία επερχόμενη ιδιότητα. Η ευφυής συμπεριφορά μπορεί να δημιουργηθεί χωρίς κάποια σαφή εσωτερική νοητική αναπαράσταση και χωρίς κάποια ρητό συλλογισμό, αλλά από την αλληλεπίδραση απλών συμπεριφορών.

2.4.3 Αρχιτεκτονική Πεποίθησης-Επιθυμίας-Πρόθεσης

Η αρχιτεκτονική Πεποίθησης-Επιθυμίας-Πρόθεσης (BDI Belief-Desire-Intention) έχει προέλευση την φιλοσοφική θεωρία της πρακτικής λογικής. Η πρακτική λογική διαφέρει από την θεωρητική λογική γιατί η πρώτη σχετίζεται με την διεργασία της απόφασης του τρόπου να γίνει κάτι, ενώ η δεύτερη με την διεργασία παραγωγής γνώσης ή εξαγωγής συμπεράσματος με την χρήση των πεποιθήσεων και της γνώσης.

Η πρακτική λογική έχει δύο πτυχές. Την απόφαση του τι θέλουμε να επιτύχουμε, γνωστή και ως *περίσκεψη* και τον τρόπο που πρόκειται να το πετύχουμε, γνωστή ως *λογική μέσου-σκοπού* ή σχεδιασμό. Στην καθημερινή μας ζωή χρησιμοποιούμε συνεχώς την πρακτική λογική για να πάρουμε αποφάσεις σχετικά με το τι θα κάνουμε παρακάτω.

Οι προθέσεις είναι κομβικής σημασίας στην πρακτική λογική. Περιγράφουν καταστάσεις τις οποίες ο πράκτορες δεσμεύεται να πραγματοποιήσει και κατά συνέπεια προκαλούν ενέργειες. Η έννοια της πρόθεσης χαρακτηρίζει και το μυαλό αλλά και τις ενέργειες ενός πράκτορα περιλαμβάνοντας μία ιδιαίτερη μορφή δέσμευσης στο πλαίσιο της αντίδρασης επαναπροσδιορισμού, φανερώνουν βούληση και λογική. Η μία πρόθεση μπορεί να οδηγήσει στην δημιουργία νέων προθέσεων που πηγάζουν από αυτή. Συνεπώς, οι προθέσεις δημιουργούν ισχυρά κίνητρα για ενέργεια, κίνητρα τα οποία τοποθετούνται συνεχώς πάνω από τα κίνητρα των πεποιθήσεων και των επιθυμιών.

Η δημιουργία προθέσεων είναι κρίσιμη για την επιτυχία ενός πράκτορα. Αυτό οφείλεται στην εξάρτηση του πράκτορα από πόρους και ότι δεν μπορεί συνεχώς να σταθμίζει τις επιλογές του, αλλά χρειάζεται να κατασταλάξει σε μία κατάσταση και να αποφασίσει τι θα κάνει στη συνέχεια. Ο πράκτορας έχει την ανάγκη να συντονίσει τις μελλοντικές του ενέργειες και να δεσμευτεί σε μία σειρά ενεργειών.

Τα βασικά μέρη της αρχιτεκτονικής βασίζονται λοιπόν στη θεωρία της πρακτικής λογικής. Οι πεποιθήσεις αντιπροσωπεύουν τις πληροφορίες που έχει ο πράκτορας για το περιβάλλον, οι οποίες δεν είναι απαραίτητα σωστές, δίνοντας του έτσι μία ανακριβή εικόνα του κόσμου. Οι επιθυμίες αντιπροσωπεύουν τα κίνητρα του πράκτορα ή τις πιθανές επιλογές, ενώ οι προθέσεις αντιπροσωπεύουν τις δεσμεύσεις του πράκτορα, δηλαδή τις επιλογές που ο πράκτορας οφείλει να πραγματοποιήσει. Τα επτά συστατικά της αρχιτεκτονικής είναι τα εξής:

- Μηχανισμός αναθεώρησης πεποιθήσεων
- Σύνολο τρεχόντων πεποιθήσεων
- Μηχανισμός δημιουργίας επιλογών
- Σύνολο επιθυμιών
- Μηχανισμός φιλτραρίσματος
- Σύνολο τρεχόντων προθέσεων
- Μηχανισμός επιλογής ενεργειών

Η κατάσταση του πράκτορα σε οποιαδήποτε στιγμή περιγράφεται από την τριάδα

(B, D, I)

όπου B οι πεποιθήσεις (Beliefs)
D οι επιθυμίες (Desires)
I οι προθέσεις (Intentions)

Ο *μηχανισμός αναθεώρησης πεποιθήσεων* του πράκτορα αντιστοιχίζει τις τρέχουσες πεποιθήσεις και τις προσλαμβάνουσες P σε ένα νέο σύνολο πεποιθήσεων:

brf: P(Beliefs) x P -> P(Beliefs)

Ο *μηχανισμός δημιουργίας επιλογών* που αντιπροσωπεύει την διεργασία λογικής μέσου-σκοπού, αντιστοιχίζει ένα σύνολο πεποιθήσεων και ένα σύνολο προθέσεων σε ένα σύνολο επιθυμιών:

options: P(Beliefs) x P(Intentions) -> P(Desires)

Ο *μηχανισμός φιλτραρίσματος* που αντιπροσωπεύει την διεργασία περίσκεψης του πράκτορα, αντιστοιχίζει ένα σύνολο πεποιθήσεων, επιθυμιών και προθέσεων σε ένα νέο σύνολο προθέσεων:

filter: P(Beliefs) x P(Desires) x P(Intentions) -> P(Intentions)

Ο *μηχανισμός επιλογής ενεργειών* επιστρέφει προθέσεις που αντιστοιχούν απευθείας σε ενέργειες:

asf: P(Intentions) -> A

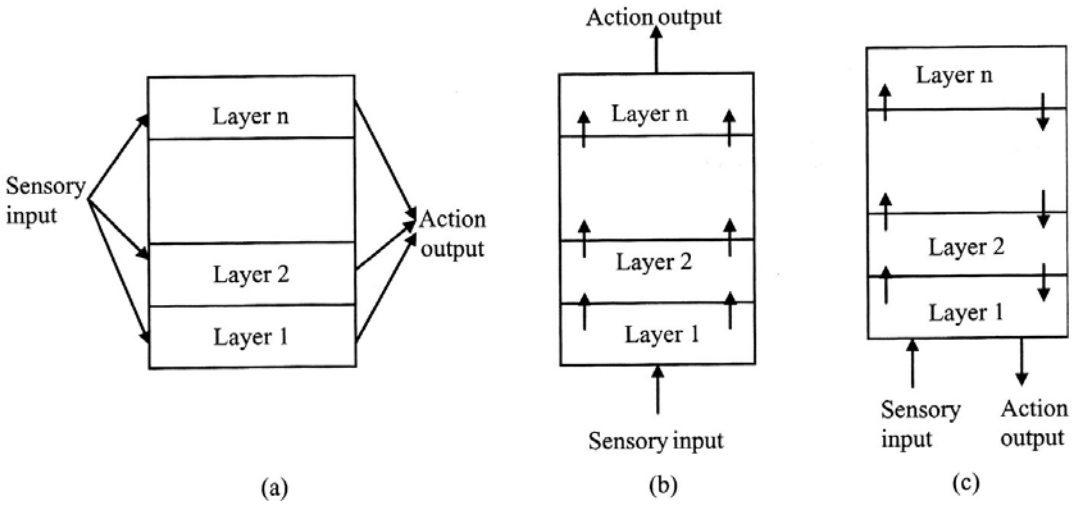
2.4.4 Υβριδική αρχιτεκτονική

Ιδανικά θα επιθυμούσαμε οι πράκτορες να εμφανίζουν τόσο συμπεριφορά αντίδρασης, όσο και πρόληψης προκειμένου να επιτύχουν τους στόχους τους. Μία προφανής λύση είναι να συνδυαστούν τα προτερήματα των δύο προσεγγίσεων και να δημιουργηθούν υβριδικοί πράκτορες. Ένα συμβουλευτικό εξάρτημα θα είναι υπεύθυνο να διατηρεί ένα μοντέλο του κόσμου, να λαμβάνει αποφάσεις χρησιμοποιώντας συμβολική λογική και να καταστρώνει σχέδια για να επιτύχει τους στόχους του πράκτορα, ενώ ταυτόχρονα ένα εξάρτημα αντίδρασης θα επιτρέπει στον πράκτορα να αντιδρά άμεσα στα εξωτερικά ερεθίσματα. Ένας τρόπος επίτευξης αυτού είναι μέσω της δομής αλληλεπιδρώντων στρωμάτων, τουλάχιστον δύο, ένα για κάθε εξάρτημα. Σε αυτή την περίπτωση η λήψη αποφάσεων γίνεται

μέσω της διαίρεσης σε πολλά στρώματα λογισμικού, όπου το κάθε στρώμα θα χρησιμοποιεί λογική διαφορετικής αφαίρεσης.

- **Οριζόντια Διαστρωμάτωση**. Στην οριζόντια υλοποίηση κάθε στρώμα συνδέεται απευθείας με είσοδο αισθητήρων και έξοδο ενέργειας, όπως φαίνεται στο Σχήμα 2.8(a). Για n διαφορετικές συμπεριφορές θα πρέπει να υλοποιηθούν n στρώματα, τα οποία ανταγωνίζονται μεταξύ τους προκειμένου να αποκτήσουν τον έλεγχο του πράκτορα. Λόγω της πιθανότητας πρόκλησης ασυνάρτητης συμπεριφοράς απαιτείται η παρουσία μίας συνάρτησης διαμεσολάβησης μεταξύ των στρωμάτων.
- **Κατακόρυφη Διαστρωμάτωση**. Μέσα στην κατακόρυφη υλοποίηση η ροή πληροφορίας και ο έλεγχος υπόκεινται σε περιορισμούς. Υπάρχουν δύο τρόποι κατακόρυφης διαστρωμάτωσης, ο κατακόρυφος μονής διέλευσης και ο κατακόρυφος διπλής διέλευσης. Στη περίπτωση της μονής διέλευσης, η ροή πληροφορίας και ο έλεγχος μέσα στο σύστημα συμβαίνει μονόδρομα. Οι είσοδοι των αισθητήρων λαμβάνονται από το χαμηλότερο επίπεδο και η ροή και ο έλεγχος μεταβιβάζονται διαδοχικά στα ανώτερα επίπεδα. Το ανώτερο στρώμα είναι υπεύθυνο για την ενέργεια, όπως φαίνεται στο Σχήμα 2.8(b). Στην δεύτερη περίπτωση η ροή και έλεγχος είναι αμφίδρομα. Ενώ οι είσοδοι των αισθητήρων λαμβάνονται στο χαμηλότερο επίπεδο, η ενέργεια που προκύπτει στο ανώτερο επίπεδο μεταβιβάζεται προς τα κάτω, όπως φαίνεται στο Σχήμα 2.8 (c).

Το πλεονέκτημα της κατακόρυφης διαστρωμάτωσης είναι η μειωμένη πολυπλοκότητα, όμως υστερεί σε ευελιξία στο ενδεχόμενο βλάβης ενός στρώματος που αποτελεί κρίκο της αλυσίδας.



Σχήμα 2.8: Ροή ελέγχου και πληροφοριών σε αρχιτεκτονικές οριζόντιας και κατακόρυφης διαστρωμάτωσης: (a) οριζόντια διαστρωμάτωση, (b) κατακόρυφη διαστρωμάτωση μονής διέλευσης (c) κατακόρυφη διαστρωμάτωση διπλής διέλευσης

(Mueller et al. 1995 p.265 – Springer Science and Business Media)

2.5 Γωνία θέασης των πρακτόρων

Οι πράκτορες και τα συστήματα πολλαπλών πρακτόρων θεωρούνται ως το νέο είδος λογισμικού που θα αποτελέσει οδηγό στην ανάπτυξη περισσότερων σύνθετων και ανοιχτών συστημάτων ικανών για δυναμική αλληλεπίδραση. Την ίδια στιγμή, αρκετοί δυσκολεύονται να αναγνωρίσουν την καινοτομία σε σχέση με τις άλλες προσεγγίσεις. Για το λόγο αυτό στις επόμενες παραγράφους ακολουθούν ανάλογες συγκρίσεις μεθοδολογιών (Wooldridge, 2009).

2.5.1 Πράκτορες και αντικείμενα

Στην περίπτωση των πρακτόρων ένα σύστημα αναπαρίσταται ως μία συλλογή από χαλαρά συνδεδεμένους πράκτορες που αλληλεπιδρούν. Παρόμοια με τα αντικείμενα, οι πράκτορες έχουν μία ταυτότητα, μία κατάσταση και συμπεριφορά, τα οποία όμως είναι πιο σύνθετα από αυτά των αντικειμένων. Η κατάσταση ενός πράκτορα μπορεί να περιγραφεί με τους όρους γνώση, πεποίθηση, επιθυμία, πρόθεση και υποχρέωση. Η συμπεριφορά του συνδέεται με τους όρους σχέδιο για την επίτευξη στόχων, ενέργεια, αντίδραση σε γεγονότα, ακόμα και σε ρόλους. Συνεπώς, η συμπεριφορά στην περίπτωση ενός πράκτορα δεν χαρακτηρίζεται από απευθείας αντιστοίχιση εισόδου σε συγκεκριμένη εξόδου, δηλαδή του τι θα πρέπει να κάνει, αλλά στο πλαίσιο του πως θα αποφασίσει τον τρόπο για να κάνει τι.

Οι πράκτορες έχουν και άλλες διαφορές σε σύγκριση με τα αντικείμενα. Πρωτίστως, οι πράκτορες εμφανίζουν αυτονομία, δηλαδή έχουν έλεγχο της κατάστασής τους, της εκτέλεσης και τελικά της συμπεριφοράς τους. Ενώ τα αντικείμενα μπορεί να θεωρηθεί ότι έχουν κάποιο έλεγχο της κατάστασής τους (π.χ. μεταβλητές που είναι δηλωμένες private), δεν έχουν κανέναν έλεγχο στην εκτέλεση και την συμπεριφορά τους. Αν το αντικείμενο i καλέσει μία μέθοδο ενός αντικειμένου j η οποία είναι δηλωμένη ως public, τότε το j δεν έχει άλλη επιλογή από το να την εκτελέσει. Στην αντίστοιχη περίπτωση μεταξύ δύο πρακτόρων, ο δεύτερος πράκτορας δέχεται αίτημα εκτέλεσης στο οποίο έχει την επιλογή να αρνηθεί.

Η δεύτερη σημαντική διαφορά μεταξύ αντικειμένων και πρακτόρων είναι ότι οι δεύτεροι εμφανίζουν συμπεριφορά με χαρακτηριστικά την επίτευξη στόχων, την δυνατότητα αντίδρασης και την κοινωνικότητα. Οι πράκτορες επικοινωνούν με σύνθετους διαλόγους και όχι με απλή ανταλλαγή μηνυμάτων. Είναι επίμονοι, έχουν αυτογνωσία και μπορούν να μάθουν και να προσαρμοστούν καλύτερα στο περιβάλλον τους. Αυτά τα χαρακτηριστικά ευφυούς συμπεριφοράς δεν συναντώνται στα αντικείμενα.

Πέρα όμως από τα παραπάνω, ο πιο συνήθης τρόπος κατασκευής πρακτόρων είναι η χρήση αντικειμενοστραφών γλωσσών. Θα πρέπει εδώ να γίνει σαφές το ότι ενώ μπορούμε να κατασκευάσουμε πράκτορες από αντικείμενα, τα αντικείμενα από μόνα τους ανήκουν στην αντικειμενοστραφή μεθοδολογία και δεν είναι πράκτορες. Τα ενεργά αντικείμενα είναι πιο κοντά στην έννοια του πράκτορα, αφού έχουν κάποιο βαθμό αυτονομίας, εξακολουθούν όμως να απουσιάζουν οι υπόλοιπες ιδιότητες που αναφέρθηκαν παραπάνω.

2.5.2 Πράκτορες και έμπειρα συστήματα

Ένα έμπειρο σύστημα μπορεί να χαρακτηριστεί ευφυές, όπως ένας πράκτορας, όμως τα έμπειρα συστήματα δεν έχουν έλεγχο των ενεργειών τους. Παρέχουν πληροφορίες συγκεκριμένου πεδίου γνώσης και συνήθως δεν έχουν προσαρμοστικότητα. Η αλληλεπιδραστική τους ικανότητα είναι αρκετά περιορισμένη. Πέραν της εισόδου/εξόδου απευθείας από πληκτρολόγιο/οθόνη, συνήθως δεν αλληλεπιδρούν με το περιβάλλον ή άλλα έμπειρα συστήματα, άρα δεν θεωρούνται ότι εμφανίζουν κοινωνική συμπεριφορά. Επίσης, δεν ικανοποιούν ούτε τα υπόλοιπα γνωρίσματα αντιπροσώπευσης των πρακτόρων, όπως η αυτονομία και η εκδήλωση αντιδράσεων ή πρόληψης.

2.5.3 Πράκτορες, Υπηρεσίες Ιστού και ο Σημασιολογικός Ιστός

Η αίτηση από τον χρήστη για μία υπηρεσία στον ιστό προϋποθέτει αρχικά τον εντοπισμό της υπηρεσίες, την ταυτοποίηση με την αιτούμενη υπηρεσία και στη συνέχεια την εκτέλεση. Ιδανικά, για να επιτύχουν τα μέγιστα οφέλη των Υπηρεσιών Ιστού και του Σημασιολογικού Ιστού, οι χρήστες θα πρέπει να είναι σε θέση να αναγνωρίσουν τις ανάγκες τους, ώστε να τις αναθέσουν σε κάποιον πράκτορα που θα εντοπίσει την κατάλληλη υπηρεσία, η οποία ικανοποιεί καλύτερα του στόχους του χρήστη. Έτσι, οι πράκτορες διευκολύνουν την ομαλή ολοκλήρωση των υπηρεσιών ιστού, ώστε η πολυπλοκότητα εκτέλεσης της αίτησης να είναι διάφανη στον χρήστη. Οι διαφορές που εντοπίζονται μεταξύ πρακτόρων και υπηρεσιών ιστού είναι οι παρακάτω.

- Μία υπηρεσία είναι στατική και δεν προσαρμόζεται. Γνωρίζει μόνο τον εαυτό της και την δική της λειτουργικότητα. Οι πράκτορες από την πλευρά τους μπορούν να μάθουν για τον χρήστη, άλλους πράκτορες και υπηρεσίες και πιθανόν να μπορούν να προσφέρουν κάποιου είδους προσωποποιημένες υπηρεσίες ανάλογα με τις ανάγκες του χρήστη, λαμβάνοντας υπόψη και νέα δεδομένα που τυχόν προκύψουν.
- Οι υπηρεσίες είναι παθητικές μέχρι να τις καλέσει κάποιος. Αντίθετα, οι πράκτορες είναι αυτόνομοι και προληπτικοί κάτι που τους επιτρέπει να αναζητούν ενεργητικά βοήθεια για τον χρήστη, για παράδειγμα ελέγχοντας περιοδικά για νέες πληροφορίες σε ένα πεδίο θεματολογίας που είναι στα ενδιαφέροντα του χρήστη ή ενημερώνοντας τον την στιγμή που κάποιο προϊόν είναι διαθέσιμο σε κάποιο κατάστημα. Μία υπηρεσία επίσης θα εκτελεστεί από την στιγμή που θα υπάρξει σχετική αίτηση, ενώ ο πράκτορας θα αποφασίσει ο ίδιος εάν για τους λόγους του θα προχωρήσει ή όχι στην ικανοποίηση της αίτησης.
- Η επικοινωνία των υπηρεσιών μεταξύ τους είναι σε χαμηλό επίπεδο συντακτικού. Ανταλλάσουν μεταξύ τους δεδομένα χωρίς να είναι σε θέση να καταλάβουν την σημασία τους, γιατί οι υπηρεσίες δεν είναι σχεδιασμένες να λειτουργούν σε συνδυασμό με οντολογίες. Στην περίπτωση που ο αιτών χρησιμοποιεί διαφορετική οντολογία από τον πάροχο υπηρεσίας θα προκύψει αδυναμία συνεννόησης. Οι πράκτορες χρησιμοποιούν οντολογίες και το Σημασιολογικό Ιστό για να αποφύγουν παρόμοιες συγχύσεις.
- Παρά το γεγονός ότι μία υπηρεσία μπορεί να αλληλεπιδράσει με άλλες υπηρεσίες, η διαχείριση της συνδυασμένης λειτουργικότητας συντονίζεται

από κάποιον τρίτο. Οι υπηρεσίες δεν εμφανίζουν την κοινωνική συμπεριφορά στο επίπεδο που το πετυχαίνουν οι πράκτορες.

Συμπερασματικά, οι πράκτορες προσφέρουν μεγαλύτερη ευελιξία στο πως οι Υπηρεσίες Ιστού χρησιμοποιούνται και δημιουργούνται. Συγκεκριμένα, οι πράκτορες είναι πρωταρχικής σημασίας στην υλοποίηση του οράματος του Σημασιολογικού Ιστού, αφού επιτρέπουν την ομαλή αλληλεπίδραση και ολοκλήρωση των Υπηρεσιών Σημασιολογικού Ιστού.

2.6 Μεθοδολογίες και Γλώσσες

Οι σύγχρονες μεθοδολογίες για την ανάλυση και τον σχεδιασμό συστημάτων που βασίζονται σε πράκτορες και συστήματα πολλαπλών πρακτόρων χωρίζονται σε τρεις βασικές κατηγορίες (Wooldridge, 2009) σε κάθε μία από τις οποίες αναφέρονται σχετικές γλώσσες και πλαίσια εργασίας:

- Αντικειμενοστραφείς μεθοδολογίες.
  - Gaia
  - Prometheus
  - Tropos
  - ROADMAP
  - MaSE
- Μεθοδολογίες που απευθείας επεκτείνουν ή υιοθετούν αντικειμενοστραφείς μεθοδολογίες.
  - AAII
  - AUML
- Μεθοδολογίες που υιοθετούν μοντέλα μηχανικής της γνώσης ή άλλες τεχνικές.
  - MAS-Common KADS
  - DESIRE

Οι πράκτορες έχουν στηριχθεί σε αρκετές προϋπάρχουσες τεχνολογίες της επιστήμης της πληροφορικής και πιο συγκεκριμένα της τεχνητής νοημοσύνης και προσφέρουν ένα επίπεδο αφαίρεσης που δεν το συναντούσαμε στις μέχρι τώρα μεθοδολογίες. Εξακολουθούν όμως να υπάρχουν πολλά συστήματα λογισμικού που δεν βασίζονται σε πράκτορες, γνωστά και ως *παραδοσιακά συστήματα*. Το ερώτημα είναι πως σε έναν συνεχώς αναπτυσσόμενο κόσμο με αυξανόμενη συνδεσιμότητα θα μπορέσουν τα παραδοσιακά συστήματα να συνυπάρξουν και να συνεργαστούν με τους πράκτορες. Η διαδικασία μετατροπής ενός συστήματος σε πράκτορα/ες ή η προσπάθεια συμβατότητάς του ώστε να αλληλεπιδράσει με συστήματα πρακτόρων καλείται *πρακτοροποίηση*. Ένα παραδοσιακό σύστημα μπορεί να αποκτήσει συμβατότητα με πράκτορες με τρεις τρόπους.

- **Μέσω ενός μετασχηματιστή**. Ο μετασχηματιστής είναι ένα λογισμικό που μεσολαβεί μεταξύ παραδοσιακού συστήματος και περιβάλλοντος και μεταφράζει τα μηνύματα του πράκτορα σε είσοδο κατάλληλη για το παραδοσιακό σύστημα και αντίστροφα.
- **Μέσω περιβλήματος**. Το περίβλημα είναι ένα λογισμικό που εσωκλείει ένα παραδοσιακό σύστημα. Δεν περιορίζεται σε απλή αμφίδρομη μετάφραση μηνυμάτων, αλλά μπορεί και να τροποποιεί υπάρχουσες δομές δεδομένων,

ακόμα και να εισάγει νέο κώδικα στο παραδοσιακό σύστημα. Αν και αυτή η εναλλακτική είναι πιο αποδοτική από αυτή του μετασχηματιστή, απαραίτητη συνθήκη για την υλοποίησή της είναι να είναι προσπελάσιμος ο κώδικας του παραδοσιακού συστήματος.

- **Με επαναπρογραμματισμό**. Η πιο ακραία μέθοδος πρακτοροποίησης είναι να επανασχεδιαστεί και να προγραμματιστεί το παραδοσιακό σύστημα σε μορφή πράκτορα. Η αποδοτικότητα αυτής της εναλλακτικής είναι ανάλογη του κόστους.

Καμία από τις παραπάνω μεθόδους δεν είναι απλή. Η ετερογένεια προκαλεί αρκετά προβλήματα, αφού οι διάφοροι προγραμματιστές μπορεί να επιλέξουν ποικίλες γλώσσες και τρόπους διεπαφής. Υπάρχει πάντοτε το ενδεχόμενο να συναντήσει κάποιος δυσνόητο κώδικα, όταν αυτός έχει δημιουργηθεί από κάποιον άλλο. Ένα δεύτερο θέμα είναι το υψηλό αρχικό κόστος επένδυσης ενός οργανισμού για την ανάπτυξη λογισμικού ώστε να θεωρηθεί πολυτέλεια το να αφιερώσει περισσότερα χρήματα στον εκσυγχρονισμό των συστημάτων του μόνο και μόνο για να παρακολουθεί τις τεχνολογικές εξελίξεις στην πληροφορική. Τέλος, θα πρέπει να αναφερθεί και η διστακτικότητα των ανθρώπων στην υιοθέτηση καινοτόμων τεχνολογιών που δεν έχουν ακόμα ευρεία αποδοχή.

3. ΣΥΣΤΗΜΑΤΑ ΠΟΛΛΑΠΛΩΝ ΠΡΑΚΤΟΡΩΝ

Ένα Σύστημα Πολλαπλών Πρακτόρων (ΣΠΠ) μπορεί να θεωρηθεί ως ένα χαλαρά συνδεδεμένο δίκτυο αυτόνομων πρακτόρων που μπορούν να συνεργαστούν για την επίλυση προβλημάτων, τα οποία ξεπερνούν το πεδίο, τους πόρους και τις ικανότητες του κάθε ένα πράκτορα μεμονωμένα. Στο πλαίσιο αυτό ένας πράκτορας μπορεί να αυξήσει την αποδοτικότητά του (Fasli, 2007). Παραδείγματα ΣΠΠ είναι ομάδες ρομπότ που παίζουν ποδόσφαιρο (RoboCup), ηλεκτρονικές αγορές (e-commerce) και ένα πλήθος κατανεμημένων διαδικασιών ελέγχου σε ένα έξυπνο κτίριο.

3.1 Γνωρίσματα Συστημάτων Πολλαπλών Πρακτόρων

Τα ΣΠΠ διαφέρουν από τα συστήματα ενός πράκτορα στα εξής:

- **Περιβάλλον**. Η λήψη αποφάσεων σε ένα ΣΠΠ είναι μία συλλογική διαδικασία και οι ενέργειες των υπόλοιπων πρακτόρων μπορούν να παρέμβουν στο πλάνο δράσης του ενός.
- **Γνώση/ειδίκευση/ικανότητες/σχεδιασμός**. Όλα αυτά είναι κατανεμημένα ή διαμοιρασμένα μεταξύ των διάφορων πρακτόρων του ΣΠΠ, οι οποίοι συχνά δεν είναι ομογενείς.
- **Έλεγχος.** Ο έλεγχος είναι επίσης κατανεμημένος. Τα δεδομένα είναι αποκεντρωμένα και οι υπολογισμοί συμβαίνουν ασύγχρονα. Ο κάθε πράκτορας λαμβάνει αποφάσεις που τον αφορούν πάντα όμως μέσα στο πλαίσιο του συνολικού συντονισμού του ΣΠΠ.
- **Αλληλεπίδραση.** Η αλληλεπίδραση των πρακτόρων ακολουθεί κανόνες για την αποφυγή συγκρούσεων.

Μία νέα έννοια που εισάγεται στα ΣΠΠ είναι αυτή της συνάφειας, δηλαδή ο βαθμός στον οποίο το σύστημα λειτουργεί σαν μονάδα. Η συνάφεια εκτιμάται συνήθως εξωτερικά από κάποιον παρατηρητή που καλεί να κρίνει κατά πόσο το ΣΠΠ φαίνεται συμπεριφέρεται με συνάφεια στην διαδικασία επίτευξης ενός στόχου. Η επιτυχία σημαίνει καλή συνάφεια.

Αν και ο ορισμός ενός ΣΠΠ αναφέρεται ρητά σε συστήματα που αποτελούνται από υπολογιστικούς πράκτορες, συχνά εμφανίζεται η ανάγκη αλληλεπίδρασης με τον άνθρωπο. Ο ορισμός λοιπόν, θα πρέπει να καλύψει και τις περιπτώσεις ύπαρξης του ανθρώπινου παράγοντα.

3.1.1 Πλεονεκτήματα

Η χρήση της τεχνολογίας ΣΠΠ προσφέρει πλήθος πλεονεκτημάτων.

1. Επεκτασιμότητα και ευελιξία, αφού το σύστημα αποτελείται από πολλούς πράκτορες με διαφορετική εξειδίκευση και ικανότητες. Η εισαγωγή νέων δυνατοτήτων στο σύστημα μπορεί να γίνει σχετικά εύκολα προσθέτοντας νέους πράκτορες με ανάλογες ικανότητες. Επίσης, οι πράκτορες μπορούν να προσαρμοστούν σε αλλαγές στο περιβάλλον.
2. Σθεναρότητα και αξιοπιστία, καθώς όταν ένας πράκτορας πάθει βλάβη, δεν προκαλεί συνολική παύση λειτουργίας, αλλά εμφανίζεται μία υποβάθμιση της συνολικής απόδοσης.

3. Υπολογιστική αποδοτικότητα και ταχύτητα λόγω υπολογισμών που εκτελούνται ασύγχρονα και παράλληλα.
4. Προγραμματισμός και συντηρησιμότητα, καθώς είναι γενικά ευκολότερος ο προγραμματισμός και η συντήρηση ενός αρθρωτού συστήματος σε σύγκριση με ένα μονολιθικό. Επίσης, είναι ευκολότερος ο προγραμματισμός διαφορετικών πρακτόρων του ΣΠΠ από διαφορετικούς προγραμματιστές.
5. Επαναχρησιμοποίηση, καθώς οι πράκτορες που αναπτύσσονται για ένα ΣΠΠ μπορούν να λύσουν κάποιο παρόμοιο πρόβλημα σε άλλο σύστημα.
6. Μειωμένο κόστος, αφού οι μεμονωμένοι πράκτορες έχουν φθηνότερο κόστος σε σχέση με το συνολικό σύστημα.

3.1.2 Κλειστά Συστήματα Πολλαπλών Πρακτόρων

Τα συστήματα αυτού του τύπου βασίζονται σε στατικό σχεδιασμό με προκαθορισμένα συστατικά και λειτουργίες. Αυτό προϋποθέτει γνώση των ιδιοτήτων του συστήματος εκ των προτέρων και ύπαρξη μίας κοινής γλώσσας επικοινωνίας των πρακτόρων. Ο κάθε πράκτορας δημιουργείται για συγκεκριμένο μέρους του συνολικού στόχου, έχοντας ανάλογες δυνατότητες επίλυσης προβλημάτων και γνώσεις και είναι δυνατό να εργαστούν παράλληλα πολλοί προγραμματιστές. Ένα παράδειγμα τέτοιου συστήματος είναι ένα ΣΠΠ που εξυπηρετεί τις ανάγκες ενός οργανισμού, όπου κάθε πράκτορες αντιπροσωπεύει ένα διαφορετικό τμήμα.

3.1.3 Ανοιχτά Συστήματα Πολλαπλών Πρακτόρων

Σε ένα ανοιχτό ΣΠΠ, το σύστημα συχνά δεν έχει προηγούμενο στατικό σχεδιασμό παρά μόνο τους πράκτορες που το συνθέτουν. Οι πράκτορες δεν είναι απαραίτητα ενήμεροι για τους υπόλοιπους πράκτορες ή την εξειδίκευση και τις υπηρεσίες που αυτοί μπορούν να παρέχουν. Για τον λόγο αυτό υπάρχει ανάγκη ύπαρξης ενός μηχανισμού εντοπισμού άλλων πρακτόρων μέσα στο σύστημα. Ένα παράδειγμα ανοιχτού ΣΠΠ είναι μία ηλεκτρονική αγορά. Οι πράκτορες που αντιπροσωπεύουν πελάτες δεν είναι απαραίτητα ενήμεροι για τους διαθέσιμους προμηθευτές και τις υπηρεσίες και τα προϊόντα τους. Ο εντοπισμός των τελευταίων από τους πρώτους γίνεται από ειδικά σχεδιασμένους ενδιάμεσους πράκτορες που λειτουργούν ως κατάλογοι (Χρυσός Οδηγός ή μεσίτες).

3.2 Αλληλεπίδραση

Ένα από τα σημαντικότερα θέματα στα ΣΠΠ είναι αυτό της αλληλεπίδρασης. Η αλληλεπίδραση συμβαίνει όταν δύο ή περισσότεροι πράκτορες δημιουργούν μία δυναμική σχέση μέσω ενός συνόλου αμοιβαίων ενεργειών.

3.2.1 Στοιχεία αλληλεπίδρασης

1. **Στόχοι**. Οι στόχοι δύο ή περισσοτέρων πρακτόρων μπορεί να είναι συμβατοί ή ασύμβατοι. Στην πρώτη περίπτωση οι πράκτορες λειτουργούν συνεργαζόμενοι, ενώ στην δεύτερη ο ένας εμποδίζει τον άλλο και βρίσκονται σε σύγκρουση.
2. **Πόροι**. Η διαθεσιμότητα πόρων σε ένα περιβάλλον είναι κρίσιμη για το είδος της αλληλεπίδρασης που θα αναπτυχθεί. Εάν οι πόροι είναι περιορισμένοι, οι πράκτορες αναγκάζονται να ανταγωνίζονται ο ένας τον

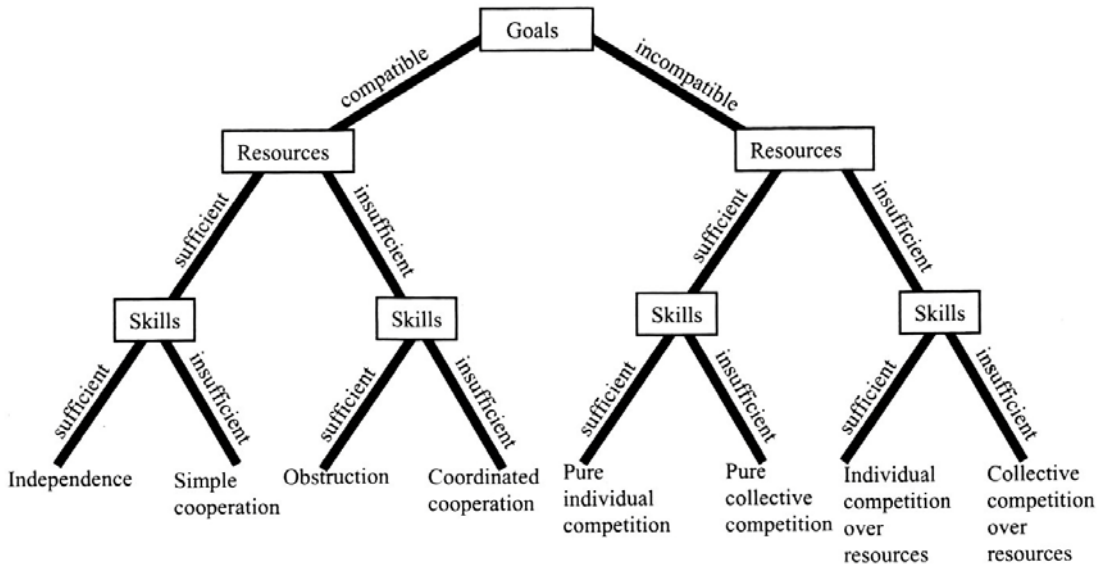
άλλο προκειμένου να αποκτήσουν πρόσβαση σε αυτούς. Σε τέτοιες περιπτώσεις η σύγκρουση είναι αναπόφευκτη και οι πράκτορες θεωρούν τους υπόλοιπους ως αντίπαλους που παρεμβαίνουν στην επίτευξη των στόχων τους.

3. **Εξειδίκευση/Ικανότητες/Δυνατότητες**. Η σχέση μεταξύ εξειδίκευσης/ικανοτήτων/δυνατοτήτων και στόχου προς επίτευξη είναι το τρίτο θεμελιώδες στοιχείο που χαρακτηρίζει το είδος αλληλεπίδρασης των πρακτόρων. Υπάρχουν στόχοι που για να επιτευχθούν απαιτούν την συμβολή της εξειδίκευσης/ικανοτήτων/δυνατοτήτων ενός πλήθους πρακτόρων, όπως στο κτίσιμο ενός σπιτιού απαιτείται η συμβολή πολλών ειδικών (αρχιτέκτονα, πολιτικού μηχανικού, μηχανολόγου μηχανικού κλπ) και τεχνητών (χτίστη, υδραυλικού, ελαιοχρωματιστή κλπ).

3.2.2 Τρόποι αλληλεπίδρασης

Οι πράκτορες μπορούν να αλληλεπιδρούν με διάφορους τρόπους. Ο βαθμός αλληλεπίδρασης είναι το πόσο συντονίζονται οι πράκτορες μεταξύ τους, αποφεύγουν συγκρούσεις, αποτρέπουν αδιέξοδα και διατηρούν ασφαλείς συνθήκες. Χαρακτηρίζονται ως ιδίου-ενδιαφέροντος/ανταγωνιστικοί ή συνεργατικοί/μη-ανταγωνιστικοί. Οι πράκτορες ιδίου-ενδιαφέροντος έχουν ασύμβατους στόχους και κύρια μέριμνά τους είναι η μεγιστοποίηση της δική τους λειτουργικότητας, χωρίς να ενδιαφέρονται ιδιαίτερα για την κοινωνία πρακτόρων σαν σύνολο. Στον αντίποδα βρίσκονται οι συνεργατικοί πράκτορες που διακρίνονται για τους συμβατούς μεταξύ τους στόχους, οι οποίοι ενεργούν με τέτοιον τρόπο ώστε να μεγιστοποιήσουν την λειτουργικότητας στα πλαίσια της κοινωνίας της οποίας είναι μέλη. Συνεπώς, συνεργασία σημαίνει συντονισμός μεταξύ συνεργατικών πρακτόρων και ανταγωνισμός σημαίνει συντονισμός μεταξύ πρακτόρων ιδίου-ενδιαφέροντος.

Η γραφική απεικόνιση όλων των δυνατών τρόπων αλληλεπίδρασης ως αποτέλεσμα των τριών θεμελιωδών στοιχείων που τις χαρακτηρίζουν φαίνονται στο Σχήμα 3.1



Σχήμα 3.1: Στοιχεία και τρόποι αλληλεπίδρασης

3.2.3 Πρωτόκολλα αλληλεπίδρασης

Η αλληλεπίδραση είναι το αποτέλεσμα ενεργειών επικοινωνίας ή φυσικών ενεργειών που μεταβάλλουν το περιβάλλον και εκτελούνται από τουλάχιστον δύο πράκτορες. Το πιο σύνηθες είναι το ενδιαφέρον να εστιάζεται όχι τόσο σε τυχαίες αλληλεπιδράσεις, αλλά σε αυτές που επιτρέπουν τους πράκτορες να συντονίζονται μεταξύ τους προς την κατεύθυνση ολοκλήρωσης των καθηκόντων τους, είτε μέσω συνεργασίας, είτε μέσω ανταγωνισμού. Τα πρωτόκολλα αλληλεπίδρασης θέτουν τους κανόνες που διευθετούν την αλληλεπίδραση που συμβαίνει μεταξύ δύο πρακτόρων και καθορίζουν τα μηνύματα ή τις ενέργειες που είναι πιθανά να εμφανιστούν σε κάθε κατάσταση. Το σύνολο αυτών των πιθανών μηνυμάτων και ενεργειών είναι σε κάθε περίπτωση πεπερασμένο. Όταν λοιπόν, ένας πράκτορας συμφωνεί να χρησιμοποιήσει ένα πρωτόκολλο, δεσμεύεται να τηρήσει τους κανόνες και τις συμβάσεις του.

Τα πρωτόκολλα αλληλεπίδρασης είναι τριών ειδών, πρωτόκολλα επικοινωνίας, πρωτόκολλα συνεργασίας ή πρωτόκολλα διαπραγμάτευσης. Τα πρωτόκολλα επικοινωνίας παρέχουν κανόνες για την δομημένη μεταβίβαση μηνυμάτων για την συγκρότηση διαλόγων με νόημα. Το δεύτερο και τρίτο είδος εξυπηρετούν σε περιπτώσεις συνεργατικής και ανταγωνιστικής λειτουργίας πρακτόρων, αντίστοιχα.

3.3 Επικοινωνία Πρακτόρων

Η επικοινωνία είναι κομβικής σημασίας σε συστήματα πολλαπλών πρακτόρων, όπου κάθε πράκτορας μπορεί να έχει ρόλο ενεργητικό, παθητικό ή ακόμα και μίγμα των δύο (Fasli, 2007). Η επικοινωνία έχει τρεις πτυχές:

1. Συντακτικό : πως δομούνται τα σύμβολα της επικοινωνίας
2. Σημασιολογία : τι δηλώνουν τα σύμβολα
3. Πραγματολογία : πως μεταφράζονται τα σύμβολα

3.3.1 Θεωρία των Λεκτικών Ενεργειών (Speech Act)

Η θεωρία αυτή υποστηρίζει ότι η ανταλλαγή επικοινωνίας είναι μία μορφή ενέργειας, συνεπώς αυτό που εκφράζει ένας ομιλητής δεν είναι μία πρόταση που δηλώνει αν κάτι ισχύει ή όχι, αλλά αποτελεί μία *λεκτική ενέργεια*. Οι ομιλητές εκτελούν διάφορες λεκτικές ενέργειες όπως διαβεβαιώσεις, αιτήματα, προτάσεις, υποσχέσεις κλπ, οι οποίες τροποποιούν την νοητική κατάσταση του δέκτη ή ακόμα προκαλούν μία αλλαγή στα τεκταινόμενα. Οι λεκτικές ενέργειες ανάλογα με την εκφραστική τους δύναμη χωρίζονται στα παρακάτω:

- *Διαβεβαιωτικές*: δηλώσεις ή γεγονότα. Αυτές μεταφέρουν πληροφορίες από τον ομιλητή στον ακροατή (πχ «Η Μαρία είναι καθηγήτρια»).
- *Δεσμευτικές*: δεσμεύσεις. Αυτές που δεσμεύουν τον ομιλητή να προκαλέσει μία κατάσταση ή να εκτελέσει κάποια ενέργεια στο μέλλον (πχ «Θα βρίσκομαι στο γραφείο στις 5 μμ»)
- *Καθοδηγητικές*: εντολές σε μία δομή master-slave. Αυτές χρησιμοποιούνται για την έκδοση οδηγιών ή εντολών στον ακροατή (πχ «Κλείσε την πόρτα»)

- *Δηλωτικές*: Χρησιμοποιώντας αυτές τις λεκτικές ενέργειες, ο ομιλητής προκαλεί μία κατάσταση στα τεκταινόμενα απλά με μία φράση (πχ «Το μάθημα έφτασε στο τέλος»)
- *Εκφραστικές*: εκφράσεις ή συναισθήματα. Αυτές εκφράζουν την νοητική κατάσταση του ομιλητή (πχ «Σήμερα είμαι χαρούμενος»)

3.3.2 Γλώσσες επικοινωνίας πρακτόρων

Μία γλώσσα επικοινωνίας πρακτόρων ορίζεται σε τρία επίπεδα. Το κατώτερο καθορίζει την μέθοδο της αλληλοσύνδεσης, το μεσαίο τη μορφή ή το συντακτικό και το ανώτερο την ερμηνεία ή τη σημασιολογία.

Μία άλλη ανάλυση των γλωσσών αυτών διακρίνει τρία συστατικά. Την εξωτερική, την εσωτερική γλώσσα και το λεξιλόγιο. Η «εξωτερική» γλώσσα χρησιμεύει για να εκφράσει βασικές έννοιες, δηλαδή τα είδη έκφρασης που χρησιμοποιεί ο πράκτορας στην επικοινωνία. Αποτελεί μία μορφή «φακέλου» που δηλώνει την συγκεκριμένη εκφραστική δύναμη του μηνύματος που περικλείει. Η «εσωτερική» γλώσσα (συντακτικό) είναι η γλώσσα (λογικής ή αναπαράστασης) στην οποία είναι γραμμένο το μήνυμα. Το λεξιλόγιο περιέχει την ερμηνεία των όρων που χρησιμοποιούνται στα μηνύματα.

3.3.3 Knowledge and Query Manipulation Language (KQML)

Η KQML είναι μία «εξωτερική» γλώσσα που βασίζεται στην θεωρία των Λεκτικών Ενεργειών. Αποτελείται από τρία επίπεδα: το επίπεδο *περιεχομένου*, το επίπεδο *μηνύματος* και το επίπεδο *επικοινωνίας*. Το επίπεδο επικοινωνίας κωδικοποιεί ένα σύνολο γνωρισμάτων του μηνύματος (δηλ. ταυτότητα αποστολέα και παραλήπτη). Το επίπεδο μηνύματος χρησιμοποιείται στην κωδικοποίηση του μηνύματος (δηλ. εκφραστικό είδος, οντολογία κλπ). Τέλος, το επίπεδο περιεχομένου περιέχει το ίδιο το μήνυμα και μπορεί να είναι σε οποιαδήποτε γλώσσα αναπαράστασης.

Η δομή ενός μηνύματος σε KQML αποτελείται από το είδος έκφρασης (δηλ. την λεκτική ενέργεια) και μία λίστα χαρακτηριστικών/τιμών:

(KQML-performative
:sender <word>
:receiver <word>
:in-reply-to <word>
:language <word>
:ontology <word>
:content <expression>
…)

Ένα παράδειγμα μηνύματος KQML τύπου **tell** φαίνεται παρακάτω:

(KQML-tell
:sender Agent435
:receiver Agent450
:in-reply-to id-msg-005
:language KIF

:ontology cycbase
:content (salary 9876543 32765))

Η σημασιολογία των ειδών έκφρασης της KQML δίνεται στο πλαίσιο συνθηκών προϋπόθεσης, μετα-υπόθεσης και ολοκλήρωσης. Αυτά τα τρία περιγράφονται σε κάθε γλώσσα που έχει τελεστές περιγραφής της νοητικής κατάστασης των πρακτόρων, όπως πεποιθήσεις (BEL), γνώση (KNOW), επιθυμίες (WANT), προθέσεων (INT) καθώς και περιγραφές ενεργειών. Κάθε γλώσσα έχει περιορισμούς στους συνδυασμούς νοητικών συμπεριφορών που συνθέτουν αυτές τις καταστάσεις.

Ένα παράδειγμα διατύπωσης των τριών συνθηκών σε ένα είδος έκφρασης **tell** φαίνεται παρακάτω:

Tell(A, B, X)
Pre(A): BEL(A,X) ^KNOW(A, WANT(B, KNOW(B, S)))
Pre(B): INT(B, KNOW(B, S))
Post(A): KNOW(A, KNOW(B, BEL(A, X))
Post(B): KNOW(B, BEL(A, X))
Completion: KNOW(B, BEL(A, X))

Είναι φανερό από την σημασιολογία του tell ότι ένας πράκτορας δεν μπορεί να προσφέρει πληροφορίες σε άλλο πράκτορα χωρίς πρώτα να του ζητηθεί. Επίσης, προκειμένου να συνεννοηθούν οι δύο πράκτορες, θα πρέπει να έχουν κοινή «εσωτερική» γλώσσα και ίδια οντολογία.

3.3.4 FIPA ACL

Στα πλαίσια της προσπάθειας τυποποίησης πρωτοκόλλων αλληλεπίδρασης πρακτόρων η FIPA πρότεινε την FIPA Agent Communication Language (FIPA ACL). Όπως και η KQML βασίζεται σε λεκτικές ενέργειες και τα μηνύματα θεωρούνται επικοινωνιακές ενέργειες (ΕΕ). Το συντακτικό στις δύο γλώσσες είναι επίσης παρόμοιο και περιέχει ένα πυρήνα βασικών ΕΕ, ενώ πρόσθετες ενέργειες μπορεί να προκύψουν από σύνθεση των βασικών. Τελικά, όλα τα είδη έκφρασης μπορούν να οριστούν με χρήση των δύο βασικών **tell** και **request**. Ένα παράδειγμα της **inform** ακολουθεί παρακάτω:

(inform
:sender Agent435
:receiver Agent450
:language Prolog
:ontology companyx
:content "salary (9876543,32765)"
)

Η σημασιολογία κάθε είδους έκφρασης ορίζεται στα πλαίσια των δύο τύπων *Γλώσσας Σημασιολογίας* που είναι οι *Προϋποθέσεις Επιτευξιμότητας* και τα *Λογικά Αποτελέσματα*. Για κάθε επικοινωνιακή ενέργεια a, οι προϋποθέσεις επιτευξιμότητας περιγράφουν τις απαραίτητες συνθήκες που θα πρέπει να ισχύουν

ώστε ο αποστολέας να στείλει την a. Το λογικό αποτέλεσμα μίας επικοινωνιακής ενέργειας αναπαριστά το αποτέλεσμα που ο πράκτορας αναμένει από την εκτέλεση της ενέργειας, ενώ μπορεί επίσης να ορίζει συνθήκες που μπορεί να ισχύουν για τον παραλήπτη. Με άλλα λόγια το λογικό αποτέλεσμα περιγράφει τον σκοπό του μηνύματος, άρα και τον στόχο του αποστολέα.

3.3.6 Knowledge Interchange Format Language (KIF)

Η Knowledge Interchange Format Language είναι μία γλώσσα περιεχομένου βασισμένη σε λογική πρώτου βαθμού με επεκτάσεις προκειμένου να υποστηρίξει μη-μονότονη συλλογιστική και ορισμούς. Είναι μία ιδιαίτερα εκφραστική γλώσσα που χρησιμοποιείται στην αναπαράσταση γνώσης σε δηλωτική μορφή. Για παράδειγμα, το παρακάτω κατηγόρημα περιγράφει ότι ο μισθός του υπαλλήλου με αριθμό 013-87-8734 που εργάζεται στο οικονομικό τμήμα είναι 35000€:

(salary 013-87-8734 finance 35000)

Στην KIF είναι δυνατό να κωδικοποιηθούν περισσότερο σύνθετες σχέσεις. Η γλώσσα περιέχει επίσης θεμελιώδεις λογικούς τελεστές για την έκφραση άρνησης (**not**), διάζευξης (**or**), σύνδεσης (**and**), ποσοτικοποίησης (**exists**, **forall**) κλπ. Η παρακάτω διατύπωση εκφράζει την ύπαρξη ενός πουλιού πάνω σε κλαδί:

(exists((?x bird) (?y tree)) (on ?x ?y))

Η διατύπωση μίας σχέσης μεταξύ δύο αντικειμένων, αλλά και κανόνες περιγράφονται με ευκολία, όπως το ακόλουθο παράδειγμα που περιγράφει την έννοια «μητέρα»:

(defrelation mother(?x?y):= (and (female ?x) (parent x? y?)))

3.3.7 Διάλογοι

Οι πράκτορες χρησιμοποιούν την επικοινωνία για να αλληλεπιδράσουν σε ένα μεγάλο εύρος καταστάσεων. Μία ακολουθία ανταλλαχθέντων μηνυμάτων συνθέτουν ένα διάλογο ή μία συζήτηση. Υπάρχουν ποικίλοι λόγοι για την έναρξη ενός διαλόγου. Οι διάλογοι διακρίνονται σε έξι βασικές κατηγορίες, όπως φαίνεται στον Πίνακα 3.1

Κατηγορία	Αρχική κατάσταση	Στόχος συμμετέχοντα	Κύριος στόχος
Πειθούς	σύγκρουση άποψης	να πείσει άλλον πράκτορα	να ξεκαθαρίσει/ξεδιαλύνει κάποιο θέμα
Διερεύνησης	ανάγκη πληροφοριών	να βρει στοιχεία ή αποδείξεις	απόκτηση γνώσης
Διαπραγμάτευσης	σύγκρουση συμφερόντων	να πετύχει την καλύτερη δυνατή συμφωνία	να καταλήξει σε συμφωνία
Αναζήτησης πληροφοριών	προσωπική άγνοια	να αποκτήσει ή να προσφέρει προσωπική γνώση	διανομή πληροφοριών
Περίσκεψης	ανάγκη να επιλέξει ενέργεια	να επηρεάσει το αποτέλεσμα	να καταλήξει σε απόφαση
Εριστικός	σύγκρουση	να προκαλέσει άλλον πράκτορα	να πετύχει συμβιβασμό
Ανάμικτος	διάφορες	διάφοροι	διάφοροι

Πίνακας 3.1: Κατηγοριοποίηση διαλόγων κατά Walton και Krabbe (1995)

Τα πρωτόκολλα που ελέγχουν την αλληλεπίδραση των πρακτόρων όταν αυτοί συμμετέχουν σε ένα διάλογο ονομάζονται συχνά και *παιχνίδια διαλόγου*. Συνεπώς, σε ένα παιχνίδι διαλόγου κάθε πράκτορας «κάνει μία κίνηση» λέγοντας κάτι με βάση ένα σύνολο κανόνων που ισχύει για κάθε κατηγορία διαλόγου όπως αυτές φαίνονται στον Πίνακα 3.1.

3.4 Οντολογίες

Το κοινό λεξιλόγιο και η ορολογία σε ένα πεδίο ή περιβάλλον συνθέτουν μία οντολογία. Η οντολογία είναι η προδιαγραφή των εννοιών, των αντικειμένων και των χαρακτηριστικών τους, των σχέσεων σε ένα συγκεκριμένο πεδίο γνώσης (Fasli, 2007). Το αντικείμενο της οντολογίας είναι η μελέτη της ταξινόμησης των πραγμάτων, των χαρακτηριστικών και των μεταξύ τους σχέσεων που υπάρχουν σε ένα πεδίο(domain) D. Η προκύπτουσα οντολογία αφορά μία συγκεκριμένη οπτική του D και συνδέεται με την χρήση κάποιας γλώσσας(language) L.

Κατά την ανάπτυξη μίας οντολογίας γίνεται αναγνώριση των θεμελιωδών εννοιών του πεδίου καθώς και των ιδιοτήτων τους, ακολουθεί η οργάνωσή τους με βάση τα χαρακτηριστικά και της αναγνώρισης των μεταξύ τους σχέσεων. Η ιεραρχία κλάσεων χτίζεται λοιπόν παράλληλα με την αναγνώριση. Η αρχή γίνεται με τις πιο «αδρές» έννοιες και αναδρομικά το μοντέλο εμπλουτίζεται με περισσότερο λεπτομέρεια. Το πρόβλημα καθορίζει τον βαθμό λεπτομέρειας, ο οποίος θα πρέπει να είναι «τόσο όσο».

Στη συνέχεια η οντολογία θα πρέπει να ελεγχθεί για την αποφυγή συντακτικών, λογικών ή σημασιολογικών αντιφάσεων. Για την δουλειά αυτή είναι πολύτιμη η συνδρομή ειδικών από το συγκεκριμένο πεδίο, οι οποίοι θα είναι σε θέση να επικυρώσουν την αρτιότητα των εννοιών και των σχέσεων. Σε πολλές περιπτώσεις οι οντολογίες περνούν από διαδοχικές φάσεις και εκδόσεις μέχρι να καταλήξουν στην τελική δημοσιεύσιμη τους μορφή. Ακόμα και μετά από αυτό το στάδιο και ενώ έχει ξεκινήσει η χρήση, είναι αναμενόμενο να προκύψει ανάγκη συντήρησης ή ενημέρωσης (εκσυγχρονισμού) της, αλλά και σύνδεσή με κάποια άλλη οντολογία.

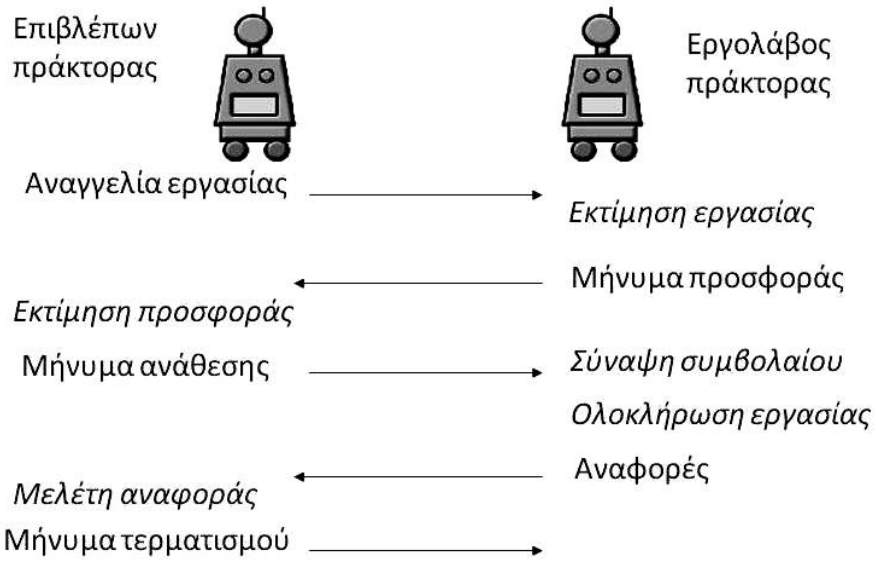
3.5 Συνεργατική Επίλυση Προβλημάτων

Η συνεργατική επίλυση προβλημάτων απαιτεί πρωτίστως αποδόμηση, δηλαδή την διαίρεση μίας σύνθετης εργασίας σε μικρότερα και πιο εύκολα στη διαχείριση βήματα. Ο τρόπος εργασίας για να γίνει η αποδόμηση αυτή εξαρτάται από το πεδίο του συγκεκριμένου προβλήματος. Μία προσέγγιση του θέματος είναι η αποδόμηση να γίνει στη φάση υλοποίησης του ΣΠΠ, ώστε να διεκπεραιωθεί από τον σχεδιαστή του συστήματος. Κατά το χρόνο εκτέλεσης οι πράκτορες αποδομούν πολύπλοκες εργασίες χρησιμοποιώντας ιεραρχικό σχεδιασμό. Μία άλλη προσέγγιση είναι να διαιρεθούν και να κατανεμηθούν στους πράκτορες ανάλογα με τις γνώσεις και τις ικανότητές τους μέσα στο σύστημα. Εναλλακτικά, η διανομή μπορεί να γίνει με χωρικά κριτήρια, ανάλογα με την κατανομή των δεδομένων, των πόρων και των κόμβων αποφάσεων. Για παράδειγμα, εργασίες που σχετίζονται με μία αποθήκη εγγράφων μπορούν να ανατεθούν με τον πράκτορα που είναι κοντύτερα σε αυτό τον πόρο.

Το ContractNet είναι ένα πρωτόκολλο αλληλεπίδρασης μεταξύ πρακτόρων για συνεργατική επίλυση προβλημάτων. Το πρωτόκολλο προσομοιάζει το μηχανισμό που λειτουργεί στον επιχειρηματικό κόσμο για την εργολαβία και υπεργολαβία δραστηριοτήτων. Το ContractNet παρέχει λύση στο πρόβλημα *σύνδεσης*, δηλαδή στην εξεύρεση του καταλληλότερου πράκτορα για μία δεδομένη εργασία.

Στα πλαίσια του ContractNet δύο είναι οι ρόλοι που μπορεί να παίξει ένας πράκτορας, του επικεφαλής και του εργολάβου. Ο επικεφαλής πράκτορας έχει μία εργασία που επιθυμεί να εκτελεστεί, ενώ ο εργολάβος έχει τις ικανότητες να την εκτελέσει. Οι ρόλοι δεν είναι περιοριστικοί, με την έννοια ότι για διαφορετικά συμβόλαια (εργασίες) ένας πράκτορας μπορεί να εναλλάσσει ρόλους, ενώ είναι δυνατόν ο εργολάβος να αναθέσεις σε έναν τρίτο πράκτορα (υπεργολάβο) την εργασία και να είναι ταυτόχρονα επικεφαλής ως προς τον τελευταίο. Στο ContractNet εξυπακούεται ότι ο επικεφαλής πράκτορας θα πρέπει να είναι σε θέση να αποδομήσει αρχικά τις εργασίες και τα προβλήματα και αργότερα να συνθέσει λύσεις από υπο-λύσεις καθώς αυτά επιστρέφονται από τους εργολάβους. Συνεπώς, οι πράκτορες οφείλουν να έχουν γνώσεις.

Η ακολουθία των μηνυμάτων που ανταλλάσσονται στα πλαίσια του ContractNet φαίνεται στο Σχήμα 3.3:

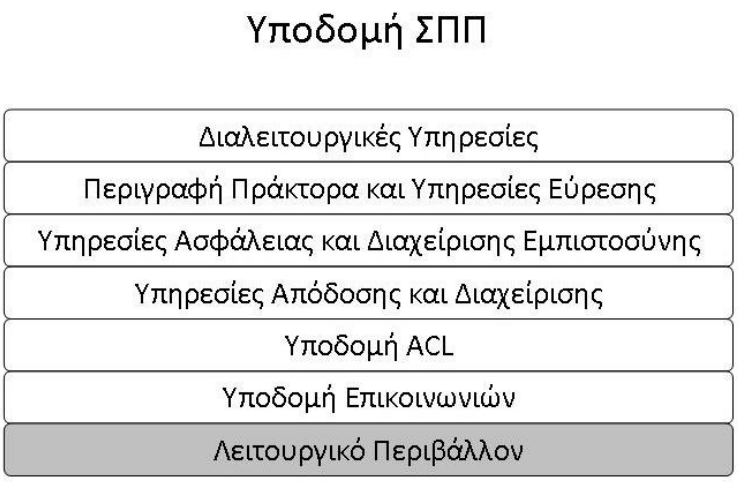


Σχήμα 3.4: Ανταλλαγή μηνυμάτων μεταξύ επιβλέποντα και εργολάβου πράκτορα στο πρωτόκολλο ContractNet

Η αναγγελία εργασιών (task announcement) μπορεί να γίνει προς πολλούς δυνητικούς εργολάβους (ευρεία εκπομπή). Εάν ο επικεφαλής πράκτορας έχει κάποια γνώση σχετικά με το προφίλ των εργολάβων μπορεί να εκδώσει μία περιορισμένη εκπομπή. Στην περίπτωση που γνωρίζει ποιος πράκτορας είναι ο καταλληλότερος, μπορεί να κάνει ακόμα και αναγγελία από-σημείο-προς-σημείο. Τέλος, υπάρχει και η δυνατότητα της απευθείας ανάθεσης, χωρίς ενδιάμεση διαπραγμάτευση, εάν ο επικεφαλής είναι απόλυτα σίγουρος για κάποιον εργολάβο πράκτορα, το οποίο αποδεικνύεται ιδιαίτερα αποδοτικό για ορισμένες εργασίες.

3.6 Απαιτήσεις Υποδομών για Ανοιχτά Συστήματα Πολλαπλών Πρακτόρων

Παρά το γεγονός ότι τα ανοιχτά ΣΠΠ δεν βασίζονται σε κάποιο προηγούμενο στατικό σχεδιασμό, δεν είναι ορθό να θεωρήσουμε αυτά τα συστήματα ως μία απλή συλλογή πρακτόρων που συνυπάρχουν σε ένα κοινό περιβάλλον. Οι πράκτορες, στην διαδικασία βελτίωσης των δυνατοτήτων τους, έρχονται κοντά ώστε να εκμεταλλευτούν τις υπηρεσίες των υπολοίπων, να μοιραστούν πόρους, εξειδίκευση και πληροφορίες, να εκτελέσουν σύνθετες εργασίες ή να επιλύσουν ένα δύσκολο πρόβλημα. Για να διευκολυνθεί η δημιουργία, λειτουργία και αλληλεπίδραση των πολλών πρακτόρων απαιτείται μία υποδομή. Μία υποδομή ΣΠΠ αποτελείται από ένα σύνολο υπηρεσιών, συμβάσεων και γνώσης που φαίνονται στο Σχήμα 3.4.



Σχήμα 3.4: Απαιτήσεις υποδομής για ένα ανοιχτό ΣΠΠ

Η υποδομή ενός ΣΠΠ αντιμετωπίζει τον κάθε πράκτορα σαν ένα μαύρο κουτί, δηλαδή τίποτε δεν είναι γνωστό για τον εσωτερικό τρόπο λειτουργίας του, τις δυνατότητες επίλυσης προβλημάτων και τα εξαρτήματα του. Ο σχεδιαστής και ο προγραμματιστής έχουν την ευθύνη να εξασφαλίσουν ότι κάθε πράκτορας έχει τα κατάλληλα εξαρτήματα που θα του επιτρέψουν να λειτουργήσει και να αλληλεπιδράσει μέσα στο ΣΠΠ και να εκμεταλλευτεί τις διαθέσιμες υπηρεσίες της υποδομής. Για παράδειγμα, ακόμα και αν το ΣΠΠ προσφέρει περιγραφή και υπηρεσίες αναζήτησης, ο σχεδιαστής και ο προγραμματιστής οφείλουν να υλοποιήσουν τις απαραίτητες μεθόδους που θα επιτρέπουν στον πράκτορα να δηλώσει την παρουσία του και να δηλώσει τις δυνατότητές και υπηρεσίες του. Αποτυχία εκτέλεσης των παραπάνω οδηγεί σε έναν απομονωμένο και συνεπώς όχι ιδιαίτερα χρήσιμο πράκτορα.

4. ΠΕΡΙΓΡΑΦΗ ΠΡΟΒΛΗΜΑΤΟΣ

Το πρόβλημα το οποίο καλείται να λυθεί στα πλαίσια αυτής της εργασίας αφορά την εκτίμηση της αποδοτικότητας διαφορετικών σεναρίων αποκομιδής απορριμμάτων μίας αστικής συνοικίας. Η σύνδεση του προβλήματος με την τεχνολογία των ευφυών πρακτόρων προκύπτει από την φύση του, αφού ανήκει στο πεδίο της διαχείρισης επιχειρήσεων. Το αντικείμενο του προβλήματος καθώς και η χρονική διάσταση της εξέλιξης των γεγονότων επιτρέπει την υλοποίηση του αντίστοιχου περιβάλλοντος πρακτόρων. Πρόκειται για ένα περιβάλλον πλήρως παρατηρήσιμο, στατικό με διακριτές καταστάσεις και συνύπαρξη πολλαπλών πρακτόρων. Με δεδομένα όλα τα παραπάνω πλέον κατάλληλο κρίνεται ένα Κλειστό Σύστημα Πολλαπλών Πρακτόρων όπου ο τρόπος αλληλεπίδρασης μεταξύ κάποιων πρακτόρων είναι ανταγωνιστικός (κάδοι - απορριμματοφόρα), ενώ μεταξύ άλλων συνεργατικός (ιδιωτικά - δημόσια απορριμματοφόρα). Οι βασικές έννοιες, οντότητες και μετρικές που περιγράφουν το πρόβλημα αναλύονται παρακάτω.

4.1 Τοπολογία συνοικίας – Εξέλιξη σεναρίου

Η συνοικία είναι ένας χώρος δύο διαστάσεων που αποτελείται από οικοδομικά τετράγωνα που χωρίζονται από δρόμους κυκλοφορίας οχημάτων. Στις γωνίες των τετραγώνων βρίσκονται οι θέσεις που μπορούν να τοποθετηθούν οι κάδοι, ενώ στην βορειοανατολική γωνία της συνοικίας βρίσκεται η χωματερή που δέχεται τα απορρίμματα. Ο χρόνος κυλάει με ρυθμό λεπτού.

4.2 Απορρίμματα - Κάδοι

Τα απορρίμματα εκφράζονται ποσοτικά με μονάδες απορρίματος (waste units) και αποθέτονται από τους κατοίκους σε δημοτικούς κάδους (bins). Η χωρητικότητα του κάθε κάδου είναι συγκεκριμένη και είναι ίδια για όλους τους κάδους της συνοικίας. Στην διάρκεια της ημέρας και με βήμα ώρας, οι κάτοικοι αποθέτουν τα απορρίμματά τους στους κάδους του οικοδομικού τους τετραγώνου. Το βήμα αύξησης του περιεχομένου των κάδων της συνοικίας είναι τυχαίο και όχι ίδιο για όλους τους κάδους, αν και περιορίζεται εντός κάποιου εύρους δεδομένου ότι υπάρχει ομοιομορφία στην πυκνότητα των κατοίκων και των συνήθειών τους. Όταν το περιεχόμενο ενός κάδου ξεπεραστεί, παύει να δέχεται νέα απορρίμματα έως ότου αδειάσει.

4.3 Απορριμματοφόρα

Υπεύθυνα για την αποκομιδή των απορριμμάτων είναι τα απορριμματοφόρα που εξυπηρετούν την συνοικία. Τα απορριμματοφόρα είναι δύο τύπων, ιδιωτικά και δημοτικά και η κίνησή τους στους δρόμους γίνεται με βήμα λεπτού σε όλη την διάρκεια της μέρας. Όλα τα απορριμματοφόρα έχουν μία τιμή **μέγιστης χωρητικότητας**, η οποία μπορεί να είναι διαφορετική ανάλογα με τον τύπο. Με τον τρόπο αυτό είναι δυνατό να θεωρηθεί ότι η χωρητικότητα των ιδιωτικών είναι π.χ. μεγαλύτερη από αυτή των δημοσίων, εφόσον στην περίπτωση αυτή πρόκειται για μεγαλύτερα σε όγκο οχήματα. Όταν ένα απορριμματοφόρο φτάσει την μέγιστη χωρητικότητά του, μεταφέρεται στην χωματερή και αδειάζει (μηδενίζει) το φορτίο του. Η δεύτερη παράμετρος που αφορά τα απορριμματοφόρα είναι το **πλήθος** των οχημάτων κάθε τύπου, δίνοντας έτσι την δυνατότητα τροποποίησης της σχέσης

μεγέθους των δύο στόλων που εξυπηρετούν την συνοικία. Τέλος, προκειμένου να δοθεί μία εκτίμηση οικονομικού κόστους της λειτουργίας ενός σεναρίου του μοντέλου, η κυκλοφορία κάθε απορριμματοφόρου συνδέεται με ένα **κόστος λειτουργίας** που στην περίπτωση των ιδιωτικών είναι ανάλογο των μονάδων απορριμμάτων που έχουν συλλέξει, ενώ στην περίπτωση των δημοσίων είναι ανάλογο του χρόνου που βρίσκονται σε λειτουργία. Το μοντέλο βασίζεται στην θεώρηση ότι τα ιδιωτικά απορριμματοφόρα εργάζονται εργολαβικά και αμείβονται με τον όγκο απορριμμάτων που καταφέρνουν να συγκεντρώσουν, ενώ στα δημόσια το κόστος είναι το άθροισμα του ωρομίσθιου των δημοτικών υπαλλήλων και του κόστους των καυσίμων ανά ώρα.

Η εκτίμηση του αποτελέσματος των διάφορων σεναρίων αποκομιδής όπως αυτά καθορίζονται από την τροποποίηση των παραμέτρων του μοντέλου γίνεται με την βοήθεια μετρικών. Οι τιμές των μετρικών υπολογίζονται κάθε χρονική στιγμή και παρουσιάζονται δυναμικά σε διαγράμματα και σε πλαίσιο κειμένου.

4.4 Διαγράμματα

Τα διαγράμματα απεικονίζουν γραφικά την πορεία του κόστους αποκομιδής, τον όγκο των μονάδων απορριμμάτων που έχουν συλλεχθεί από τα απορριμματοφόρα και τον όγκο των μονάδων απορριμμάτων που βρίσκονται ακόμα στους κάδους και αναμένουν αποκομιδή. Στα δύο πρώτα διαγράμματα γίνεται διαχωρισμός των τιμών ανάλογα με τον τύπο απορριμματοφόρων.

4.5 KPI’s

Στο πλαίσιο κειμένου εμφανίζεται στο τέλος κάθε 24ώρου δύο ειδικότεροι Δείκτες Απόδοσης (Key Performance Indices). Ο πρώτος είναι ο **υπολογισμός κόστους ανά μονάδα απορριμμάτων** που έχει συλλεχθεί μέχρι στιγμής. Ο δεύτερος είναι το **ποσοστό των μονάδων απορριμμάτων που βρίσκονται ακόμα στους κάδους** σε σχέση με το άθροισμα όλων των απορριμμάτων της συνοικίας (φορτωμένοι σε απορριμματοφόρα + μη συλλεγμένοι μέσα σε κάδους).

5. ΠΕΡΙΒΑΛΛΟΝ NetLogo

5.1 Γενικά

Η NetLogo είναι ένα προγραμματιστικό περιβάλλον μοντελοποίησης με σκοπό την προσομοίωση συστημάτων πολλαπλών πρακτόρων (ΣΠΠ) που απαρτίζονται από μεγάλους πληθυσμούς πρακτόρων. Δημιουργήθηκε από τον Uri Wilensky το 1999 και έκτοτε υπόκειται σε συνεχείς βελτιώσεις από το Center for Connected Learning and Computer-Based Modeling (http://ccl.northwestern.edu/netlogo/).

Τα υπό μελέτη μοντέλα εξελίσσονται σε συνάρτηση με το χρόνο επιτρέποντας τη διερεύνηση των σχέσεων που προκύπτουν τόσο σε χαμηλό επίπεδο συμπεριφοράς μεταξύ ατόμων, όσο και σε μακρο-μεγέθη που μεταβάλλονται ως αποτέλεσμα των πρώτων (Sakellariou et al, 2008).

Το σύστημα προσφέρει μία απλή γλώσσα προγραμματισμού και δυνατότητες δημιουργίας GUI τα οποία επιτρέπουν την εύκολη δημιουργία οπτικοποιημένων μοντέλων ΣΠΠ. Παρέχεται ένα πλούσιο μενού βασικών εντολών που υποστηρίζει δεκαδικούς αριθμούς κινητής υποδιαστολής, γεννήτριες τυχαίων αριθμών και γραφική αναπαράσταση. Το περιβάλλον θεωρείται ένα χρήσιμο εργαλείο για ταχεία προτυποποίηση και αρχική δοκιμή ΣΠΠ κυρίως για συστήματα με πράκτορες που κινούνται και δρουν εντός ενός πεπερασμένου χώρου. Επίσης, είναι αρκετά χρήσιμο για εκπαιδευτική χρήση.

Οι κύριες οντότητες της NetLogo είναι τα patches, τα turtles και ο observer. Ο observer είναι αυτός που ελέγχει το πείραμα, στο οποίο τα patches και turtles έχουν ενεργή συμμετοχή. Τα patches είναι «στατικοί» πράκτορες δηλαδή δομικά συστατικά του χώρου μέσα στον οποίο «υπάρχουν» τα turtles που με την σειρά τους έχουν δυνατότητα κίνησης, «ζωής» και αλληλεπίδρασης. Και τα δύο παραπάνω μπορούν να εξετάζουν τον κόσμο γύρω τους, για παράδειγμα μπορούν να διαπιστώσουν την ύπαρξη άλλων πρακτόρων, να διαβάσουν την κατάσταση των γειτόνων τους και να τροποποιήσουν το περιβάλλον. Ίσως το χαρακτηριστικό που αυξάνει την δυνατότητα αποτύπωσης του πραγματικού κόσμου σε ένα μοντέλο είναι η ύπαρξη μεταβλητών που ανήκουν αποκλειστικά σε κάποια patches/turtles. Αυτό σημαίνει ότι για τα μεν patches είναι εφικτό να μοντελοποιούν ένα σύνθετο περιβάλλον που απαιτεί την χρήση πολλών μεταβλητών, για τα δε turtles ότι κάθε πράκτορας μπορεί να έχει την δική του κατάσταση.

Η γλώσσα προγραμματισμού επιτρέπει την ακριβή περιγραφή της συμπεριφοράς κάθε πράκτορα και του ελέγχου κατά την εκτέλεση. Ο έλεγχος και η εκτέλεση των πρακτόρων είναι ευθύνη της οντότητας του observer που «ζητάει» από κάθε πράκτορα να εκτελέσει συγκεκριμένες υπολογιστικές εντολές.

Η αξία του περιβάλλοντος της NetLogo έγκειται στο ότι αναδεικνύει όλες αυτές τις βασικές ιδιότητες των πρακτόρων που αναφέρθηκαν στα προηγούμενα κεφάλαια, αφού κάθε πράκτορας στην NetLogo:

- **αντιλαμβάνεται** το περιβάλλον του και **αντιδρά** σε αυτό
- εκτελεί τις **δικές του εντολές**
- είναι **αυτόνομος**

Η ανταλλαγή μηνυμάτων μεταξύ των πρακτόρων γίνεται κατά τρόπο παρόμοιο με αυτό που περιγράφεται σε προηγούμενο κεφάλαιο, όπου αναλύεται η γλώσσα FIPA ACL. Η συμπεριφορά των turtles μέσα στο χώρο καθορίζεται από την αλληλεπίδρασή τους με τα στατικά patches, δηλαδή την ανταλλαγή μηνυμάτων με σκοπό την εξακρίβωση της κατάστασης των τελευταίων. Αυτό σε συνδυασμό με τις προθέσεις των turtles είναι που καθορίζει το επόμενο βήμα, άρα και την μετάβαση του περιβάλλοντος σε μία νέα κατάσταση. Ακολουθεί μία αντιπαράθεση διατύπωση μηνύματος σε FIPA ACL

```
[“inform” “sender:truck” “receiver:bins-around” “bin-load:” “0”]
```

με την αντίστοιχη έκφρασή του στη NetLogo να είναι η

```
ask bins-around  <- αίτημα προς πληθυσμό πρακτόρων
    [
      set bin-load 0
    ]
```

Η NetLogo επιτρέπει στον χρήστη να ανοίξει υπάρχουσες προσομοιώσεις και να πειραματιστεί ανακαλύπτοντας τις μεταβολές συμπεριφορών υπό διαφορετικές συνθήκες. Για τον σκοπό αυτό υπάρχει προεγκατεστημένη μία Βιβλιοθήκη Μοντέλων (Models Library) με μία μεγάλη συλλογή έτοιμων προσομοιώσεων που μπορούν να εκτελεστούν ως έχουν ή ακόμα και να υποστούν τροποποιήσεις. Τα μοντέλα εξετάζουν προβλήματα και συστήματα από πολλές φυσικές και κοινωνικές επιστήμες όπως η βιολογία και η ιατρική, η φυσική και η χημεία, τα μαθηματικά και η πληροφορική, τα οικονομικά και την ψυχολογία.

Η εκμάθηση του περιβάλλοντος και η εξοικείωση με τις βασικές λειτουργίες μπορεί να γίνει εύκολα με την βοήθεια πληθώρας εκπαιδευτικού υλικού που βρίσκεται διαθέσιμο στο διαδίκτυο. Αρχίζοντας από το εκπαιδευτικό υλικό στον ιστότοπο της NetLogo, ο ενδιαφερόμενος μπορεί να μελετήσει τα:

- Tutorial #1: Models
  http://ccl.northwestern.edu/netlogo/docs/tutorial1.html
- Tutorial #2: Commands
  http://ccl.northwestern.edu/netlogo/docs/tutorial2.html
- Tutorial #3: Procedures
  http://ccl.northwestern.edu/netlogo/docs/tutorial3.html

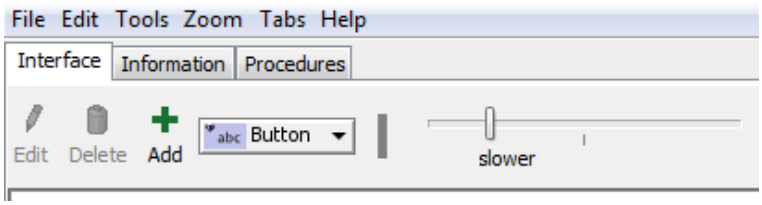
Μία δεύτερη ηλεκτρονική πηγή είναι η σειρά βιντεοδιαλέξεων του Gabriel Wurzer στο κανάλι που διατηρεί στο YouTube. Το περιβάλλον της NetLogo περιγράφεται σε 16 σύντομες βιντεοδιαλέξεις
http://www.youtube.com/watch?v=XJ-gO-yAwHU&feature=relmfu

καθώς και μία νεότερη αυτοτελής παρουσίαση του ίδιου με τίτλο «Introductory Lecture on NetLogo - Agents in Archeology Workshop 2011»

http://www.youtube.com/watch?v=nGEYV4BEzEM&list=UU4u7zjoQmBiXJ1fAx2vFwbw&index=3&feature=plcp

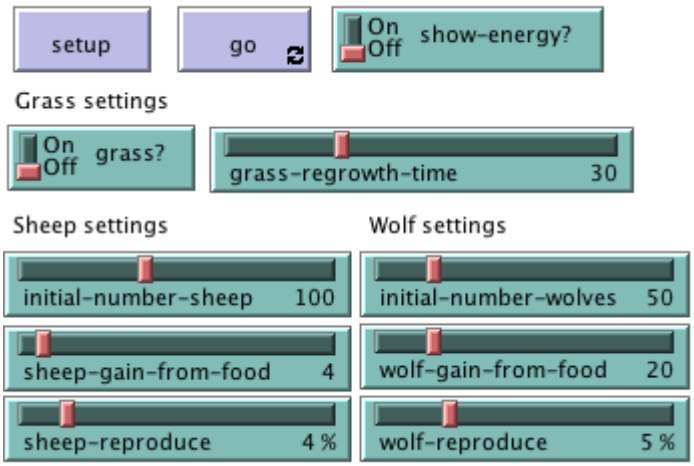
5.2 Διαχείριση Μοντέλων

Κάθε μοντέλο (NetLogo User Manual, 2012) αποτελείται από τρεις καρτέλες (**tabs**) (Σχήμα 5.1). Στην πρώτη βρίσκεται η γραφική αναπαράσταση του μοντέλου (**Interface**), όπου ο χρήστης μπορεί να μεταβάλει παραμέτρους, να εκτελέσει το μοντέλο και να παρακολουθήσει την έκβαση. Η δεύτερη καρτέλα (**Info**) αποτελεί μία εισαγωγή στο μοντέλο. Εδώ αναλύεται το σύστημα και το τι αυτό μοντελοποιεί, πως δημιουργήθηκε το μοντέλο και πως χρησιμοποιείται. Σε αυτή την καρτέλα μπορούν να συμπεριληφθούν ακόμα και προτάσεις για επέκταση/βελτίωση του μοντέλου ή αναφορές σε ιδιαίτερα εργαλεία της NetLogo που χρησιμοποιήθηκαν κατά την υλοποίηση. Η τελευταία καρτέλα (**Procedures**) περιέχει τον κώδικα του μοντέλου, όπου κάποιος μπορεί να δει, να τροποποιήσει και να προσθέσει λειτουργικότητα.



Σχήμα 5.1: Tabs στο περιβάλλον της NetLogo

Ανοίγοντας ένα μοντέλο, στην καρτέλα Interface ο χρήστης έχει την δυνατότητα να αρχικοποιήσει το περιβάλλον με τη βοήθεια του κουμπιού **Setup** και στη συνέχει να εκκινήσει την εκτέλεση με το κουμπί **Go** (Σχήμα 5.2). Ο κόσμος μέσα στον οποίο «ζουν» και αλληλεπιδρούν οι πράκτορες προκειμένου να εκτελεστεί η προσομοίωση είναι ένας χώρος δύο διαστάσεων (**world**), που ορίζεται με την βοήθεια ζεύγους συντεταγμένων x και y με το κέντρο του να αντιστοιχεί στο σημείο (0,0).

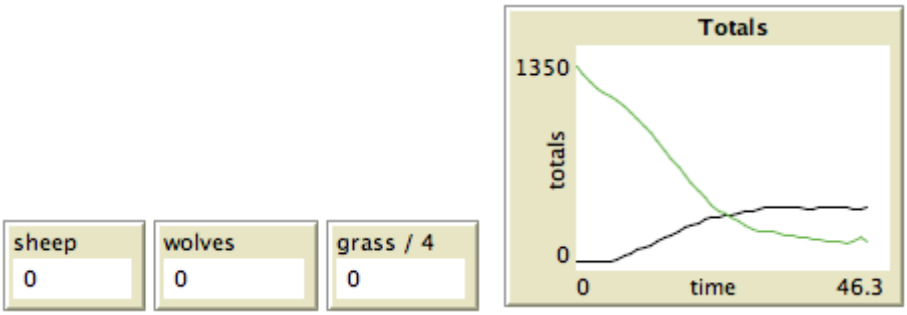


Σχήμα 5.2: Κουμπιά, διακόπτες και επιλογείς στο Interface Tab.

Ένας επιλογέας ταχύτητας (**Speed Slider**) ρυθμίζει την ταχύτητα (χρόνο) εξέλιξης της προσομοίωσης (Σχήμα 5.1), ώστε ο χρήστης να μπορεί είτε να παρακολουθεί με λεπτομέρεια κάθε στιγμιότυπο του φαινομένου (χαμηλή ταχύτητα),είτε να μεταβαίνει γρήγορα σε προχωρημένη χρονική στιγμή (υψηλή ταχύτητα).

Ανάλογα με την προσομοίωση ο χρήστης έχει στην διάθεσή του διακόπτες (**switches**) και επιλογείς (**sliders**) που ελέγχουν τις αρχικές συνθήκες του «κόσμου» και την συμπεριφορά των πρακτόρων (Σχήμα 5.2). Μεταβάλλοντας τα παραπάνω σε κάθε εκτέλεση, δημιουργεί και διαφορετικά σενάρια εκτέλεσης.

Από την έκβαση κάθε προσομοίωσης προκύπτουν αριθμητικά στοιχεία που μπορούν να εμφανιστούν είτε σε ενδείξεις μετρητών (**monitors**) είτε ακόμα και να αναπαρασταθούν γραφικά (**plots**) (Σχήμα 5.3).



Σχήμα 5.3: Μετρητές και γραφικές παραστάσεις στο Interface Tab.

6. ΜΟΝΤΕΛΟ ΕΠΙΛΥΣΗΣ ΠΡΟΒΛΗΜΑΤΟΣ ΑΠΟΚΟΜΙΔΗΣ ΑΣΤΙΚΩΝ ΑΠΟΡΡΙΜΜΑΤΩΝ

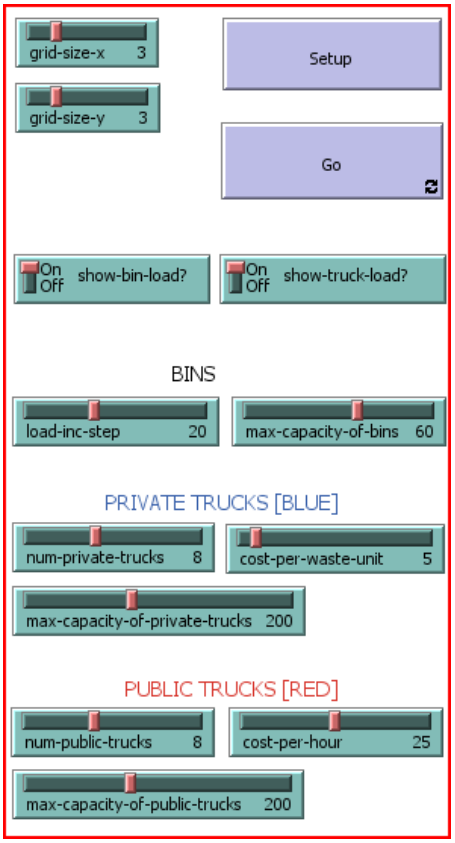
EmptyBinCity

Στο κεφάλαιο αυτό περιγράφεται το μοντέλο επίλυσης του προβλήματος. Αρχικά, γίνεται η περιγραφή της διεπαφής με την χρήστη και ακολουθεί αναφορά και επεξήγηση του κώδικα που την υλοποιεί.

6.1 Διεπαφή μοντέλου (Interface Tab)

Η διεπαφή του μοντέλου χωρίζεται σε τρεις περιοχές. Αριστερά βρίσκονται συγκεντρωμένα τα εργαλεία παραμετροποίησης (control panel) τα κουμπιά ελέγχου, οι διακόπτες και οι επιλογείς αρχικών τιμών/παραμέτρων. Στο κέντρο βρίσκεται ο «κόσμος» του μοντέλου (world) που αναπαριστά μία συνοικία πόλης. Στην δεξιά πλευρά της διεπαφής βρίσκονται γραφικές παραστάσεις μεγεθών και ένα πλαίσιο κειμένου.

6.1.1 Control Panel

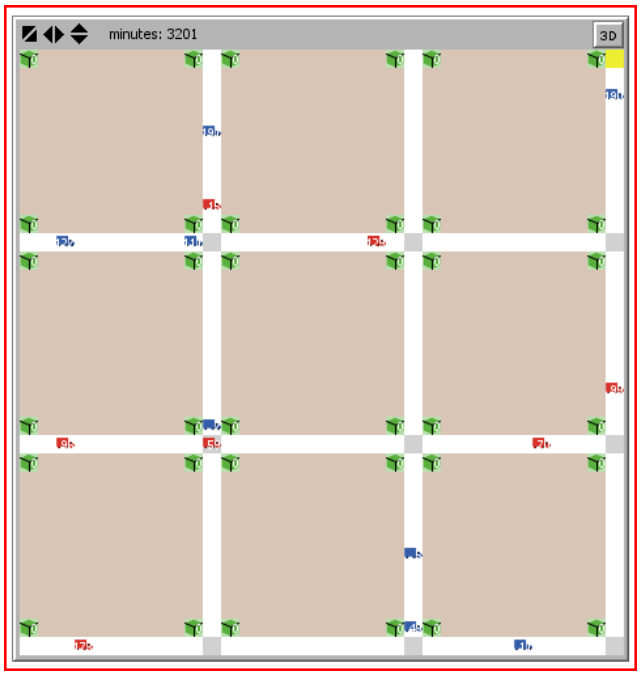


Οι διαστάσεις της συνοικίας (πλάτος x ύψος) ορίζονται με την βοήθεια δύο επιλογέων διαστάσεων, τους *grid-size-x* και *grid-size-y* που μπορούν να πάρουν τιμές από 1 έως 9. Οι διακόπτες *show-bin-load* και *show-truck-load* ενεργοποιούν την ένδειξη μονάδων απορριμμάτων που περιέχει αντίστοιχα κάθε κάδος ή απορριμματοφόρο και εμφανίζεται κατά την διάρκεια εκτέλεσης με τη μορφή ετικέτας (tag).

Οι παράμετροι που σχετίζονται με τους πράκτορες που λειτουργούν ως κάδοι είναι οι επιλογείς τιμών *load-inc-step* και *max-capacity-of-bins* και ελέγχουν το εύρος βήματος αύξησης του περιεχομένου των κάδων που συμβαίνει σε ωριαία βάση και τη μέγιστη χωρητικότητα του κάδου αντίστοιχα. Και τα δύο εκφράζονται σε μονάδες απορριμμάτων.

Ακολουθούν οι παράμετροι των απορριμματοφόρων (ιδιωτικά και δημόσια). Για τους δύο στόλους ορίζεται ξεχωριστά το πλήθος *num-private-trucks* και *num-public-trucks* και η μέγιστη χωρητικότητα οχήματος σε μονάδες απορριμμάτων (*max-capacity-of-private-trucks* και *max-capacity-of-public-trucks*). Επιπλέον, η έννοια του κόστους στα μεν ιδιωτικά οχήματα είναι συνάρτηση των συλλεχθέντων μονάδων απορριμμάτων και μετριέται σε €/μονάδα απ/των (*cost-per-waste-unit*), ενώ στα δημόσια το κόστος είναι συνάρτηση του χρόνου λειτουργίας/κυκλοφορίας του οχήματος και μετριέται σε €/ώρα (*cost-per-hour*).

6.1.2 World



Η συνοικία αποτελείται από οικοδομικά τετράγωνα με θέσεις κάδων πράσινου χρώματος στις γωνίες τους. Οι λευκοί αριθμοί στους κάδους αναπαριστούν το περιεχόμενό τους σε μονάδες απορριμμάτων.

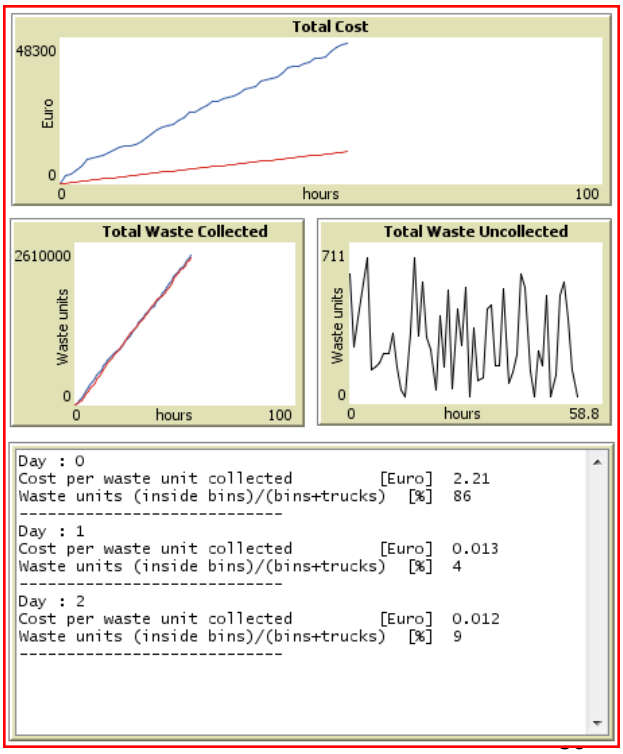
Ανάμεσα στα οικοδομικά τετράγωνα υπάρχουν δρόμοι κυκλοφορίας οχημάτων πάνω στους οποίους με γκρι χρώμα ορίζονται οι διασταυρώσεις οδών. Στο βόρειο-ανατολικό άκρο της πόλης βρίσκεται η χωματερή της πόλης με κίτρινο χρώμα.

Τα απορριμματοφόρα κινούνται προς όλες τις κατευθύνσεις πάνω στο οδικό δίκτυο και ο αριθμός που αναγράφεται πάνω τους είναι το τρέχον φορτίο τους σε μονάδες απορριμμάτων. Με μπλε χρώμα είναι ο στόλος των ιδιωτικών οχημάτων αποκομιδής, ενώ με κόκκινο των δημοσίων.

Όταν οι κάδοι ξεπερνούν την χωρητικότητά τους, το χρώμα τους γίνεται μαύρο και παύουν να δέχονται νέα απορρίμματα μέχρι να αδειάσουν από κάποιο όχημα. Παρομοίως, όταν ένα απορριμματοφόρο γεμίσει, το χρώμα του γίνεται μαύρο και μεταφέρεται στην χωματερή, όπου και αδειάζει όλο του το περιεχόμενο πριν ξεκινήσει εκ νέου για την αποκομιδή της συνοικίας.

Στο άνω περιθώριο του παραθύρου φαίνεται ο χρόνος σε λεπτά (*minutes*).

6.1.3 Plots



Το πρώτο διάγραμμα με τίτλο **Total Cost** απεικονίζει γραφικά την πορεία του κόστους αποκομιδής και διαχωρίζει το κόστος λειτουργίας των ιδιωτικών οχημάτων (μπλε καμπύλη) από τα δημόσια (κόκκινη καμπύλη). Στο επόμενο διάγραμμα με τίτλο **Total Waste Collected** απεικονίζονται οι μονάδες απορριμμάτων που έχουν συλλεχθεί από τα απορριμματοφόρα (ιδιωτικά και δημόσια αντίστοιχα με διαφορετικό χρώμα). Το διάγραμμα με τίτλο **Total Waste Uncollected** αναπαριστά τον όγκο των μονάδων απορριμμάτων που βρίσκονται

ακόμα στους κάδους και αναμένουν αποκομιδή.

Το πλαίσιο κειμένου που ακολουθεί ενημερώνεται στο τέλος κάθε 24ώρου και δίνει αναφορά για την πορεία των δύο Δεικτών Απόδοσης (Key Performance Indices). Ο πρώτος είναι ο **υπολογισμός κόστους ανά μονάδα απορριμμάτων** που έχει συλλεχθεί μέχρι στιγμής. Ο δεύτερος είναι το **ποσοστό των μονάδων απορριμμάτων που βρίσκονται ακόμα στους κάδους** σε σχέση με το άθροισμα όλων των απορριμμάτων της συνοικίας (φορτωμένοι σε απορριμματοφόρα + μη συλλεγμένοι μέσα σε κάδους).

6.2 Κώδικας μοντέλου (Procedures Tab)

Ο κώδικας του μοντέλο μπορεί να αναλυθεί σε τέσσερα μέρη. Το πρώτο μέρος αφορά στον ορισμό των γενικών μεταβλητών (*globals*) του προγράμματος καθώς και των επιμέρους ορισμό των οντοτήτων πρακτόρων του μοντέλου και των εσωτερικών τους μεταβλητών.

- **bins-own**
- **patches-own**
- **pritrucks-own**
- **pubtrucks-own**

Ένα παράδειγμα κώδικα ορισμού πράκτορα καθώς και σχόλια κόκκινου χρώματος είναι το παρακάτω :

```
pritrucks-own
[
  up-car? <- boolean μεταβλητή που σχετίζεται με την κίνηση
  load    <- αριθμητική μεταβλητή που αποθηκεύει το φορτίο ιδιωτικού
απορριμματοφόρου
  cost    <- αριθμητική μεταβλητή που αποθηκεύει το κόστος λειτουργίας
ιδιωτικού απορριμματοφόρου
  full?   <- boolean μεταβλητή που σχετίζεται με το φορτίο, true για γεμάτο
]
```

Στο δεύτερο μέρος ακολουθεί ο κώδικας αρχικοποίησης του μοντέλου με βάση τις ρυθμίσεις του χρήστη, αλλά και οι μέθοδοι (procedures) ορισμού της συμπεριφοράς των πρακτόρων. Ειδικότερα :

- **setup**: κεντρική μέθοδος αρχικοποίησης του μοντέλου
- **setup-globals** : μέθοδος αρχικοποίησης των γενικών μεταβλητών
- **setup-patches** : μέθοδος αρχικοποίησης των στατικών πρακτόρων (patches), δηλαδή των οδών, των διασταυρώσεων και των θέσεων των κάδων
- **setup-bins** : μέθοδος αρχικοποίησης των πρακτόρων που λειτουργούν ως κάδοι απορριμμάτων (turtles)
- **setup-pritrucks** : μέθοδος αρχικοποίησης των κινούμενων πρακτόρων που αντιπροσωπεύουν τα ιδιωτικά απορριμματοφόρα (turtles)
- **setup-pubtrucks** : μέθοδος αρχικοποίησης των κινούμενων πρακτόρων που αντιπροσωπεύουν τα δημόσια απορριμματοφόρα (turtles)
- **put-on-empty-road** : μέθοδος τοποθέτησης των απορριμματοφόρων σε τυχαίες θέσεις του οδικού δικτύου

Ένα παράδειγμα κώδικα αρχικοποίησης πράκτορα είναι το παρακάτω :

```
to setup-bins

  create-bins bin-population  <- εντολή δημιουργίας πληθυσμού πρακτόρων
  [
    move-to one-of bin-spots with [not any? turtles-on self] <- τοποθέτηση
πράκτορα-κάδου σε άδεια θέση με χρήση built-in εντολών της NetLogo

    set shape "box"                    <- αλλαγή της ιδιότητας shape
    set color green                    <- αλλαγή της ιδιότητας color
    set size 1                         <- αλλαγή της ιδιότητας size
    set bin-load random load-inc-step <- αλλαγή του περιεχομένου του κάδου
(bin-load) μέσω της τιμής load-inc-step που έχει δώσει ο χρήστης στο
interface

    display-waste-in-bins <- κλήση άλλης μεθόδου για εμφάνισης περιεχομένου
  ]

end
```

Το τρίτο μέρος απαρτίζεται από τις μεθόδους εκείνες που ελέγχουν την εκτέλεση του μοντέλου.

- **go** : κεντρική μέθοδος που συντονίζει την εκτέλεση και ενεργοποιείται από το κουμπί go του interface
- **display-waste-in-bins** : μέθοδος εμφάνισης περιεχομένου κάδων
- **advance-trucks** : μέθοδος που ελέγχει την κίνηση των απορριμματοφόρων πάνω στους δρόμους
- **check-and-collect-waste** : μέθοδος που ελέγχει την αλληλεπίδραση κάδων και απορριμματοφόρων, όταν βρεθούν σε γειτονική απόσταση
- **display-truck-load** : μέθοδος εμφάνισης φορτίου απορριμματοφόρων
- **check-capacity** : μέθοδος που χρησιμοποιούν τα απορριμματοφόρα για να εξακριβώσουν ότι δεν έχουν υπερβεί την μέγιστη χωρητικότητα τους
- **move-to-depot** : μέθοδος που μεταφέρει τα πλήρη απορριμματοφόρα στην χωματερή

Ένα παράδειγμα κώδικα μεθόδου χρόνου εκτέλεσης είναι το παρακάτω :

```
to check-and-collect-waste

    ; check if there are bins in the vicinity
    let bins-around (bins in-radius 1)

    ; calculate neighboring bins' load
    let waste-around sum [bin-load] of bins-around

    ; collect waste on the truck
    set load load + waste-around


    ; set neighboring bins to be empty of waste
    ask bins-around  <- αίτημα προς πληθυσμό πρακτόρων
    [
      set bin-load 0
    ]

    ; increase cost for private trucks on a waste unit basis
    if ( member? self pritrucks )
    [
      set cost cost + (waste-around * cost-per-waste-unit)
    ]

end
```

Το τελευταίο μέρος του κώδικα συγκεντρώνει τις μεθόδους που διαχειρίζονται τις γραφικές παραστάσεις και το κείμενο διαλόγου στο δεξί μέρος του Interface.

- **do-plotting** : κύρια μέθοδος γραφικής αναπαράστασης
- **plot-new-value [name-of-plot pen value]** : μέθοδος που επεμβαίνει στην καμπύλη <pen> της γραφικής παράστασης <name-of-plot> δίνοντας την τιμή <value> την τρέχουσα χρονική στιγμή
- **report-kpis** : μέθοδος που τροφοδοτεί με νέα παράγραφο κειμένου το πλαίσιο κειμένου και ενημερώνει τον χρήστη για τις μετρικές του μοντέλου

Ως παράδειγμα κώδικα μεθόδου διαχείρισης διαγράμματος κατά τον χρόνο εκτέλεσης παρατίθεται η δεύτερη παρακάτω :

```
to plot-new-value [ name-of-plot pen value ]

  set-current-plot name-of-plot
  set-current-plot-pen pen
  plot value

end
```

Ο κώδικας υλοποίησης των οικοδομικών τετραγώνων και των οδών βασίστηκε στον αντίστοιχο κώδικα του μοντέλου Traffic Grid από την βιβλιοθήκη μοντέλων (Models Library) της NetLogo 4.1.3.

7. ΣΥΜΠΕΡΑΣΜΑΤΑ

Η μελέτη και εφαρμογή της τεχνολογίας των ευφυών πρακτόρων έδειξε ότι μπορεί να χρησιμοποιηθεί πετυχαίνοντας σε μεγάλο βαθμό μία πιστή αναπαράσταση του πραγματικού κόσμου. Τόσο στον σχεδιασμό του «κόσμου» μέσα στον οποίο ζουν και αλληλεπιδρούν οι πράκτορες, όσο και στην λεπτομέρεια περιγραφής της συμπεριφοράς των ίδιων των πρακτόρων σε συνδυασμό με την δυνατότητα ύπαρξης διαφορετικών πληθυσμών ετερογενών οντοτήτων-πρακτόρων η τεχνολογία αποδεικνύεται επαρκής και ανθεκτική.

Το δεύτερο αξιοσημείωτο στοιχείο της μεθοδολογίας, το οποίο συνδέεται με τον τρόπο που αυτή μιμείται τον πραγματικό κόσμο, είναι η δυνατότητα γρήγορης μοντελοποίησης με καλής ποιότητας εποπτεία του για χώρους δύο διαστάσεων. Έχοντας ξεκαθαρίσει τους ρόλους των οντοτήτων που συμμετέχουν στο υπό εξέταση μοντέλο προκύπτει σχεδόν φυσικά η μετατροπή τους σε στατικούς (patches) ή κινητούς (turtles) πράκτορες οδηγώντας ομαλά σε ταχύτατη προτυποποίηση. Τα παραγόμενα μοντέλο μπορούν να αποτελέσουν τον σκελετό ή τον πυρήνα για κάποιον συγγενή τρόπο υλοποίησης.

Μεγάλη συμβολή στην ταχύτητα ανάπτυξης του μοντέλου που εξετάστηκε στην παρούσα εργασία είχε και η ευκολία εξοικείωσης με το περιβάλλον της NetLogo καθώς και η εκμάθηση της γλώσσας υψηλού επιπέδου που υποστηρίζει την υλοποίηση των πρακτόρων. Η ύπαρξη πλούσιου υλικού υποστήριξης τόσο στον ιστότοπο του The Center for Connected Learning (CCL) and Computer-Based Modeling του Northwestern University, όσο και σχετικών εκπαιδευτικών video στο YouTube καθιστούν τη NetLogo χρήσιμη και για εκπαιδευτικούς σκοπούς.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Fasli M. (2007) Agent Technology for e-Commerce, John Wiley & Sons, p.p1-111

Sakellariou, I., Kefalas,P., Stamatopoulou, I. (2008), Teaching Intelligent Agents using NetLogo, Proceedings of the ACM-IFI IEEIII 2008, Informatics Education Europe III Conference, Venice, Italy, December 4-5 2008

Wooldridge M J. (2009) An Introduction to MultiAgent Systems, John Wiley & Sons, 2nd ed p.p1-105

NetLogo User Manual (2012), http://ccl.northwestern.edu/netlogo/faq.html, (ανακτήθηκε στις 15 Ιουλίου 2012)

ΠΑΡΑΡΤΗΜΑ 1. - ΚΩΔΙΚΑΣ ΣΕ ΠΕΡΙΒΑΛΛΟΝ NETLOGO

```
breed [bins bin]
breed [pritrucks pritruck]
breed [pubtrucks pubtruck]

globals
[
  grid-x-inc                ;; the amount of patches in between two roads in the x
direction
  grid-y-inc                ;; the amount of patches in between two roads in the y
direction
  bin-population           ;  the number of bins
  new-load                 ;  waste units to be added per tick per bin
  total-waste-coll-pri     ;  waste units loaded on private trucks
  total-waste-coll-pub     ;  waste units loaded on public trucks
  total-waste-uncoll       ;  waste units still inside bins

  ;; patch agentsets
  intersections ;; agentset containing the patches that are intersections
  roads         ;; agentset containing the patches that are roads
  bin-spots     ;  agentset containing the patches with spots for bins
]

bins-own
[
  bin-load        ; waste units inside bin
]

patches-own
[
  intersection?   ;; true if the patch is at the intersection of two roads
]

pritrucks-own
[
  up-car?   ;; true if the turtle moves downwards and false if it moves to the right
  load      ;; the current load of waste on a truck
  cost      ;; the cost of operation
  full?     ;; true if the waste on the truck has reached the max-capacity
]

pubtrucks-own
[
  up-car?   ;; true if the turtle moves downwards and false if it moves to the right
  load      ;; the current load of waste on a truck
  cost      ;; the cost of operation
  full?     ;; true if the waste on the truck has reached the max-capacity
]


;;;;;;;;;;;;;;;;;;;;;;
;; Setup Procedures ;;
;;;;;;;;;;;;;;;;;;;;;;

;-------------------------------------------------------------------------------------
;; Initialize the display by giving the global and patch variables initial values.
;; Create num-cars of turtles if there are enough road patches for one turtle to
;; be created per road patch. Set up the plots.
to setup

  ca
  setup-globals

  ;; First we ask the patches to draw themselves and set up a few variables
  setup-patches
  setup-bins

  if (num-private-trucks + num-public-trucks > 4 * ( grid-size-x + grid-size-y ) )
  [
    user-message (word "There are too many trucks for the amount of "
                       "road.  Either increase the amount of roads "
                       "by increasing the GRID-SIZE-X or "
                       "GRID-SIZE-Y sliders, or decrease the "
                       "number of trucks by lowering the NUMBER slider.\n"
                       "The setup has stopped.")
    stop
  ]

  ;; Now create the turtles and have each created turtle call the functions setup-cars
and set-car-color
  create-pritrucks num-private-trucks
  [
    setup-pritrucks
  ]
```

```
  create-pubtrucks num-public-trucks
  [
    setup-pubtrucks
  ]

end


;-------------------------------------------------------------------------------
;; Initialize the global variables to appropriate values
to setup-globals

  set grid-x-inc world-width / grid-size-x
  set grid-y-inc world-height / grid-size-y

  set total-waste-coll-pri 0
  set total-waste-coll-pub 0
  set total-waste-uncoll 0

end


;-------------------------------------------------------------------------------
;; Make the patches have appropriate colors, set up the roads and intersections
agentsets,
;; and initialize the traffic lights to one setting
to setup-patches

  ;; initialize the patch-owned variables and color the patches to a base-color
  ask patches
  [
    set intersection? false
    set pcolor brown + 3
  ]

  set roads patches with
    [(floor((pxcor + max-pxcor - floor(grid-x-inc - 1)) mod grid-x-inc) = 0) or
    (floor((pycor + max-pycor) mod grid-y-inc) = 0)]

  set intersections roads with
    [(floor((pxcor + max-pxcor - floor(grid-x-inc - 1)) mod grid-x-inc) = 0) and
    (floor((pycor + max-pycor) mod grid-y-inc) = 0)]

  set bin-spots patches with
  [
    (
      (floor((pxcor + max-pxcor - floor(grid-x-inc - 1)) mod grid-x-inc) = 1) and
      (floor((pycor + max-pycor) mod grid-y-inc) = 1)
    )
    or
    (
      (floor((pxcor + max-pxcor - floor(grid-x-inc - 1)) mod grid-x-inc) = grid-x-inc
- 1) and
      (floor((pycor + max-pycor) mod grid-y-inc) = 1)
    )
    or
    (
      (floor((pxcor + max-pxcor - floor(grid-x-inc - 1)) mod grid-x-inc) = 1) and
      (floor((pycor + max-pycor) mod grid-y-inc) = grid-y-inc - 1)
    )
    or
    (
      (floor((pxcor + max-pxcor - floor(grid-x-inc - 1)) mod grid-x-inc) = grid-x-inc
- 1) and
      (floor((pycor + max-pycor) mod grid-y-inc) = grid-y-inc - 1)
    )
  ]

  ask roads [ set pcolor white ]

  ask intersections [ set pcolor gray + 3 ]

  ask bin-spots [ set pcolor 58 ]

  ; depot patch is yellow
  ask roads with [ ( pxcor = 16 ) and ( pycor = 16 ) ]
  [
    set pcolor yellow
  ]

  set bin-population count patches with [pcolor = 58]

  setup-intersections

end
```

```
;-------------------------------------------------------------------------------
;; Give the intersections appropriate values
to setup-intersections

  ask intersections
  [
    set intersection? true
  ]

end


;-------------------------------------------------------------------------------
to setup-bins

  create-bins bin-population
  [
    move-to one-of bin-spots with [not any? turtles-on self]
    set shape "box"
    set color green
    set size 1
    set bin-load random load-inc-step

    display-waste-in-bins
  ]

end


;-------------------------------------------------------------------------------
;; Initialize the turtle variables to appropriate values and place the turtle on an
empty road patch.
to setup-pritrucks  ;; turtle procedure

  set color blue
  set shape "truck"
  put-on-empty-road
  set load 0
  set cost 0
  set full? false
  set label ""

  ifelse intersection?
  [
    ifelse random 2 = 0
    [ set up-car? true ]
    [ set up-car? false ]
  ]
  [
    ; if the turtle is on a vertical road (rather than a horizontal one)
    ifelse (floor((pxcor + max-pxcor - floor(grid-x-inc - 1)) mod grid-x-inc) = 0)
    [ set up-car? true ]
    [ set up-car? false ]
  ]
  ifelse up-car?
  [ set heading 180 ]
  [ set heading 90 ]

end


;-------------------------------------------------------------------------------
;; Initialize the turtle variables to appropriate values and place the turtle on an
empty road patch.
to setup-pubtrucks  ;; turtle procedure

  set color red
  set shape "truck"
  put-on-empty-road
  set load 0
  set cost 0
  set full? false
  set label ""

  ifelse intersection?
  [
    ifelse random 2 = 0
    [ set up-car? true ]
    [ set up-car? false ]
  ]
  [
    ; if the turtle is on a vertical road (rather than a horizontal one)
    ifelse (floor((pxcor + max-pxcor - floor(grid-x-inc - 1)) mod grid-x-inc) = 0)
    [ set up-car? true ]
    [ set up-car? false ]
  ]
```

```
  ifelse up-car?
  [ set heading 180 ]
  [ set heading 90 ]

end


;-------------------------------------------------------------------------------
;; Find a road patch without any turtles on it and place the turtle there.
to put-on-empty-road  ;; turtle procedure

  move-to one-of roads with [not any? turtles-on self]

end



;;;;;;;;;;;;;;;;;;;;;;;
;; Runtime Procedures ;;
;;;;;;;;;;;;;;;;;;;;;;;


;-------------------------------------------------------------------------------
;; Run the simulation
to go

  ; change bins' content every hour = 60 minutes (ticks)
  if ( ticks mod 60 = 0 )
  [
    ; "update" garbage disposed in the bins
    set new-load random load-inc-step

    ask bins
    [
      ifelse (bin-load < max-capacity-of-bins)
      [
        set bin-load bin-load + new-load   ; increases bin-load if waste units less
than 100
        set color green  ; bins that are not full yet  stay green
      ]
      [
        set color black  ; FULL bins (waste units equal and more than 100) turn black
      ]
    ]
  ]

  display-waste-in-bins

  ; move garbage trucks
  ask pritrucks [ advance-trucks ]
  ask pubtrucks [ advance-trucks ]

  ; check and collect waste from bins
  ask pritrucks [ check-and-collect-waste ]
  ask pritrucks [ display-truck-load ]
  ask pritrucks [ check-capacity]

  ask pubtrucks [ check-and-collect-waste ]
  ask pubtrucks [ display-truck-load ]
  ask pubtrucks [ check-capacity]

  ; calculate Key Performance Indices
  set total-waste-coll-pri total-waste-coll-pri + sum [ load ] of pritrucks
  set total-waste-coll-pub total-waste-coll-pub + sum [ load ] of pubtrucks
  set total-waste-uncoll sum [ bin-load ] of bins

  ; refresh plots every hour = 60 minutes (ticks)
  if ( ticks mod 60 = 0 )
  [
    do-plotting
  ]

  ; refresh output every 24 hours = 1440 minutes (ticks)
  if ( ticks mod 1440 = 0 )
  [
    report-kpis
  ]

  tick

end


;-------------------------------------------------------------------------------
;; Display units of waste inside a bin
to display-waste-in-bins
```

```
  ask bins [ set label "" ]
  if show-bin-load? [
    ask bins [ set label round bin-load ]
  ]

end


;-------------------------------------------------------------------------------------
;; Driving trucks
to advance-trucks
    ; decide new route on intersection
    if intersection?
    [
      let new-heading random 3
      ; new-heading 0 maintains current direction

      ; new-heading 1 turns right
      if (new-heading = 1)
      [
        rt 90
      ]

      ; new-heading 2 turns left
      if (new-heading = 2)
      [
        lt 90
      ]
    ]

    ; drives one patch ahead
    if (not full?)
    [
      fd 1

      ; increase cost for public trucks on a tick basis
          if ( member? self pubtrucks )
          [
            set cost cost + ( cost-per-hour / 60 )
          ]
    ]
end



;-------------------------------------------------------------------------------------
;; Checking and collecting from bins
to check-and-collect-waste

    ; check if there are bins in the vicinity
    let bins-around (bins in-radius 1)

    ; calculate neighbouring bins' load
    let waste-around sum [bin-load] of bins-around

    ; collect waste on the truck
    set load load + waste-around


    ; set neighbouring bins to be empty of waste
    ask bins-around
    [
      set bin-load 0
    ]

    ; increase cost for private trucks on a waste unit basis
    if ( member? self pritrucks )
    [
      set cost cost + (waste-around * cost-per-waste-unit)
    ]

end



;-------------------------------------------------------------------------------------
;; Display load of each garbage truck
to display-truck-load

    set label ""
    if (show-truck-load?)
    [
      set label load
    ]

end
```

```
;-------------------------------------------------------------------------------
;; Check current load versus max-capacity of each garbage truck
to check-capacity

    if ( member? self pritrucks )
    [
      if ( load > max-capacity-of-private-trucks )
      [
        set full? true
        set color 0
        move-to-depot
      ]
    ]

    if ( member? self pubtrucks )
    [
      if ( load > max-capacity-of-public-trucks )
      [
        set full? true
        set color 0
        move-to-depot
      ]
    ]

end


;-------------------------------------------------------------------------------
;; Move a FULL garbage truck to the depot
to move-to-depot

    ; move to upper-right patch with coordinates (16,16)
    setxy 16 16
    set heading 180

    ; unload waste
    set load 0
    set full? false

    ; restore original color
    if ( member? self pritrucks )
    [
        set color blue
    ]

    if ( member? self pubtrucks )
    [
        set color red
    ]

end




;;;;;;;;;;;;;;;;;;;;;;;;
;; Plotting Procedures ;;
;;;;;;;;;;;;;;;;;;;;;;;;


;-------------------------------------------------------------------------------
;; plot the data from this pass thru the go procedure
to do-plotting

  plot-new-value "Total Cost" "public-cost-pen" sum [ cost ] of pubtrucks
  plot-new-value "Total Cost" "private-cost-pen" sum [ cost ] of pritrucks
  plot-new-value  "Total  Waste  Collected"  "public-waste-collected-pen"  total-waste-
coll-pub
  plot-new-value  "Total  Waste  Collected"  "private-waste-collected-pen"  total-waste-
coll-pri
  plot-new-value "Total Waste Uncollected" "waste-uncollected-pen" total-waste-uncoll

end


;-------------------------------------------------------------------------------
to plot-new-value [ name-of-plot pen value ]

  set-current-plot name-of-plot
  set-current-plot-pen pen
  plot value

end
```

```
;-------------------------------------------------------------------------------
;; report basic statistics in the output area to monitor and assess the garbage
collection process
to report-kpis

  output-type "Day : " output-print ticks / 1440
  output-type "Cost per waste unit collected         [Euro]  " output-print precision
( ( sum [ cost ] of pubtrucks + sum [ cost ] of pritrucks ) / ( total-waste-coll-pub +
total-waste-coll-pri ) ) 3
  output-type "Waste units (inside bins)/(bins+trucks)   [%]   " output-print round (
100 * total-waste-uncoll / ( sum [ load ] of pubtrucks + sum [ load ] of pritrucks +
total-waste-uncoll ) )
  output-print "----------------------------"

end
```

ΠΑΡΑΡΤΗΜΑ 2. – ΕΓΧΕΙΡΙΔΙΟ ΧΡΗΣΗΣ

Καλωσορίσατε!

Το *EmptyBinCity* είναι ένα λογισμικό βασισμένο στην πλατφόρμα NetLogo και έχει σκοπό την εκτίμηση της αποδοτικότητας διαφορετικών σεναρίων αποκομιδής απορριμμάτων μίας αστικής συνοικίας. Αναπτύχθηκε στα πλαίσια εκπόνησης πτυχιακής εργασίας του Τμήματος Πληροφορικής του Α.Τ.Ε.Ι Θεσσαλονίκης από τον φοιτητή Χρήστο Τζιβανάκη με επιβλέπων καθηγητή τον Δρ. Δημοσθένη Σταμάτη.

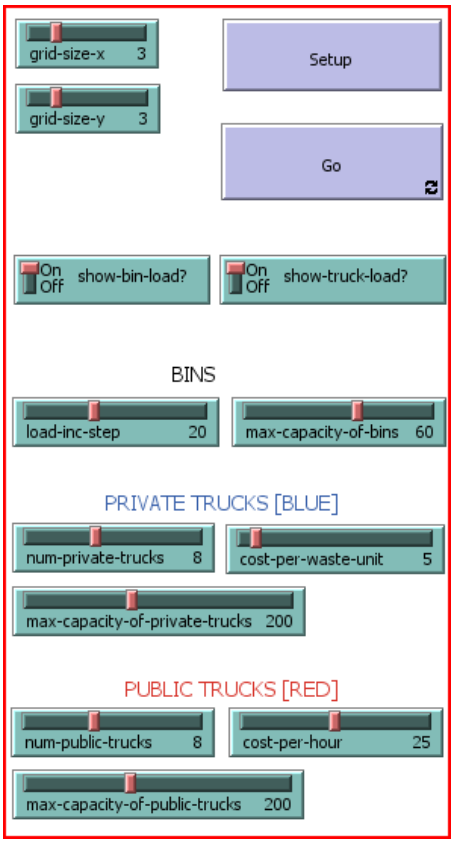
Α. Εγκατάσταση

Το *EmptyBinCity* λειτουργεί σε περιβάλλον Microsoft Windows έκδοση 7, Vista, 2000 και XP. Απαιτεί να υπάρχει εγκατεστημένο το Runtime Environment της Java 6. Για την εκτέλεση, ανοίξτε τον φάκελο με το αρχείο **EmptyBinCity.nlogo** και κάντε διπλό κλικ. Φροντίστε στον ίδιο φάκελο να υπάρχει και το αρχείο **NetLogoLite.jar**. Το λογισμικό τρέχει μέσα από τους περισσότερους Internet Browser (Mozilla Firefox, Google Chrome και Internet Explorer).

Β. Περιγραφή περιβάλλοντος εργασίας

Το περιβάλλον εργασίας του *EmptyBinCity* χωρίζεται σε τρεις περιοχές. Αριστερά βρίσκονται συγκεντρωμένα τα εργαλεία παραμετροποίησης (control panel) με τα κουμπιά ελέγχου, τους διακόπτες και τους επιλογείς αρχικών τιμών/παραμέτρων. Στο κέντρο βρίσκεται ο «κόσμος» του μοντέλου (world) που αναπαριστά μία συνοικία πόλης. Στην δεξιά πλευρά της διεπαφής βρίσκονται γραφικές παραστάσεις μεγεθών και ένα πλαίσιο κειμένου που ενημερώνονται δυναμικά με αριθμητικά μεγέθη και στατιστικά κατά τον χρόνο εκτέλεσης.

B1. Control Panel



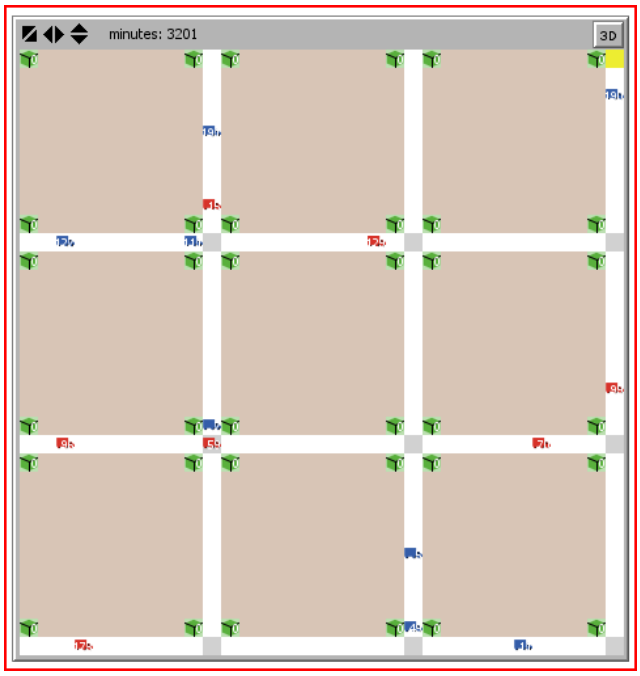
Οι διαστάσεις της **συνοικίας** (πλάτος x ύψος) ορίζονται με την βοήθεια δύο επιλογέων διαστάσεων, τους *grid-size-x* και *grid-size-y* που μπορούν να πάρουν τιμές από 1 έως 9. Οι διακόπτες *show-bin-load* και *show-truck-load* ενεργοποιούν την ένδειξη μονάδων απορριμμάτων που περιέχει αντίστοιχα κάθε κάδος ή απορριμματοφόρο και εμφανίζεται κατά την διάρκεια εκτέλεσης.

Οι παράμετροι που σχετίζονται με τους **κάδους** είναι οι επιλογείς τιμών *load-inc-step* και *max-capacity-of-bins* και ελέγχουν το εύρος βήματος αύξησης του περιεχομένου των κάδων που συμβαίνει σε ωριαία βάση και τη μέγιστη χωρητικότητα του κάδου αντίστοιχα. Και τα δύο εκφράζονται σε μονάδες απορριμμάτων.

Ακολουθούν οι παράμετροι των **απορριμματοφόρων** (**ιδιωτικά** και **δημόσια**). Για τους δύο στόλους ορίζεται ξεχωριστά το πλήθος *num-private-trucks* και *num-public-trucks* και η μέγιστη χωρητικότητα οχήματος σε μονάδες απορριμμάτων (*max-capacity-of-private-trucks* και *max-capacity-of-public-trucks*). Επιπλέον, η έννοια του κόστους στα μεν ιδιωτικά οχήματα είναι συνάρτηση των συλλεχθέντων μονάδων απορριμμάτων και μετριέται σε €/μονάδα απ/των (*cost-per-waste-unit*), ενώ στα δημόσια το κόστος είναι συνάρτηση του χρόνου λειτουργίας/κυκλοφορίας του οχήματος και μετριέται σε €/ώρα (*cost-per-hour*).

Το πλήθος του συνόλου των απορριμματοφόρων (ιδιωτικά και δημόσια) δεν πρέπει να υπερβαίνει το τετραπλάσιο του πλήθους των δρόμων κυκλοφορίας (κάθετοι και οριζόντιοι). Το αποτέλεσμα είναι να διατηρείται στην αρχική φάση σχεδιασμού του σεναρίου μία μέγιστη αναλογία 4 οχημάτων αποκομιδής ανά 1 εξυπηρετούμενη οδό για λόγους αποφυγής κυκλοφοριακής συμφόρησης. Να σημειωθεί ότι κατά την εκτέλεση τα απορριμματοφόρα κινούνται ελεύθερα σε όλους τους δρόμους χωρίς τον περιορισμό 4 προς 1.

B.2 World



Η συνοικία αποτελείται από οικοδομικά τετράγωνα με θέσεις κάδων πράσινου χρώματος στις γωνίες τους. Οι λευκοί αριθμοί στους κάδους αναπαριστούν το περιεχόμενό τους σε μονάδες απορριμμάτων.

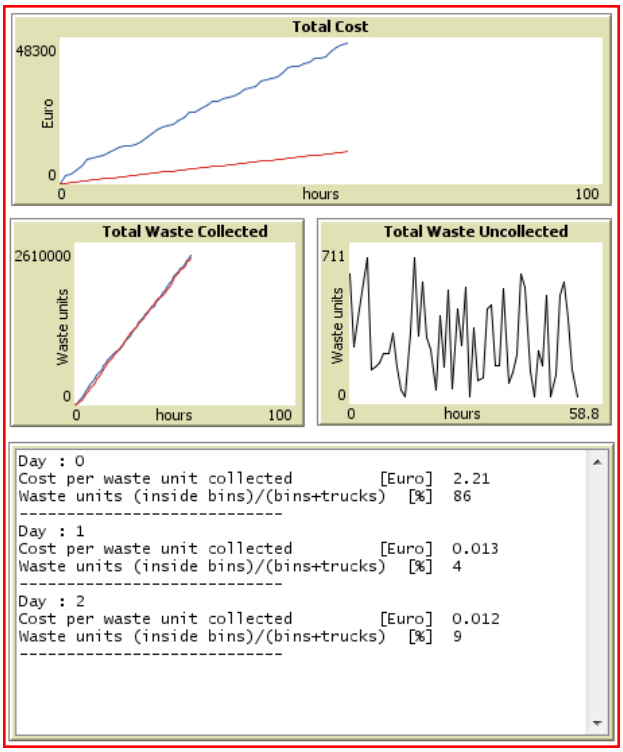
Ανάμεσα στα οικοδομικά τετράγωνα υπάρχουν δρόμοι κυκλοφορίας οχημάτων πάνω στους οποίους με γκρι χρώμα ορίζονται οι διασταυρώσεις οδών. Στο βόρειο-ανατολικό άκρο της πόλης βρίσκεται η χωματερή της πόλης με κίτρινο χρώμα.

Τα απορριμματοφόρα κινούνται προς όλες τις κατευθύνσεις πάνω στο οδικό δίκτυο και ο αριθμός που αναγράφεται πάνω τους είναι το τρέχον φορτίο τους σε μονάδες απορριμμάτων. Με μπλε χρώμα είναι ο στόλος των ιδιωτικών οχημάτων αποκομιδής, ενώ με κόκκινο των δημοσίων.

Όταν οι κάδοι ξεπερνούν την χωρητικότητά τους, το χρώμα τους γίνεται μαύρο και παύουν να δέχονται νέα απορρίμματα μέχρι να αδειάσουν από κάποιο όχημα. Παρομοίως, όταν ένα απορριμματοφόρο γεμίσει, το χρώμα του γίνεται μαύρο και μεταφέρεται στην χωματερή, όπου και αδειάζει όλο του το περιεχόμενο πριν ξεκινήσει εκ νέου για την αποκομιδή της συνοικίας.

Στο άνω περιθώριο του παραθύρου φαίνεται ο χρόνος σε λεπτά (*minutes*).

Β.3 Διαγράμματα Εξέλιξης και Αξιολόγηση Απόδοσης



Το πρώτο διάγραμμα με τίτλο **Total Cost** απεικονίζει γραφικά την πορεία του κόστους αποκομιδής και διαχωρίζει το κόστος λειτουργίας των ιδιωτικών οχημάτων (μπλε καμπύλη) από τα δημόσια (κόκκινη καμπύλη). Στο επόμενο διάγραμμα με τίτλο **Total Waste Collected** απεικονίζονται οι μονάδες απορριμμάτων που έχουν συλλεχθεί από τα απορριμματοφόρα (ιδιωτικά και δημόσια αντίστοιχα με διαφορετικό χρώμα). Το διάγραμμα με τίτλο **Total Waste Uncollected** αναπαριστά τον όγκο των μονάδων απορριμμάτων που βρίσκονται ακόμα στους κάδους και αναμένουν αποκομιδή.

Το πλαίσιο κειμένου που ακολουθεί ενημερώνεται στο τέλος κάθε 24ώρου και δίνει αναφορά για την πορεία των δύο Δεικτών Απόδοσης (Key Performance Indices) του μοντέλου. Ο πρώτος είναι ο **υπολογισμός κόστους ανά μονάδα απορριμμάτων** που έχει συλλεχθεί μέχρι στιγμής. Ο δεύτερος είναι το **ποσοστό των μονάδων απορριμμάτων που βρίσκονται ακόμα στους κάδους** σε σχέση με το άθροισμα όλων των απορριμμάτων της συνοικίας (φορτωμένοι σε απορριμματοφόρα + μη συλλεγμένοι μέσα σε κάδους).

Γ. Εκτέλεση

Αφού ολοκληρωθεί η επιλογή των παραμέτρων του μοντέλου, ο χρήστης επιλέγει στο Control Panel το κουμπί

προκειμένου να αρχικοποιηθεί το μοντέλο. Η αρχικοποίηση φαίνεται στην κεντρική οθόνη και περιλαμβάνει την εμφάνιση της συνοικίας (οικοδομικά τετράγωνα, δρόμοι κυκλοφορίας, θέσεις κάδων, χωματερή), την δημιουργία των κάδων με ένα αρχικό περιεχόμενο σε απορρίμματα και την δημιουργία και τοποθέτηση σε τυχαίες θέσεις πάνω στους δρόμους άδειων απορριμματοφόρων.

Η εκτέλεση του σεναρίου ξεκινάει επιλέγοντας στο Control Panel το κουμπί

το οποίο εμφανίζεται πατημένο

για όσο χρονικό διάστημα ο χρήστης επιθυμεί. Για παύση εκτέλεσης, αρκεί απλό πάτημα του ίδιου κουμπιού. Με τον ίδιο τρόπο μπορεί να συνεχίσει την εκτέλεση από το σημείο που ζητήθηκε παύση.

Κατά την διάρκεια της εκτέλεσης είναι δυνατό να αυξομειωθεί η ρυθμός που κυλάει ο χρόνος του μοντέλου μεταβάλλοντας τον επιλογέα που βρίσκεται στο πάνω πλαίσιο της οθόνης με τη συνοικία.

Εάν ο χρήστης αποφασίσει να δοκιμάσει νέο σενάριο με διαφορετικές παραμέτρους εκτέλεσης πρέπει να σταματήσει την εκτέλεσης πατώντας **Go**, να τροποποιήσει τις παραμέτρους που τον ενδιαφέρουν και να ξαναπατήσει **Setup** για την νέα αρχικοποίηση.