ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ – WEB INTELLIGENCE

Ευφυείς Πράκτορες ως Διαπραγματευτές για την κατάστρωση Προγράμματος Ωρών και Μαθημάτων

(Intelligent Agents as negotiators for solving the timetabling problem)

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΣΩΚΡΑΤΗ ΠΕΠΕ

Επιβλέπων : Δημοσθένης Σταμάτης
Καθηγητής, Τμήμα Μηχανικών Πληροφορικής ΑΤΕΙΘ

Θεσσαλονίκη, Μάϊος, 2015

ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ – WEB INTELLIGENCE

Ευφυείς Πράκτορες ως Διαπραγματευτές για την κατάστρωση Προγράμματος Ωρών και Μαθημάτων (Intelligent Agents as negotiators for solving the timetabling problem)

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΣΩΚΡΑΤΗ ΠΕΠΕ

Επιβλέπων : Δημοσθένης Σταμάτης
Καθηγητής, Τμήμα Μηχανικών Πληροφορικής ΑΤΕΙΘ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις Choose a date.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Δημοσθένης Σταμάτης Καθηγητής Α.Τ.Ε.Ι.Θ.	Χρήστος Ηλιούδης Αναπληρωτής Καθηγητής.Α.Τ.Ε.Ι.Θ.	Αντώνης Σιδηρόπουλος Καθηγητής Εφαρμογών Α.Τ.Ε.Ι.Θ.

Θεσσαλονίκη, Μάϊος, 2015

Σωκράτης Πεπές

.....

Click here to enter text.

Click here to enter text.

Περίληψη

Η τεχνολογία των ευφυών πρακτόρων λογισμικού αποκτά ολοένα και περισσότερους υποστηρικτές στις μέρες μας. Η χρήση λοιπόν ενός πολυπρακτορικού συστήματος, στο οποίο οι πράκτορες καθηγητών και φοιτητών συνεργάζονται και αλληλοεπιδρούν με τον πράκτορα του προγραμματιστή με σκοπό την κατασκευή ενός Προγράμματος Ωρών και Μαθημάτων (Ωρολογίου Προγράμματος) που θα ικανοποιεί όλους τους συμμετέχοντες, αποκτά ιδιαίτερο ενδιαφέρον. Σκοπός αυτής της διπλωματικής εργασίας ήταν η μελέτη της χρήσης των πολυπρακτορικών συστημάτων για τη δημιουργία τέτοιου είδους προγραμμάτων. Ως ειδική μελέτη περίπτωσης δημιουργήθηκε το ωρολόγιο πρόγραμμα για το τμήμα Μηχανικών Πληροφορικής του ΑΤΕΙ Θεσσαλονίκης. Έπειτα από στατιστική έρευνα στο τμήμα για τον τρόπο κατασκευής του προγράμματος, υλοποιήθηκαν τρεις κλάσεις πρακτόρων (προγραμματιστής, καθηγητές, φοιτητές), οι οποίοι επικοινωνούν και συνεργάζονται μεταξύ τους ανταλλάσσοντας FIPA-ACL μηνύματα. Αυτό το γεγονός πραγματοποιείται υλοποιώντας τρεις φάσεις διαπραγμάτευσης. Στις δυο πρώτες οι πράκτορες καθηγητών και σπουδαστών αποστέλλουν ελεύθερα τις προτιμήσεις τους, ενώ στην τρίτη καλούνται να επιλέξουν μέσω ενός πρωτοκόλλου πειθούς (persuasion protocol) που χρησιμοποιεί ο προγραμματιστής από μια λίστα με πιθανές λύσεις του προβλήματος. Η διαδικασία υλοποιήθηκε σε γλώσσα προγραμματισμού Java, με χρήση της πλατφόρμας JADE και του περιβάλλοντος eclipse. Στα πλαίσια της διπλωματικής αναδείχθηκε η χρησιμότητας της τεχνολογίας πρακτόρων και της διαπραγμάτευσης μεταξύ αυτών για την επίλυση σύνθετων προβλημάτων. Μετά την υλοποίηση της εφαρμογής αναδείχθηκαν τα πλεονεκτήματα της τεχνικής αυτής σε σχέση με άλλες προγενέστερες τεχνικές.

Λέξεις Κλειδιά: Ωρολόγιο πρόγραμμα, Πράκτορες, Πολυπρακτορικά Συστήματα, Διαπραγμάτευση, JADE

Abstract

Intelligent Agents technology is gaining a lot of supporters nowadays. Therefore, using a multi-agent system, in which Teacher and Student Agents cooperate and interact with the Programmer Agent to generate a timetable program that would satisfy all participants, is extremely interesting. The aim of this thesis was to study the use of Multi-Agent Systems in solving the Timetabling Problem.As a case study we created the timetable program for the Department of Information Technology, of ATEI Thessaloniki. After a statistical invastigation in the department, we implemented three Agent classes (Programmer Agent, Teachers Agents and Students Agents) who communicate and cooperate by exchanging FIPA-ACL messages. The system developed is based on three negotiation phases. In the first two phases, the Teachers and Students Agents send their preferences freely, while in the third phase the Teacher Agents are advised to choose timeslots through a persuasion protocol, which is used by the Programmer Agent from a list of possible solutions to the problem. The whole process is implemented in the programming language Java, using the JADE platform and the environment of eclipse. In the context of the thesis we showed the usefulness of the Agents technology and the negotiation between them to solve such complex problems as that of the timetabling. The evaluation of the application developed showed clearly the advandages of this technique over other previous techniques.

Keywords: Timetable program, Agents, Multi-Agent System, Negotiation, JADE

Ευχαριστίες

Αισθάνομαι την ανάγκη να ευχαριστήσω αρχικά τον επιβλέποντα Καθηγητή μου κ. Δημοσθένη Σταμάτη που με ενέπνευσε και μου κίνησε το ενδιαφέρον να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα, όπως είναι οι Ευφυείς Πράκτορες. Πραγματικά νοιώθω ότι έχω δικαιωθεί με την επιλογή του θέματος της Διπλωματικής μου. Η συνεργασία μας όλο αυτό τον καιρό υπήρξε άψογη, καθώς με ενθάρρυνε και μου δημιουργούσε κίνητρα για να προχωρήσω την έρευνα ακόμη περισσότερο.

Θα ήθελα επίσης να ευχαριστήσω τον Αναπληρωτή Καθηγητή, κ. Ηλιούδη Χρήστο και τον Καθηγητή Εφαρμογών κ. Σιδηρόπουλο Αντώνιο, οι οποίοι αποτελούν την εξεταστική επιτροπή που μελέτησε και αξιολόγησε αυτή τη διπλωματική εργασία. Η βοήθεια τους υπήρξε πολύτιμη.

Ένα ιδιαίτερο ευχαριστώ αξίζει στον συνάδελφο-συμφοιτητή μου κ. Καλπαξίδη Ηλία για την άψογη συνεργασία που είχαμε σε όλη την διάρκεια των σπουδών μας, καθώς επίσης και κατά τη διάρκεια της διπλωματικής, λόγω του κοινού θέματος που είχαμε επιλέξει. Ιδιαίτερη μνεία αξίζει επίσης και στη συνάδελφο-συμφοιτήτρια Ελισάβετ Γρηγορίου για όλη τη βοήθεια και τη στήριξη που μου παρείχε σε όλη τη διάρκεια του μεταπτυχιακού. Με ανιδιοτέλεια, ήτανε πάντα πρόθυμη να σταθεί δίπλα μου και να με βοηθήσει-συμβουλέψει σε οτιδήποτε χρειάστηκα. Αισθάνομαι πραγματικά πως έχω κερδίσει δυο καλούς φίλους.

Θα ήθελα στη συνέχεια να ευχαριστήσω θερμά τους συναδέλφους-συμφοιτητές, Κοτρότσιο Κωνσταντίνο, Τσελεμέγκο Γρηγόρη και Τσιτσιρούδη Νίκη για όλη τη βοήθεια που μου προσέφεραν σε ότι πρόβλημα αντιμετώπισα στο θέμα της υλοποίησης του κώδικα της διπλωματικής. Η βοήθεια τους ήταν πολύτιμη,Ένα μεγάλο ευχαριστώ και σε όλα τα υπόλοιπα παιδιά του μεταπτυχιακού, με τα οποία περάσαμε αξέχαστες στιγμές και οι αναμνήσεις θα μας συντροφεύουν για πάντα.

Τέλος, ίσως και το σημαντικότερο, θα ήθελα να ευχαριστήσω την οικογένειά μου. Τους γονείς μου που ήταν πάντοτε δίπλα μου, να με αγαπούν, να με ανέχονται και να με συμβουλεύουν σε οτιδήποτε έχω επιχειρήσει στη ζωή μου. Εύχομαι να τους έχω δικαιώσει και να είναι πάντοτε υγιείς και δυνατοί στο πλευρό μου. Ευχαριστώ πολύ την αδελφή μου και τα ανίψια μου για όλη την αγάπη και τη στήριξη που μου προσφέρουν απλόχερα όλα αυτά τα χρόνια.

Η διπλωματική εργασία είναι εξαιρετικά αφιερωμένη στην οικογένειά μου και σε όλους τους ανθρώπους που με αγαπούν και μου συμπαραστέκονται.

Πίνακας περιεχομένων

Περιεχόμενα

Πίνακας Εικόνων

Πίνακας Πινάκων

1. Εισαγωγή

1.1 Ευφυείς πράκτορες ως διαπραγματευτές για την κατάστρωση προγράμματος ωρών και μαθημάτων

Η δημιουργία του ωρολόγιού προγράμματος για Ανώτατα Εκπαιδευτικά Ιδρύματα αποτελεί ένα συνδυαστικό πρόβλημα βελτιστοποίησης NP-hard (Non-deterministc Polynomial-time hard) τάξης. Αυτό το πρόβλημα, το οποίο εμφανίζεται στην αρχή κάθε ακαδημαϊκού εξαμήνου περιλαμβάνει την ανάθεση μιας σειράς πόρων (μαθημάτων, καθηγητών, φοιτητών) σε έναν προκαθορισμένο αριθμό χρονοθυρίδων (timeslots) και σε ένα σταθερό αριθμό αιθουσών. Προκειμένου να υπάρξει μια ορθή και αποτελεσματική λύση του προβλήματος, θα πρέπει αυτό να λαμβάνει υπόψιν του και να ικανοποιεί μια σειρά από περιορισμούς (constraints). Οι περιορισμοί αυτοί μπορεί να είναι είτε ισχυροί (hard constrains), είτε χαλαροί (soft constrains). Οι ισχυροί περιορισμοί θα πρέπει να ικανοποιηθούν στο σύνολο τους, γιατί η μη τήρηση τους ενδέχεται να οδηγήσει σε μια μη αποδεκτή λύση του προβλήματος. Μέσω των χαλαρών περιορισμών μπορούμε να αυξήσουμε την ικανοποίηση των συμμετεχόντων πλευρών και να μετρήσουμε το βαθμό ποιότητας του προγράμματος που δημιουργείται (Yang et. al., 2011).

Οι πράκτορες (Agents) είναι πολύ δημοφιλείς στις μέρες μας για την επίλυση σύνθετων προβλημάτων, όπως αυτό της δημιουργίας του ωρολόγιού προγράμματος. Ένας πράκτορας μπορεί να οριστεί ως «Οτιδήποτε αντιλαμβάνεται το περιβάλλον του μέσω αισθητήρων (sensors) και ενεργεί σε αυτό μέσω ενεργοποιητών (actuators)». Ένα σύστημα πολλαπλών πρακτόρων είναι απλά ένα σύστημα που περιέχει περισσότερους από έναν πράκτορες. Σε ένα πολυπρακτορικό σύστημα (Multi Agent System/MAS), η ομάδα των πρακτόρων που το αποτελεί, αλληλοεπιδρά μεταξύ τους για την επίτευξη των στόχων για τους οποίους έχουν σχεδιαστεί. Οι επιμέρους στόχοι του κάθε πράκτορα ενδέχεται να συμφωνούν ή να διαφωνούν με τους γενικούς στόχους της ομάδας συνολικά.

Τα οφέλη της χρήσης των πολυπρακτορικών συστημάτων έχουν αποδειχτεί από την πληθώρα των εφαρμογών που τα έχουν χρησιμοποιήσει. Μεταξύ άλλων έχουν αναπτυχθεί βιομηχανικές εφαρμογές σε τομείς που περιλαμβάνουν τον έλεγχο της διαδικασίας και της μεταποίησης. Ένα παράδειγμα είναι το Σύστημα Λογικής Διασύνδεσης (Procedural

Reasoning System, PRS), το οποίο χρησιμοποιείται για το σύστημα OASIS που διαχειρίζεται την εναέρια κυκλοφορία στο αεροδρόμιο του Σίδνεϋ. Αποτελεί τη βάση του Swarmm , δηλαδή ενός πολυπρακτορικού συστήματος προσομοίωσης για τον Οργανισμό Αμυντικής Επιστήμης και Τεχνολογίας της Αυστραλίας (D'Inverno et. al., 2004). Χρησιμοποιείται επίσης σε πληθώρα επιχειρήσεων λογισμικού για τη λειτουργία τηλεφωνικών κέντρων ή υπηρεσιών διαδικτύου. Το MAS έχει εφαρμοστεί επίσης και σε πολλούς άλλους τομείς, όπως παρακολούθηση πολλαπλών ρομπότ, διασυνδεδεμένα ηλεκτρικά οχήματα, εφαρμογές κινητής τηλεφωνίας και πολλούς άλλους.

Τα πλεονεκτήματα και τα μειονεκτήματα της χρήσης MAS, είναι παρόμοια με εκείνα της αντικειμενοστραφούς σχεδίασης και ανάλυσης (Object-Oriented Design and Analysis, OODA) και του αντικειμενοστραφούς προγραμματισμού (Object-Oriented Programming, OOP) στην ανάπτυξη λογισμικού. Η χρήση των πρακτόρων επιτρέπει την ένα-προς-ένα χαρτογράφηση του πραγματικού κόσμου του χρήστη με τη χρήση ενός λογισμικού το οποίο μπορεί να επιταχύνει τον κύκλο σχεδίασης του λογισμικού (Tan et. al., 2012).

1.2 Αντικείμενο διπλωματικής

Η δημιουργία του Ωρολογίου Προγράμματος είναι ένα από τα γνωστά προβλήματα χρονοπρογραμματισμού, το οποίο μπορεί να περιγραφεί ως η κατανομή των διαθέσιμων πόρων σε όλους τους επιμέρους παράγοντες που το αποτελούν, λαμβάνοντας υπόψιν κάποιους προκαθορισμένους περιορισμούς, έτσι ώστε να μεγιστοποιηθεί η πιθανότητα ορθής κατανομής των πόρων και να ελαχιστοποιηθεί η παραβίαση των περιορισμών που έχουν τεθεί (Yang et. al., 2011).

Σε σύγκριση με όλα τα υπόλοιπα γνωστά προβλήματα χρονοπρογραμματισμού, αυτό για τη δημιουργία ωρολογίου προγράμματος για εκπαιδευτικούς σκοπούς, είναι αυτό το οποίο έχει μελετηθεί περισσότερο και έχει μεγάλη επίπτωση σε όλα τα ενδιαφερόμενα μέρη (καθηγητές, φοιτητές, διοικητικούς υπαλλήλους). Το πρόβλημα αυτό μπορεί να οριστεί ως η ανάθεση ενός συνόλου μαθημάτων M = {M1, M2, ...}, σε έναν περιορισμένο αριθμό χρονοθυρίδων (χρονικά διαστήματα) X = {X1, X2, ...} και σε έναν προκαθορισμένο αριθμό αιθουσών A={A1,A2,…} . Επίσης όλα τα μαθήματα θα πρέπει να αντιστοιχιστούν σε μια σειρά καθηγητών K={K1,K2,…}, και σε μια σειρά φοιτητών Φ={Φ1,Φ2,…} . Το σύνολο όλων αυτών των πόρων υπόκειται σε ένα σύνολο περιορισμών. Οι περιορισμοί που τίθενται κατηγοριοποιούνται συνήθως σε Ισχυρούς περιορισμούς (Hard constrains), δηλαδή περιορισμούς που δεν θα πρέπει να παραβιαστούν κάτω από οποιεσδήποτε συνθήκες και σε Χαλαρούς περιορισμούς (Soft constrains), δηλαδή τις ιδιαίτερες προτιμήσεις και επιθυμίες των ενδιαφερόμενων μερών.

Η πολυπλοκότητα και η πρόκληση που παρουσίασαν τα προβλήματα αυτά, προκύπτει από το γεγονός ότι υπάρχει μια μεγάλη ποικιλία περιορισμών, μερικοί από τους οποίους έρχονται σε αντίθεση (σύγκρουση) μεταξύ τους.

Κατά τη διάρκεια των χρόνων έχουν προταθεί πολλές μέθοδοι για τη επίλυση του προβλήματος, όπως η *θεωρία των γράφων* (theory of graphs), *ο ακέραιος γραμμικός προγραμματισμός* (integer linear programming), τεχνικές *ικανοποίησης περιορισμών* (constraint satisfaction), καθώς επίσης και μια σειρά από *μεταευριστικές* (metaheuristic) μεθόδους, όπως η simulated annealing, η tabu search και διάφοροι γενετικοί αλγόριθμοι (genetic algorithms). Μεγαλύτερο ενδιαφέρον παρουσιάζει όμως η επίλυση τους με τη χρήση πρακτόρων λογισμικού (Software Agents) λόγω της δυνατότητας προσομοίωσης με πραγματικές συνθήκες των εμπλεκόμενων και της διαπραγμάτευσης μεταξύ αυτών για την αποτελεσματική αντιμετώπιση του προβλήματος.

Ένας πράκτορας είναι ένα σύστημα υπολογιστή, το οποίο βρίσκεται σε κάποιο περιβάλλον και είναι ικανό να επιτελέσει αυτόνομη δράση σε αυτό, προκειμένου να επιτύχει τους στόχους για τους οποίους έχει σχεδιαστεί. Η αυτονομία χρησιμοποιείται για να εκφράσει το γεγονός ότι οι πράκτορες είναι σε θέση να ενεργήσουν (να εκτελέσουν ενέργειες) χωρίς την παρέμβαση του ανθρώπου ή άλλου συστήματος. Μερικά σημαντικά χαρακτηριστικά των πρακτόρων είναι η προσαρμοστικότητα στις αλλαγές του περιβάλλοντος και τη συνεργασία τους με άλλους πράκτορες. Οι πράκτορες αλληλοεπιδρούν προκειμένου να ενταχθούν σε πιο σύνθετες ομάδες και να συγκροτήσουν συστήματα πολλαπλών πρακτόρων ή όπως αλλιώς λέγονται πολυπρακτορικά συστήματα (Multi Agent Systems, MAS). Ένας ευφυής πράκτορας είναι ικανός να επιτελεί ευέλικτη αυτόνομη δράση προκειμένου να επιτύχει τους στόχους για τους οποίους σχεδιάστηκε. Η ελαστικότητα σημαίνει τρία πράγματα: προνοητικότητα (συμπεριφορά σύμφωνα με τους στόχους του), αντιδραστικότητα (ανταπόκριση στις αλλαγές), και κοινωνική ικανότητα (αλληλεπίδραση με άλλους παράγοντες) (Neruda et. al., 2012)

Τα πολυπρακτορικά συστήματα απαιτούν από τους πράκτορες που τα αποτελούν να συμπεριφέρονται με ένα συγκεκριμένο τρόπο. . Η ομάδα των πρακτόρων αλληλοεπιδρούν μεταξύ τους, προκειμένου να επιτύχουν τους επιμέρους στόχους για τους οποίους έχουν σχεδιαστεί. Μια σημαντική δυνατότητα, η οποία βοηθάει τη συνεργασία μεταξύ των πρακτόρων είναι η διαπραγμάτευση. Σε ένα σύστημα, διαπραγμάτευση είναι η αμοιβαία επικοινωνία και ο συντονισμός μεταξύ των πρακτόρων, προκειμένου αυτοί να καταλήξουν σε ένα συμπέρασμα το οποίο ικανοποιεί όλα τα επιμέρους συμφέροντα των πρακτόρων (Chen 2014).

Στα πλαίσια αυτής της διπλωματικής αναπτύχθηκε ένα πολυπρακτορικό σύστημα στο περιβάλλον της Jade για τη δημιουργία του ωρολογίου προγράμματος μαθημάτων του

τμήματος Μηχανικών Πληροφορικής του Αλεξάνδρειου Τεχνολογικού Ιδρύματος Θεσσαλονίκης. Για το σκοπό αυτό έχει τεθεί ένας αριθμός ισχυρών και χαλαρών περιορισμών και έχουν αναπτυχθεί τρείς διαφορετικές κλάσεις πρακτόρων. Ο ProgramAgent για τη δημιουργία και τον συντονισμό του προγράμματος, οι TeacherAgents, οι οποίοι είναι οι πράκτορες των καθηγητών και οι StudentAgents, οι οποίοι είναι οι πράκτορες των σπουδαστών. Οι πράκτορες αυτοί αλληλοεπιδρούν μεταξύ τους, ανταλλάσσοντας FIPA-ACL μηνύματα (περισσότερες πληροφορίες στο κεφάλαιο 4), συνεργάζονται και διαπραγματεύονται με σκοπό τη δημιουργία ενός προγράμματος μαθημάτων, το οποίο θα ικανοποιεί όλα τα συμβαλλόμενα μέρη. Η διαπραγμάτευση των πρακτόρων αποτελείται από τρία διαφορετικά διακριτά στάδια. Στα δυο πρώτα οι πράκτορες των καθηγητών και των σπουδαστών αποστέλλουν ελεύθερα τις προτιμήσεις τους, και στο τρίτο αν δεν έχει καταστεί δυνατή η δημιουργία του κοινά αποδεκτού τελικού προγράμματος, ο ProgramAgent χρησιμοποιεί ένα πρωτόκολλό πειθούς (persuasion protocol) προκειμένου να προτείνει χρονοθυρίδες για την τελική επίλυση του προβλήματος.

1.2.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Μελετήσαμε το πρόβλημα της δημιουργία ωρολογίου προγράμματος.
2. Μελετήσαμε μεθοδολογίες επίλυσης του ωρολογίου προγράμματος.
3. Μελετήσαμε την τεχνολογία των πρακτόρων και των πολυπρακτορικών συστημάτων.
4. Μελετήσαμε το πρότυπο της FIPA και την πλατφόρμας της JADE.
5. Πραγματοποιήσαμε στατιστική έρευνα με τη χρήση ερωτηματολογίων για την ανίχνευση των προτιμήσεων καθηγητών και σπουδαστών όσον αφορά τον τρόπο δημιουργίας του ωρολογίου προγράμματος.
6. Επεξεργαστήκαμε και αξιολογήσαμε τα αποτελέσματα που προέκυψαν από τα ερωτηματολόγια με τη βοήθεια του SPSS.
7. Υλοποιήσαμε ένα πολυπρακτορικό σύστημα στο περιβάλλον της JADE για τη δημιουργία του ωρολογίου προγράμματος μαθημάτων σε Ανώτατο Εκπαιδευτικό Ίδρυμα.
8. Προτείναμε και υλοποιήσαμε ένα πρωτόκολλο διαπραγμάτευσης το οποίο βασίζεται σε τρεις διακριτές φάσεις διαπραγμάτευσης μεταξύ των πρακτόρων.
9. Αξιολογήσαμε το σύστημα με τη βοήθεια μιας συνάρτησης αξιολόγησης και του Τεστ Ευχρηστίας (Usability Testing).
10. Αναφέραμε μελλοντικές επεκτάσεις για βελτίωση της εφαρμογής.

1.3 Οργάνωση κειμένου

Η διπλωματική αυτή έχει οργανωθεί ως εξής:

Στο κεφάλαιο 2 περιγράφεται το πρόβλημα δημιουργίας του ωρολογίου προγράμματος. Το κεφάλαιο 3 αναφέρεται σε σχετικές εργασίες και υλοποιήσεις για την επίλυση του προβλήματος. Στο κεφάλαιο 4 υπάρχει το θεωρητικό υπόβαθρο που απαιτείται για τη επίλυση του προβλήματος με τη χρήση πολλαπλών πρακτόρων. Στο κεφάλαιο 5 γίνεται η ανάλυση των απαιτήσεων της εφαρμογής που θα αναπτυχθεί και στο κεφάλαιο 6 περιγράφεται η υλοποίηση της. Στο κεφάλαιο 7 γίνεται η αξιολόγηση του συστήματος που αναπτύχθηκε και τέλος στο κεφάλαιο 8 αναφέρονται τα συμπεράσματα και οι μελλοντικές επεκτάσεις.

2. Δημιουργία Ωρολογίου Προγράμματος

Το πρόβλημα της δημιουργίας ωρολογίων προγραμμάτων απασχολεί εδώ και πάρα πολλά χρόνια τους επιστήμονες σε διάφορους τομείς. Η εφαρμογή του σε νοσοκομεία, τρένα, σχολεία, πανεπιστήμια και σε διάφορους άλλους τομείς αποτελεί αντικείμενο μελάτης. Το πρόβλημα δημιουργίας ωρολογίου προγράμματος μαθημάτων είναι αυτό που μας απασχολεί στα πλαίσια αυτής της διπλωματικής.

Στην απλούστερη μορφή του το πρόβλημα έγκειται στην κατασκευή ενός προγράμματος για κάθε καθηγητή και για κάθε τμήμα, το οποίο θα ικανοποιεί τις απαιτήσεις των καθηγητών και ταυτόχρονα θα υπακούει σε ορισμένους βασικούς περιορισμούς, όπως είναι «Κανένας καθηγητής, όπως και κανένας φοιτητής δεν μπορεί να είναι σε δυο αίθουσες ταυτόχρονα». Οι περιορισμοί του καθηγητή μπορούν να οριστούν ως ο αριθμός των χρονικών περιόδων, στις οποίες ο καθηγητής είναι διαθέσιμος για να συναντηθεί με την τάξη. Ως ένα χαρακτηριστικό παράδειγμα θα μπορούσαμε να αναφέρουμε τη δημιουργία ενός προγράμματος για ένα Σχολείο, το οποίο αποτελείται από τέσσερις καθηγητές, τέσσερα τμήματα και τα μαθήματα πραγματοποιούνται κατά τη διάρκεια τεσσάρων περιόδων κατά τη διάρκεια μιας εβδομάδας. Κάθε καθηγητής καλείται να διδάξει σε κάθε τάξη για μια χρονική περίοδο μέσα στην εβδομάδα. Είναι κατανοητό λοιπόν, ότι οι καθηγητές και οι τάξεις θα πρέπει να αντιστοιχηθούν πλήρως. Μια από τις πολλές διαθέσιμες λύσεις του προβλήματος παρουσιάζεται στην παρακάτω εικόνα.

	PERIODS			
CLASS	1	2	3	4
1	1	2	3	4
2	4	3	2	1
3	2	1	4	3
4	3	4	1	2

Εικόνα 1.Εβδομαδιαίο πρόγραμμα μαθημάτων Σχολείου (Barraclough, 1965)

Το ίδιο απλό πρόβλημα μπορεί να χρησιμοποιηθεί, προκειμένου να καταλάβουμε τις δυσκολίες που ενδέχεται να εμφανιστούν κατά τη δημιουργία του προγράμματος. Ας υποθέσουμε πως δημιουργείται το πρόγραμμα και σχηματίζεται η λύση της ακόλουθης εικόνας.

	PERIODS			
CLASS	1	2	3	4
1	1	2	3	4
2	2	3	4	1
3		4	1	3
4	3	1	2	

Εικόνα 2.Εβδομαδιαίο πρόγραμμα μαθημάτων, το οποίο είναι αδύνατο να ολοκληρωθεί (Barraclough, 1965)

Είναι προφανές ότι δεν είναι δυνατόν να γεμίσουν τα εναπομείναντα κενά, χωρίς να υπάρχει κάποια σύγκρουση. Στην πραγματικότητα βέβαια, για τη δημιουργία του ωρολογίου προγράμματος μαθημάτων σε ένα κανονικό Σχολείο, οι συνολικές απαιτήσεις δεν περιορίζονται στις επιθυμίες ενός και μόνο καθηγητή (Barraclough, 1965).

Μια δυσκολία στη διαδικασία δημιουργίας του προγράμματος προκύπτει από το γεγονός ότι κατά τη διάρκεια μιας χρονιάς, περισσότεροι από ένας καθηγητές διδάσκουν σε ένα τμήμα, τα μαθήματα σε όλα τα τμήματα πραγματοποιούνται ταυτόχρονα και επίσης ότι δεν πρέπει κάποιο τμήμα να έχει κενό κατά τη διάρκεια μιας ημέρας. Λαμβάνοντας υπόψιν και τις τυχόν απαιτήσεις των καθηγητών, η δυσκολία δημιουργίας του προγράμματος συνεχώς μεγαλώνει και ενδέχεται σε ορισμένες περιπτώσεις να είναι και αδύνατο να βρεθεί μια κοινά αποδεκτή λύση.

2.1 Ωρολόγιο πρόγραμμα μαθημάτων για Ανώτατα Εκπαιδευτικά Ιδρύματα

Το πρόβλημα δημιουργίας ωρολόγιού προγράμματος μαθημάτων για Πανεπιστημιακά Ιδρύματα είναι αυτό το οποίο έχει μελετηθεί ευρέως και από θεωρητική και από πρακτική σκοπιά. Είναι μια από τις πιο σημαντικές και χρονοβόρες διαδικασίες, οι οποίες απασχολούν ανά τακτά χρονικά διαστήματα (Τρίμηνο, Εξάμηνο, Έτος) όλα τα Πανεπιστημιακά Ιδρύματα στον κόσμο. Η ποιότητα του ωρολόγιού προγράμματος έχει μεγάλη επίπτωση σε ένα ευρύ φάσμα δραστηριοτήτων όλων των εμπλεκόμενων πλευρών, συμπεριλαμβανομένου των καθηγητών, των φοιτητών και των διοικητικών υπαλλήλων (Qu et. al., 2009).

Το ωρολόγιο πρόγραμμα μαθημάτων είναι μια ιδιαίτερη περίπτωση του γενικότερου προβλήματος δημιουργίας προγραμμάτων. Στην απλούστερη μορφή του είναι ένα πρόβλημα προγραμματισμού μιας σειράς εκδηλώσεων (διαλέξεις, σεμινάρια, εργαστήρια) σε ένα σύνολο αιθουσών διδασκαλίας και σε σύνολο χρονοθυρίδων, τα οποία διδάσκονται από ένα σύνολο καθηγητών και τα παρακολουθούν ένα σύνολο φοιτητών. Οι εκδηλώσεις θα πρέπει να είναι οργανωμένες με τέτοιο τρόπο ώστε κανένας φοιτητής ή καθηγητής να μην είναι σε περισσότερες από μια αίθουσες ταυτόχρονα και επίσης να υπάρχει αρκετός χώρος σε κάθε

αίθουσα για τον αριθμό των φοιτητών που παρακολουθούν την εκδήλωση. Επίσης θα πρέπει να μην συμβαίνουν δυο γεγονότα στην ίδια αίθουσα ταυτόχρονα (Malim et.al., 2006).

Η δομή των προσφερόμενων μαθημάτων ενδέχεται να είναι αυτή που εμφανίζει τα περισσότερα προβλήματα. Παρόλο που το πρόβλημα που αντιμετωπίζουμε ονομάζεται «Δημιουργία ωρολόγιού προγράμματος μαθημάτων», στην πραγματικότητα οι επιμέρους συναντήσεις (μαθήματα) είναι αυτές που δημιουργούν το μάθημα. Ένα μάθημα είναι ένα σύνολο από επιμέρους συναντήσεις μιας ομάδας φοιτητών, οι οποίοι έχουν εγγραφεί στο συγκεκριμένο μάθημα, το οποίο διδάσκεται από έναν καθηγητή, και οι οποίες πραγματοποιούνται σε μια αίθουσα, σε προκαθορισμένα χρονικά διαστήματα μέσα σε μια εβδομάδα. Τα μαθήματα συνήθως αποτελούνται από έναν αριθμό εβδομαδιαίων συναντήσεων, είτε διαλέξεων, είτε εργαστηρίων, είτε σεμιναρίων. Οι μαθητές εγγράφονται σε μια σειρά μαθημάτων σύμφωνα με τις ανάγκες του ακαδημαϊκού τους προγράμματος.

Η δημιουργία του προγράμματος γίνεται δημιουργώντας μια λογική δομή ανάμεσα στα δεδομένα τα οποία αντιπροσωπεύουν όλα τα επιμέρους κομμάτια του μαθήματος (καθηγητές, μαθητές, τάξεις, χρονικά διαστήματα) καθώς και τις σχέσεις μεταξύ αυτών και του υπόλοιπου συνόλου των μαθημάτων, λαμβάνοντας υπόψιν τους τυχόν περιορισμούς που τίθενται. Πολλά μαθήματα αποτελούνται από διάφορες ορισμένες υποενότητες, όπως σεμινάρια ή εργαστήρια και οι οποίες συνδέονται με το συγκεκριμένο μάθημα. Το ίδιο μάθημα μπορεί να αποτελείται από διαφορετικές υποενότητες ανάλογα με τον καθηγητή. Για παράδειγμα κάποιος καθηγητής επιθυμεί κάποιες ώρες στο εργαστήριο, ενώ κάποιος άλλος όχι.

Το πρόβλημα δημιουργίας ωρολόγιού προγράμματος μαθημάτων μπορεί να θεωρηθεί ότι αποτελείται από τρία υποπροβλήματα. Την ανάθεση μαθημάτων σε καθηγητές, την ανάθεση γεγονότων σε χρονοθυρίδες και την ανάθεση αιθουσών σε γεγονότα. Οι καθηγητές αναλαμβάνουν μια σειρά εκδηλώσεων για κάθε μάθημα. Ένα μάθημα μπορεί να το διδάσκουν περισσότεροι από ένας καθηγητές. Στη συνέχεια γίνεται ανάθεση προκαθορισμένων χρονοθυρίδων σε εβδομαδιαία βάση για το σύνολο των εκδηλώσεων του μαθήματος (διαλέξεις και εργαστήρια). Έπεται η εκχώρηση αιθουσών σε όλα τα γεγονότα του μαθήματος. Και τέλος οι φοιτητές επιλέγουν τα μαθήματα, τα οποία θα παρακολουθήσουν. Ως εκ τούτου, το πρόβλημα δημιουργίας ωρολόγιού προγράμματος μαθημάτων είναι ένα πρόβλημα εκχώρησης τιμών σε μια προκαθορισμένη τετράδα (a,b,c,d), όπου a ανήκει στα γεγονότα (events), b ανήκει στις χρονοθυρίδες (timeslots), c ανήκει στις αίθουσες (rooms) και d ανήκει στους καθηγητές (professors). Ένα γεγονός a ξεκινά στη χρονοθυρίδα b και διδάσκεται στην αίθουσα c από τον καθηγητή d (Malim et.al., 2006). Το πρόβλημα έγκειται στην αντιστοίχιση χρόνου και πηγών για την δημιουργία συναντήσεων και την ταυτόχρονη ικανοποίηση όσο το δυνατόν περισσότερων περιορισμών (Nouri et. al., 2013).

Πρόκειται για ένα πρόβλημα που έχει απασχολήσει τους ερευνητές εδώ και πάρα πολλά χρόνια. Κατά καιρούς έχουν προταθεί διάφοροι μέθοδοι αντιμετώπισης. Ένα μεγάλο μέρος των εργασιών σε αυτό τον τομέα έχει γίνει με τη χρήση τεχνητών συνόλων δεδομένων ή βασίζονται σε πραγματικά προβλήματα, τα οποία έχουν σε μεγάλο βαθμό απλοποιηθεί. Οι κύριες διαφορές ανάμεσα στα προβλήματα που μελετήθηκαν και στις πραγματικές συνθήκες είναι η πρόσθετη πολυπλοκότητα που επιβάλλεται από τη δομή των προγραμμάτων, την ποικιλία των περιορισμών καθώς και την κατανεμημένη ευθύνη για τη συλλογή των πληροφοριών που απαιτούνται για την επίλυση αυτών των προβλημάτων σε πανεπιστημιακό επίπεδο. Το πρόβλημα δημιουργίας ωρολογίου προγράμματος μαθημάτων για ένα πανεπιστήμιο ενδέχεται επίσης να απαιτεί την επίλυση πολλαπλών υποπροβλημάτων με διαφορετικά χαρακτηριστικά το καθένα. Ως εκ τούτου η διαδικασία επίλυσης δεν θα πρέπει να είναι προσαρμοσμένη σε ένα μόνο τύπο προβλήματος. Το πλήρες πρόβλημα αποσυντίθεται σε μια σειρά υποπροβλημάτων, τα οποία πρέπει να επιλυθούν σε επίπεδο ακαδημαϊκού τμήματος και στα οποία ελέγχονται η πόροι που απαιτούνται για την αποδοτικότερη επίλυση του. Αρκετά άλλα προβλήματα, όπως ο διαμοιρασμός των πόρων ή η αλληλεπίδραση των φοιτητών είναι καίριας σημασίας. Μια σημαντική παράμετρος για τον σχεδιασμό του συστήματος είναι η υποστήριξη της κατανεμημένης κατασκευής των προγραμμάτων για κάθε τμήμα, παράλληλα με τον κεντρικό συντονισμό του συνολικού προβλήματος. Αυτό αντανακλά στην κατανεμημένη διαχείριση των εκπαιδευτικών πόρων για το κάθε τμήμα του πανεπιστημίου.

Η δημιουργία του προγράμματος είναι ένα πρόβλημα κατανομής των πόρων. Ως εκ τούτου στα περισσότερα πανεπιστημιακά ιδρύματα η ευθύνη της κατασκευής του ωρολογίου προγράμματος κατανέμεται στα επιμέρους τμήματα, λαμβάνοντας υπόψιν τις κτηριακές και φυσικές εγκαταστάσεις, όπως επίσης και όλους τους άλλους πόρους που απαιτούνται για την δημιουργία του. Η παροχή υποστήριξης στην κατανεμημένη αυτή εργασία είναι πολύ σημαντική γιατί η δημιουργία του ωρολογίου προγράμματος για το κάθε τμήμα ξεχωριστά , δίνει μεγαλύτερο έλεγχο από ότι η συνολική και κεντρική δημιουργία των προγραμμάτων, με δεδομένο ότι το κάθε τμήμα έχει καλύτερη γνώση των αναγκών και των μαθημάτων που υπάρχουν και μεγαλύτερη γνώση των εγκαταστάσεων που διαθέτει για τη διεξαγωγή κάθε μαθήματος. Η επιλογή μαθημάτων από τους φοιτητές, τα οποία ανήκουν σε συγκεκριμένο ακαδημαϊκό έτος, ή η ύπαρξη προαπαιτούμενων μαθημάτων για την επιλογή κάποιου μαθήματος ενδέχεται να δημιουργήσει πολλαπλές συγκρούσεις. Το πρόβλημα όμως γίνεται ακόμα πιο πολύπλοκο αν ο μαθητής πρέπει να επιλέξει μαθήματα από πολλαπλές ακαδημαϊκές μονάδες. Τότε απαιτείται πρόσθετος συντονισμός για την υλοποίηση του προγράμματος (Tan et. al., 2012).

2.2 Περιορισμοί

Προκειμένου να βρεθεί η καλύτερη δυνατή λύση, θα πρέπει να ληφθούν υπόψιν ένα σύνολο περιορισμών οι οποίοι τίθενται για την αντιμετώπιση του προβλήματος και οι οποίοι αν τηρηθούν, θα οδηγήσουν στην αποτελεσματική επίλυση του προβλήματος. Οι περιορισμοί αυτοί ταξινομούνται σε δυο κατηγορίες. Η πρώτη κατηγορία είναι οι ισχυροί περιορισμοί (hard constraints) και η δεύτερη οι χαλαροί περιορισμοί (soft constraints).

2.2.1 Ισχυροί περιορισμοί

Ισχυροί Περιορισμοί (Hard Constrains). Οι ισχυροί περιορισμοί δεν επιτρέπεται να παραβιαστούν υπό οποιεσδήποτε συνθήκες (κυρίως λόγω των φυσικών περιορισμών). Θα πρέπει σε κάθε περίπτωση να ικανοποιηθεί το σύνολο τους, γιατί η παραβίαση τους θα οδηγήσει σε μη αποδεκτή λύση (Tan et. al., 2012). Οι κυριότεροι και πιο συνηθισμένοι ισχυροί περιορισμοί για τη δημιουργία του ωρολόγιού προγράμματος σε ένα ανώτερο εκπαιδευτικό ίδρυμα είναι:

- Ένας καθηγητής ή ένας μαθητής μπορεί να βρίσκεται μόνο σε ένα μάθημα σε μια συγκεκριμένη χρονική στιγμή. Περισσότερες από μια διαλέξεις, οι οποίες δίνονται από έναν καθηγητή δεν μπορούν να πραγματοποιηθούν ταυτόχρονα
- Σε έναν καθηγητή δεν μπορεί να ανατεθεί κανένα γεγονός σε συγκεκριμένες χρονοθυρίδες (timeslots) κατά τις οποίες αυτός δεν είναι διαθέσιμος.
- Ο αριθμός των χρονοθυρίδων που ανατίθενται σε κάθε μάθημα για μια εβδομάδα πρέπει να είναι ίσος με τον εβδομαδιαίο αριθμό ωρών του μαθήματος.
- Δυο γεγονότα του ίδιου μαθήματος δεν μπορούν να γίνονται ταυτόχρονα.
- Ο αριθμός των γεγονότων που συμβαίνουν ταυτόχρονα δεν θα πρέπει να υπερβαίνει τον συνολικό αριθμό των αιθουσών.
- Μια αίθουσα μπορεί να φιλοξενήσει μόνο ένα γεγονός σε μια συγκεκριμένη χρονοθυρίδα.
- Δυο διαλέξεις δεν μπορούν να πραγματοποιηθούν στην ίδια αίθουσα και την ίδια χρονική στιγμή (Malim et. al., 2006).
- Μια διάλεξη ομάδας δεν μπορεί να γίνεται την ίδια περίοδο με μια άλλη, ή οποία δεν είναι διάλεξη ομάδας, αλλά ανήκουν και οι δυο στο ίδιο επίπεδο σπουδών (πχ. ακαδημαϊκό εξάμηνο, έτος).
- Ο αριθμός των φοιτητών που έχουν εγγραφεί σε ένα μάθημα θα πρέπει να είναι μικρότερος ή ίσος με τη χωρητικότητα της αίθουσας στην οποία πραγματοποιείται (Nouri et. al., 2013).

Το κάθε Πανεπιστήμιο μπορεί να θέσει τους δικούς του ιδιαίτερους ισχυρούς περιορισμούς, οι οποίοι θα βασίζονται στις δικές του ανάγκες και απαιτήσεις. Κάθε ωρολόγιο πρόγραμμα που αποτυγχάνει να ικανοποιήσει αυτούς τους περιορισμούς θεωρείται μη αποδεκτό (Malim et. al., 2006).

2.2.2 Χαλαροί περιορισμοί

Χαλαροί Περιορισμοί (Soft Constrains). Οι χαλαροί περιορισμοί είναι ιδιαίτερες επιθυμίες και προτιμήσεις των εμπλεκόμενων μερών, οι οποίες όμως δεν είναι τόσο κρίσιμες για τη δημιουργία του ωρολογίου προγράμματος. Στην πράξη είναι σχεδόν αδύνατο να βρεθούν εφικτές λύσεις, οι οποίες θα ικανοποιούν όλους τους χαλαρούς περιορισμούς. Αυτοί ποικίλουν και συχνά έρχονται σε αντίθεση μεταξύ τους και είναι περισσότεροι σε αριθμό και σε γενικές γραμμές πιο πολύπλοκοι από τους ισχυρούς περιορισμούς. Κάποιοι συνηθισμένοι χαλαροί περιορισμοί είναι:

- Ο κάθε καθηγητής να αναλαμβάνει τα μαθήματα που προτιμά
- Μαθήματα της ίδιας ομάδας φοιτητών θα πρέπει να ανατίθενται σε διαδοχικές χρονοθυρίδες.
- Σε μαθήματα, τα οποία περιλαμβάνουν και θεωρητικό και εργαστηριακό μέρος, το θεωρητικό μέρος θα πρέπει να προηγείται στην εβδομάδα από το εργαστηριακό.
- Ένα μάθημα θα πρέπει ή δεν θα πρέπει να πραγματοποιείται σε κάποια συγκεκριμένη χρονοθυρίδα.
- Όλα τα μαθήματα θα πρέπει να ανατίθενται αίθουσες με τέτοιο τρόπο, ώστε να μεγιστοποιηθεί η χρησιμοποίηση της αίθουσας (Malim et. al., 2006).
- Κάποιος καθηγητής προτιμά πρωινά από βραδινά μαθήματα.
- Ο καθηγητής επιθυμεί τα μαθήματα του να είναι συνεχόμενα και να μην έχει κενά διαστήματα ανάμεσα σε αυτά.

Πρόκειται στην ουσία για τους περιορισμούς, οι οποίοι μας επιτρέπουν να συγκρίνουμε το πόσο καλή είναι μια δοσμένη λύση σε σχέση με άλλες παρόμοιες λύσεις. Αυτό γίνεται ελέγχοντας πόσοι χαλαροί περιορισμοί έχουν ικανοποιηθεί με την κάθε λύση (Nouri et. al., 2013).

Για την αποτελεσματική δημιουργία του ωρολογίου προγράμματος υπάρχουν δυο είδη ισχυρών περιορισμών που πρέπει να υλοποιηθούν. Ισχυροί περιορισμοί των διαθέσιμων πόρων (π.χ. Ένας καθηγητής δεν μπορεί να είναι σε δυο αίθουσες ταυτόχρονα, ή δεν μπορούν να γίνονται στην ίδια τάξη δυο μαθήματα την ίδια χρονική στιγμή) και ισχυροί περιορισμοί διανομής, οι οποίοι εκφράζουν απαιτούμενες ή απαγορευμένες σχέσεις μεταξύ πολλών

τάξεων (π.χ. Δυο διαλέξεις του ίδιου μαθήματος δεν μπορούν να γίνονται την ίδια χρονική στιγμή). Υπάρχουν επίσης και τρεις τύποι χαλαρών περιορισμών. Η πρώτη αφορά χρόνους και αίθουσες, η δεύτερη απαιτήσεις των φοιτητών και των καθηγητών και η τρίτη αφορά τους περιορισμούς διανομής των διαθέσιμων πόρων, οι οποίοι εκφράζουν προτιμήσεις σε τάξεις και μαθήματα (Murray et. al., 2007).

Συνήθως στη δημιουργία ενός ωρολόγιου προγράμματος δεν λαμβάνονται υπόψιν οι ατομικές προτιμήσεις γιατί είναι δύσκολο να συλληφθούν και να μοντελοποιηθούν. Απλές προτιμήσεις του στυλ ¨Προτιμώ πρωινά μαθήματα και όχι βραδινά ¨ είναι εύκολο να μοντελοποιηθούν. Αντιθέτως προτάσεις του στυλ ¨Προτιμώ βραδινά μαθήματα παρά τρία πρωινά στη σειρά¨ παρουσιάζουν δυσκολίες στη μοντελοποίηση τους (Tan et. al., 2012).

Ο μεγάλος αριθμός και η ποικιλία των περιορισμών που προκύπτουν τόσο στη δομή του μαθήματος, όσο και διάφορες ειδικές απαιτήσεις αυξάνουν τη δυσκολία δημιουργίας ενός προγράμματος, το οποίο θα ικανοποιεί όλα τα εμπλεκόμενα μέρη. Ο κάθε δημιουργός του προγράμματος μπορεί να ορίσει τους δικούς του ισχυρούς και χαλαρούς περιορισμούς (Murray et. al., 2007).

3. *Σχετικές υλοποιήσεις*

. Η ιστορία των ερευνών και των λύσεων που έχουν προταθεί ευρέως για την επίλυση του προβλήματος της δημιουργίας ωρολογίου προγράμματος ξεκινά από το μακρινό 1960. Λόγω της φύσης και της δομής του, το ωρολόγιο πρόβλημα αποτελεί ένα από τα πλέον δύσκολα προβλήματα, εξαιτίας της πολυπλοκότητας του και των συνδυασμών που απαιτούνται για την επίλυση του. Αυτό συμβαίνει επειδή είναι δύσκολο να βρεθεί η βέλτιστη λύση που απαιτείται, καθώς αυξάνονται οι διαθέσιμοι πόροι και οι περιορισμοί που τίθενται. Στην πραγματικότητα, όλα τα προβλήματα που σχετίζονται με την κατασκευή ενός χρονοδιαγράμματος αποτελούν NP-complete προβλήματα. Κατά καιρούς έχουν προταθεί διάφορες μέθοδοι για την επίλυση του.

Στις πρώτες προσπάθειες χρησιμοποιήθηκε η θεωρία των γράφων (theory of graphs), ο ακέραιος γραμμικός προγραμματισμός (integer linear programming) και τεχνικές ικανοποίησης περιορισμών (constraint satisfaction). Υπάρχουν επίσης μια σειρά από μεταευριστικές (metaheuristic) μεθόδους που έχουν χρησιμοποιηθεί για την επίλυση του. Τέτοιες είναι η simulated annealing, η tabu search και διάφοροι γενετικοί αλγόριθμοι (genetic algorithms) (Zhang et. al., 2005).

3.1 Θεωρία των γράφων

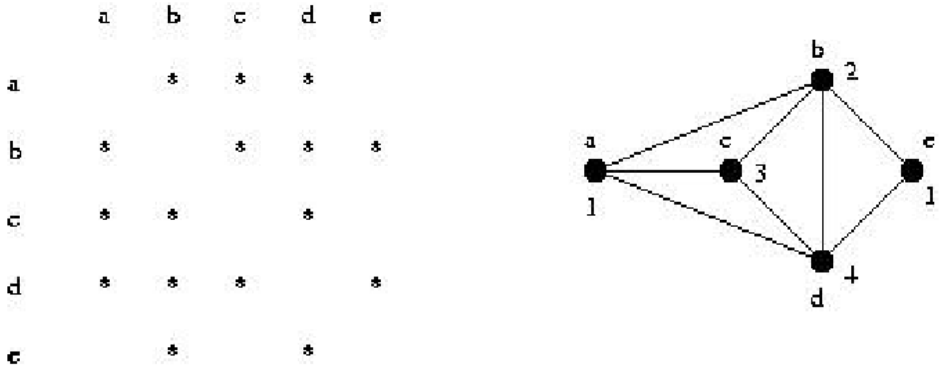
Ο γράφος στον απλούστερο ορισμό του είναι η οπτική αναπαράσταση των σχέσεων που αναπτύσσουν ορισμένες ποσότητες, σχεδιασμένες σε σχέση με ένα σύνολο αξόνων. Ένας άλλος ορισμός που κινείται στο ίδιο εννοιολογικό πλαίσιο της οπτικής αναπαράστασης αναγνωρίζει το γράφο ως απεικόνιση αποτελούμενη από ένα σύνολο σημείων (κορυφών ή κόμβων) που συνδέονται με γραμμές (ακμές). Σε μία άλλη εκδοχή είναι ένα σύνολο από κόμβους (κορυφές) που ενώνονται μεταξύ τους με ακμές και ορίζεται από τον τρόπο με τον οποίο συνδέονται οι κορυφές (κόμβοι) (Graph theory, 2015).

Ο χρωματισμός γραφήματος (graph coloring) είναι μια ειδική περίπτωση της επισήμανσης γράφου (graph labeling). Πρόκειται στην ουσία για μια ανάθεση χρωμάτων σε όλα τα στοιχεία του γράφου, τα οποία υπόκεινται σε ορισμένους περιορισμούς. Στην απλούστερη μορφή του, είναι ένας τρόπος χρωματισμού των κορυφών του γράφου, με τρόπο ώστε να μην υπάρχουν δυο γειτονικές κορυφές, οι οποίες να μοιράζονται το ίδιο χρώμα. Αυτό το γεγονός ονομάζεται χρωματισμός κορυφών (vertex coloring). Με παρόμοιο τρόπο, ο χρωματισμός

ακμών (edge coloring), εκχωρεί ένα χρώμα σε κάθε ακμή, με τέτοιο τρόπο ώστε να μην υπάρχουν δυο παρακείμενες ακμές με το ίδιο χρώμα (Graph coloring, 2015).

Τα παραδοσιακά πρότυπα χρωματισμού γραφήματος για τη δημιουργία ωρολόγιού προγράμματος περιλαμβάνουν γράφους στους οποίους μια κορυφή αντιπροσωπεύει ένα μάθημα το οποίο πρέπει να προγραμματιστεί και μια ακμή ένα ζευγάρι μαθημάτων τα οποία έρχονται σε σύγκρουση (conflict), δηλαδή δεν μπορούν να προγραμματιστούν την ίδια χρονική περίοδο. Το χρώμα της κορυφής αντιπροσωπεύει την χρονική περίοδο για την οποία προγραμματίζεται το μάθημα. Αν δυο κορυφές v και w ενός γράφου μοιράζονται μια ακμή vw, τότε χρωματίζονται με διαφορετικά χρώματα. Ο ελάχιστος αριθμός χρωμάτων που απαιτούνται για να χρωματιστούν σωστά οι κορυφές του γράφου ονομάζεται χρωματικός αριθμός του γράφου και συμβολίζεται με χ(G).

Στο παρακάτω σχήμα απεικονίζεται ένα στιγμιότυπο απλού γράφου, στο οποίο υπάρχουν πέντε μαθήματα, a,b,c,d και e, τα οποία σχεδιάζονται. Στο διάγραμμα στα αριστερά, ένα * αντιπροσωπεύει ένα ζευγάρι μαθημάτων, τα οποία συγκρούονται, για παράδειγμα προγραμματίζονται την ίδια χρονική στιγμή. Η αιτία της σύγκρουσης θα μπορούσε να είναι ότι διδάσκονται από τον ίδιο καθηγητή, ή ότι απαιτούν την ίδια τάξη. Όταν δοθεί ο κατάλογος των μαθημάτων a,b,c,d και e και μια λίστα των πιθανών συγκρούσεων μπορεί να δημιουργηθεί ένα πρόγραμμα χωρίς συγκρούσεις μαθημάτων με τη μετατροπή του πίνακα στα αριστερά, στον αντίστοιχο γράφο συγκρούσεων στα δεξιά και την εύρεση του ελάχιστου χρωματισμού.



Εικόνα 3.Δημιουργία γράφου συγκρούσεων (Redl, 2007)

Κατά καιρούς έχουν αναπτυχθεί και παρουσιαστεί διάφορα μαθηματικά και υπολογιστικά μοντέλα, τα οποία βασίζονται σε αυτή την παραδοσιακή προσέγγιση για την επίλυση του προβλήματος της δημιουργίας ωρολογίου προγράμματος για ανώτερα εκπαιδευτικά ιδρύματα. Αυτά συνήθως χρησιμοποιούν διάφορες ευριστικές τεχνικές (heuristic techniques) για το χρωματισμό του γραφήματος, οι οποίες περιλαμβάνουν επίσης και την ικανοποίηση μια σειράς ουσιαστικών και προνομιακών συνθηκών, όπως αυτές που παρουσιάστηκαν προηγουμένως (Redl, 2007).

3.2 Ακέραιος γραμμικός προγραμματισμός (Integer Linear Programming)

Ο γραμμικός προγραμματισμός (Linear Programming) είναι μια μέθοδος για την επίτευξη του καλύτερου αποτελέσματος (όπως μέγιστο κέρδος, ή ελάχιστο κόστος) σε ένα μαθηματικό μοντέλο, στο οποίο οι απαιτήσεις αναπαρίστανται με γραμμικές σχέσεις (Linear programming, 2015). Ένα πρόβλημα ακέραιου προγραμματισμού (Integer Programming) είναι μια μαθηματική αναπαράσταση της σκοπιμότητας του προβλήματος, στην οποία μερικές ή όλες οι παράμετροι έχουν αντικατασταθεί από ακεραίους αριθμούς. Στις περισσότερες περιπτώσεις αναφερόμαστε σε ακέραιο γραμμικό προγραμματισμό, στον οποίο γίνεται χρήση μιας αντικειμενικής συνάρτησης (objective function) και οι περιορισμοί οι οποίοι τίθενται είναι γραμμικοί (Integer programming, 2015).

Οι Bakir και Aksop (Bakir et. al., 2008) προτείνουν μια εφαρμογή του ακέραιου γραμμικού προγραμματισμού για τη δημιουργία του ωρολογίου προγράμματος, στην οποία εκχωρούνται μαθήματα σε χρονικές περιόδους και τάξεις και μοντελοποιείται το πρόβλημα ως μοντέλο 0 και 1. Στη συνέχεια με τη βοήθεια της αντικειμενικής συνάρτησης υπολογίζεται η δυσαρέσκεια καθηγητών και φοιτητών. Για παράδειγμα θα μπορούσε να μοντελοποιηθεί το πρόβλημα με τη χρήση δυο μεταβλητών απόφασης, όπως φαίνεται παρακάτω:

$$\forall i \in I, \forall j \in J, \forall k \in K, \forall l \in L_k, \forall m \in M_{kl}, \forall n \in N_m$$

$$\chi_{i,j,k,l,m,n} = \begin{cases} 1, \text{ αν το μάθημα m διδάσκεται απο τον καθηγητή l στην ομάδα} \\ \text{των φοιτητών κ και παραγματοποιείται την ημέρα i} \\ \text{στην χρονοθυρίδα j και στην αίθουσα n} \\ \\ 0, \text{ σε οποιαδήποτε άλλη περίπτωση} \end{cases}$$

$$\forall i \in I, \forall j \in J, \forall k \in K \text{ - } \{4,8\}$$

$$Z_{i,j,k} = \begin{cases} 1, \text{ όταν βασικά μαθήματα της ομάδας φοιτητών k επικαλύπτονται με} \\ \text{μαθήματα που διδάσκονται σε άλλη κλάση την i ημέρα και j χρονοθυρίδα} \\ \\ 0, \text{ σε οποιαδήποτε άλλη περίοδο} \end{cases}$$

Η αντικειμενική συνάρτηση κατασκευάζεται ώστε να ελαχιστοποιείται η συνολική δυσαρέσκεια και αποτελείται από πέντε όρους. Η συνολική δυσαρέσκεια μπορεί να οριστεί ως εξής:

{Συνολική Δυσαρέσκεια} = {Δυσαρέσκεια φοιτητών στην κανονική ομάδα} +

{Δυσαρέσκεια φοιτητών στη δευτερεύουσα ομάδα}+

{Δυσαρέσκεια των καθηγητών} +

{Δυσαρέσκεια των καθηγητών εκτός σχολής} +

{Δυσαρέσκεια των φοιτητών που αποτυγχάνουν}(Bakir et. al., 2008).

3.3 Τεχνικές Ικανοποίησης Περιορισμών (Constraint Satisfaction)

Η μεθοδολογία επίλυσης που βασίζεται στην ικανοποίηση περιορισμών εφαρμόζεται συνήθως σε προβλήματα δημιουργίας προγραμμάτων. Ένα πρόβλημα ικανοποίησης περιορισμών και βελτιστοποίησης (Constraint Satisfaction and Optimization Problem, CSOP) αποτελείται από ένα σύνολο μεταβλητών, οι οποίες ανήκουν σε πεπερασμένους τομείς (finite domains), ένα σύνολο ισχυρών περιορισμών, οι οποίοι περιορίζουν το σύνολο τιμών που μπορούν να έχουν οι μεταβλητές την ίδια χρονική στιγμή και μια συνάρτηση στόχου. Σε μια ολοκληρωμένη λύση CSOP μια τιμή ανατίθεται σε κάθε μεταβλητή, με τέτοιο τρόπο ώστε να ικανοποιούνται όλοι οι ισχυροί περιορισμοί. Ο στόχος μπορεί να εκφραστεί από τους χαλαρούς περιορισμούς και προσπαθεί να βρει την καλύτερη λύση, η οποία παραβιάζει τον μικρότερο αριθμό χαλαρών περιορισμών.

Όλες οι συναντήσεις σε ένα μάθημα μπορούν να αναπαρασταθούν με τη χρήση μιας μεταβλητής. Οι περισσότερες συναντήσεις μιας τάξης διδάσκονται στην ίδια αίθουσα, από τον ίδιο καθηγητή και τις ίδιες ώρες μιας ημέρας στη διάρκεια μιας εβδομάδας. Οι συναντήσεις διαχωρίζονται από ένα μοναδικό αριθμό ημέρας. Ένα τυπικό παράδειγμα είναι ένα μάθημα, το οποίο διδάσκεται δυο φορές την εβδομάδα (για παράδειγμα Δευτέρα και Πέμπτη), στην ίδια αίθουσα από τον ίδιο καθηγητή. Η ημέρα της εβδομάδας, η διάρκεια και η εναρκτήρια ώρα κωδικοποιούνται σε μεταβλητές στη δημιουργία του ωρολόγιού προγράμματος. Μια τιμή κωδικοποιεί επίσης τον καθηγητή και την αίθουσα. Επίσης κωδικοποιούνται οι προτιμήσεις σε χρόνο, αίθουσα, μάθημα (χαλαροί περιορισμοί). Σε ένα μάθημα τοποθετούνται μόνο έγκυρες τιμές για καθηγητές, μαθητές και τάξη. Για παράδειγμα εάν το μάθημα θεωρείται εργαστηριακό, τότε θα πρέπει να τοποθετηθεί σε αίθουσα που διαθέτει εργαστηριακό εξοπλισμό (Murray et. al., 2007).

Τα προβλήματα ικανοποίησης περιορισμών (Constraint Satisfaction Problem, CSP) είναι τα προβλήματα που ορίζονται ως ένα σύνολο αντικειμένων, τα οποία πρέπει να ικανοποιούν μια σειρά από περιορισμούς. Η επίλυση τους κατατάσσεται σε δυο κύριες ομάδες:

- Πλήρεις μεθόδους (Complete methods), οι οποίες στοχεύουν στην εξερεύνηση ολόκληρου του χώρου αναζήτησης προκειμένου είτε να βρεθεί το σύνολο των λύσεων είτε να αποδειχθεί ότι το CSP δεν είναι συνεπές. Στη συνέχεια, εφόσον βρεθεί το

σύνολο των λύσεων, πραγματοποιούνται τεχνικές οπισθοδρόμησης (backtracking search), προκειμένου να οδηγηθούμε στην τελική λύση.

- Ελλιπείς μέθοδοι (Incoplete methods), οι οποίες βασίζονται στη χρήση ευριστικών μεθόδων, παρέχοντας μια πιο αποτελεσματική διερεύνηση του χώρου αναζήτησης για την εύρεση της τελικής λύσης. Ένα παράδειγμα είναι η χρήση τεχνικών τοπικής αναζήτησης (Local Search Techniques).

Ο κύριος στόχος των διαδικασιών επεξεργασίας των περιορισμών είναι να αναζητήσουν διαθέσιμες χρονοθυρίδες και αίθουσες, οι οποίες πληρούν όλες τις απαραίτητες προϋποθέσεις για το συγκεκριμένο μάθημα. Κάθε φορά που γίνεται μια αναζήτηση χρησιμοποιείται μια συνάρτηση καταλληλόλητας (fitness function), η οποία δημιουργεί βαθμολογίες για τις διαθέσιμες επιλογές στην προσπάθεια να αξιοποιηθούν κατάλληλα οι διαθέσιμες χρονοθυρίδες και αίθουσες. Αυτό οφείλεται στο γεγονός ότι οι αίθουσες έχουν διαφορετικό είδος εγκαταστάσεων, προσβασιμότητας και τοποθεσίας. Επίσης υπάρχουν περισσότερο ευνοϊκές χρονοθυρίδες, οι οποίες συγκεντρώνουν τις προτιμήσεις των χρηστών και άλλες λιγότερες ευνοϊκές. Αν βρεθούν ελεύθερες χρονοθυρίδες και αίθουσες, τότε αυτές επιλέγονται για το μάθημα, ενώ σε διαφορετική περίπτωση ακολουθείται μια διαδικασία υπαναχώρησης στις προηγούμενες επιλογές. Τροποποιούνται οι προηγούμενες επιλογές, σύμφωνα με τη βαθμολογία και πραγματοποιείται η νέα αναζήτηση στο χώρο των διαθέσιμων λύσεων. Η διαδικασία συνεχίζεται μέχρι να ολοκληρωθεί το πρόγραμμά και να τοποθετηθούν σε αυτό όλα τα μαθήματα (Irene et. al., 2008)

3.4 Μεταευριστικές μέθοδοι (Metaheuristic methods)

Ωστόσο, οι παραπάνω μέθοδοι, σε πολλές περιπτώσεις, αδυνατούσαν να δώσουν λύση σε όλα τα στιγμιότυπα και σε όλους τους περιορισμούς που τίθενται στο πρόβλημα. Γι΄ αυτό το λόγο στη συνέχεια αναπτύχθηκαν και μελετήθηκαν νέες μέθοδοι, οι οποίες ονομάζονται μεταευριστικές (metaheuristics methods) σε μια προσπάθεια καλύτερης αντιμετώπισης του. Τέτοιες μέθοδοι είναι η Adaptive Tabu search, η simulated annealing και διάφοροι γενετικοί αλγόριθμοι (genetic algorithms). Οι μέθοδοι αυτοί διαθέτουν μηχανισμούς, οι οποίοι επιτρέπουν μια καλύτερη εξερεύνηση του χώρου αναζήτησης.

3.4.1 Adaptive Tabu Search

Η μέθοδος Tabu Search χρησιμοποιεί διάφορες μεταευριστικές (metaheuristic)μεθόδους αναζήτησης, οι οποίες χρησιμοποιώντας διάφορες μεθόδους τοπικής αναζήτησης ψάχνουν για τη μαθηματική βελτιστοποίηση της επίλυσης του προβλήματος. Οι τοπικές αναζητήσεις ψάχνουν για μια πιθανή λύση του προβλήματος ελέγχοντας τους άμεσους γείτονες της

(δηλαδή παρόμοιες λύσεις με την υπάρχουσα εκτός από μια δυο μικρές λεπτομέρειες), με την ελπίδα να βρουν μια βελτιωμένη λύση.

Οι Zhipeng Lu και Jin-Kao Hao (Lü et. al., 2010) ανέπτυξαν ένα αλγόριθμο, ο οποίος ακολουθεί ένα γενικό πλαίσιο, το οποίο αποτελείται από τρεις φάσεις. Προετοιμασία (Initialization), εντατικοποίηση (Intensification) και διαφοροποίηση (Diversification).

- Προετοιμασία (Initialization): Η φάση της προετοιμασίας στοχεύει κυρίως στην κατασκευή ενός αρχικού, εφικτού προγράμματος, χρησιμοποιώντας γρήγορους άπληστους ευριστικούς αλγορίθμους.
- Εντατικοποίηση (Intensification): Μόλις ολοκληρωθεί αυτή η φάση και δημιουργηθεί το αρχικό πρόγραμμα, χρησιμοποιείται ένας προσαρμοστικός μηχανισμός εντατικοποίησης. Σε αυτή τη φάση η βασική μηχανή αναζήτησης βασίζεται στο tabu search, στο οποίο έχουν εισαχθεί δυο διακριτές γειτονικές δομές. Η μία είτε ανταλλάσει δυο μαθήματα μεταξύ τους, είτε μετακινεί κάποιο μάθημα σε μια άλλη κενή θέση. Η δεύτερη δομή καθορίζεται από απλές ή διπλές αλυσίδες Kempe, οι οποίες αφορούν δυο διακριτές περιόδους. Ο tabu search εξερευνά το χώρο αναζήτησης χρησιμοποιώντας επαλειμμένες αντικαταστάσεις της τρέχουσας λύσης με κάποια άλλη γειτονική λύση, η οποία δεν έχει επιλεγεί πρόσφατα. Με αυτό τον τρόπο υπάρχει βέβαια ο κίνδυνος η λύση που επιλέγεται να είναι χειρότερη από αυτήν που αντικαθίσταται. Ο tabu search βασίζεται στην πεποίθηση ότι η ευφυής αναζήτηση πρέπει να βασίζεται συστηματικά στην προσαρμοστική μνήμη και τη μάθηση.
- Διαφοροποίηση (Diversification): Όταν ο tabu search δεν μπορεί να βελτιώσει πλέον τη λύση, τότε ενεργοποιείται η φάση της διαφοροποίησης, στην οποία πραγματοποιούνται επαναλαμβανόμενες τοπικές αναζητήσεις, προκειμένου να βρεθεί η βέλτιστη τοπική λύση (Lü et. al., 2010).

3.4.2 Γενετικοί Αλγόριθμοι (Genetic Algorithms)

Οι γενετικοί αλγόριθμοι χρησιμοποιούν τεχνικές εμπνευσμένες από τη βιολογία, όπως η κληρονομικότητα (inheritance), η μετάλλαξη (mutation), η φυσική επιλογή (natural selection) και ο ανασυνδυασμός (recombination ή crossover), προκειμένου να βρουν τη βέλτιστη λύση, μέσα στο χώρο των πιθανών λύσεων. Πρόκειται για έναν πιθανολογικό αλγόριθμο αναζήτησης, ο οποίος μετατρέπει επαναληπτικά έναν πληθυσμό (ένα σύνολο) μαθηματικών αντικειμένων, το οποίο περιέχει μια σχετικά υγιή τιμή, σε ένα νέο πληθυσμό απογόνων αντικειμένων, χρησιμοποιώντας τη Δαρβινική αρχή της φυσικής επιλογής και ενέργειες οι οποίες είναι διαμορφωμένες σύμφωνα με γενετικές πράξεις, όπως είναι ο ανασυνδυασμός και η μετάλλαξη. Ένα βασικό χαρακτηριστικό των γενετικών αλγορίθμων είναι το γεγονός ότι κωδικοποιούν τις πιθανές λύσεις του προβλήματος με τη μορφή δομών δεδομένων, όπως τα

χρωμοσώματα. Διαδικασίες ανασυνδυασμού εφαρμόζονται συνεχώς σε αυτή τη δομή, προκειμένου να παράγουν καλύτερους απογόνους, διατηρώντας παράλληλα τις κρίσιμες πληροφορίες που περιέχουν (Genetic algorithm, 2015).

Οι Aderemi O. Adewumi, Babatunde A. Sawyerr και M. Montaz Ali (Aderemi et. al., 2009) προτείνουν μια εφαρμογή των γενετικών αλγορίθμων για την δημιουργία του ωρολόγιού προγράμματος σε ένα Πανεπιστημιακό Ίδρυμα. Η εργασία αυτή ασχολείται με την κατανομή των διαφόρων μαθημάτων σε δοσμένες αίθουσας διδασκαλίας. Ο κύριος στόχος είναι η εξασφάλιση της ικανοποίησης όλων των ισχυρών περιορισμών. Στην προκειμένη περίπτωση, οι διαλέξεις πραγματοποιούνται κατά τη διάρκεια μιας εβδομάδας πέντε ημερών και η κάθε ημέρα αποτελείται από δέκα χρονοθυρίδες (από τις 08:00 εως τις 18:00).

Στο παράδειγμα αυτό κάθε πρόγραμμα για μια τάξη (Room) λαμβάνεται ως χρωμόσωμα. Έτσι υπάρχει ένας πληθυσμός με μήκος χρωμοσώματος N, όπου N είναι ο αριθμός των αιθουσών. Κάθε γονίδιο στο χρωμόσωμα περιέχει πληροφορίες σχετικά με τα μαθήματα, τα οποία έχουν προγραμματιστεί στη συγκεκριμένη αίθουσα κατά τη διάρκεια μιας εβδομάδας.

Ολόκληρο το πρόγραμμα μπορεί να θεωρηθεί ως μια σειρά από αίθουσες. Κάθε αίθουσα αναπαρίσταται από ένα πίνακα (matrix). Το πρόγραμμα για ολόκληρο το Εκπαιδευτικό Ίδρυμα είναι μια συλλογή από πίνακες, έναν για κάθε αίθουσα.

Οι δομές δεδομένων που απαιτούνται για τη συνολική δημιουργία του ωρολόγιού προγράμματος είναι τρία διαφορετικά αρχεία εισόδου. Ένα για το μάθημα, ένα για την αίθουσα και ένα για τον καθηγητή. Άλλες σημαντικές είσοδοι που δίνονται από το χρήστη αφορούν τον αριθμό των αιθουσών, των διαλέξεων και των μαθημάτων.

Στην εικόνα που ακολουθεί, παρουσιάζεται μια δομή δεδομένων, όπου το C_i αντιπροσωπεύει το μάθημα i, όπου i=1,2,3,…,m, και m είναι ο αριθμός των μαθημάτων. Με 0 αναπαρίσταται μια κενή χρονοθυρίδα.

Room 1					
Time	Mon	Tue	Wed	Thu	Fri
8-10	C1	0	0	0	0
10-12	C5	0	0	0	C2
12-1	C12	0	0	0	0
…	…	…	…	…	…
4-6	0	0	C9	0	0

……

……

Room N					
Time	Mon	Tue	Wed	Thu	Fri
8-10	0	0	C3	C6	0
10-12	C15	0	0	0	C16
12-1	C4	0	0	0	0
…	…	…	…	…	…
4-6	C17	0	C18	0	C20

Εικόνα 4.Μια απεικόνιση για τα μαθήματα του πληθυσμού των τάξεων (Aderemi et. al., 2009)

Όταν ξεκινάει ο αλγόριθμός, μια διαδικασία αρχικοποίησης δημιουργεί ένα τυχαίο πληθυσμό λύσεων, προκειμένου ο γενετικός αλγόριθμος να ξεκινήσει να εργάζεται επαναληπτικά. Αυτό γίνεται ελέγχοντας αν σε κάθε μάθημα και σε κάθε αίθουσα έχει ανατεθεί μια χρονοθυρίδα. Αν έχει συμβεί αυτό, τότε ανατίθεται στη μεταβλητή που τα αντιπροσωπεύει μια boolean μεταβλητή true, σε διαφορετική περίπτωση, η τιμή της μεταβλητής είναι false. Αν στο τέλος της διαδικασίας η τιμή αυτή παραμένει false, τότε της ανατίθεται μια τυχαία τιμή για το μάθημα ή την αίθουσα από αυτές που απομένουν.

Σε όλη την επαναληπτική διαδικασία της δημιουργίας του προγράμματος, χρησιμοποιείται μια συνάρτηση καταλληλότητας (fitness function), προκειμένου να μετρηθεί ο βαθμός παραβίασης των ισχυρών περιορισμών που έχουν δοθεί. Σε κάθε περιορισμό έχει ανατεθεί μια τιμή ως βάρος (weight), προκειμένου να τιμωρηθούν οι παραβιάσεις τους. Χρησιμοποιείται επίσης ένας γραμμικός συνδυασμός του αριθμού των περιορισμών που παραβιάζονται, ως ένα μέτρο σύγκρισης των χρωμοσωμάτων μέσα σε ένα πληθυσμό (Aderemi et. al., 2009).

3.4.3 Simulated Annealing

Η Simulated Annealing, SA (Προσομοίωση Ανοποίησης), είναι μια συμπαγής και ισχυρή τεχνική, η οποία παρέχει εξαιρετικές λύσεις σε απλά και πολλαπλά προβλήματα αντικειμενικής βελτιστοποίησης με ουσιαστική μείωση του χρόνου υπολογισμού. Βασίζεται σε μια εφαρμογή της θερμοδυναμικής, με την οποία τα μέταλλα έχουν ψυχθεί και ανοποιηθεί. Η SA χρησιμοποιείται για την επίλυση προβλημάτων βελτιστοποίησης, στα οποία ένα επιθυμητό συνολικό ελάχιστο/μέγιστο είναι κρυμμένο ανάμεσα σε πολλά τοπικά ελάχιστα/μέγιστα. Βρίσκει τη βέλτιστη λύση, χρησιμοποιώντας μια σημείο-προς-σημείο (point-to-point) επανάληψη, παρά μια απλή αναζήτηση σε ένα πληθυσμό ατόμων (Suman et. al., 2005).

Ας υποθέσουμε ότι υπάρχει ένας χώρος λύσεων S του προβλήματος, ο οποίος είναι πεπερασμένος και μια αντικειμενική συνάρτηση (objective function) f. Το πρόβλημα ελαχιστοποίησης έγκειται στην εύρεση μιας λύσης, ή μιας κατάστασης $i \in S$, η οποία ελαχιστοποιεί την τιμή της f στο S. Ξεκινώντας τη διαδικασία βρίσκεται μια αρχική λύση. Στη συνέχεια δημιουργείται μια νέα λύση, στην γειτονιά της αρχικής και υπολογίζεται η τιμή της f για αυτήν. Αν παρατηρηθεί μείωση της τιμής της f, τότε η νέα λύση γίνεται αποδεκτή και η ίδια διαδικασία επαναλαμβάνεται. Προκειμένου να αποφύγει να παγιδευτεί σε ένα τοπικό ελάχιστο, η simulated annealing αποδέχεται μερικές φορές και τη χειρότερη κίνηση. Η αποδοχή ή η απόρριψη της χειρότερης λύσης ελέγχεται από μια συνάρτηση πιθανοτήτων (probability function). Η πιθανότητα να επιλεγεί μια κίνηση, η οποία προκαλεί μια αύξηση δ στην τιμή της f ονομάζεται συνάρτηση αποδοχής (acceptance function) και συνήθως ορίζεται ως $\exp(-\delta/T)$, όπου T είναι μια παράμετρος ελέγχου, η οποία αντιστοιχεί στη θερμοκρασία

της φυσικής ανοποίησης (physical annealing). Αυτή η διαδικασία αποδοχής/απόρριψης, σημαίνει ότι μια μικρή αύξηση στην τιμή της f, είναι πιο πιθανό να γίνει αποδεκτή σε σύγκριση με μια μεγαλύτερη αύξηση. Ο αλγόριθμος ξεκινά με μια υψηλή τιμή θερμοκρασίας T και προχωρεί επιχειρώντας έναν ορισμένο αριθμό κινήσεων σε κάθε θερμοκρασία, πριν την μειώσει και συνεχίσει τη διαδικασία.

Κατά καιρούς έχουν προταθεί πολλές παραλλαγές της SA. Οι Ceshia, Gaspero και Scahrf (Ceschia et. al., 2012) προτείνουν μια εφαρμογή της για την δημιουργία του ωρολόγιού προγράμματος, η οποία βασίζεται σε μια πιθανολογική αποδοχή και ένα γεωμετρικό σύστημα ψύξης. Σε αυτή τη μέθοδο, σε κάθε επανάληψη της διαδικασίας αναζήτησης επιλέγεται ένας τυχαίος γείτονας. Η κίνηση αυτή πραγματοποιείται είτε πρόκειται για μια βελτιωτική κίνηση, είτε σύμφωνα με μια εκθετική πιθανότητα μείωσης του χρόνου. Εάν το κόστος της κίνησης είναι $\Delta F>0$, τότε η κίνηση είναι αποδεκτή με πιθανότητα $e^{-\Delta F/T}$, όπου T είναι μια παράμετρος μείωσης του χρόνου, η οποία ονομάζεται θερμοκρασία (Temperature). Σε κάθε θερμοκρασία, ένας αριθμός N γειτόνων της τρέχουσας λύσης δειγματίζεται και η νέα λύση γίνεται αποδεκτή σύμφωνα με την προηγούμενη κατανομή πιθανότητας. Η τιμή του T τροποποιείται χρησιμοποιώντας ένα γεωμετρικό σχήμα, όπως $T_i+1=\beta*T_i$, στο οποίο η παράμετρος $\beta>1$ καλείται ρυθμός ψύξης (cooling rate). Η αναζήτηση ξεκινά με θερμοκρασία T_0 και σταματά όταν φτάσει τη θερμοκρασία T_{min}. Διαφορετικές ρυθμίσεις στις παραμέτρους της SA, οδηγούν σε διαφορετικούς χρόνους λειτουργίας (Ceschia et. al., 2012).

3.5 Αξιολόγηση

Όλες οι προηγούμενοι μέθοδοι προσπαθούν να επιλύσουν το πρόβλημα της δημιουργίας ωρολογίου προγράμματος. Και πράγματι, οι περισσότερες από αυτές τα καταφέρνουν αποτελεσματικά σε ορισμένες περιπτώσεις. Στο σύνολο τους όμως παρουσιάζουν διάφορα προβλήματα.

Στη θεωρία των γράφων, στον ακέραιο γραμμικό προγραμματισμό και στις τεχνικές ικανοποίησης περιορισμών υπάρχει μεγάλη εξάρτηση από τη σωστή ρύθμιση των παραμέτρων που απαιτούνται για την επίλυση του προβλήματος. Επίσης ο τρόπος ενσωμάτωσης και αξιοποίησης της γνώσης που απαιτείται (δηλαδή η σκληρή κωδικοποίηση των ισχυρών και των χαλαρών περιορισμών) αποτελούν τα μεγαλύτερα μειονεκτήματα των προηγούμενων μεθόδων.

Περισσότερο αποτελεσματικές είναι οι μεταευριστικές μέθοδοι, ωστόσο και αυτές αποτελούν μια εξατομικευμένη προσέγγιση για την επίλυση ενός συγκεκριμένου προβλήματος, με συνέπεια να μην είναι σε θέση να αντιμετωπίσουν κάποιες ειδικές περιπτώσεις του προγράμματος, είτε άλλα παρόμοια προβλήματα δημιουργίας ωρολογίων προγραμμάτων. Επίσης και εδώ η σωστή ρύθμιση των παραμέτρων που απαιτούνται παίζει πολύ σημαντικό

ρόλο στην επίλυση του προβλήματος. Πρόκειται επίσης για μη ντετερμινιστικές μεθόδους, οι οποίες δεν παρέχουν καμία εγγύηση για την εύρεση της βέλτιστης λύσης.

4. *Ευφυείς Πράκτορες: Θεωρητικό υπόβαθρο*

Είναι γνωστή σε όλους η έννοια του πράκτορα, όπως αυτή χρησιμοποιείται στην καθημερινή ζωή, για παράδειγμα, ο ρόλος που παίζει ένας ταξιδιωτικός πράκτορας ή ένας κτηματομεσίτης. Και οι δύο έχουν αντιπροσωπευτικό ρόλο, δηλαδή ενεργούν για λογαριασμό κάποιου άλλου. Στην περίπτωση του ταξιδιωτικού πράκτορα για λογαριασμό των ξενοδοχείων και των αεροπορικών εταιρειών, ενώ στην περίπτωση του κτηματομεσίτη για λογαριασμό του ιδιοκτήτη. Επιπλέον, εκτός από το χαρακτήρα του μεσολαβητή που εμφανίζουν οι παραπάνω πράκτορες, έχουν ένα εξίσου σημαντικό χαρακτηριστικό: εμφανίζουν διαφορετικό βαθμό αυτονομίας. Για παράδειγμα ο κτηματομεσίτης αναλαμβάνει να κλείσει ένα ραντεβού για επίδειξη ενός σπιτιού σε κάποιον πιθανό αγοραστή, χωρίς να εμπλέκει στη διαδικασία αυτή τους ιδιοκτήτες, ενώ ίσως έχει και την εξουσιοδότηση να διαπραγματευτεί και την τιμή πώλησης. Ένα τρίτο σημαντικό χαρακτηριστικό του κάθε πράκτορα είναι ο βαθμός της αντίδρασης και της προνοητικότητας που υπάρχει στη συμπεριφορά του. Για παράδειγμα, ένας κτηματομεσίτης που απλά τοποθετεί μία πινακίδα με το σήμα "Πωλείται" έξω από μία ιδιοκτησία και περιμένει τους πελάτες να εμφανιστούν στο γραφείο του, βασίζεται περισσότερο στην αντίδραση. Αντίθετα, ένας κτηματομεσίτης ο οποίος διαφημίζει γενικά την προσφορά ιδιοκτησίας στον τοπικό τύπο ενεργεί προνοητικά προκαλώντας το ενδιαφέρον πιθανών πελατών ακόμα και πριν να έχει έτοιμη για διάθεση κάποια ιδιοκτησία. Οι έννοιες της αντίδρασης (reactiveness) και της προνοητικότητας (proactiveness) δεν είναι αντίθετες. Ο ίδιος πράκτορας μπορεί να εμφανίσει υψηλούς βαθμούς αντίδρασης και προνοητικότητας σε διαφορετικές χρονικές στιγμές. Τέλος, οι προαναφερθέντες πράκτορες εμφανίζουν και άλλα χαρακτηριστικά, όπως η δυνατότητα μάθησης, η συνεργατικότητα και η κινητικότητα.

Ένας ευφυής πράκτορας είναι μια οντότητα που αντιλαμβάνεται το περιβάλλον μέσα στο οποίο βρίσκεται με τη βοήθεια αισθητήρων (sensors), είναι μέρος του περιβάλλοντος αυτού, κάνει συλλογισμούς και δρα πάνω σε αυτό με τη βοήθεια μηχανισμών δράσης (effectors) για την επίτευξη κάποιων στόχων. (Βλαχάβας κ. α., 2011, Coen, 1997, Kautz et. al., 1994).

Ένα *πολυπρακτορικό σύστημα* (*multi-agent system*) είναι ένα σύστημα που σχεδιάστηκε και υλοποιήθηκε ως ένα σύνολο πρακτόρων που αλληλοεπιδρούν, δηλαδή συνεργάζονται, συντονίζονται, διαπραγματεύονται, κτλ. Τα συστήματα πολλαπλών πρακτόρων μαζί με την κατανεμημένη επίλυση προβλημάτων (distributed problem solving) αποτελούν τους βασικούς

τομείς της *κατανεμημένης Τεχνητής Νοημοσύνης*, (*distributed artificial intelligence*). Η κατανεμημένη επίλυση προβλημάτων ασχολείται με το πώς ένα συγκεκριμένο πρόβλημα μπορεί να επιλυθεί από έναν αριθμό επεξεργαστικών μονάδων που συνεργάζονται διαμοιράζοντας γνώση για το πρόβλημα καθώς και τις επιμέρους λύσεις. Τα πολυπρακτορικά συστήματα είναι ουσιαστικά ένα δίκτυο από "χαλαρά συνδεδεμένους" πράκτορες που δρουν μαζί για να επιλύσουν προβλήματα που είναι πέραν των δυνατοτήτων και της γνώσης ενός μόνο πράκτορα. Αν και σε πρώτη ματιά θα φαινόταν ότι δεν υπάρχει διαφορά μεταξύ των δύο περιοχών, εντούτοις, η συνεργασία στα πολυπρακτορικά συστήματα είναι δυναμική με την έννοια ότι οι οντότητες που αλληλοεπιδρούν είναι αυτόνομες, άρα αποφασίζουν για το πότε και πώς θα συνεργαστούν (Βλαχάβας κ. α., 2011).

4.1 Πράκτορες

Οι πράκτορες είναι πολύ δημοφιλείς στις μέρες μας. Ένας αυτόνομος πράκτορας είναι ένα μέρος του συστήματος, το οποίο αισθάνεται το περιβάλλον στο οποίο βρίσκεται και δρα σε αυτό με την πάροδο του χρόνου, προσπαθώντας να πετύχει τους δικούς του στόχους και να αλληλοεπιδράσει με αυτό που αισθάνεται. Από τη στιγμή που ένας ευφυής πράκτορας είναι σε θέση να πραγματοποιήσει αυτόνομες δράσεις, είναι ικανός να εργαστεί για λογαριασμό κάποιου εκπροσώπου, προκειμένου να εκπληρώσει τους στόχους για τους οποίους έχει σχεδιαστεί. Με άλλα λόγια, ένας χρήστης το μόνο που χρειάζεται είναι να ορίσει τους στόχους του πράκτορα και στη συνέχεια αυτός θα χρησιμοποιήσει τις δικές του μεθόδους, προκειμένου να πετύχει τους στόχους του, σύμφωνα με το περιβάλλον στο οποίο βρίσκεται (Ahmad et. al., 2002).

Ως πράκτορα μπορούμε να θεωρήσουμε ένα σύστημα το οποίο προσπαθεί να ικανοποιήσει ένα σύνολο στόχων μέσα σε ένα πολύπλοκο, δυναμικό περιβάλλον. Ο πράκτορας βρίσκεται μέσα στο περιβάλλον, μπορεί να αντιληφθεί το περιβάλλον του μέσω διάφορων αισθητήρων (sensors) και να ενεργήσει στο περιβάλλον μέσω διαφόρων οργάνων δράσης (actuators). Ένας ανθρώπινος πράκτορας έχει μάτια, αυτιά και άλλα όργανα-αισθητήρες και έχει πόδια, στόμα και άλλα μέλη του σώματος για την εξάσκηση δράσης. Ένας μηχανικός πράκτορας υποκαθιστά με κάμερες και υπέρυθρους ανιχνευτές πεδίου τους αισθητήρες ενώ επιτυγχάνει δράση με διάφορα μοτέρ. Ένας πράκτορας λογισμικού έχει κωδικοποιημένες σειρές χαρακτήρων ως όργανα αίσθησης και δράσης. Οι στόχοι που έχει ο κάθε πράκτορας μπορούν να πάρουν πολλές διαφορετικές μορφές. Μπορεί να είναι τελικοί στόχοι ή καταστάσεις τις οποίες ο πράκτορας προσπαθεί να επιτύχει, μία αμοιβή την οποία ο πράκτορας προσπαθεί να μεγιστοποιήσει, εσωτερικές ανάγκες ή κίνητρα τα οποία ο πράκτορας πρέπει να κρατήσει μέσα σε ορισμένα πλαίσια και όρια (Βλαχάβας κ. α., 2011).

Υπάρχουν πολλοί τύποι πρακτόρων, οι οποίοι οργανώνονται σύμφωνα με τη λειτουργικότητα και τα χαρακτηριστικά τους. Κάποιοι από αυτούς είναι Collaborative Agents, Interface Agents, Mobile Agents, Information Agents, Reactive Agents, Hybrid Agents, Heterogeneous Agent Systems και Smart Agents.

Οι συνεργατικοί πράκτορες (Collaborative Agents) διαθέτουν αυτόνομα και συνεργατικά χαρακτηριστικά. Η συνεργασία τους βασίζεται στις επιθυμίες και τις ανάγκες της οντότητας που εκπροσωπούν (Ahmad et. al., 2002).

4.1.1 Πράκτορες με εσωτερική κατάσταση

Στην κατηγορία των πρακτόρων με εσωτερική κατάσταση, ανήκουν οι πράκτορες οι οποίοι διατηρούν μία εσωτερική συμβολική αναπαράσταση του περιβάλλοντός τους και χρησιμοποιώντας κλασσικές συλλογιστικές καθώς και τεχνικές ταυτοποίησης προτύπων και χειρισμού συμβόλων, προσπαθούν να επιτύχουν τους στόχους τους. Η συγκεκριμένη κατηγορία περιλαμβάνει τους πράκτορες που είναι βασισμένοι στη λογική (deliberative agents) και αυτούς με πεποιθήσεις-επιθυμίες-προθέσεις (BDI agents).

Κύριο χαρακτηριστικό της πρώτης αρχιτεκτονικής είναι ότι η ενέργεια την οποία θα εκτελέσει ο πράκτορας προκύπτει βάσει μίας διαδικασίας λογικής απόδειξης. Έτσι, οι πράκτορες της κατηγορίας αυτής έχουν μία βάση γνώσης, στην οποία διατηρούν την αντίληψή τους για τον πραγματικό κόσμο με τη μορφή λογικών προτάσεων και ένα σύνολο κανόνων συμπερασμού, οι οποίοι αναπαριστούν τις ενέργειες που μπορεί να εκτελέσει ένας πράκτορας. Λόγω βέβαια του γεγονότος ότι ο πράκτορας μαθαίνει μέσα από το περιβάλλον του, παρουσιάζεται μία αδυναμία στην αναπαράσταση και στη συλλογιστική της έννοιας του χρόνου και επιπλέον η διαδικαστική γνώση που απαιτείται (ουσιαστικά η απάντηση στην ερώτηση "τι πρέπει να γίνει") δεν εμπεριέχεται με φυσικό τρόπο στη συλλογιστική (Βλαχάβας κ. α., 2011, Craenen et. al., 2010).

4.1.2 Πράκτορες με πεποιθήσεις – επιθυμίες – προθέσεις (BDI)

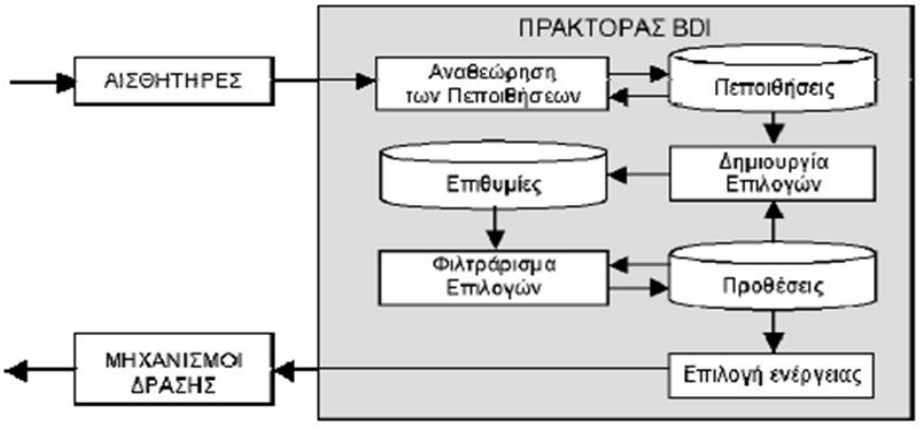
Οι πράκτορες με πεποιθήσεις - επιθυμίες - προθέσεις (Belief – Desire – Intension) έχουν μία πολυπλοκότερη αναπαράσταση του περιβάλλοντος και σχεδιάζουν την επίτευξη των στόχων τους. Η εσωτερική κατάσταση των πρακτόρων αυτών αποτελείται από πεποιθήσεις, επιθυμίες, προθέσεις, στόχους και πλάνα.

- Οι πεποιθήσεις αποτελούν την άποψη και τη γνώση που έχει ο πράκτορας για το περιβάλλον του, η οποία ενδέχεται να είναι και εσφαλμένη.
- Οι επιθυμίες αφορούν την κρίση του πράκτορα για τις μελλοντικές καταστάσεις του περιβάλλοντός του, όπως για παράδειγμα αν μία μελλοντική κατάσταση είναι ή όχι

επιθυμητή. Στο επίπεδο αυτό δεν εξετάζεται αν κάποια επιθυμητή κατάσταση είναι εφικτή, ενώ είναι δυνατό να υπάρχει σύγκρουση μεταξύ των επιθυμητών καταστάσεων.

- Οι στόχοι αποτελούν υποσύνολο των επιθυμιών, για την επίτευξη των οποίων ο πράκτορας μπορεί να ενεργήσει. Οι στόχοι πρέπει να είναι εφικτοί και να μην περιέχουν συγκρούσεις μεταξύ τους.
- Οι προθέσεις είναι υποσύνολο των στόχων, τις οποίες ο πράκτορας προσπαθεί να επιτύχει τη δεδομένη χρονική στιγμή. Θα πρέπει να σημειωθεί ότι στην πλειοψηφία των περιπτώσεων δεν είναι δυνατό να επιτευχθούν όλοι οι στόχοι ταυτόχρονα, οπότε και επιλέγεται ένα υποσύνολό τους, που αποτελεί τις προθέσεις, βάσει ορισμένων κριτηρίων ιεράρχησης.
- Τα πλάνα αποτελούν τους τρόπους με τους οποίους ο πράκτορας θα επιτύχει τις προθέσεις του.

Οι προθέσεις παίζουν κυρίαρχο ρόλο στη συλλογιστική διαδικασία ενός πράκτορα, καθώς από τη στιγμή που ο πράκτορας διαμορφώνει μία πρόθεση, θα πρέπει να αποφασίσει και πώς θα την υλοποιήσει. Επίσης, δημιουργούν δεσμεύσεις στις μελλοντικές ενέργειες του πράκτορα, καθώς η προσπάθεια για την υλοποίηση μίας πρόθεσης αποκλείει την προσπάθεια για την επίτευξη προθέσεων που είναι ασύμβατες με αυτή. Επιπλέον οι προθέσεις επηρεάζουν τις πεποιθήσεις του πράκτορα πάνω στις οποίες βασίζεται η μελλοντική συλλογιστική του, με την έννοια ότι θα πρέπει ο πράκτορας να πιστεύει ότι οι προθέσεις του είναι υλοποιήσιμες, αλλιώς θα ενεργεί παράλογα (Βλαχάβας κ. α., 2011, Craenen et. al., 2010).



Εικόνα 5.Η αρχιτεκτονική BDI (Βλαχάβας κ. α., 2011)

4.1.3 Περιβάλλοντα Πρακτόρων - Διαδικασία μάθησης

Όπως αναφέραμε, πράκτορας είναι οτιδήποτε μπορεί να αντιληφθεί το περιβάλλον του μέσω αισθητήρων και να αντιδράσει πάνω στο περιβάλλον μέσω μηχανισμών δράσης. Ως εκ τούτου, οι πράκτορες είναι υπολογιστικά συστήματα τα οποία δρουν σε ένα πολύπλοκο περιβάλλον, αντιλαμβάνονται και δρουν αυτόνομα πάνω σε αυτό, πετυχαίνοντας έτσι ένα σύνολο από στόχους για τους οποίους έχουν κατασκευαστεί.

Οι ευφυείς πράκτορες κάνουν συνεχώς τις εξής τρεις λειτουργίες: α) αντιλαμβάνονται τις δυναμικές συνθήκες του περιβάλλοντος, β) δρουν πάνω στο περιβάλλον ώστε να το αλλάξουν και γ) συλλογίζονται ώστε να ερμηνεύσουν αυτά που αντιλαμβάνονται, να λύσουν προβλήματα, να συμπεράνουν και να καθορίσουν τη δράση τους. Οι πράκτορες λογισμικού είναι προγράμματα που διενεργούν διάλογο, διαπραγματεύονται και συντονίζουν τη ροή των πληροφοριών.

Το κεντρικό ερώτημα στο οποίο καλείται να απαντήσει κάθε υλοποίηση είναι το πώς ο πράκτορας καθορίζει την επόμενη ενέργεια που πρέπει να κάνει δεδομένων της κατάστασης του περιβάλλοντος και του στόχου τον οποίο καλείται να επιτύχει. Μία πρώτη προσέγγιση στο ερώτημα αυτό δίνεται από τις αφηρημένες αρχιτεκτονικές πρακτόρων, δηλαδή στο ποια μπορεί να είναι τα μέρη (modules) των πρακτόρων και ποια είναι η ροή ελέγχου και δεδομένων μεταξύ των μερών αυτών. Φαίνεται λοιπόν από όλους τους ορισμούς που δόθηκαν ότι ένας πράκτορας αντιλαμβάνεται ένα περιβάλλον, μαθαίνει από αυτό και τελικά δρα μέσα σε αυτό. Τα περιβάλλοντα των πρακτόρων κατηγοριοποιούνται ως εξής:

- Προσβάσιμα ή Μη Προσβάσιμα (Accessible - Inaccessible), ανάλογα με το αν υπάρχει διαθέσιμη πλήρης, ακριβής και ανανεωμένη πληροφορία.
- Αιτιοκρατικά ή Μη Αιτιοκρατικά (Deterministic - Non-Deterministic), αναφορικά με το αν μία συγκεκριμένη ενέργεια έχει πάντα συγκεκριμένα αποτελέσματα.
- Επεισοδιακά ή Μη Επεισοδιακά (Episodic - Non-Episodic), αναφορικά με το αν το περιβάλλον χωρίζεται ή όχι σε διακριτά και ανεξάρτητα επεισόδια. Οι ενέργειες του πράκτορα σε ένα επεισόδιο δεν έχουν επίδραση στα υπόλοιπα. Σε ένα επεισοδιακό περιβάλλον δεν υπάρχει ανάγκη να εξετάζει ο πράκτορας την επίδραση που θα έχουν οι τρέχουσες ενέργειες σε μελλοντικές καταστάσεις.
- Στατικά ή Δυναμικά (Static - Dynamic), σε σχέση με το αν εμφανίζονται αλλαγές χωρίς την παρέμβαση του πράκτορα.
- Διακριτά ή Συνεχή (Discrete - Continuous), ως προς την ύπαρξη ή όχι ενός πεπερασμένου αριθμού ενεργειών και δεδομένων στο μηχανισμό αντίληψης του πράκτορα (Coen, 1997, Kautz et. al., 1994, Craenen et. al., 2010).

4.2 Συστήματα Πολλαπλών Πρακτόρων (MAS)

Ένα σύστημα πολλαπλών πρακτόρων είναι απλά ένα σύστημα που περιέχει περισσότερους από έναν πράκτορες. Η διαφορά ανάμεσα στα πολυπρακτορικά συστήματα και τους απλούς πράκτορες, είναι ότι τα πολυπρακτορικά συστήματα απαιτούν από τους πράκτορες που τα αποτελούν να συμπεριφέρονται με ένα συγκεκριμένο τρόπο. Η ομάδα των πρακτόρων αλληλοεπιδρούν μεταξύ τους, προκειμένου να επιτύχουν τους επιμέρους στόχους για τους οποίους έχουν σχεδιαστεί (Chen, 2014).

Μια σημαντική δυνατότητα, η οποία βοηθάει τη συνεργασία μεταξύ των πρακτόρων είναι η διαπραγμάτευση. Σε ένα σύστημα, διαπραγμάτευση είναι η αμοιβαία επικοινωνία και ο συντονισμός μεταξύ των πρακτόρων, προκειμένου αυτοί να καταλήξουν σε ένα συμπέρασμα το οποίο ικανοποιεί τα επιμέρους συμφέροντα τους (Chen, 2014). Πρόκειται για τη διαδικασία λήψης αποφάσεων στην οποία δυο ή περισσότερα μέρη λαμβάνουν τις δικές τους αποφάσεις και αλληλοεπιδρούν μεταξύ τους με στόχο το αμοιβαίο κέρδος. Προτάσεις αποστέλλονται σε άλλα μέρη και μια νέα πρόταση ενδέχεται να δημιουργηθεί μετά τη λήψη της αντιπρότασης. Η διαδικασία συνεχίζεται μέχρις ότου επιτευχθεί μια συμφωνία, είτε ένα ή περισσότερα μέρη σταματήσουν τη διαπραγμάτευση (Chiu et. al., 2004). Οι επιμέρους στόχοι του κάθε πράκτορα ενδέχεται να συμφωνούν ή να διαφωνούν με τους στόχους της υπόλοιπης ομάδας. Οι πράκτορες έχουν περιορισμένη αντίληψη και γνώση του κόσμου στον οποίο βρίσκονται (Wangmaeteekul, 2011).

Μια δυσκολία στη χρήση της διαπραγμάτευσης, προκειμένου να επιτευχθούν αμοιβαία οφέλη, είναι το γεγονός ότι πρόκειται για μια δαπανηρή και χρονοβόρα διαδικασία, η οποία επιβαρύνει το συντονισμό. Θα πρέπει λοιπόν να υπάρχουν κάποιοι περιορισμοί, είτε χρονικοί, είτε αριθμός φάσεων διαπραγμάτευσης, είτε συνδυασμός τους, στη διαδικασία του σχεδιασμού και της διαπραγμάτευσης. Η διαπραγμάτευση μπορεί να είναι είτε για ανταλλαγή εργασίας, είτε για κατανομή των πόρων. Σε ένα πολυπρακτορικό σύστημα δεν υπάρχει κατ΄ ανάγκη κεντρικός έλεγχος, ούτε κοινή γνώση, ούτε κοινοί στόχοι, ούτε κριτήρια επιτυχίας. Με αυτό τον τρόπο, παρέχεται η δυνατότητα για πραγματικό ανταγωνισμό μεταξύ των πρακτόρων (Kraus et. al., 1995).

Το πιο κοινό πρόβλημα σε μια γενική διαδικασία διαπραγμάτευσης είναι η αναποτελεσματικότητα (Inefficiency). Τα πολυπρακτορικά συστήματα παρέχουν τη δυνατότητα να ξεπεραστεί αυτή η δυσκολία. Οι πράκτορες σε ένα πολυπρακτορικό σύστημα ενεργούν συλλογικά ως μια κοινότητα και συνεργάζονται – ανταγωνίζονται, προκειμένου να επιτύχουν ο καθένας τους στόχους για τους οποίους έχει σχεδιαστεί., καθώς επίσης και όλοι μαζί τον κοινό στόχο. Αυτά τα συναγωνιστικά και ανταγωνιστικά χαρακτηριστικά ταιριάζουν με την πραγματική φύση της διαπραγμάτευσης. Επιπλέον τα βασικά χαρακτηριστικά των πρακτόρων καθιστούν το πολυπρακτορικό σύστημα ιδανικό για την υποστήριξη των

διαπραγματεύσεων. Οι Wooldridge και Jennings (Wooldridge et. al., 1995) συνοψίζουν αυτά τα χαρακτηριστικά ως εξής:

- Αυτονομία (Autonomy. Οι πράκτορες είναι σε θέση να λειτουργούν χωρίς την άμεση παρέμβαση των ανθρώπων, ή άλλων παραγόντων και να έχουν κάποιας μορφής έλεγχο πάνω στις ενέργειες και στην εσωτερική τους κατάσταση.
- Κοινωνική Ικανότητα (Social Ability). Οι πράκτορες είναι σε θέση να αλληλεπιδράσουν με άλλους πράκτορες (ενδεχομένως και ανθρώπους), μέσω κάποιας γλώσσας επικοινωνίας πρακτόρων (πχ. KQML, FIPA ACL).
- Αντιδραστικότητα (Reactivity). Οι πράκτορες είναι σε θέση να αντιλαμβάνονται το περιβάλλον μέσα στο οποίο βρίσκονται και να ανταποκρίνονται άμεσα σε όποιες αλλαγές συμβαίνουν σε αυτό, βασιζόμενοι σε μια εσωτερική διαδικασία μάθησης που διαθέτουν.
- Προ-δραστικότητα (Pro-activeness). Οι πράκτορες δεν ενεργούν απλώς ως αντίδραση στο περιβάλλον τους, αλλά αναλαμβάνουν επίσης πρωτοβουλίες προσπαθώντας να επιτύχουν το στόχο τους.

Από τη στιγμή που δεν υπάρχει κάποιος κεντρικός έλεγχος, οι πράκτορες θα πρέπει να πείσουν ο ένας τον άλλον προκειμένου να εκτελεστεί μια εργασία. Ως εκ τούτου, το κεντρικό σημείο σε ένα πολυπρακτορικό σύστημα είναι η διαπραγμάτευση. Οι πράκτορες διαπραγματεύονται απευθείας μεταξύ τους και πρέπει να καταλήξουν σε κάποια συμφωνία εντός ενός προκαθορισμένου χρονικού πλαισίου (Ren et. al., 2003).

Οι πράκτορες πρέπει να ακολουθούν ορισμένα πρωτόκολλα επικοινωνίας προκειμένου να έχουν την δυνατότητα να ανταγωνιστούν αποτελεσματικά. Πρέπει να υπάρχει μια κοινή γλώσσα επικοινωνίας, η οποία θα επιτρέπει σε κάθε πράκτορα να καταλάβει και να ανταποκριθεί στις αιτήσεις των άλλων πρακτόρων. Μια τέτοια τυποποιημένη γλώσσα επικοινωνίας πρακτόρων, η οποία παρέχει την απαιτούμενη αλληλεπίδραση είναι η KQML (Knowledge Query and Manipulation Language). Η KQML είναι μια γλώσσα με την οποία οι πράκτορες μπορούν να επικοινωνούν και να ανταλλάσουν πληροφορίες. Είναι ανεξάρτητη από τη μορφή της πληροφορίας και έτσι οι εκφράσεις της συχνά περιέχουν υπο-εκφράσεις άλλων γλωσσών περιεχομένου.

Ένας χρήστης μπορεί να ρυθμίσει τη διαμόρφωση και τους περιορισμούς του πράκτορα του, χωρίς αυτό να γίνει γνωστό στον υπόλοιπο κόσμο. Η μη αποκάλυψη της πραγματικής αξίας μιας προτίμησης κατά της διαδικασία της επίλυσης είναι κάτι το οποίο οι άλλες τεχνικές επίλυσης που περιεγράφηκαν προηγουμένως δεν διαθέτουν. Στις προηγούμενες τεχνικές για την επίλυση του ωρολογίου προγράμματος, όλοι οι περιορισμοί είναι γνωστοί ευρέως επειδή όλοι οι υπολογισμοί γίνονται σε ένα και μόνο κεντρικό σημείο. Το πολυπρακτορικό σύστημα

επιτρέπει την ύπαρξη τοπικών περιορισμών, οι οποίοι είναι γνωστοί μόνο στον πράκτορα του χρήστη και όχι σε όλους τους υπόλοιπους.

Αξιοποιώντας τα χαρακτηριστικά των πρακτόρων, μια σειρά από πράκτορες μπορούν να χρησιμοποιηθούν για την επίλυση προβλημάτων με αλληλοσυγκρουόμενους στόχους και περιορισμούς, όπως συμβαίνει στην περίπτωση του ωρολόγιού προγράμματος. Αυτό γίνεται μέσω της διαπραγμάτευσης (Negotiation) και της συνεργασίας (Collaboration) και οφείλεται στο γεγονός ότι οι πράκτορες προσφέρουν ένα μοντέλο κατανεμημένης δημιουργίας του προγράμματος, με την έννοια κατανεμημένο να προκύπτει από την παρουσία ξεχωριστών αυτόνομων συστατικών, τα οποία δεν επικοινωνούν με κάθε λεπτομέρεια, αλλά χρειάζεται να ανταλλάξουν μόνο κάποιες κρίσιμες πληροφορίες. Οι λύσεις προέρχονται από τη συλλογή και επεξεργασία των εκδηλώσεων ενδιαφέροντος και του βαθμού συμφωνίας των επιμέρους πρακτόρων. Στην πραγματικότητα η λύση προέρχεται μέσω μιας διαδικασίας διαπραγμάτευσης, στην οποία όλα τα εμπλεκόμενα μέρη αναζητούν τις καλύτερες λύσεις και ψάχνουν τρόπους να ικανοποιήσουν τα αιτήματα ενός άλλους πράκτορα. Ως εκ τούτου, λαμβάνοντας υπόψιν τις σημαντικές και ισχυρές ιδιότητες που παρέχονται από την τεχνολογία πρακτόρων, είναι σαφές ότι η εφαρμογή ενός μοντέλου πολλαπλών πρακτόρων για τη δημιουργία του ωρολόγιού προγράμματος, έχει την ικανότητα να αντιμετωπίσει τις δύσκολες καταστάσεις και πραγματικές ανάγκες που προκύπτουν (Wangmaeteekul, 2011).

4.3 Τεχνικές Διαπραγμάτευσης

Ένα πολυπρακτορικό σύστημα είναι ένα σύστημα που σχεδιάστηκε και υλοποιήθηκε ως ένα σύνολο πρακτόρων που αλληλοεπιδρούν, δηλαδή συνεργάζονται, συντονίζονται και διαπραγματεύονται. Στον πολυπρακτορικό σχεδιασμό, οι πράκτορες συντάσσουν ένα πλάνο ενεργειών, βάσει του οποίου θα επιλύσουν το πρόβλημα. Κατά τη διάρκεια της εκτέλεσης το πλάνο αναθεωρείται βάσει των νέων στοιχείων και αποτελεσμάτων. Η διαπραγμάτευση (negotiation) αφορά τη διαδικασία με την οποία οι πράκτορες με προσωπικούς, πιθανά αλληλοσυγκρουόμενους/ανταγωνιστικούς στόχους θα φτάσουν σε κοινά ωφέλιμες συμφωνίες. Έχουν προταθεί κατά καιρούς διάφορες τεχνικές και πρωτόκολλα διαπραγμάτευσης όπως, α) πρωτόκολλα που βασίζονται στη θεωρία των παιγνίων (game theory based negotiation), β) πρωτόκολλα που βασίζονται στη θεωρία πλάνων και γ) πρωτόκολλα που βασίζονται στην κατανεμημένη ικανοποίηση περιορισμών (distributed constraint satisfaction). Μία από τις απλούστερες κατηγορίες πρωτοκόλλων διαπραγμάτευσης είναι τα πρωτόκολλα πλειστηριασμού (auctions protocols) που περιγράφονται παρακάτω.

Οι πλειστηριασμοί αποτελούν ουσιαστικά πρωτόκολλα μέσω των οποίων διακινούνται αγαθά, όπως και στον πραγματικό κόσμο. Στόχος είναι να καθοριστεί τόσο η τιμή του αγαθού προς διάθεση, όσο και σε ποιον πράκτορα θα διατεθεί. Σε έναν πλειστηριασμό υπάρχουν δύο

εμπλεκόμενα μέρη: ο εκπλειστηριαστής (auctioneer) ο οποίος θέλει να διαθέσει το συγκεκριμένο αγαθό στη μεγαλύτερη δυνατή τιμή και οι πλειοδότες (bidders), οι οποίοι αγωνίζονται για την απόκτηση του αγαθού που διατίθεται, φυσικά στη χαμηλότερη δυνατή τιμή. Οι πλειοδότες κάνουν μία ή περισσότερες προσφορές (bids) και το αγαθό διατίθεται στον πράκτορα που έδωσε τη μεγαλύτερη προσφορά. Όπως είναι προφανές κάθε εμπλεκόμενος πράκτορας σε έναν πλειστηριασμό έχει τον προσωπικό στόχο που πρέπει να ικανοποιήσει, ο οποίος είναι ανταγωνιστικός ως προς τους στόχους των υπόλοιπων πρακτόρων. Οι μηχανισμοί με τους οποίους καθορίζεται ο τελικός πλειοδότης διαφέρουν στα ακόλουθα σημεία:

- Αν οι προσφορές είναι ανοικτές (open cry) ή κλειστές (sealed bid), δηλαδή αν οι πράκτορες γνωρίζουν τις προσφορές των άλλων πλειοδοτών - πρακτόρων.
- Αν γίνεται μόνο ένας κύκλος προσφορών (one shot) ή περισσότεροι. Στη δεύτερη περίπτωση μπορεί η τιμή των προσφορών σε κάθε κύκλο να αυξάνεται ή να μειώνεται.
- Ο τρόπος με τον οποίο καθορίζεται η τελική τιμή του αγαθού, δηλαδή αν ο τελικός πλειοδότης καλείται να καταβάλλει την τιμή της προσφοράς του ή τη δεύτερη χαμηλότερη τιμή.

Υπάρχουν τέσσερα είδη πλειστηριασμών: οι πλειστηριασμοί κλειστών προσφορών ενός κύκλου (one shot sealed bid auctions), οι πλειστηριασμοί Αγγλικού τύπου (English auctions), οι πλειστηριασμοί Ολλανδικού τύπου (Dutch auctions) και οι πλειστηριασμοί Vickrey (Vickrey auctions).

- Πλειστηριασμοί κλειστών προσφορών ενός κύκλου: Αποτελούν την απλούστερη μορφή πλειστηριασμών. Οι πλειοδότες καταθέτουν τις κλειστές προσφορές τους στον εκπλειστηριαστή, ο οποίος είναι υπεύθυνος για τον έλεγχο των προσφορών και την ανάδειξη του νικητή. Το αγαθό διατίθεται στον πράκτορα με τη μέγιστη προσφορά στην τιμή που προσέφερε.
- Πλειστηριασμοί Αγγλικού τύπου: Είναι ίσως το πλέον συχνό είδος πλειστηριασμών στον πραγματικό κόσμο, για παράδειγμα είναι το είδος του πλειστηριασμού που υιοθετείται στο διαδίκτυο, όπως για παράδειγμα στο e-Bay. Οι προσφορές είναι ανοιχτές και ο πλειστηριασμός λαμβάνει χώρα σε περισσότερους του ενός κύκλους. Οι προσφορές ξεκινούν από μία χαμηλή τιμή και οι πράκτορες έχουν δικαίωμα να κάνουν μία προσφορά με τιμή μεγαλύτερη της τρέχουσας. Μέσα από διαδοχικούς κύκλους προσφορών η τιμή ανεβαίνει έτσι ώστε κάποια στιγμή οι προσφορές σταματούν. Ο νικητής του πλειστηριασμού είναι εκείνος ο οποίος έχει κάνει τη μεγαλύτερη (τελευταία) προσφορά και καλείται να καταβάλλει το αντίστοιχο τίμημα που προσέφερε.

- Πλειστηριασμοί Ολλανδικού τύπου: Παρόμοιοι με τους πλειστηριασμούς Αγγλικού τύπου, με τη διαφορά ότι ο εκπλειστηριαστής ξεκινά τη διαδικασία από μία αρκετά υψηλή τιμή την οποία ανακοινώνει στους πλειοδότες αναμένοντας προσφορά από κάποιον πλειοδότη. Σε περίπτωση που κανένας από τους πλειοδότες δεν προχωρήσει σε προσφορά, τότε ο εκπλειστηριαστής μειώνει την τιμή και ξεκινά ένας νέος κύκλος πλειστηριασμού. Το αγαθό διατίθεται σε εκείνον που κάνει την πρώτη προσφορά.
- Πλειστηριασμοί Vickrey: Αποτελούν ίσως το πλέον ενδιαφέρον και σπάνιο είδος πλειστηριασμών. Είναι πλειστηριασμοί ενός κύκλου, κλειστού τύπου, όπου νικητής είναι εκείνος που κατέθεσε τη μεγαλύτερη προσφορά αλλά καλείται να καταβάλλει το τίμημα της δεύτερης μεγαλύτερης προσφοράς. Αν και αυτό φαίνεται περίεργο με την πρώτη ματιά, εξασφαλίζει ότι οι πράκτορες θα προσφέρουν τη μέγιστη δυνατή τους τιμή για το αγαθό, δηλαδή τη μέγιστη τιμή την οποία μπορούν να καταβάλλουν για το αγαθό βάσει των ιδιαίτερών τους κριτηρίων (Βλαχάβας κ. α., 2011, Craenen et. al., 2010)

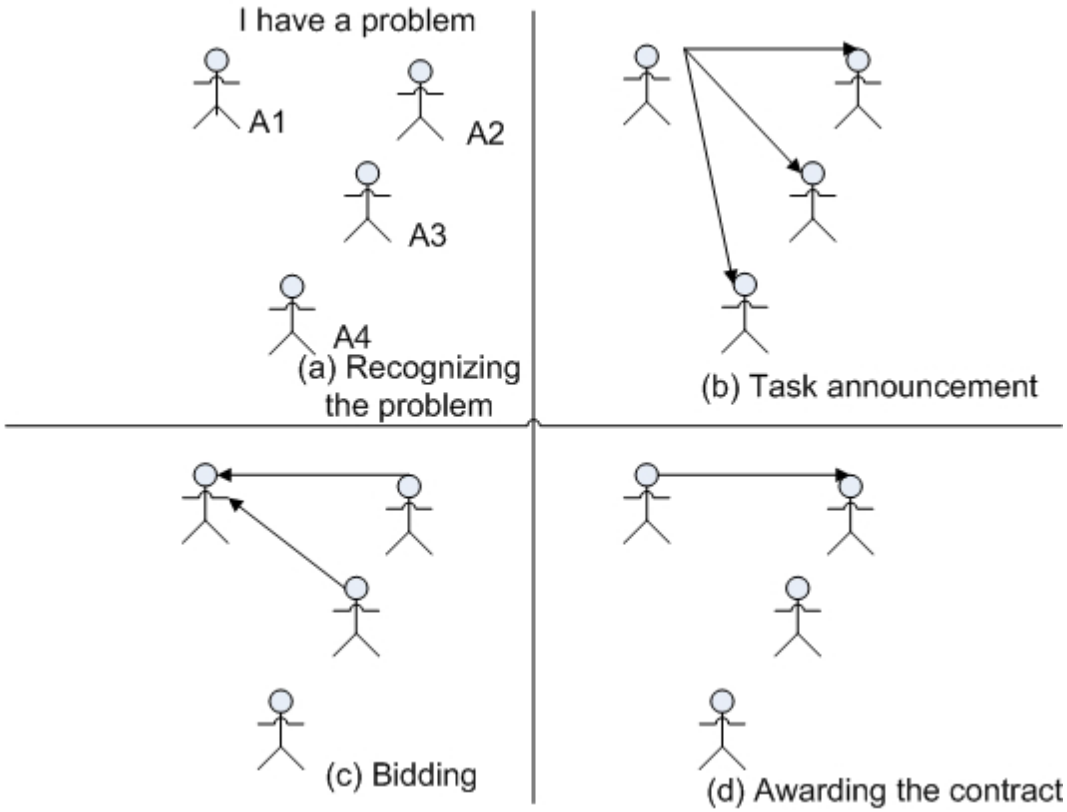
4.3.1 Contract Net Protocol

Οι πράκτορες εκφράζουν έμμεσα τις προτιμήσεις τους, προσφέροντας πόρους στους άλλους πράκτορες ή κάνοντας προσφορά για υπάρχοντες πόρους που προσφέρονται από τους άλλους πράκτορες. Αυτή η διαδικασία γίνεται μέσω του μηχανισμού Contract Net Protocol.

Το Contract Net Protocol (CNP) είναι ένας μηχανισμός συντονισμού, ο οποίος χρησιμοποιείται από ένα διαχειριστή, ο οποίος ανακοινώνει τις προσφορές για μια σύμβαση. Στη συνέχεια συλλέγει τις προσφορές από τους αναδόχους πράκτορες και ανακοινώνει την ανάθεση σε κάποιον πράκτορα. Η διαδικασία αυτή συνεχίζεται μέχρι την ολοκλήρωση των στόχων. Το πρωτόκολλό αλληλεπίδρασης καθορίζει τέσσερα βήματα για κάθε σύμβαση μεταξύ διαχειριστή και αναδόχου:

1. Ο διαχειριστής στέλνει μια πρόταση υποβολής προτάσεων (Call For Proposal, CFP).
2. Κάθε ανάδοχος βλέπει τις CFP και αν είναι στις δυνατότητες του, τότε κάνει προσφορά.
3. Ο διαχειριστής επιλέγει την καλύτερη προσφορά και συνάπτει σύμβαση με τον ανάδοχο.
4. Ο διαχειριστής απορρίπτει όλες τις υπόλοιπες προσφορές.

Η δομή και ο τρόπος λειτουργίας του Contract Net Protocol φαίνεται στην παρακάτω εικόνα.



Εικόνα 6.Contract Net Protocol (Wangmaeteekul, 2011)

Όλοι οι πράκτορες έχουν τη δυνατότητα να στέλνουν CFP και να υποβάλουν προσφορές. Σε κάθε επανάληψη ένας πράκτορας επιλέγεται και αναλαμβάνει το ρόλο του διαχειριστή. Στην περίπτωση του ωρολόγιού προγράμματος, οι CFP αφορούν χρονοθυρίδες που προσφέρονται από τους άλλους πράκτορες. Οι υπόλοιποι πράκτορες λαμβάνουν το ρόλο του αναδόχου και εξετάζουν τη χρονοθυρίδα που προσφέρεται. Στη συνέχεια είτε κάνουν προσφορά για αυτήν είτε προτείνουν μια άλλη για ανταλλαγή. Ο διαχειριστής επιλέγει την προσφορά που μεγιστοποιεί την ικανότητα του και ενημερώνει κατάλληλα το ωρολόγιο πρόγραμμα (Wangmaeteekul, 2011).

4.4 FIPA

Η FIPA (Foundation for Intelligent Physical Agents) είναι ένας οργανισμός της IEEE Standard Society, ο οποίος προωθεί πρότυπα, τα οποία έχουν σχέση με την τεχνολογία πρακτόρων, καθώς και με θέματα διαλειτουργικότητας των προτύπων αυτών σε σχέση με άλλες τεχνολογίες. Η FIPA δημιουργήθηκε αρχικά το 1996 στην Ελβετία, ως ένας οργανισμός για την παραγωγή προτύπων και προδιαγραφών λογισμικού για ετερογενείς και αλληλεπιδραστικούς πράκτορες, καθώς επίσης και για συστήματα τα οποία βασίζονται σε πράκτορες. Στις 8 Ιουνίου 2005 έγινε δεκτός από την IEEE ως ο παγκόσμιος οργανισμός δημιουργίας προτύπων για πράκτορες και για πολυπρακτορικά συστήματα.

Πολλές από τις ιδέες που δημιουργήθηκαν και αναπτύχθηκαν από τη FIPA, έρχονται συνεχώς στο προσκήνιο ως νέες γενιές της Web / Internet τεχνολογίας. Δημιουργούνται πρότυπα για πράκτορες και πολυπρακτορικά συστήματα στο ευρύτερο πλαίσιο της ανάπτυξης εφαρμογών λογισμικού. Με λίγα λόγια, η τεχνολογία των πρακτόρων προσπαθεί να συνεργαστεί, προκειμένου να ενσωματωθεί σε άλλες τεχνολογίες, οι οποίες δεν χρησιμοποιούν πράκτορες. Οι προδιαγραφές -της FIPA αντιπροσωπεύουν μια συλλογή από πρότυπα, τα οποία προορίζονται για την προώθηση της διαλειτουργικότητας των ετερογενών πρακτόρων και των υπηρεσιών που εκπροσωπούν (www.fipa.org).

4.4.1 FIPA-ACL

Οι πιο γνωστές γλώσσες επικοινωνίας μεταξύ των πρακτόρων είναι οι KQML και FIPA-ACL. Η FIPA έχει υιοθετήσει την FIPA-ACL ως την επίσημη γλώσσα επικοινωνίας πρακτόρων. Ένα μήνυμα FIPA-ACL αποτελείται από μια ή περισσότερες παραμέτρους, οι οποίες είναι απαραίτητες για την επικοινωνία. Το πλήθος και το είδος των παραμέτρων ποικίλει ανάλογα με την περίσταση. Οι παράμετροι που ενδέχεται να υπάρχουν σε ένα τέτοιο μήνυμα φαίνονται στον παρακάτω πίνακα.

Πίνακας 1 Παράμετροι μηνύματος ACL (Wiley, 2004)

Παράμετρος	**Περιγραφή**
performative	Τύπος της επικοινωνίας
sender	Ταυτότητα αποστολέα
receiver	Ταυτότητα παραλήπτη
reply-to	Πράκτορας που προκάλεσε το μήνυμα
content	Περιεχόμενο μηνύματος
language	Γλώσσα παραμέτρου περιεχομένου
encoding	Κωδικοποίηση μηνύματος περιεχομένου
ontology	Αναφορά σε οντολογία
protocol	Πρωτόκολλο αλληλεπίδρασης
conversation-id	Μοναδική ταυτότητα της επικοινωνίας
reply-with	Έκφραση που χρησιμοποιείται από τον παραλήπτη για την αναγνώριση του μηνύματος
in-reply-to	Αναφορά σε προηγούμενο μήνυμα
reply-by	Αναφορά σε ώρα και ημερομηνία που αναμένεται απάντηση

Η μόνη υποχρεωτική παράμετρος είναι η Performative. Ωστόσο, τα περισσότερα ACL μηνύματα περιέχουν επίσης αποστολές, παραλήπτη και περιεχόμενο. Η FIPA-ACL ορίζει την επικοινωνία με όρους που δηλώνουν το είδος της λειτουργίας ή της ενέργειας που επιτελεί και ονομάζεται επικοινωνιακή πράξη (Communicative Act, CA). Το σύνολο των τυποποιημένων CA που χρησιμοποιούνται φαίνεται στον ακόλουθο πίνακα:

Πίνακας 2. Επικοινωνιακές πράξεις FIPA (Wiley, 2004)

Επικοινωνιακές πράξεις FIPA	**Περιγραφή**
Accept Proposal	Αποδοχή πρότασης που είχε υποβληθεί νωρίτερα για την εκτέλεση κάποιας ενέργειας
Agree	Συμφωνία για την εκτέλεση κάποιας ενέργειας, πιθανόν στο μέλλον
Cancel	Ενημέρωση σε έναν πράκτορα ότι ο αποστολέας δεν επιθυμεί πλέον την εκτέλεση κάποιας συμφωνημένης ενέργειας
Call for Proposal	Πρόσκληση για την υποβολή προτάσεων όσον αφορά μια συγκεκριμένη ενέργεια
Confirm	Επιβεβαίωση από τον αποστολέα ότι μια δοσμένη πρόταση είναι αληθινή, όταν ο παραλήπτης έχει αμφιβολίες
Disconfirm	Ο αποστολέας ενημερώνει τον παραλήπτη ότι μια δοσμένη πρόταση είναι ψευδής, όταν ο παραλήπτης έχει αμφιβολίες
Failure	Ενημέρωση του παραλήπτη ότι μια προηγούμενη ενέργεια του ήταν αποτυχημένη
Inform	Ενημέρωση του παραλήπτη ότι ελήφθη η πρόταση ελήφθη και ήταν αληθινή
Inform If	Μακροεντολή δράσης προκειμένου να ενημερωθεί ή όχι ο παραλήπτης για την ορθότητα μιας πρότασης
Inform Ref	Μακροεντολή δράση προκειμένου να ενημερώθει ο αποδέκτης για κάποιο αντικείμενο που πιστεύεται από τον αποστολέα ότι αντιστοιχεί σε μια συγκεκριμένη Περιγραφή,
Not Understood	Ενημέρωση του παραλήπτη ότι δεν κατάλαβε μια ενέργεια που μόλις πραγματοποιήθηκε (για παράδειγμα ένα μήνυμα που έλαβε)
Propagate	Ο αποστολέας θέλει ο παραλήπτης να αναγνωρίσει τους πράκτορες που περιγράφονται στο συγκεκριμένο

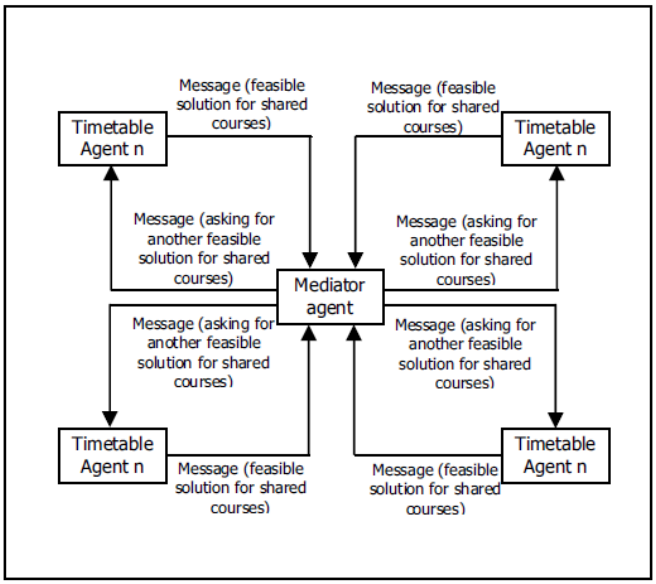
	ενσωματωμένο μήνυμα και να αποστείλει τα ίδιο σε αυτούς
Propose	Υποβολή πρότασης για την εκτέλεση κάποιας ενέργειας κάτω από ορισμένες προϋποθέσεις
Proxy	Ο αποστολέας θέλει ο παραλήπτης να επιλέξει τον πράκτορα στόχο που υποδηλώνεται από μια περιγραφή που δίνεται και να του αποστείλει ένα ενσωματωμένο μήνυμα
Query If	Ερώτηση προς τον αποστολέα εάν μια συγκεκριμένη πρόταση είναι αληθινή ή όχι
Query Ref	Ερώτηση στον παραλήπτη σχετικά με ένα αντικείμενο που αναφέρεται σε μια αναφορά ενός μηνύματος
Refuse	Άρνηση εκτέλεσης κάποιας ενέργειας και αναφορά των λόγων που οδήγησαν σε αυτή
Reject Proposal	Απόρριψη μιας πρότασης για την εκτέλεση κάποιας ενέργειας κατά τη διάρκεια μιας διαπραγμάτευσης
Request	Ο αποστολέας ζητά από τον παραλήπτη να εκτελέσει κάποια ενέργεια
Request When	Ο αποστολέας ζητάει από τον παραλήπτη να εκτελέσει κάποια ενέργεια όταν πραγματοποιηθεί μια δεδομένη πρόταση
Request Whenever	Ο αποστολέας ζητάει από τον παραλήπτη να εκτελεί κάποια ενέργεια όποτε μια συγκεκριμένη πρόταση γίνεται πραγματικότητα
Subscribe	Ενημερώσει για την τιμή μιας αναφοράς και αίτημα να ενημερώνεται για οποιαδήποτε αλλαγή στην ίδια τιμή

4.5 Σχετικές υλοποιήσεις με χρήση πολυπρακτορικών συστημάτων

Τα τελευταία χρόνια, η χρήση της τεχνολογίας πρακτόρων κερδίζει συνεχώς έδαφος ως μία ενδεδειγμένη λύση για την επίλυση του προβλήματος δημιουργίας ωρολόγιού προγράμματος για Ανώτατα Εκπαιδευτικά Ιδρύματα. Οι πράκτορες είναι ικανοί να πραγματοποιήσουν αυτόνομες δράσεις και να εργαστούν για λογαριασμό του εκπροσώπου τους προκειμένου να υλοποιήσουν τους στόχους για τους οποίους έχουν σχεδιαστεί. Έχουν επίσης τη δυνατότητα να συνεργάζονται μεταξύ τους και να διαπραγματεύονται προκειμένου να επιλύσουν

αλληλοσυγκρουόμενες προσδοκίες και περιορισμούς που τίθενται. Βασίζονται σε μια κατανεμημένη επίλυση του προβλήματος, η οποία βελτιώνει σημαντικά τους χρόνους επεξεργασίας. Προσφέρουν επίσης τη δυνατότητα διαπραγμάτευσης, το οποίο δεν ήταν εφικτό με τις προηγούμενες τεχνικές επίλυσης, οι οποίες συζητήθηκαν στο κεφάλαιο 3. Γι' αυτό το λόγο και προσπαθώντας επίσης να διορθώσουν τις ατέλειες των προηγούμενων τεχνικών, έχουν γραφτεί αρκετές εργασίες και άρθρα τα οποία προτείνουν διαφορετικές προσεγγίσεις για την επίλυση του προβλήματος. Ορισμένες από αυτές περιγράφονται παρακάτω.

1. Οι Obit, Landa-Silva, Ouelhadz, Vun και Alfred (Obit et. al., 2011) έχουν αναπτύξει ένα πολυπρακτορικό σύστημα για τη δημιουργία ωρολόγιού προγράμματος σε ένα πανεπιστημιακό ίδρυμα το οποίο αποτελείται από έναν αριθμό τμημάτων. Η προσέγγιση αυτή, η αρχιτεκτονική της οποία φαίνεται στο παρακάτω σχήμα (Εικόνα 7), αποτελείται από δυο τύπους πρακτόρων:



Εικόνα 7. Σχεδιασμός πολυπρακτορικού συστήματος (Obit et. al., 2011)

- Πράκτορας Μεσάζοντας (Mediator Agent, MA). Ο πράκτορας αυτός λειτουργεί ως ενδιάμεσος μεταξύ των πρακτόρων του προγράμματος (TAs). Το κύριο καθήκον του είναι ο συντονισμός της τοπικής κατασκευής του προγράμματος και η αποφυγή των συγκρούσεων μεταξύ των TAs.
- Πράκτορες προγράμματος (Timetabling Agents, TAs). Το κύριο καθήκον τους είναι να ψάχνουν για τοπικές βελτιώσεις του προγράμματος, από τη δική τους οπτική γωνία. Κατά τη διάρκεια κατασκευής του τοπικού προγράμματος, ο κάθε πράκτορας εξετάζει τους περιορισμούς που επιβάλλονται από τον MA σε σχέση με τα κοινά μαθήματα,

όπως επίσης και τους δικούς του τοπικούς περιορισμούς. Εφόσον υπάρχει συμφωνία στα κοινά μαθήματα, τότε ο ΤΑ αναλαμβάνει να βελτιώσει την τοπική λύση.

Το προτεινόμενο μοντέλο αποτελείται από τρία κύρια μέρη:

- Γενικό Προγραμματισμό. Σε αυτό το στάδιο ο ΜΑ παράγει το πρόγραμμα μαθημάτων που μοιράζονται τα επιμέρους τμήματα. Ο ΜΑ και οι TAs συνεργάζονται σε μια κατανεμημένη εργασία αναζήτησης και προσπαθούν να καταλήξουν σε μια συμφωνία για τυχόν αλλαγές, οι οποίες πρέπει να πραγματοποιηθούν σε κοινές χρονοθυρίδες μαθημάτων, προκειμένου να βελτιωθεί η ποιότητα του προγράμματος.
- Τοπικός προγραμματισμός. Οι TAs δημιουργούν τις δικές τους τοπικές λύσεις, χρησιμοποιώντας εποικοδομητικές ευριστικές τεχνικές (constructive heuristic techniques). Σε αυτό το στάδιο οι TAs αποφεύγουν την παραβίαση των ισχυρών περιορισμών που έχουν τεθεί, δεν ασχολούνται όμως με τους χαλαρούς περιορισμούς. Αυτό σημαίνει ότι το πρόγραμμα που δημιουργείται σε αυτή τη φάση είναι εφικτό για το επίπεδο του τμήματος, χωρίς να λαμβάνει όμως υπόψιν του τα κοινά μαθήματα με τα υπόλοιπα τμήματα και τους χαλαρούς περιορισμούς του προγράμματος. Στη συνέχεια οι TAs στέλνουν αυτή την αρχική τους λύση στον ΜΑ και ξεκινά μια διαδικασία διαπραγμάτευσης ανάμεσα στον ΜΑ και τους TAs προκειμένου να φτιαχτεί το κοινό ωρολόγιο πρόγραμμα. Μετά από μια διαδικασία συγκρούσεων και εξεύρεσης λύσεων, οι TAs εφαρμόζουν μια διαδικασία εξελικτικής αναζήτησης προκειμένου να βρουν τη βέλτιστη λύση.
- Διαπραγμάτευση μεταξύ των πρακτόρων. Ο ΜΑ περιμένει μέχρι να λάβει από όλους τους TAs τις αρχικές τους λύσεις. Μόλις τις λάβει, ελέγχει για την ύπαρξη μαθημάτων σε κοινές χρονοθυρίδες. Αν υπάρχουν συγκρούσεις, τότε στέλνει ένα μήνυμα σε όσους TAs εμπλέκονται, ζητώντας τους εναλλακτικές χρονοθυρίδες. Η διαδικασία διαπραγμάτευσης συνεχίζεται μέχρι τη στιγμή που δεν υπάρχουν πλέον συγκρούσεις, ή μέχρι τη στιγμή που οι χαλαροί περιορισμοί δεν μπορούν να διορθωθούν περαιτέρω. Κατά τη διάρκεια της διαπραγμάτευσης ο ΜΑ λαμβάνει υπόψιν του τους χαλαρούς περιορισμούς για να ελέγχει την κατανομή των κοινών μαθημάτων (Obit et. al., 2011).

2. Οι Eddine Nouri και Belkahla Driss (Nouri et. al., 2013) προτείνουν μια διαφορετική προσέγγιση, η οποία αποτελείται από τρεις κλάσεις πρακτόρων:

- Η πρώτη κλάση αποτελείται από τους πράκτορες καθηγητών (Teacher Agents, TAs), η οποία χωρίζεται σε τρεις κατηγορίες καθηγητών. Αυτές είναι οι C1: Professor, Associate Professor, C2: Assistant Professor και C3: Contractual.
- Η δεύτερη κλάση αποτελείται από τους πράκτορες αιθουσών (Classroom Agents, CAs), οι οποίοι χωρίζονται τρεις τύπους αιθουσών ανάλογα με το είδος της αίθουσας.

- Η Τρίτη κλάση αποτελείται από τρεις πράκτορες. 2 πράκτορες διασύνδεσης (Interface Agents), οι οποίοι ονομάζονται I.A1 και I.A2 και έναν πράκτορα ιστορικού, ο οποίος ονομάζεται H.A.

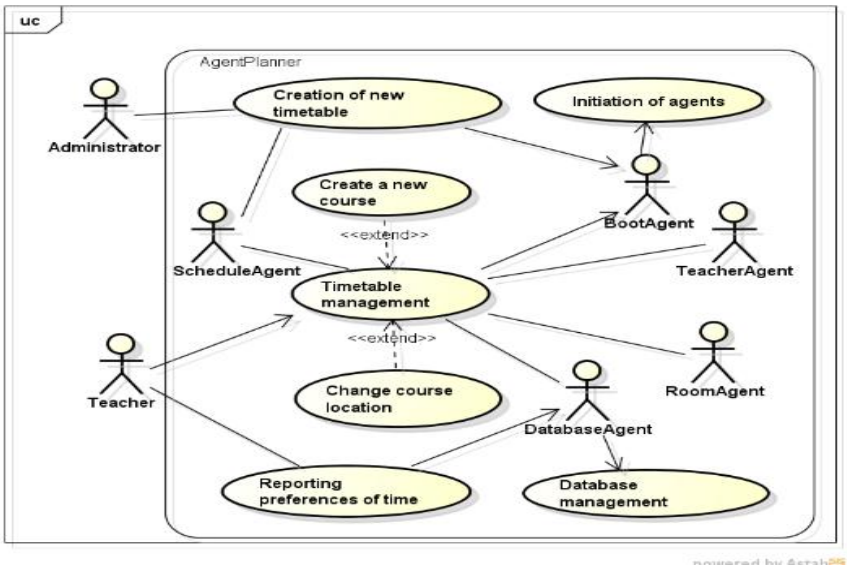
Το μοντέλο υλοποιείται σε τρεις φάσεις:

I. Φάση αρχικοποίησης (Initialization phase). Σε αυτή τη φάση ο I.A1 αρχικοποιεί την εκτέλεση των πρακτόρων του συστήματος. Στην πραγματικότητα επιτρέπει την εκτέλεση όλων των πρακτόρων, με βάση τις αρχικές παραμέτρους που ορίζονται από τον χρήστη.

II. Φάση διαπραγμάτευσης (Negotiation phase). Πρόκειται για τον πυρήνα του μοντέλου. Βασίζεται σε ένα σύστημα ανταλλαγής μηνυμάτων μεταξύ δυο κλάσεων πρακτόρων, των CA και TA, προκειμένου να επιτευχθεί μια συμφωνία μεταξύ τους, τηρώντας όλους τους ισχυρούς περιορισμούς του προβλήματος. Οι TAs ξεκινούν τη διαπραγμάτευση στέλνοντας όλες τις προτάσεις κατανομής, οι οποίες ανακτήθηκαν από τη βάση προτιμήσεων τους στους CAs, προκειμένου να κάνουν μια κράτηση στην αίθουσα που τους ενδιαφέρει για την ημέρα και την ώρα που επιθυμούν. Οι CAs θα λάβουν και θα αναλύσουν τις προτιμήσεις των TAs. Θα ζητήσουν από τον H.A. να εξακριβώσει την ύπαρξη αλληλοεπικαλύψεων των χρονικών περιόδων για τον ίδιο TA. Έτσι σε περίπτωση που δεν εμφανίζονται συγκρούσεις θα επικυρώσουν την κράτηση, ενώ σε διαφορετική περίπτωση θα αποστείλουν στον TA μια νέα αντιπρόταση. Ο κάθε CA μπορεί να έχει περισσότερους από έναν ενδιαφερόμενους TAs για την ίδια χρονοθυρίδα. Σε αυτή την περίπτωση, η αίθουσα μπορεί να αντικατασταθεί από μια άλλη, η οποία διαθέτει τα ίδια χαρακτηριστικά. Εφαρμόζονται επίσης πολλοί τύποι κριτηρίων για την πραγματοποίηση μιας κράτησης, προκειμένου να υπάρξει μείωση των συγκρούσεων μεταξύ των TAs. Τα κριτήρια αυτά μπορεί να είναι το πλήθος των φοιτητών για μια συνεδρία, η κατηγορία του καθηγητή, ο τύπος της τάξης, ο τύπος της συνεδρίας και πολλά άλλα.

III. Διαβίβαση των τελικών αποτελεσμάτων (Transmission of final results). Όταν ο TA λάβει τις λύσεις ως απάντηση στα μηνύματα του, τελειώνει τη φάση των διαπραγματεύσεων και μεταδίδει τα τελικά αποτελέσματα στον I.A1 δημιουργώντας τη φόρμα με το πρόγραμμα του καθηγητή. Στη συνέχεια ο I.A2 τελειώνει τη διαδικασία με τη δημιουργία του τελικού προγράμματος για όλες τις αίθουσες διδασκαλίας (Nouri et. al., 2013).

3. Οι Tkaczyk, Ganzha και Paprzycki έχουν αναπτύξει το AgentPlanner (Tkaczyk et. al., 2013), ένα πολυπρακτορικό σύστημα για τη δημιουργία και διαχείριση του ωρολόγιού προγράμματος καθώς επίσης και μιας σειράς πρόσθετων λειτουργιών. Το σύστημα αποτελείται από τους ακόλουθους πράκτορες (Εικόνα 8):



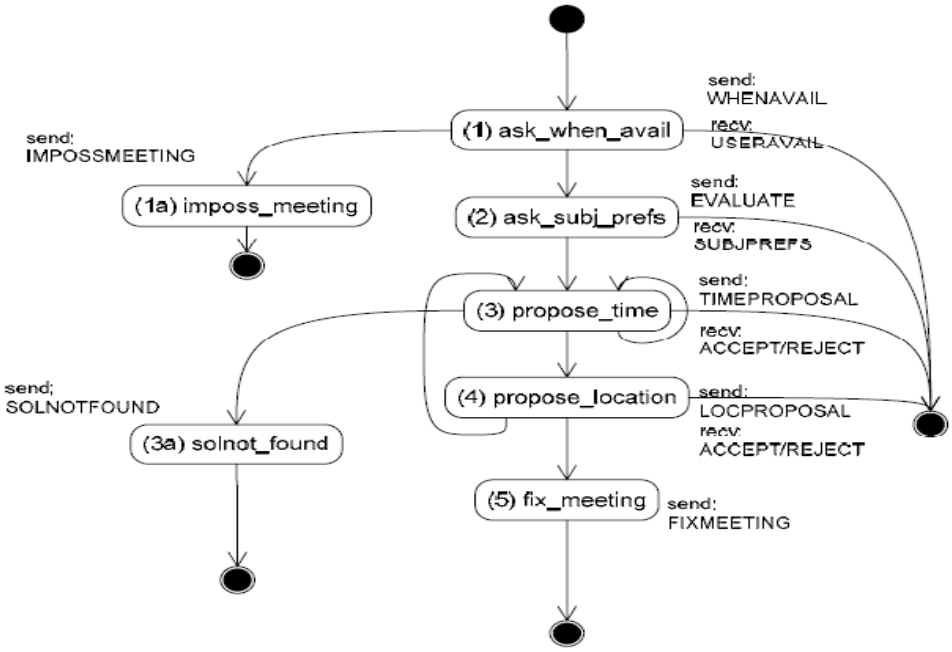
Εικόνα 8.Το διάγραμμα περιπτώσεων χρήσης του AgentPlanner (Tkaczyk et. al., 2013)

- Boot Agent. Η μόνη του λειτουργία είναι η δημιουργία και η εκκίνηση των υπολοίπων πρακτόρων του συστήματος.
- Database Agent. Είναι υπεύθυνος για την επικοινωνία του συστήματος και της βάσης δεδομένων του προγράμματος.
- Rooms Agent. Αντιπροσωπεύουν τις αίθουσες στη διαδικασία προγραμματισμού. Ανακτούν, φιλτράρουν και αποθηκεύουν πληροφορίες σχετικά με τις αίθουσες (τύπος αίθουσας, αριθμός θέσεων) από τη βάση δεδομένων.
- Teacher Agent. Ενεργεί για λογαριασμό του καθηγητή τόσο κατά τη δημιουργία όσο και κατά την διαχείριση του προγράμματος. Αποθηκεύει πληροφορίες σχετικές με τον καθηγητή, όπως λίστα προτιμήσεων (μαθήματα και ομάδες φοιτητών που διδάσκει), λίστα των αιθουσών που πληρούν τις απαιτήσεις του μαθήματος (οι οποίες λαμβάνονται από τους Rooms Agent) και τα αποτελέσματα μιας συνάρτησης αξιολόγησης του τρέχοντος προγράμματος. Ο αλγόριθμος της συνάρτησης αξιολόγησης λαμβάνει υπόψιν του τις προτεραιότητες μαθημάτων και διαλέξεων, τις προτιμήσεις του καθηγητή και την τρέχουσα κατάσταση του προγράμματος, προκειμένου να αξιολογήσει και να βαθμολογήσει τις διαθέσιμες επιλογές.
- Schedule Agent. Είναι ο κεντρικός πράκτορας του αλγορίθμου διαπραγμάτευσης. Γνωρίζει ποιοι Teacher Agents παίρνουν μέρος στις τρέχουσες διαπραγματεύσεις και έχει πρόσβαση στη βάση δεδομένων του προγράμματος μέσω του Database Agent. Όλα τα δεδομένα αποθηκεύονται στη βάση δεδομένων και έτσι ο Schedule Agent γνωρίζει ποια αίθουσα είναι ελεύθερη και ποια καταλυμένη, αν ο κάθε καθηγητής είναι διαθέσιμος για μια χρονοθυρίδα. Υπάρχει όμως το μειονέκτημα ότι σε μεγάλα προβλήματα χρονοπρογραμματισμού ο Schedule Agent ενδέχεται να καθυστερεί λόγω μεγάλου φόρτου εργασίας.

Το AgentPlannner εφαρμόζεται ως ένα σύστημα πελάτη – διακομιστή (client – server), όπου όλες οι εργασίες που αφορούν τη δημιουργία και τη διαχείριση του προγράμματος, τρέχουν στην πλευρά του διακομιστή, ενώ από την πλευρά του πελάτη γίνεται μόνο η αποστολή των αιτημάτων και η αναθεώρηση και πρόσβαση των αποτελεσμάτων.

Κατά τη διαδικασία δημιουργίας του προγράμματος κεντρικό ρόλο παίζει ο Schedule Agent, ο οποίος έχει πρόσβαση στο ωρολόγιο πρόγραμμα, το οποίο αρχικά είναι κενό. Ο Schedule Agent διαπραγματεύεται το πρόγραμμα με τους Teacher Agents χρησιμοποιώντας πληροφορίες από τους Rooms Agents. Οι διαπραγματεύσεις χωρίζονται σε γύρους. Ο αριθμός των γύρων δεν είναι μεγαλύτερος από το συνολικό αριθμό των δραστηριοτήτων (μαθήματα, εργαστήρια, ομάδες εργασίας όλων των καθηγητών). Κατά τη διάρκεια ενός γύρου ο Teacher Agent στέλνει στο Schedule Agent ένα αίτημα ζητώντας μια αίθουσα και μια συγκεκριμένη χρονοθυρίδα για μια δραστηριότητα. Ο Schedule Agent λαμβάνει υπόψιν του όλες τις ληφθείσες αιτήσεις και αποδέχεται ορισμένες από αυτές, τοποθετώντας τη δραστηριότητα στο ωρολόγιο πρόγραμμα, ενώ ταυτόχρονα απορρίπτει άλλες, αναμένοντας νέες προτάσεις. Η απόφαση εξαρτάται από το εάν είναι ελεύθερη στο πρόγραμμα η συγκεκριμένη δραστηριότητα, ή έρχεται σε σύγκρουση με κάποια άλλη δραστηριότητα. Συνήθως πολλοί Teacher Agents ζητούν την ίδια θέση. Στην περίπτωση αυτή ο Schedule Agent επιλέγει αυτόν με την καλύτερη τιμή της συνάρτησης αξιολόγησης. Ο γύρος ολοκληρώνεται είτε όταν η λίστα των αιτήσεων είναι κενή (έχουν προγραμματιστεί όλες οι δραστηριότητες), είτε όταν οι μη τοποθετημένες αιτήσεις δεν μπορούν να ικανοποιηθούν. Πριν την έναρξη του επόμενου γύρου, ο Schedule Agent λαμβάνει μηνύματα από τους Teacher Agents σχετικά με αιτήματα τα οποία έχουν απορριφθεί. Στη συνέχεια διακόπτει τις τρέχουσες διαπραγματεύσεις και τρέχει έναν αλγόριθμο αναδιοργάνωσης του προγράμματος για την ανάπτυξη των απορριφθέντων δραστηριοτήτων στο τρέχων πρόγραμμα (Tkaczyk et. al., 2013).

4. Οι Babkin, Abdulrab και Babkina προτείνουν το AgentTime (Babkin et. al., 2009). Σε αυτή την προσέγγιση υπάρχουν δυο τύποι πρακτόρων. Οι πράκτορες διοργανωτές (Agents Organizers) και οι πράκτορες συμμετέχοντες (Agents Participants). Οι πράκτορες ταξινομούνται επίσης ως πράκτορες καθηγητών, ομάδων και αιθουσών. Οι πράκτορες καθηγητών παίζουν το ρόλο του διοργανωτή. Οι πράκτορες ομάδων καθώς και οι πράκτορες αιθουσών παίζουν το ρόλο του συμμετέχοντα. Οι αριθμοί των πρακτόρων καθηγητών και των πρακτόρων ομάδων αντιστοιχούν στους πραγματικούς αριθμούς εκπαιδευτικών και ομάδων σε ένα Πανεπιστημιακό Ίδρυμα. Ο πράκτορας αίθουσας αντιστοιχεί στο σύνολο των αιθουσών διδασκαλίας που χρησιμοποιούνται στο πρόγραμμα. Μια συλλογική αναζήτηση για την εύρεση του καλύτερου χρόνου και τοποθεσίας πραγματοποιείται και η οποία περιλαμβάνει επικοινωνία μεταξύ των διαφόρων πρακτόρων. Για κάθε μάθημα ο πράκτορας καθηγητή εκτελεί μια σειρά δράσεων, η οποία φαίνεται στο ακόλουθο διάγραμμα (Εικόνα 9):



Εικόνα 9.Το διάγραμμα καταστάσεων του πράκτορα καθηγητή (Babkin et. al., 2009)

Ο πράκτορας μεταβαίνει στις ακόλουθες καταστάσεις:

- Ask_when_avail. Αυτή είναι η πρώτη κατάσταση του πράκτορα καθηγητή. Ο πράκτορας αποστέλλει σε όλους τους πράκτορες ομάδων ένα ερώτημα WHENAVAIL, μαζί με το αναγνωριστικό του μαθήματος, ζητώντας από αυτούς τους διαθέσιμους χρόνους για το μάθημα. Οι πράκτορες των ομάδων θα απαντήσουν με ένα μήνυμα USERAVAIL, με το οποίο ενημερώνουν τον πράκτορα καθηγητή αν είναι ελεύθεροι και διαθέτουν ελεύθερους χρόνους για το μάθημα. Μόλις όλοι οι πράκτορες στείλουν την απάντηση τους, ο πράκτορας καθηγητής βρίσκει τα κοινά χρονικά διαστήματα από τα μηνύματα που έλαβε. Αν δεν υπάρχουν κοινοί χρόνοι, τότε ο πράκτορας μεταβαίνει στην κατάσταση imposs_meeting.
- Imposs_meeting. Όταν ο πράκτορας βρίσκεται σε αυτή την κατάσταση, αυτό σημαίνει ότι δεν κατάφερε να βρει λύσεις με τους χρόνους.
- Ask_subj_prefs. Σε αυτή την κατάσταση ο πράκτορας καθηγητή ζητά από τους πράκτορες ομάδων τις προτιμήσεις τους σε χρόνο και τοποθεσία. Αυτοί αποστέλλουν τις προτιμήσεις τους με ένα μήνυμα SUBJPREFS. Ο πράκτορας καθηγητή ταξινομεί τος ληφθείσες προτιμήσεις σύμφωνα με κάποια προκαθορισμένα κριτήρια και δημιουργεί τη λίστα προτιμήσεων.
- Prospose_time. Ο πράκτορας καθηγητή επιλέγει των πρώτη χρονοθυρίδα από την ταξινομημένη λίστα προτιμήσεων και την αποστέλλει μαζί με το αναγνωριστικό του μαθήματος στους πράκτορες ομάδων μέσω ενός μηνύματος TIMEPROPOSAL. Οι πράκτορες ομάδων ελέγχουν την πρόταση και εάν έχουν ελεύθερη τη διαθέσιμη χρονοθυρίδα, τότε στέλνουν ένα μήνυμα αποδοχής (ACCEPT) στον πράκτορα

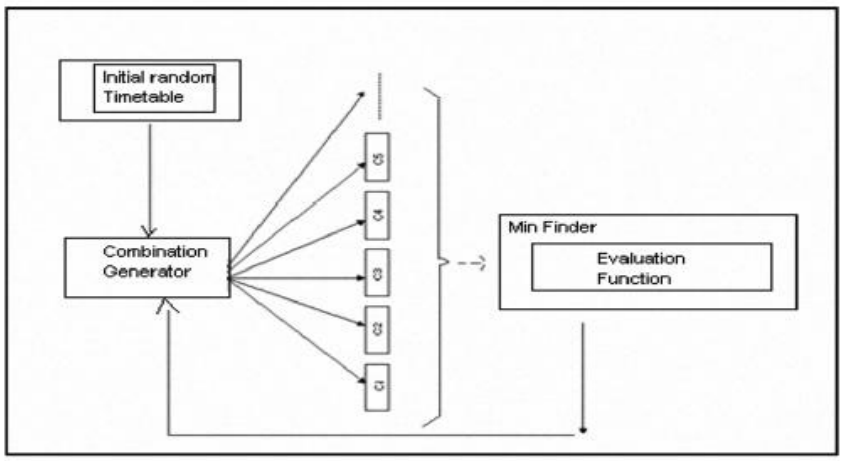
καθηγητή. Στην αντίθετη περίπτωση αποστέλλεται ένα αρνητικό μήνυμα (REJECT). Αν ο πράκτορας καθηγητή λάβει αρνητική απάντηση, τότε αποστέλλει την επόμενη χρονοθυρίδα από την ταξινομημένη λίστα. Η διαδικασία συνεχίζεται μέχρι ο πράκτορας καθηγητής να λάβει καταφατικές απαντήσεις. Μόλις λάβει καταφατικές απαντήσεις, μεταβαίνει στην επόμενη κατάσταση Propose_location. Αν όλες οι επιλογές της λίστας απορριφθούν, τότε ο πράκτορας μεταβαίνει στην κατάσταση Solnot_found.

- Solnot_found. Όταν ο πράκτορας καθηγητή βρίσκεται σε αυτή την κατάσταση, τότε αυτό σημαίνει ότι δεν κατάφερε να βρει λύσεις.
- Propose_location. Ο πράκτορας καθηγητή στέλνει μια ταξινομημένη λίστα με τις αίθουσες διδασκαλίας στον πράκτορα αιθουσών μέσω ενός μηνύματος LOCPROPOSAL. Αυτό το μήνυμα περιέχει επίσης το αναγνωριστικό του μαθήματος και το αναγνωριστικό της χρονοθυρίδας που επιλέχθηκε. Ο πράκτορας αιθουσών ψάχνει στη δική του λίστα για διαθέσιμες αίθουσες με τα ανωτέρω χαρακτηριστικά. Μόλις τα βρει, αποστέλλει στον πράκτορα καθηγητή ένα μήνυμα αποδοχής (ACCEPT) με το αναγνωριστικό της αίθουσας. Σε περίπτωση αποτυχίας αποστέλλει ένα αρνητικό μήνυμα (REJECT). Μόλις ο πράκτορας καθηγητή λάβει καταφατική απάντηση, μεταβαίνει στην κατάσταση Fix_meeting. Αν λάβει αρνητική απάντηση, τότε μεταβαίνει και πάλι στην κατάσταση Propose_time.
- Fix_meeting. Εάν ο πράκτορας καθηγητή βρεθεί σε αυτή την κατάσταση, τότε αυτό σημαίνει ότι τόσο η αίθουσα, όσο και η χρονοθυρίδα για το συγκεκριμένο μάθημα έχουν βρεθεί με επιτυχία. Στη συνέχεια αποστέλλεται ένα μήνυμα FIXMEETING με τα αναγνωριστικά του μαθήματος, της αίθουσας και της χρονοθυρίδας στους πράκτορες ομάδων και στους πράκτορες αιθουσών. Εάν αυτοί συμφωνούν, τότε κανονίζεται το μάθημα και τοποθετείται στο πρόγραμμα. Σε διαφορετική περίπτωση αποστέλλεται στον πράκτορα καθηγητή ένα μήνυμα ακύρωσης CANCELMEETING και η διαδικασία επαναλαμβάνεται.

Όταν ο πράκτορας καθηγητή βρίσκεται σε μια από τις προηγούμενες καταστάσεις και δεν λάβει κάποια απάντηση από τους άλλους πράκτορες μέσα σε ένα προκαθορισμένο χρονικό διάστημα, τότε τοποθετεί το συγκεκριμένο μάθημα σε μια άλλη λίστα, προκειμένου να προσπαθήσει ξανά αργότερα. Η διαδικασία επαναλαμβάνεται για όλη τη λίστα των μαθημάτων του πράκτορα καθηγητή. Μόλις όλοι οι πράκτορες καθηγητή τελειώσουν επιτυχώς με τις λίστες μαθημάτων τους, τότε το πρόγραμμα θεωρείται ολοκληρωμένο (Babkin et. al., 2009).

5. Οι Nandhini και Kanmani (Nandhini et. al., 2009) προτείνουν ένα άλλο μοντέλο για τη δημιουργία του ωρολόγιού προγράμματος, στο οποίο προκειμένου να κατανεμηθεί ο φόρτος εργασίας, χρησιμοποιούνται δυο συνεργαζόμενοι πράκτορες, οι οποίοι έχουν διαφορετικά χαρακτηριστικά και στόχους. Οι πράκτορες αυτοί είναι (Εικόνα 10):

- CombinationGenerator. Ο πράκτορας αυτός σχεδιάζεται με στόχο να δημιουργεί το μέγιστο πλήθος δυνατών συνδυασμών για το πρόγραμμα.
- minFinder. Ο πράκτορας αυτός βρίσκει το συνδυασμό με την καλύτερη τιμή μιας συνάρτησης αξιολόγησης από όλους τους συνδυασμούς που δημιούργησε ο CombinationGenerator.



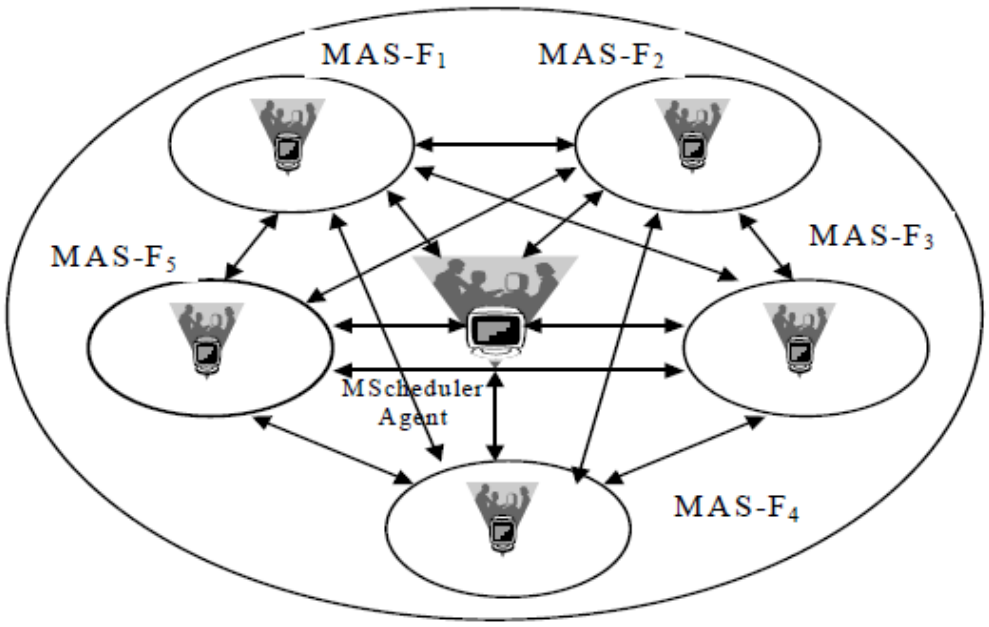
Εικόνα 10.Η αλληλεπίδραση του πολυπρακτορικού συστήματος (Nandhini et. al., 2009)

Ο CombinationGenerator σχεδιάζεται λαμβάνοντας υπόψιν του όλους τους ισχυρούς περιορισμούς που τέθηκαν για τη δημιουργία του προγράμματος και κάνει χρήση κάποιων προεπιλεγμένων ευριστικών τεχνικών, βρίσκει όλους τους δυνατούς συνδυασμούς που πληρούν τους προηγούμενους περιορισμούς. Προκειμένου να υπάρχει μια ισονομία στο πρόγραμμα, δυο διαδοχικές χρονοθυρίδες δεν θα πρέπει να έχουν το ίδιο θέμα και ο φόρτος εργασίας των καθηγητών κατά τη διάρκεια μιας εβδομάδας θα πρέπει να κατανέμεται ομοιόμορφα, δηλαδή τα μαθήματα να μοιράζονται σε όλες τις ημέρες της εβδομάδας, όλοι οι καθηγητές να έχουν και πρωινά και απογευματινά μαθήματα κ.α.

Ο πράκτορας minFinder σχεδιάζεται λαμβάνοντας υπόψιν του όλους τους χαλαρούς περιορισμούς που τέθηκαν για τη δημιουργία του προγράμματος, προκειμένου να βρει το πρόγραμμα το οποίο παραβιάζει τους λιγότερους από αυτούς. Προκειμένου να πετύχει το σκοπό του χρησιμοποιεί μια συνάρτηση αξιολόγησης για να αξιολογήσει τους προτεινόμενους συνδυασμούς. Η συνάρτηση αυτή ανάλογα με τη σπουδαιότητα που έχουν οι χαλαροί περιορισμοί, τους αναθέτει μια τιμή βάρους και κρίνοντας εάν ικανοποιήθηκαν ή όχι μια άλλη τιμή 0 ή 1. Ο minFinder υπολογίζει και επιλέγει το συνδυασμό με την μικρότερη τιμή. Στη συνέχεια ο συνδυασμός αυτός αποστέλλεται και πάλι στον CombinationGenerator προκειμένου να υποστεί περαιτέρω αναζήτηση (Nandhini et. al., 2009).

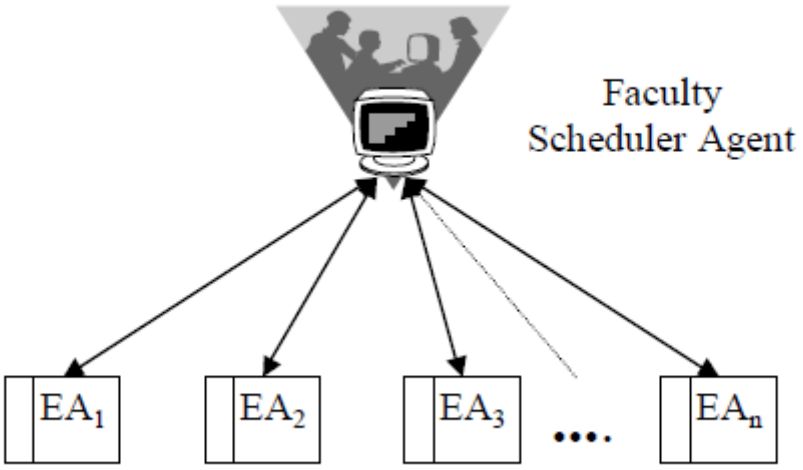
6. Ο M. Oprea (Oprea, 2006) πρότεινε ένα πολυπρακτορικό σύστημα (MAS) για τη δημιουργία του ωρολόγιού προγράμματος σε ένα πανεπιστήμιο με πολλές σχολές. Στο μοντέλο που χρησιμοποιείται θεωρεί ότι το πανεπιστήμιο αποτελείται από πέντε σχολές και κάθε σχολή έχει έναν αριθμό ειδικοτήτων καθηγητών. Το πρόγραμμα για κάθε ειδικότητα γίνεται από έναν πράκτορα, ο οποίος ονομάζεται Specialization Course Scheduler. Αυτός λαμβάνει μια λίστα προτιμήσεων από τον προσωπικό πράκτορα κάθε καθηγητή που διδάσκει ένα μάθημα στη συγκεκριμένη ειδικότητα. Ο κατάλογος προτιμήσεων περιλαμβάνει τις προτιμήσεις του καθηγητή ταξινομημένες σύμφωνα με τις επιθυμίες του, καθώς επίσης και μια λίστα με ημέρες και ώρες στις οποίες δεν μπορεί να παρευρεθεί. Αφού δημιουργήσει το πρόγραμμα για κάθε σχολή, αυτό αποστέλλεται στον Faculty Scheduler Agent προκειμένου να δημιουργηθεί το πρόγραμμα με τις ημέρες και ώρες σε επίπεδο σχολής. Μόλις δημιουργηθεί το πρόγραμμα για κάθε σχολή ξεκινάει η κατανομή των αιθουσών σε επίπεδο πανεπιστημίου. Όταν όλα τα μαθήματα έχουν κατανεμηθεί στα κατάλληλα χρονικά διαστήματα και στις κατάλληλες αίθουσες, τότε το ωρολόγιο πρόγραμμα έχει ολοκληρωθεί. Κάθε φορά που παρουσιάζεται ένα πρόβλημα, ξεκινάει μια διαδικασία επικοινωνίας, η οποία περιλαμβάνει διαπραγμάτευση.

Κάθε σχολή έχει ένα πολυπρακτορικό σύστημα (MAS-Fi), το οποίο προγραμματίζει τα μαθήματα της σχολής. Ο κύριος πράκτορας προγραμματισμού, ο οποίος κάνει την κατανομή στις αίθουσες σε επίπεδο πανεπιστημίου είναι ο MScheduler Agent. Η αρχιτεκτονική του συστήματος φαίνεται στο επόμενο σχήμα (Εικόνα 11).



Εικόνα 11.Η αρχιτεκτονική του συστήματος (Oprea, 2006)

Επειδή ο κάθε καθηγητής μπορεί να διδάσκει σε περισσότερες από μια σχολές, τα πολυπρακτορικά συστήματα κάθε σχολής θα πρέπει να επικοινωνούν μεταξύ τους. Στο επόμενο σχήμα (Εικόνα 12) παρουσιάζεται το πολυπρακτορικό σύστημα (MAS) σε επίπεδο σχολής, το οποίο περιλαμβάνει τον τοπικό πράκτορα δημιουργό του προγράμματος (Faculty Scheduler agent) και έναν αριθμό ειδικών βοηθών (Expert Assistants, EA) για κάθε ειδικότητα της σχολής.



Εικόνα 12.Το MAS σε επίπεδο σχολής (Oprea, 2006)

Για κάθε ειδικότητα υπάρχει ένας ειδικός βοηθός, ο οποίος αναλαμβάνει να πραγματοποιήσει όλες τις δραστηριότητες που σχετίζονται με αυτή την ειδικότητα, συμπεριλαμβανομένου και του σχεδιασμού του προγράμματος στο επίπεδο της σχολής όσον αφορά τις ημέρες και τις ώρες των μαθημάτων. Χρησιμοποιώντας μια σειρά από περιορισμούς, οι οποίοι τίθενται από την αρχή, οι Faculty Scheduler Agents, οι οποίοι ενεργούν αυτόνομα, μπορούν να σχεδιάσουν το ωρολόγιο πρόγραμμα λαμβάνοντας υπόψιν τους τις ιδιαίτερες προτιμήσεις των καθηγητών. Στην ιδανική περίπτωση, όλες οι προτιμήσεις των καθηγητών γίνονται δεκτές. Στην πράξη όμως, δεν συμβαίνει αυτό και οι πράκτορες δεν καταλήγουν σε συμφωνία. Γι' αυτό το λόγο οι πράκτορες ποσοτικοποιούν και ταξινομούν τις προτιμήσεις των καθηγητών. Στη συνέχεια ακολουθείται μια διαδικασία διαπραγμάτευσης. Σε περίπτωση που αυτή δεν έχει επιτυχία ένα πρωτόκολλο πειθούς (persuasion protocol), όπως αυτό που παρουσιάστηκε από τους Ito και Shindani το 1997 (Ito et. al., 1997). Το πλεονέκτημα του πρωτοκόλλου είναι το γεγονός ότι οι πράκτορες μπορούν να φτάσουν σε περισσότερες συμφωνίες σε σχέση με τα υπάρχοντα πρωτόκολλα διαπραγμάτευσης.

Δυο συνηθισμένα παραδείγματα κρίσιμων καταστάσεων που ενδέχεται να παρουσιαστούν είναι:

- Στο σχεδιασμό του προγράμματος σε επίπεδο σχολής υπάρχει σύγκρουση στις προτιμήσεις (ημέρα και ώρα) δυο ή περισσότερων καθηγητών. Σε αυτή την περίπτωση ξεκινάει μια διαπραγμάτευση μεταξύ του ειδικού βοηθού της ειδικότητας των

καθηγητών και των καθηγητών που συμμετέχουν στη σύγκρουση μέσω των προσωπικών τους πρακτόρων. Ο ειδικός βοηθός, τους ενημερώνει μέσω ενός μηνύματος για την σύγκρουση και αναμένει από αυτούς να του στείλουν απαντήσεις. Αν λάβει απάντηση, τότε πραγματοποιείται μια αναδιάταξη του προγράμματος. Σε διαφορετική περίπτωση ο Faculty Scheduler Agent χρησιμοποιεί το persuasion protocol προτείνοντας λύσεις.

- Στη σχεδίαση του προγράμματος σε επίπεδο πανεπιστημίου, δεν υπάρχουν διαθέσιμες αίθουσες προκειμένου να προγραμματιστούν κάποια μαθήματα σε συγκεκριμένη ημέρα και ώρα. Σε αυτή την περίπτωση ο MScheduler Agent ξεκινά μια διαδικασία διαπραγμάτευσης στέλνοντας ένα μήνυμα στους Faculty Scheduler Agents που εμπλέκονται στη σύγκρουση. Ο κάθε Faculty Scheduler Agent περνάει το μήνυμα στους ειδικούς βοηθούς και αυτοί με τη σειρά τους αν δεν μπορέσουν να το λύσουν, το μεταδίδουν στους καθηγητές προκειμένου αυτοί να μπορέσουν να διαπραγματευτούν άμεσα. Αν μετά από αυτή τη διαπραγμάτευση το πρόβλημα παραμένει, τότε ο MScheduler Agent ξεκινά το persuasion protocol προτείνοντας λύσεις στους Faculty Agents, οι οποίοι με τη σειρά τους μεταφέρουν το πρόβλημα σε χαμηλότερα επίπεδα μέχρι αυτό να επιλυθεί (Oprea, 2006).

4.5.1 Αξιολόγηση προηγούμενων τεχνικών

Όλες οι προηγούμενες προσεγγίσεις προσπαθούν να επιλύσουν το πρόβλημα δημιουργίας ωρολογίου προγράμματος για Ανώτατα Εκπαιδευτικά Ιδρύματα. Άλλες σε επίπεδο Σχολής (2, 3, 4, 5) και άλλες σε επίπεδο Ιδρύματος (1, 6). Και πράγματι φαίνεται πως τα καταφέρνουν με πολύ καλά αποτελέσματα. Πιο αποτελεσματικά δείχνει να τα καταφέρνει το Agent Planner των Tkaczyk, Ganzha και Paprzycki στο οποίο κεντρικό ρόλο παίζει ο ένας πράκτορας, ο οποίος είναι υπεύθυνος για τη δημιουργία του προγράμματος και ο οποίος συνεργάζεται με όλους τους υπόλοιπους (Βάσης Δεδομένων, Αιθουσών, Καθηγητών) προκειμένου να υλοποιηθεί το πρόγραμμα. Επίσης, πολύ σημαντικό ρόλο φαίνεται πως παίζει η συνάρτηση αξιολόγησης, η οποία χρησιμοποιείται για να αξιολογηθούν οι εισερχόμενες προτάσεις. Εξίσου αποτελεσματική φαίνεται πως είναι και η προσέγγιση του Oprea, όπου επιλύει το πρόβλημα σε επίπεδο Ιδρύματος. Σε επίπεδο Σχολής οι πράκτορες ποσοτικοποιούν και ταξινομούν τις προτιμήσεις των καθηγητών, γεγονός το οποίο φαίνεται πως προσδίδει ένα επιπλέον ατού στη δημιουργία του προγράμματος.

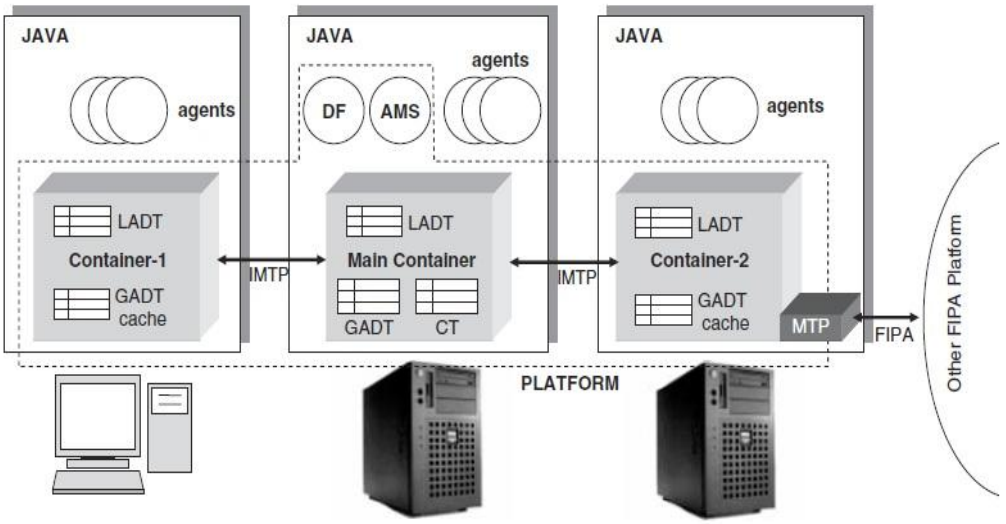
Καμία όμως από τις προηγούμενες προσεγγίσεις δεν λαμβάνει υπόψιν της τις προτιμήσεις των σπουδαστών. Θεωρούμε ότι και οι σπουδαστές θα πρέπει να μπορούν να εκφέρουν την άποψη τους κατά τη διαδικασία δημιουργίας του προγράμματος, ίσως μέσω κάποιων χαλαρών περιορισμών που θα μπορούν να θέσουν στη διάρκεια της διαδικασίας. Επίσης όλες,

με εξαίρεση αυτή του Oprea, ενδέχεται να οδηγήσουν σε μεγάλες καθυστερήσεις, λόγω του γεγονότος ότι δεν φροντίζουν να συντομεύουν τη διαδικασία σε περίπτωση πολλαπλών συγκρούσεων. Ο Oprea προτείνει ένα πρωτόκολλο πειθούς, προτείνοντας πιθανές λύσεις στους καθηγητές προκειμένου να συντομεύσει τη διαδικασία. Αυτά τα δυο μειονεκτήματα θα προσπαθήσουμε να βελτιώσουμε στα πλαίσια αυτής της διπλωματικής.

4.6 Java Agent Development Framework (JADE)

Η JADE είναι μια open source πλατφόρμα για peer-to-peer εφαρμογές, οι οποίες βασίζονται σε πράκτορες. Πρόκειται για ένα λογισμικό, που έχει βασιστεί πλήρως στη γλώσσα Java και απλοποιεί την ανάπτυξη πολυπρακτορικών συστημάτων, μέσω ενός middleware, το οποίο ακολουθεί τις προδιαγραφές της FIPA. Χρησιμοποιεί μια σειρά από γραφικά εργαλεία, τα οποία υποστηρίζουν τη φάση της ανάπτυξης και εκσφαλμάτωσης της εφαρμογής. Ένα σύστημα που έχει αναπτυχθεί σε περιβάλλον JADE μπορεί να βρίσκεται κατανεμημένο σε πολλά μηχανήματα, τα οποία παρότι υπάρχει περίπτωση να μην χρησιμοποιούν ούτε το ίδιο Λειτουργικό Σύστημα, η διαμόρφωση τους ελέγχεται απομακρυσμένα μέσω ενός γραφικού περιβάλλοντος χρήστη (Graphical User Interface, GUI). Η διαμόρφωση του συστήματος μπορεί να αλλάξει ακόμα και κατά τη διάρκεια της εκτέλεσης του, με τη μετακίνηση – μετανάστευση των πρακτόρων από ένα μηχάνημα σε ένα άλλο, όταν και όποτε αυτή απαιτείται (Jade Site).

Στο ακόλουθο σχήμα (Εικόνα 13) παρουσιάζονται τα κύρια αρχιτεκτονικά στοιχεία μιας πλατφόρμας JADE. Μια πλατφόρμα JADE αποτελείται από Agent Containers, οι οποίοι μπορούν να διανεμηθούν μέσω του διαδικτύου. Οι πράκτορες ζουν σε containers, μέσω των οποίων η Java παρέχει στην JADE όλες τις υπηρεσίες που απαιτούνται για τη φιλοξενία και εκτέλεσης των πρακτόρων. Υπάρχει ένα ειδικό container, το οποίο ονομάζεται main container, το οποίο χρησιμοποιείται ως το σημείο εκκίνησης μιας πλατφόρμας. Είναι το πρώτο container, το οποίο ξεκινάει και όλα τα άλλα containers πρέπει να δηλώσουν την ύπαρξη τους σε αυτό.



Εικόνα 13 Κύρια αρχιτεκτονικά στοιχεία μιας πλατφόρμας JADE (Wiley, 2004)

Ως σημείο εκκίνησης, το main container είναι υπεύθυνο για τα ακόλουθα:

- Τη διαχείριση του Container Table (CT), δηλαδή του μητρώου αναφοράς των αντικειμένων και των διευθύνσεων μεταφοράς όλων των containers που συνθέτουν την πλατφόρμα.
- Τη διαχείριση του Global Agent Descriptor Table (GADP), δηλαδή του μητρώου αναφοράς όλων των πρακτόρων, οι οποίοι υπάρχουν στην πλατφόρμα και περιέχει πληροφορίες για την τρέχουσα κατάσταση και θέση τους.
- Τη φιλοξενία των AMS και DF. Ο Agent Management System (AMS) είναι ο πράκτορας που εποπτεύει το σύνολο της πλατφόρμας. Πρόκειται για το σημείο επαφής όλων των πρακτόρων που πρόκειται να αλληλοεπιδράσουν και πρέπει να αποκτήσουν πρόσβαση στις υπηρεσίες white pages της πλατφόρμας. Κάθε πράκτορας εγγράφεται αυτόματα κατά την εκκίνηση του στον AMS. Ο Directory Facilitator (DF) είναι ο πράκτορας που υλοποιεί την υπηρεσία yellow pages και στον οποίο όλοι οι πράκτορες που βρίσκονται στην πλατφόρμα δηλώνουν τις υπηρεσίες που προσφέρουν ή αναζητούν τις υπηρεσίας που ψάχνουν.

Το σύνολο των εργασιών που καλείται να εκτελέσει ένας πράκτορας υλοποιείται μέσω συμπεριφορών (behaviours). Μια συμπεριφορά αντιπροσωπεύει μια σειρά από ενέργειες που εκτελεί ο πράκτορας και υλοποιείται ως ένα αντικείμενο μιας κλάσης που επεκτείνει την κλάση jade.core.behaviours.Behaviour.

Η JADE υλοποιείται πλήρως σε Java, η ελάχιστη απαίτηση του συστήματος είναι η έκδοση 5 της Java και η χρήση του Java Runtime Environment (JRE) ή του Java Development Kit (JDK). Η Jade βρίσκεται διαθέσιμη στην ιστοσελίδα http://jade.tilab.com/ (Bellifemine F.L, Caire G., Greenwood D., 2004).

5. *Ανάλυση Απαιτήσεων*

Το πρόβλημα της δημιουργίας ωρολόγιού προγράμματος για Ανώτατα Εκπαιδευτικά Ιδρύματα είναι ένα σύνθετο πρόβλημα, στο οποίο θα πρέπει να γίνει μια σωστή κατανομή των διαθέσιμων πόρων, όπως επίσης να υπάρχει και μια αποτελεσματική χρήση των περιορισμών που απαιτούνται για τη δημιουργία του προγράμματος.

Κάθε εξάμηνο, ή κάθε χρόνο σε ένα Ανώτατο Πανεπιστημιακό Ίδρυμα, κάθε επιμέρους τμήμα του, θα πρέπει να σχεδιάσει ένα νέο χρονοδιάγραμμα για τα μαθήματα του. Η επίλυση αυτού του προβλήματος συνίσταται στην τοποθέτηση αυτών των μαθημάτων (modules), τα οποία θα πρέπει να αντιστοιχηθούν με τους διαθέσιμους πόρους του Ιδρύματος. Τέτοιοι πόροι είναι οι καθηγητές, οι αίθουσες διδασκαλίας και οι ημέρες και ώρες στις οποίες θα πραγματοποιηθούν τα μαθήματα. Αυτή συνήθως είναι μια χρονοβόρα και επίπονη διαδικασία, η οποία δημιουργεί διάφορες εντάσεις εντός του Ιδρύματος.

Συνήθως η διαδικασία δημιουργίας του προγράμματος αποτελείται από δυο ξεχωριστές φάσεις:

- Αρχικά ορίζονται τα αναλυτικά προγράμματα για κάθε ενότητα και οι διαθέσιμοι πόροι (αίθουσες, καθηγητές κ.α.) ανατίθενται στα εξάμηνα του προγράμματος σπουδών.
- Στη συνέχεια, δημιουργείται ένα εφικτό χρονοδιάγραμμα, το οποίο θα πρέπει να είναι απόλυτα συμβατό με τις απαιτήσεις που έχουν οριστεί (Kambi et. al., 1996).

Η εφαρμογή, η οποία πρόκειται να αναπτυχθεί στα πλαίσια αυτής της διπλωματικής εργασίας ασχολείται με τη δεύτερη αυτή φάση.

5.1 Περιγραφή του προβλήματος και των απαιτήσεων

Στα πλαίσια αυτής της διπλωματικής, ως μελέτη περίπτωσης, αποφασίστηκε να δημιουργηθεί το ωρολόγιο πρόγραμμα μαθημάτων για το προπτυχιακό τμήμα σπουδών του Τμήματος Μηχανικών Πληροφορικής του Αλεξάνδρειου Τεχνολογικού Εκπαιδευτικού Ιδρύματος Θεσσαλονίκης

Προκειμένου να δημιουργηθεί το ωρολόγιο πρόγραμμα αποφασίστηκε ότι οι πόροι οι οποίοι πρέπει να διατεθούν είναι:

- Μάθημα (Θεωρητικό και εργαστηριακό)
- Καθηγητής που διδάσκει το μάθημα

- Ημέρα μαθήματος
- Ώρα μαθήματος
- Εξάμηνο μαθήματος

Στο τμήμα διδάσκονται μαθήματα τα οποία είναι είτε αμιγώς θεωρητικά, είτε συνδυασμός θεωρίας και εργαστηρίου. Αποφασίστηκε ότι τα θεωρητικά μαθήματα θα πρέπει να προηγούνται ωρολογιακά από τα εργοστασιακά μαθήματα ούτως ώστε να είναι ευκολότερη η παρακολούθηση τους από τους σπουδαστές του τμήματος. Δεν πρόκειται να μας απασχολήσει η αίθουσα που θα πραγματοποιηθεί το μάθημα και οι υποδομές που αυτή διαθέτει.

Στο τμήμα Μηχανικών Πληροφορικής του ΤΕΙ Θεσσαλονίκης (ΑΤΕΙΘ) διδάσκουν οι ακόλουθοι καθηγητές:

Πίνακας 3 Καθηγητές του τμήματος Μηχανικών Πληροφορικής του ΑΤΕΙΘ (http://www.it.teithe.gr/)

Καθηγητής	**Τίτλος**
Σταμάτης Δημοσθένης	Καθηγητής
Σιδηρόπουλος Αντώνης	Καθηγητής Εφαρμογών
Αδαμίδης Παναγιώτης	Καθηγητής
Δέρβος Δημήτριος	Καθηγητής
Διαμαντάρας Κωνσταντίνος	Καθηγητής
Χατζημήσιος Περικλής	Αναπληρωτής Καθηγητής
Ηλιούδης Χρήστος	Αναπληρωτής Καθηγητής
Σιάκα Κερστιν	Καθηγητής
Κεραμόπουλος Ευκλείδης	Επίκουρος Καθηγητής
Σφέτσος Παναγιώτης	Αναπληρωτής Καθηγητής
Κώστογλου Βασίλης	Καθηγητής
Τεκτονίδης Δημήτριος	Καθηγητής
Βίτσας Βασίλειος	Καθηγητής
Κλεφτούρης Δημήτρης	Καθηγητής
Αντωνίου Ευστάθιος	Επίκουρος Καθηγητής
Γουλιάνας Κων/νος	Επίκουρος Καθηγητής
Ράπτης Παναγιώτης	Επίκουρος Καθηγητής
Δεληγιάννης Ιγνάτιος	Καθηγητής
Ψαρράς Νικόλας	Καθηγητής Εφαρμογών

Σαλαμπάσης Μιχάλης	Καθηγητής

Για λόγους απλούστευσης στην εφαρμογή που θα αναπτύξουμε, οι Καθηγητές Εφαρμογών έχουν ομαδοποιηθεί από κοινού με τους Επίκουρους Καθηγητές.

Για τη δημιουργία του προγράμματος θεωρούμε ότι μια εβδομάδα αποτελείται από πέντε ημέρες στις οποίες πραγματοποιούνται μαθήματα (Δευτέρα έως και Παρασκευή) και για κάθε ημέρα, μαθήματα πραγματοποιούνται σε δεκατρείς διαθέσιμες χρονοθυρίδες. Από τις 08:00 το πρωί έως τις 21:00 το βράδυ. Για κάθε μάθημα μπορούν να διατεθούν είτε δυο, είτε τρεις συνεχόμενες χρονοθυρίδες την ίδια ημέρα, ανάλογα με τη συνολική διάρκεια του μαθήματος. Για μαθήματα άρτιου αριθμού ωρών (2, 4, …) μπορούν να διατεθούν δυο συνεχόμενες διδακτικές ώρες στη διάρκεια μιας ημέρας. Για μαθήματα με περιττό αριθμό ωρών (3, 5, ….) μπορεί να διατεθούν τρεις συνεχόμενες διδακτικές ώρες στη διάρκεια της ημέρας και οι εναπομείναντες ώρες να συνεχίσουν να διατίθενται στις υπόλοιπες ημέρες σε δίωρες χρονοθυρίδες,

Στο τμήμα Μηχανικών Πληροφορικής του ΤΕΙ Θεσσαλονίκης υπάρχουν επτά εξάμηνα σπουδών, τα οποία αποτελούνται από πέντε μαθήματα το κάθε ένα. Υπάρχουν επίσης διαθέσιμα οκτώ μαθήματα επιλογής και οι σπουδαστές του Τμήματος καλούνται να επιλέξουν ένα από αυτά στο Στ’ εξάμηνο και άλλα τρία στο Ζ’ εξάμηνο. Τα μαθήματα του προγράμματος σπουδών, όπως προκύπτει από την ιστοσελίδα της Σχολής είναι τα παρακάτω:

Πίνακας 4.Εξάμηνο Α’ (http://www.it.teithe.gr/)

Τίτλος	**Ώρες Διδασκαλίας**		**Διδάσκοντες**
	Θεωρία	**Εργαστήριο**	
Εισαγωγή στη Πληροφορική	4	2	Ηλιούδης Χρήστος
Αλγοριθμική και Προγραμματισμός	4	2	Σφέτσος Παναγιώτης
Ψηφιακά Συστήματα	4	0	Κλεφτούρης Δημήτριος
Μαθηματική Ανάλυση	5	0	Αντωνίου Ευστάθιος
Δεξιότητες Επικοινωνίας/Κοινωνικά Δίκτυα	3	2	Σιάκα Κερστιν

Πίνακας 5.Εξάμηνο Β' (http://www.it.teithe.gr/)

Τίτλος	Ώρες Διδασκαλίας		Διδάσκοντες
	Θεωρία	Εργαστήριο	
Αντικειμενοστραφής Προγραμματισμός	4	2	Αδαμίδης Παναγιώτης
Εισαγωγή στα Λειτουργικά Συστήματα	4	2	Σιδηρόπουλος Αντώνης
Διακριτά Μαθηματικά	5	0	Αντωνίου Ευστάθιος
Γλώσσες και Τεχνολογίες Ιστού	4	2	Τεκτονίδης Δημήτριος
Πληροφοριακά Συστήματα Ι	4	0	Σιάκα Κέρστιν

Πίνακας 6.Εξάμηνο Γ' (http://www.it.teithe.gr/)

Τίτλος	Ώρες Διδασκαλίας		Διδάσκοντες
	Θεωρία	Εργαστήριο	
Αριθμητική Ανάλυση & Προγραμματισμός Επιστημονικών Εφαρμογών	3	2	Αντωνίου Ευστάθιος
Δομές Δεδομένων και Ανάλυση Αλγορίθμων	4	2	Σταμάτης Δημοσθένης
Οργάνωση και Αρχιτεκτονική Υπολογιστικών Συστημάτων	3	2	Κλεφτούρης Δημήτρης
Αλληλεπίδραση Ανθρώπου-Μηχανής & Ανάπτυξη Διεπιφανειών Χρήστη	3	2	Κεραμόπουλος Ευκλείδης
Συστήματα Διαχείρισης Βάσεων Δεδομένων	3	2	Δέρβος Δημήτριος

Πίνακας 7.Εξάμηνο Δ' (http://www.it.teithe.gr/)

Τίτλος	Ώρες Διδασκαλίας		Διδάσκοντες
	Θεωρία	Εργαστήριο	
Μεθοδολογίες Προγραμματισμού	4	2	Ράπτης Παναγιώτης
Τεχνητή Νοημοσύνη:Γλώσσες και Τεχνικές	3	2	Σταμάτης Δημοσθένης
Τηλεπικοινωνίες και Δίκτυα Υπολογιστών	3	2	Βίτσας Βασίλειος
Θεωρία Λειτουργικών Συστημάτων	5	0	Χατζημήσιος Περικλής
Θεωρία Πιθανοτήτων και Στατιστική	3	2	Αντωνίου Ευστάθιος

Πίνακας 8.Εξάμηνο Ε' (http://www.it.teithe.gr/)

Τίτλος	Ώρες Διδασκαλίας		Διδάσκοντες
	Θεωρία	Εργαστήριο	
Πληροφοριακά Συστήματα ΙΙ	5	0	Σιάκα Κέρστιν
Μηχανική Λογισμικού Ι	3	2	Δεληγιάννης Ιγνάτιος
Δίκτυα Υπολογιστών	3	2	Ψαρράς Νικόλας
Ανάπτυξη Διαδικτυακών Συστημάτων & Εφαρμογών	4	2	Διαμαντάρας Κωνσταντίνος
Επιχειρησιακή Έρευνα	5	0	Κώστογλου Βασίλης

Πίνακας 9.Εξάμηνο Στ' (http://www.it.teithe.gr/)

Τίτλος	Ώρες Διδασκαλίας		Διδάσκοντες
	Θεωρία	Εργαστήριο	
Ασφάλεια Πληροφοριακών Συστημάτων	4	2	Ηλιούδης Χρήστος
Μηχανική Μάθηση	3	2	Διαμαντάρας Κων/νος Γουλιάνας Κων/νος
Τεχνολογία Βάσεων Δεδομένων	3	2	Δέρβος Δημήτριος Κεραμόπουλος Ευκλείδης
Μηχανική Λογισμικού ΙΙ	5	0	Δεληγιάννης Ιγνάτιος Σφέτσος Παναγιώτης
Μάθημα Επιλογής 1			

Πίνακας 10.Εξάμηνο Ζ' (http://www.it.teithe.gr/)

Τίτλος	Ώρες Διδασκαλίας		Διδάσκοντες
	Θεωρία	Εργαστήριο	
Ανάπτυξη και Διαχείριση Ολοκληρωμένων Πλ. Συστημάτων & Εφαρμογών	4	2	Σαλαμπάσης Μιχάλης
Τεχνολογία Πολυμέσων	3	2	Κλεφτούρης Δημήτρης
Μάθημα Επιλογής 2			
Μάθημα Επιλογής 3			
Μάθημα Επιλογής 4			

Πίνακας 11.Μαθήματα Επιλογής (http://www.it.teithe.gr/)

Τίτλος	Ώρες Διδασκαλίας		Διδάσκοντες
	Θεωρία	Εργαστήριο	
Ευφυή Συστήματα	5	0	Αδαμίδης Παναγιώτης
Προηγμένες Αρχιτεκτονικές Υπολογιστών και Παράλληλα Συστήματα	5	0	Διαμαντάρας Κωνσταντίνος
Οργάνωση Δεδομένων και Εξόρυξη Πληροφορίας	5	0	Δέρβος Δημήτριος Κεραμόπουλος Ευκλείδης
Ειδικά Θέματα Δικτύων Ι	2	4	Ψαρράς Νικόλας
Ειδικά Θέματα Δικτύων ΙΙ	2	4	Ψαρράς Νικόλας
Διαδικτυακές Υπηρεσίες Προστιθέμενης Αξίας	5	0	Ηλιούδης Χρήστος
Ασύρματα και Κινητά Δίκτυα Επικοινωνιών	5	0	Χατζημήσιος Περικλής
Γραφικά Υπολογιστών	5	0	Ράπτης Παναγιώτης

Οι σπουδαστές καλούνται να επιλέξουν τα μαθήματα των εξαμήνων τους. Για λόγους απλούστευσης της εφαρμογής που θα αναπτύξουμε, έχουμε ενσωματώσει τα μαθήματα επιλογής στα επτά διαθέσιμα εξάμηνα. Έτσι κάναμε την ακόλουθη αντιστοίχηση των διαθέσιμων μαθημάτων επιλογής:

Πίνακας 12. Αντιστοίχηση μαθημάτων επιλογής στα υπόλοιπα εξάμηνα

Μάθημα Επιλογής	Εξάμηνο
Ευφυή Συστήματα	Ζ’
Προηγμένες Αρχιτεκτονικές Υπολογιστών και Παράλληλα Συστήματα	Ζ’
Οργάνωση Δεδομένων και Εξόρυξη Πληροφορίας	Ζ’
Ειδικά Θέματα Δικτύων Ι	Ε΄
Ειδικά Θέματα Δικτύων ΙΙ	Ζ΄
Διαδικτυακές Υπηρεσίες Προστιθέμενης Αξίας	Δ’
Ασύρματα και Κινητά Δίκτυα Επικοινωνιών	Στ΄
Γραφικά Υπολογιστών	Στ’

5.2 Ανάλυση του προβλήματος (Στατιστική Έρευνα)

Προκειμένου να αναπτυχθεί με αποτελεσματικό τρόπο η εφαρμογή αποφασίσαμε να ζητήσουμε τη γνώμη των καθηγητών και των σπουδαστών του τμήματος σε σχέση με μερικά κρίσιμα ερωτήματα που θα μας απασχολήσουν κατά τη διάρκεια της εφαρμογής.

Γι' αυτό το λόγο δημιουργήθηκαν και διανεμήθηκαν δυο ερωτηματολόγια, ένα προς τους καθηγητές και ένα προς τους σπουδαστές του τμήματος, τα οποία παρατίθενται στο Παράρτημα Α. Διανεμήθηκαν συνολικά 12 ερωτηματολόγια στους καθηγητές του τμήματος και 22 στους σπουδαστές του τμήματος. Η έρευνα διενεργήθηκε κατά την περίοδο Νοεμβρίου – Δεκεμβρίου 2014.

5.2.1 Χρήση ερωτηματολόγιων για στατιστική έρευνα

Για την διεξαγωγή της παρούσας έρευνας ήταν αναγκαία η συλλογή πρωτογενών στοιχείων. Η συλλογή αυτή μπορεί να πραγματοποιηθεί με διάφορες μεθόδους και στη συγκεκριμένη εργασία χρησιμοποιήθηκε αυτή του δομημένου ερωτηματολογίου. Το ερωτηματολόγιο είναι ένα τυποποιημένο σχέδιο για τη συλλογή και την καταγραφή εξειδικευμένης και σχετικής με ένα θέμα πληροφόρησης, με σχετική πληρότητα και ακρίβεια. Με άλλα λόγια καθοδηγεί τη διαδικασία συλλογής των πληροφοριών και προωθεί την καταγραφή τους μέσω ενός συστηματικού τρόπου. Η δημιουργία του ερωτηματολογίου, λόγω των παραπάνω ιδιοτήτων που έχει, αποτελεί την πλέον κρίσιμη και σημαντική εργασία, που καθορίζει τη σημασία της στατιστικής έρευνας. Είναι σημαντικό, ένα επιτυχημένο ερωτηματολόγιο να έχει συνοχή, πληρότητα, σαφήνεια, κατάλληλη δομή, να περιλαμβάνει ερωτήματα ελέγχου, να είναι σύντομο και να περιλαμβάνει βασικές οδηγίες συμπλήρωσης και εννοιολογικές επεξηγήσεις.

Το ερωτηματολόγιο αποτελεί καθοριστικό επιστημονικό όργανο μέτρησης της έρευνας και απαραίτητο εργαλείο ενός ερευνητή. Τα στοιχεία και οι μετρήσεις που προκύπτουν από αυτό θα πρέπει να χαρακτηρίζονται από πληρότητα, ακρίβεια, σταθερότητα κ.α.. Ένα ερωτηματολόγιο θα πρέπει να έχει θεματικό περιεχόμενο που να καλύπτει απόλυτα τους γενικότερους και ειδικότερους στόχους της έρευνας (Javeau, 1996).

Στην παρούσα έρευνα δημιουργήθηκε ένα ερωτηματολόγιο σχεδιασμένο με τέτοιο τρόπο, ώστε να καλύπτει τα ερωτήματα και τους στόχους της. Ο σχεδιασμός και η δημιουργία του ερωτηματολογίου ακολουθεί όλα τα απαραίτητα στάδια που απαιτούνται. Η μέθοδος συλλογής των ερωτηματολογίων θα είναι η προσωπική, δηλαδή αυτά μοιράστηκαν προσωπικά (χέρι με χέρι).

Τα ερωτηματολόγια που χρησιμοποιήθηκαν στην έρευνα μας αποτελούνται από τα εξής δύο μέρη:

- Α μέρος ερωτηματολογίου – Γενικά στοιχεία
  Για το ερωτηματολόγιο καθηγητών Φύλο και Βαθμίδα καθηγητή.
  Για το ερωτηματολόγιο φοιτητών Φύλο και εξάμηνο σπουδών.
- Β μέρος ερωτηματολογίου – Μέτρηση Μεταβλητών.

5.2.2 Επεξεργασία αποτελεσμάτων

Για τη στατιστική τους ανάλυση επιλέχθηκε να χρησιμοποιηθεί το SPSS και συγκεκριμένα το IBM SPSS Statistics 22.

Το πρόγραμμα SPSS θέτει περιορισμούς στην εσωτερική δομή του αρχείου, τους τύπους δεδομένων, στην επεξεργασία δεδομένων και στα αρχεία που ταιριάζουν, και έτσι η διαδικασία απλουστεύεται σημαντικά. Στο SPSS, τα σύνολα δεδομένων έχουν μια δύο διαστάσεων δομή του πίνακα, όπου οι σειρές αντιπροσωπεύουν συνήθως ??? και οι στήλες αντιπροσωπεύουν μετρήσεις. Μόνο δύο τύποι δεδομένων ορίζονται: αριθμητικοί και κείμενο. Όλη η επεξεργασία των δεδομένων γίνεται διαδοχικά κατά περίπτωση μέσω του αρχείου. Τα αρχεία μπορούν να συνδυαστούν ένα προς ένα και ένα προς πολλά, αλλά όχι πολλά προς πολλά.

5.2.3 Στατιστική ανάλυση αποτελεσμάτων

Ερωτηματολόγιο καθηγητών:

Στο Α' μέρος του ερωτηματολόγιο, το οποίο αποτελείται από τα γενικά στοιχεία προκύπτουν τα ακόλουθα αποτελέσματα:

Φύλο

		Frequency	**Percent**	**Valid Percent**	**Cumulative Percent**
Valid	Άνδρας	12	100,0	100,0	100,0

Βαθμίδα

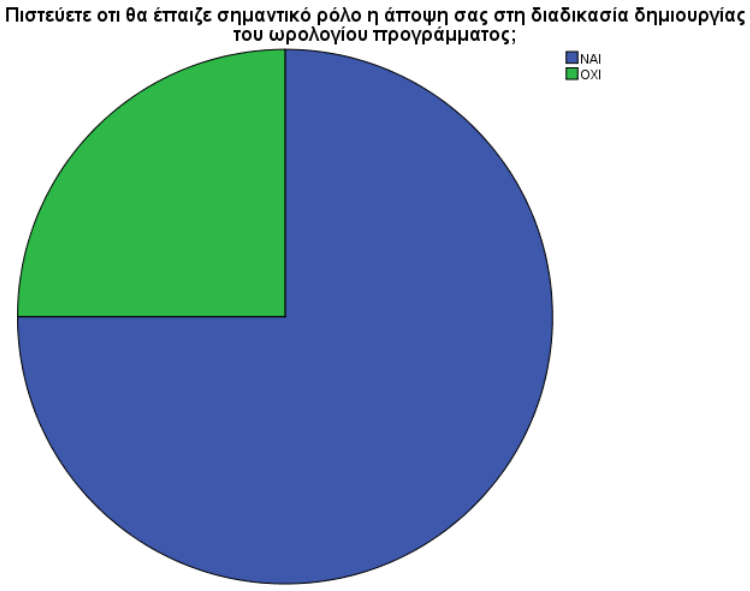
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Καθηγητής εφαρμογών - επίκουρος Καθηγητής	4	33,3	33,3	33,3
	Αναπληρωτής Καθηγητής	3	25,0	25,0	58,3
	Καθηγητής	5	41,7	41,7	100,0
	Total	12	100,0	100,0	

Παρατηρούμε λοιπόν ότι στο ΤΕΙ Πληροφορικής διδάσκουν κυρίως άνδρες. Αυτό φαίνεται και στον πίνακα με τους καθηγητές, στο τμήμα διδάσκουν 19 άνδρες καθηγητές και μόνο 1 γυναίκα. Στα ερωτηματολόγια απάντησαν μόνο 12 άνδρες καθηγητές εκ των οποίων οι 4 είναι Καθηγητές Εφαρμογών και Επίκουροι Καθηγητές, οι 3 Αναπληρωτές Καθηγητές και οι 5 Καθηγητές.

Στο Β΄ μέρος, το οποίο αποτελεί και το ουσιαστικότερο κομμάτι του ερωτηματολόγιου μας και γίνεται η μέτρηση των μεταβλητών που χρησιμοποιήθηκαν προέκυψαν τα ακόλουθα αποτελέσματα:

Πιστεύετε ότι θα έπαιζε σημαντικό ρόλο η άποψη σας στη διαδικασία δημιουργίας του ωρολογίου προγράμματος;

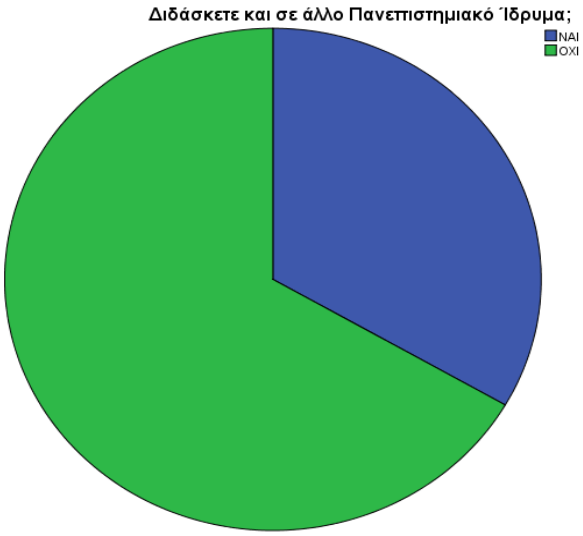
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	NAI	9	75,0	75,0	75,0
	OXI	3	25,0	25,0	100,0
	Total	12	100,0	100,0	



Παρατηρούμε λοιπόν ότι σε μεγάλη πλειοψηφία, οι καθηγητές επιθυμούν να εκφέρουν την άποψη τους κατά τη δημιουργία του προγράμματος μαθημάτων.

Διδάσκετε και σε άλλο Πανεπιστημιακό Ίδρυμα;

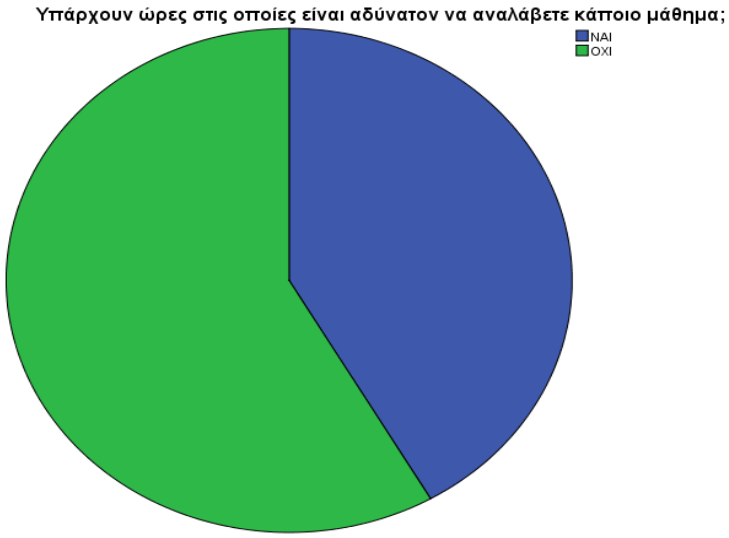
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	NAI	4	33,3	33,3	33,3
	OXI	8	66,7	66,7	100,0
	Total	12	100,0	100,0	



Τέσσερις στους δώδεκα καθηγητές διδάσκουν και σε άλλο Πανεπιστημιακό Ίδρυμα, οπότε θα πρέπει να λάβουμε πολύ σοβαρά υπόψιν μας τις ώρες που αυτοί δεν μπορούν να διδάξουν μαθήματα και κατά συνέπεια να μην τοποθετηθούν στο πρόγραμμα που πρόκειται να δημιουργήσουμε.

Υπάρχουν ώρες στις οποίες είναι αδύνατον να αναλάβετε κάποιο μάθημα;

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	NAI	5	41,7	41,7	41,7
	OXI	7	58,3	58,3	100,0
	Total	12	100,0	100,0	



Αν συνδυαστεί αυτό με το προηγούμενο αποτέλεσμα, τότε ενισχύεται η προσπάθεια που πρέπει να καταβάλουμε να μην τοποθετηθούν μαθήματα σε ώρες που ο Καθηγητής δηλώνει ότι δεν μπορεί να αναλάβει μαθήματα. Αυτό πιθανόν συμβαίνει, επειδή διδάσκει και σε άλλα Πανεπιστημιακά Ιδρύματα.

Προτιμάτε πρωινά μαθήματα ή απογευματινά;

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Πρωινά	7	58,3	58,3	58,3
	Δεν έχω πρόβλημα	5	41,7	41,7	100,0
	Total	12	100,0	100,0	

Οι περισσότεροι καθηγητές προτιμούν πρωινά μαθήματα (7/12) και οι υπόλοιποι δηλώνουν ότι δεν έχουν πρόβλημα. Σαν συμπέρασμα θα μπορούσαμε να πούμε ότι θα πρέπει να προσπαθήσουμε να τοποθετηθούν τα υπόλοιπα μαθήματα των καθηγητών, πέρα από τις προτιμήσεις τους σε πρωινές ώρες.

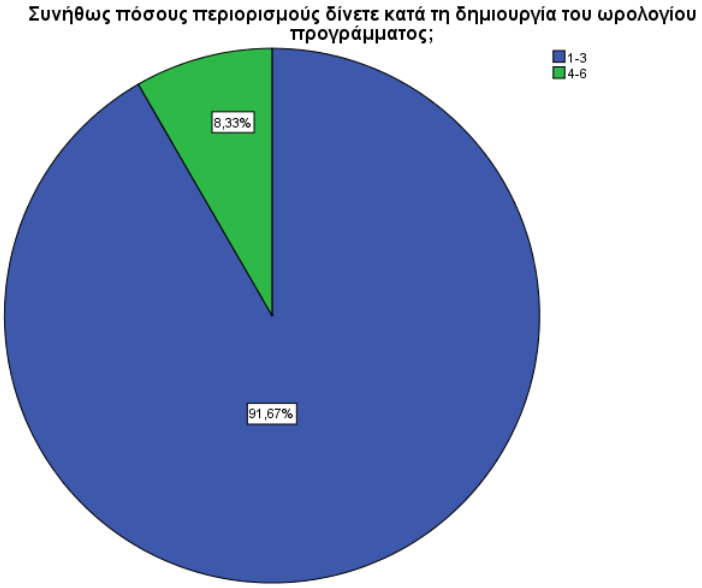
Σας ενδιαφέρει το πρόγραμμα να είναι συμπαγές;

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	NAI	8	66,7	66,7	66,7
	OXI	3	25,0	25,0	91,7
	Δεν με απασχολεί	1	8,3	8,3	100,0
	Total	12	100,0	100,0	

Παρατηρούμε λοιπόν ότι στη μεγαλύτερη πλειοψηφία τους, ποσοστό 66,7% οι καθηγητές προτιμούν ένα συμπαγές (χωρίς κενά) πρόγραμμα, 8,3% δηλώνει ότι δεν τον ενδιαφέρει, ενώ μόνο το 25% δηλώνει αντίθετο σε ένα συμπαγές πρόγραμμα. Θα πρέπει να καταβληθεί προσπάθεια λοιπόν το πρόγραμμα που θα προκύψει να είναι όσο πιο συμπαγές γίνεται.

Συνήθως πόσους περιορισμούς δίνετε κατά τη δημιουργία του ωρολογίου προγράμματος;

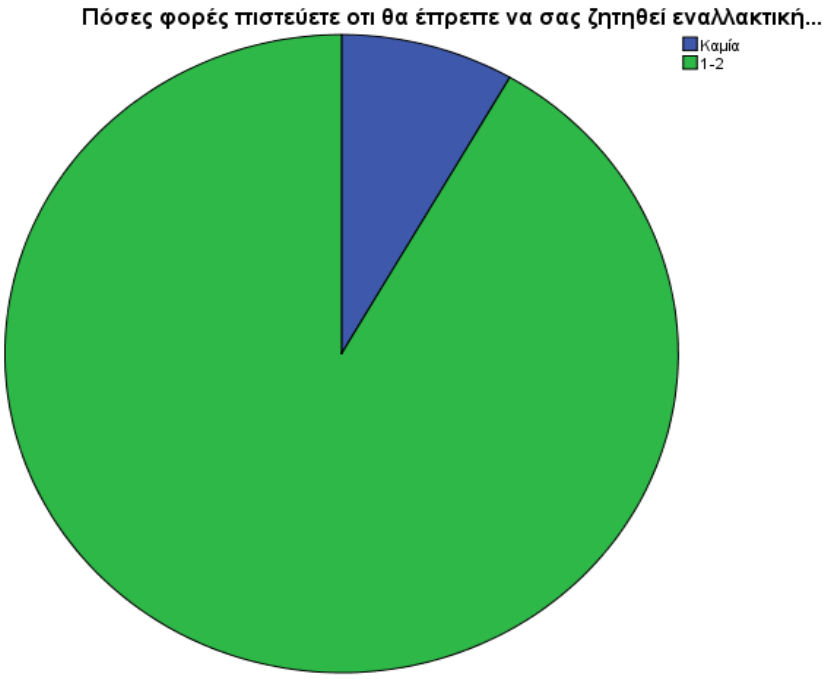
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	1-3	11	91,7	91,7	91,7
	4-6	1	8,3	8,3	100,0
	Total	12	100,0	100,0	



Οι καθηγητές στη συντριπτική πλειοψηφία τους, ποσοστό 91,7%, δηλώνουν 1-3 προτιμήσεις, ενώ μόνο ένα πολύ μικρό ποσοστό, 8,3% δηλώνει περισσότερες από 3 προτιμήσεις.

Πόσες φορές πιστεύετε ότι θα έπρεπε να σας ζητηθεί εναλλακτική...

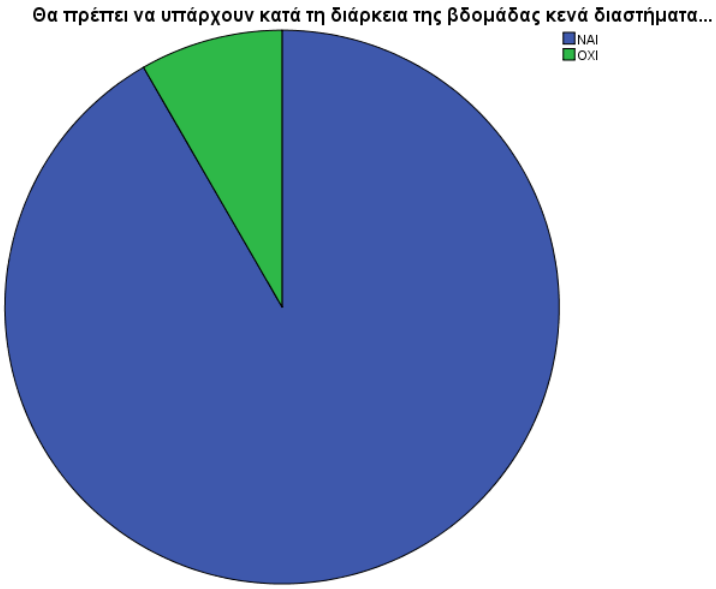
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Καμία	1	8,3	8,3	8,3
	1-2	11	91,7	91,7	100,0
	Total	12	100,0	100,0	



Και σε αυτή την ερώτηση, σε ποσοστό 91,7%, οι καθηγητές συμφωνούν ότι σε περίπτωση που δεν ικανοποιηθούν οι απαιτήσεις τους, θα πρέπει να τους ζητηθούν εναλλακτικές προτάσεις 1-2 φορές ακόμη. Κανείς δεν έχει επίλεξει περισσότερες από 3 φορές. Θα πρέπει λοιπόν να υπάρχουν μαζί με την αρχική, 3 φάσεις διαπραγμάτευσης, δηλαδή 3 φορές να τους ζητηθεί να αποστείλουν προτιμήσεις.

Θα πρέπει να υπάρχουν κατά τη διάρκεια της βδομάδας κενά διαστήματα για την τοποθέτηση αναπληρώσεων

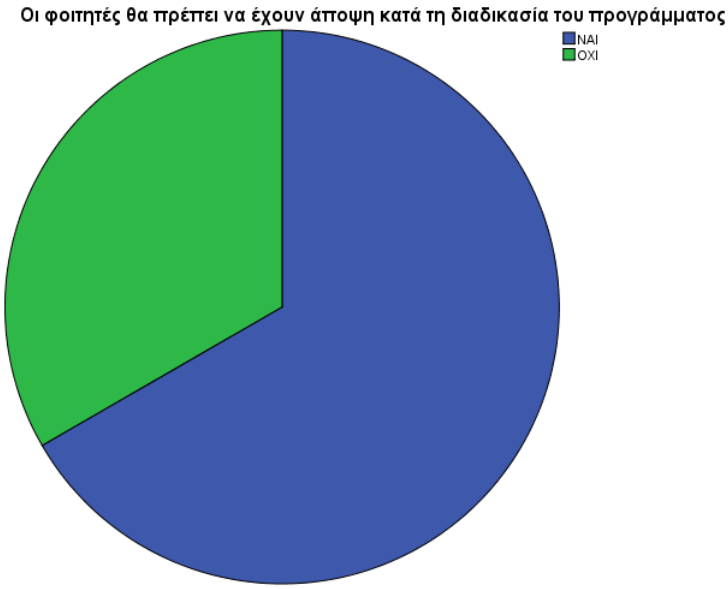
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	NAI	11	91,7	91,7	91,7
	OXI	1	8,3	8,3	100,0
	Total	12	100,0	100,0	



Στην πλειοψηφία τους οι καθηγητές, ποσοστό 91,7%, πιστεύουν ότι θα πρέπει να υπάρχει πρόβλεψη, ώστε να υπάρχει ένα κενό διάστημα ανάμεσα στην εβδομάδα, στο οποίο δεν θα τοποθετηθούν μαθήματα και το οποίο θα χρησιμοποιηθεί για την τοποθέτηση των μαθημάτων μου χάνονται κατά τη διάρκεια της χρονιάς προκειμένου να αναπληρωθούν αυτές οι ώρες.

Οι φοιτητές θα πρέπει να έχουν άποψη κατά τη διαδικασία του προγράμματος

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	NAI	8	66,7	66,7	66,7
	OXI	4	33,3	33,3	100,0
	Total	12	100,0	100,0	



Το 66,7% των καθηγητών είναι υπέρ της άποψης να εκφράζουν τη γνώμη τους και οι φοιτητές για τη δημιουργία του προγράμματος, ενώ το 33,3% διαφωνεί. Εφόσον συμφωνεί λοιπόν η πλειοψηφία των καθηγητών, θα πρέπει να ληφθεί μέριμνα ούτως ώστε και οι φοιτητές του τμήματος να εκφράζουν την άποψη τους κατά τη διαδικασία δημιουργίας του ωρολογίου προγράμματος μαθημάτων.

Ερωτηματολόγιο φοιτητών:

Στο Α' μέρος του ερωτηματολόγιο, το οποίο αποτελείται από τα γενικά στοιχεία προκύπτουν τα ακόλουθα αποτελέσματα:

Φύλο

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Άνδρας	11	50,0	50,0	50,0
	Γυναίκα	11	50,0	50,0	100,0
	Total	22	100,0	100,0	

Εξάμηνο σπουδών

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	1ο	1	4,5	4,5	4,5
	2ο	1	4,5	4,5	9,1
	3ο	6	27,3	27,3	36,4
	5ο	2	9,1	9,1	45,5
	7ο	1	4,5	4,5	50,0
	Μεγαλύτερο εξάμηνο	1	4,5	4,5	54,5
	Μεταπτυχιακός φοιτητής	10	45,5	45,5	100,0
	Total	22	100,0	100,0	

Παρατηρούμε λοιπόν ότι στο δείγμα από τους σπουδαστές που έχουμε λάβει το ποσοστό των Ανδρών και των Γυναικών είναι ίδιο, από 50%. Αυτό το γεγονός κάνει το δείγμα πολύ αντιπροσωπευτικό. Επίσης, το μεγαλύτερο ποσοστό από το δείγμα των απαντήσεων 45,5% ανήκει στους μεταπτυχιακούς φοιτητές του τμήματος, επειδή εκεί υπήρχε ευκολότερη πρόσβαση. Ένα καλό ποσοστό 27,3% ανήκει στους σπουδαστές του 3ου εξαμήνου, ενώ από τα υπόλοιπα εξάμηνα το δείγμα είναι πολύ μικρό.

Στο Β΄ μέρος, το οποίο αποτελεί και το ουσιαστικότερο κομμάτι του ερωτηματολόγιου μας και γίνεται η μέτρηση των μεταβλητών που χρησιμοποιήθηκαν, προέκυψαν τα ακόλουθα αποτελέσματα:

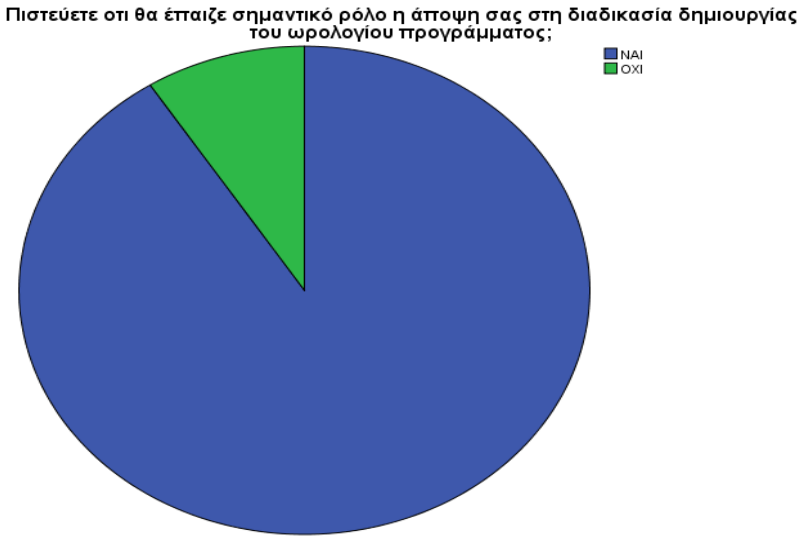
Εργάζεστε παράλληλα με τις σπουδές

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	NAI	8	36,4	36,4	36,4
	OXI	14	63,6	63,6	100,0
	Total	22	100,0	100,0	

Εδώ παρατηρούμε ότι το μεγαλύτερο ποσοστό των φοιτητών, ποσοστό 63,63% δεν εργάζεται κατά τη διάρκεια των σπουδών του και μάλιστα οι περισσότεροι από αυτούς που εργάζονται ανήκουν στους μεταπτυχιακούς φοιτητές του τμήματος. Αυτό το γεγονός μας δίνει μια μεγαλύτερη ευελιξία στη διαδικασία της δημιουργίας του προγράμματος.

Πιστεύετε ότι θα έπαιζε σημαντικό ρόλο η άποψη σας στη διαδικασία δημιουργίας του ωρολογίου προγράμματος;

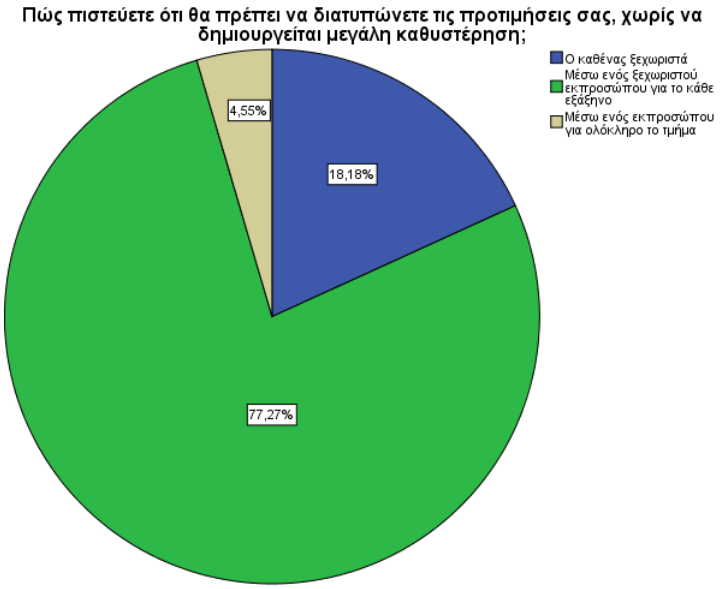
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	NAI	20	90,9	90,9	90,9
	OXI	2	9,1	9,1	100,0
	Total	22	100,0	100,0	



Σχεδόν στο σύνολο τους, σε ποσοστό 90,9%, πιστεύουν ότι θα έπρεπε να έχουν τη δυνατότητα να εκφράσουν και αυτοί τις προτιμήσεις τους κατά τη διαδικασία δημιουργίας του ωρολογίου προγράμματος. Αυτό το γεγονός ενισχύει ακόμα περισσότερο την πεποίθηση μας να συμπεριλάβουμε τις προτιμήσεις τους στην εφαρμογή που θα αναπτύξουμε.

Πώς πιστεύετε ότι θα πρέπει να διατυπώνετε τις προτιμήσεις σας, χωρίς να δημιουργείται μεγάλη καθυστέρηση;

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Ο καθένας ξεχωριστά	4	18,2	18,2	18,2
	Μέσω ενός ξεχωριστού εκπροσώπου για το κάθε εξάξηνο	17	77,3	77,3	95,5
	Μέσω ενός εκπροσώπου για ολόκληρο το τμήμα	1	4,5	4,5	100,0
	Total	22	100,0	100,0	



Εδώ παρατηρούμε ότι στο μεγαλύτερο ποσοστό τους, ποσοστό 77,3%, οι φοιτητές πιστεύουν ότι τις προτιμήσεις τους θα πρέπει να τις εκφράζει ένας εκπρόσωπος τους για κάθε εξάμηνο. Με αυτό τον τρόπο θα υπάρχουν συνολικά επτά εκπρόσωποι των φοιτητών, ένας για κάθε εξάμηνο και αυτό θα έχει σαν συνέπεια να εκφράζονται οι προτιμήσεις συνολικά και να είναι περισσότερο ευέλικτη και λιγότερο σύνθετη η εφαρμογή. Μόνο ένα ποσοστό 18,2% θα ήθελε να εκφράζει τις προτιμήσεις του ατομικά, ενώ 4,5% θα ήθελε να υπάρχει ένας εκπρόσωπος για ολόκληρο το τμήμα.

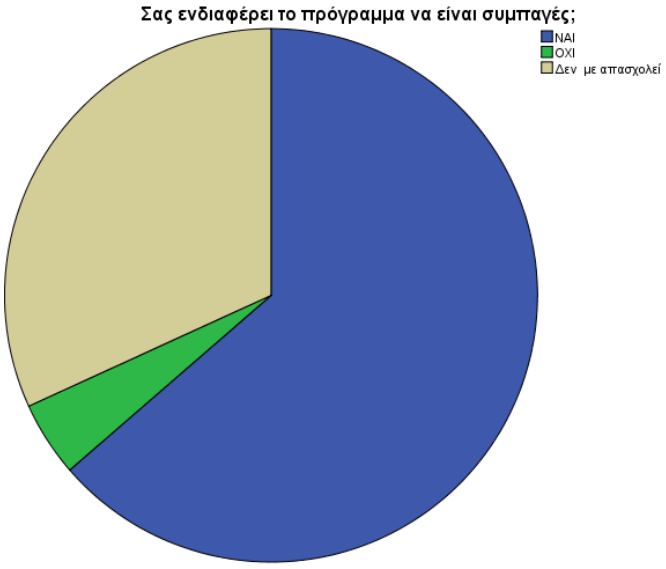
Προτιμάτε πρωινά μαθήματα ή απογευματινά;

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Πρωινά	7	31,8	31,8	31,8
	Απογευματινά	5	22,7	22,7	54,5
	Δεν έχω πρόβλημα	10	45,5	45,5	100,0
	Total	22	100,0	100,0	

Όπως παρατηρούμε το μεγαλύτερο ποσοστό των φοιτητών, 45,5%, δηλώνει ότι δεν έχει πρόβλημα αν τα μαθήματα είναι πρωινά ή όχι. 31,8% προτιμά πρωινά μαθήματα, ενώ αντίθετα απογευματινά μαθήματα προτιμά το 22,7%. Παρατηρούμε λοιπόν πως δεν μπορεί να εξαχθεί ξεκάθαρο συμπέρασμα από τις προτιμήσεις των φοιτητών, αν και υπάρχει μια μικρή τάση προς τα πρωινά μαθήματα.

Σας ενδιαφέρει το πρόγραμμα να είναι συμπαγές;

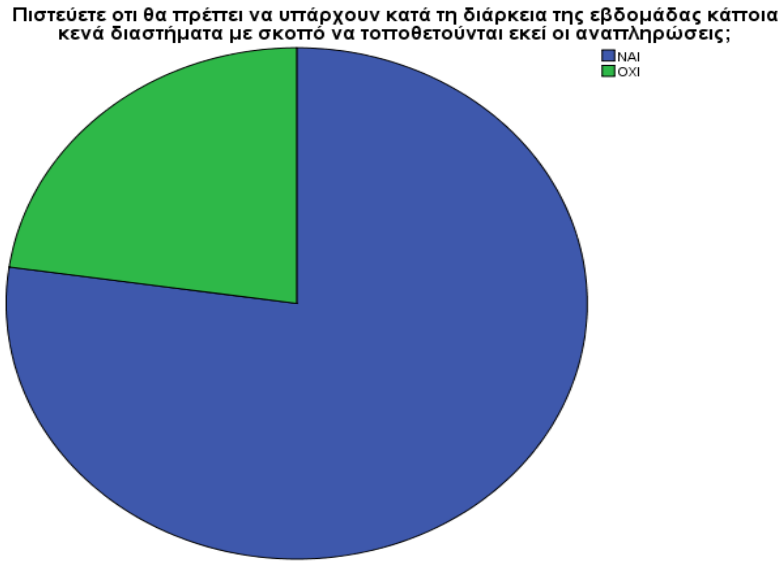
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	NAI	14	63,6	63,6	63,6
	OXI	1	4,5	4,5	68,2
	Δεν με απασχολεί	7	31,8	31,8	100,0
	Total	22	100,0	100,0	



Οι φοιτητές σε ποσοστό 63,6% προτιμούν το πρόγραμμα να είναι συμπαγές, δηλαδή χωρίς κενά. Ένα ποσοστό 31,8% δηλώνει ότι δεν το απασχολεί κάτι τέτοιο. Παρατηρούμε λοιπόν ότι σε ποσοστό 95,4% αν προσπαθήσουμε τα πρόγραμμα μαθημάτων να είναι όσο περισσότερο συμπαγές γίνετε, η συντριπτική πλειοψηφία των φοιτητών θα είναι ευχαριστημένη. Μόνο ένα ποσοστό 4,5% δεν επιθυμεί συμπαγές πρόγραμμα.

Πιστεύετε ότι θα πρέπει να υπάρχουν κατά τη διάρκεια της εβδομάδας κάποια κενά διαστήματα με σκοπό να τοποθετούνται εκεί οι αναπληρώσεις;

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	NAI	17	77,3	77,3	77,3
	OXI	5	22,7	22,7	100,0
	Total	22	100,0	100,0	



Και από εδώ, όπως και από το ερωτηματολόγιο των καθηγητών προκύπτει ότι θα πρέπει να υπάρξει πρόβλεψη ώστε να υπάρχει κάποιο κενό διάστημα ανάμεσα στην εβδομάδα, έτσι ώστε να τοποθετηθούν εκεί τυχόν αναπληρώσεις μαθημάτων που μπορεί να προκύψουν κατά τη διάρκεια της χρονιάς. Οι φοιτητές συμφωνούν σε αυτό σε ποσοστό 77,3%, ενώ μόνο ένα ποσοστό 22,7% διαφωνεί.

5.3 Συμπεράσματα από στατιστική έρευνα

Όπως προκύπτει από τη στατιστική ανάλυση των ερωτηματολόγιων, και των καθηγητών και των φοιτητών, θα πρέπει η εφαρμογή που θα αναπτυχθεί να λαμβάνει υπόψιν της και τις προτιμήσεις των καθηγητών και τις προτιμήσεις των φοιτητών. Επίσης για τις προτιμήσεις των φοιτητών αυτό θα πρέπει να γίνει μέσω ενός εκπροσώπου για κάθε εξάμηνο. Άρα συνολικά θα πρέπει να υπάρχουν επτά εκπρόσωποι φοιτητών, οι οποίοι θα εκφράζουν τις προτιμήσεις τους. Θα πρέπει επίσης να υπάρχουν συνολικά 2-3 φάσεις διαπραγμάτευσης ανάμεσα στον υπεύθυνο του προγράμματος και σε κάθε καθηγητή, προκειμένου να καταλήξουμε σε κάποια συμφωνία. Και οι δυο συμφωνούν ότι θα πρέπει να υπάρξει πρόβλεψη έτσι ώστε να υπάρχει κάποιο κενό διάστημα ανάμεσα στην εβδομάδα, ούτως ώστε να τοποθετούνται εκεί οι αναπληρώσεις των μαθημάτων. Τέλος θα πρέπει να καταβληθεί προσπάθεια ώστε το πρόγραμμα να είναι όσο πιο συμπαγές γίνεται.

6. *Επίλυση του προβλήματος με τη χρήση πολυπρακτορικού συστήματος*

Η υλοποίηση του προγράμματος πραγματοποιήθηκε σε γλώσσα προγραμματισμού Java χρησιμοποιώντας την πλατφόρμα του eclipse. Το eclipse είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment, IDE), το οποίο περιέχει μια βάση εργασίας και ένα επεκτάσιμο σύστημα plug-in για την προσαρμογή του περιβάλλοντος. Χρησιμοποιείται κυρίως για τη συγγραφή προγραμμάτων σε Java και είναι διαθέσιμο στην ιστοσελίδα https://eclipse.org/ (Eclipse, 2015).

Στο πρόγραμμα τίθενται οι ισχυροί και χαλαροί περιορισμοί, όπως αυτοί περιγράφονται στην ενότητα 6.1

Το πολυπρακτορικό σύστημα αποφασίστηκε να υλοποιηθεί στο περιβάλλον του eclipse με τη χρήση του Java Agent Development Framework (JADE), το οποίο περιέχει ενσωματωμένο το Contract Net Protocol και το οποίο θα χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής.

Προκειμένου να λειτουργήσει σωστά η εφαρμογή μας χρειάστηκε να γίνουν τα ακόλουθα βήματα:

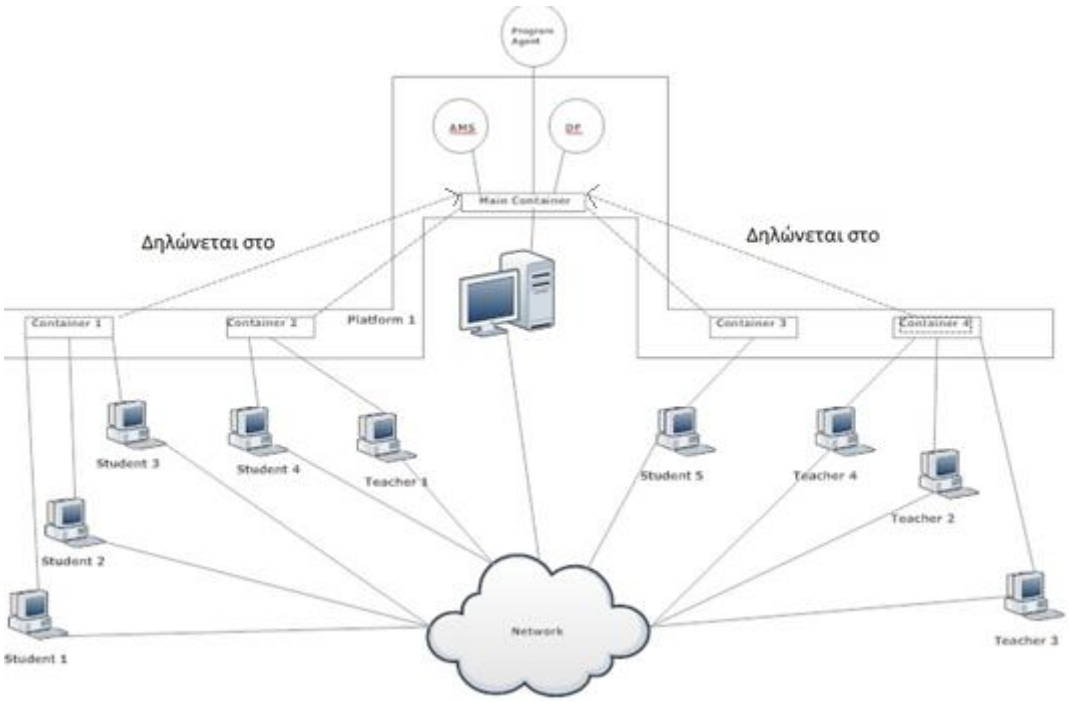
1. Τοποθέτηση των αρχείων της εφαρμογής στο workspace του eclipse.
2. Τοποθέτηση των αρχείων της JADE στο workspace του eclipse.
3. Δεξί κλικ στο Project → Properties → Java Build Path → Add External JARs. Εκεί τοποθετούμε τα jade.jar και commons-codec-1.3.jar.
4. Run Configurations → Java Application. Στο main → main class επιλέγουμε jade.Boot και στο Arguments –gui και τη λίστα των πρακτόρων που θέλουμε να εκκινήσουμε.

Η υλοποίηση βασίζεται στα συμπεράσματα που προέκυψαν από το κεφάλαιο 5 το οποίο περιέχει την ανάλυση των απαιτήσεων. Οι αποφάσεις οι οποίες πάρθηκαν αφορούν:

1. Τη δημιουργία 3 κλάσεων πρακτόρων (προγραμματιστής, καθηγητές, σπουδαστές).
2. Την ύπαρξη 3 σταδίων διαπραγμάτευσης για τους καθηγητές. Προκειμένου να υπάρξει συντόμευση της διαδικασίας και να καταλήξουμε σε μια αποδεκτή λύση, στα δυο πρώτα στάδια, οι πράκτορες των καθηγητών θα αποστέλλουν ελεύθερα τις προτιμήσεις τους και θα ενημερώνονται για την πορεία του προγράμματος, ενώ στο τρίτο στάδιο θα τους προτείνονται πιθανές λύσεις του προβλήματος για να επιλέξουν.

3. Θα υπάρχει ένας εκπρόσωπος φοιτητών για κάθε εξάμηνο. Άρα συνολικά θα δημιουργηθούν 7 πράκτορες φοιτητών (ένας για κάθε εξάμηνο σπουδών).
4. Οι σπουδαστές θα αποστέλλουν τις προτιμήσεις τους μόνο στην αρχή της διαδικασίας. Οι προτιμήσεις αυτές θα αφορούν μόνο τα διαστήματα που δεν επιθυμούν να τοποθετηθούν μαθήματα του εξαμήνου τους. Αυτές θα λαμβάνονται ως χαλαροί περιορισμοί και μετά την ολοκλήρωση του προγράμματος θα αποστέλλεται στους εκπροσώπους των σπουδαστών το πρόγραμμα του εξαμήνου τους.
5. Την ύπαρξη δυο κατηγοριών περιορισμών για τους καθηγητές. Στην πρώτη κατηγορία, η οποία θα αποτελεί και ισχυρό περιορισμό, ο καθηγητής θα δηλώνει τα χρονικά διαστήματα που δεν μπορεί να αναλάβει κάποιο μάθημα και στη δεύτερη, η οποία θα αποτελεί και χαλαρό περιορισμό, ο καθηγητής θα δηλώνει τα διαστήματα που επιθυμεί/δεν επιθυμεί να αναλάβει κάποιο μάθημα.
6. Τη δημιουργία κενού διαστήματος στο πρόγραμμα για την τοποθέτηση πιθανών αναπληρώσεων.
7. Το πρόγραμμα των μαθημάτων να γίνει όσο πιο συμπαγές γίνετε (χωρίς κενά).

Το πολυπρακτορικό σύστημα (MAS) της εφαρμογής μας εμφανίζεται στο ακόλουθο σχήμα (Εικόνα 14):



Εικόνα 14. Το πολυπρακτορικό σύστημα (MAS)

6.1 Περιορισμοί κατά την υλοποίηση του προγράμματος

Κατά τη διαδικασία της δημιουργίας του προγράμματος θα πρέπει να τεθούν κάποιοι περιορισμοί, προκειμένου το πρόγραμμα που θα δημιουργηθεί να είναι όσο το δυνατόν πιο

αποτελεσματικό και να μην δημιουργηθούν προβλήματα και συγκρούσεις με αυτό. Οι περιορισμοί αυτοί χωρίζονται σε ισχυρούς και χαλαρούς περιορισμούς.

6.1.1 Ισχυροί περιορισμοί

Οι ισχυροί περιορισμοί, δηλαδή οι περιορισμοί οι οποίοι δεν πρέπει να παραβιαστούν είναι:

- Δεν επιτρέπεται ο ίδιος καθηγητής να διδάσκει 2 μαθήματα ταυτόχρονα.
- Δεν επιτρέπεται να τοποθετηθούν δυο μαθήματα του ίδιου εξαμήνου την ίδια ημέρα και ώρα. Αυτό γίνεται προκειμένου οι σπουδαστές να μπορούν να επιλέξουν ελεύθερα όποια μαθήματα επιθυμούν.
- Δεν επιτρέπεται το ίδιο μάθημα να τοποθετηθεί και δεύτερη φορά μέσα στην ίδια ημέρα.
- Δεν επιτρέπεται να τοποθετηθούν παραπάνω από 4 μαθήματα του ίδιου εξαμήνου την ίδια ημέρα. Αυτό γίνεται με σκοπό οι σπουδαστές να μην είναι υποχρεωμένοι παρακολουθούν πολλά μαθήματα την ίδια ημέρα, γεγονός που ενδέχεται να οδηγήσει στην εξάντληση τους.
- Δεν επιτρέπεται να τοποθετηθούν παραπάνω από 2 μαθήματα του ίδιου καθηγητή την ίδια ημέρα. Αυτό γίνεται με σκοπό το πρόγραμμα του καθηγητή να κατανεμηθεί ομοιόμορφα σε όλες τις ημέρες.
- Δεν επιτρέπεται να τοποθετηθούν μαθήματα σε ημέρες και ώρες στις οποίες ο καθηγητής δηλώνει ότι δεν μπορεί να διδάξει στο τμήμα. Πιθανόν αυτό συμβαίνει επειδή ο καθηγητής διδάσκει και σε άλλο Πανεπιστημιακό Ίδρυμα.

6.1.2 Χαλαροί περιορισμοί

Πρόκειται για περιορισμούς, οι οποίοι λαμβάνονται υπόψιν κατά τη διαδικασία της δημιουργίας του προγράμματος, αλλά η παραβίαση τους δεν θα οδηγήσει σε εσφαλμένη επίλυση του. Οι χαλαροί περιορισμοί, οι οποίοι αποφασίστηκε να τεθούν είναι:

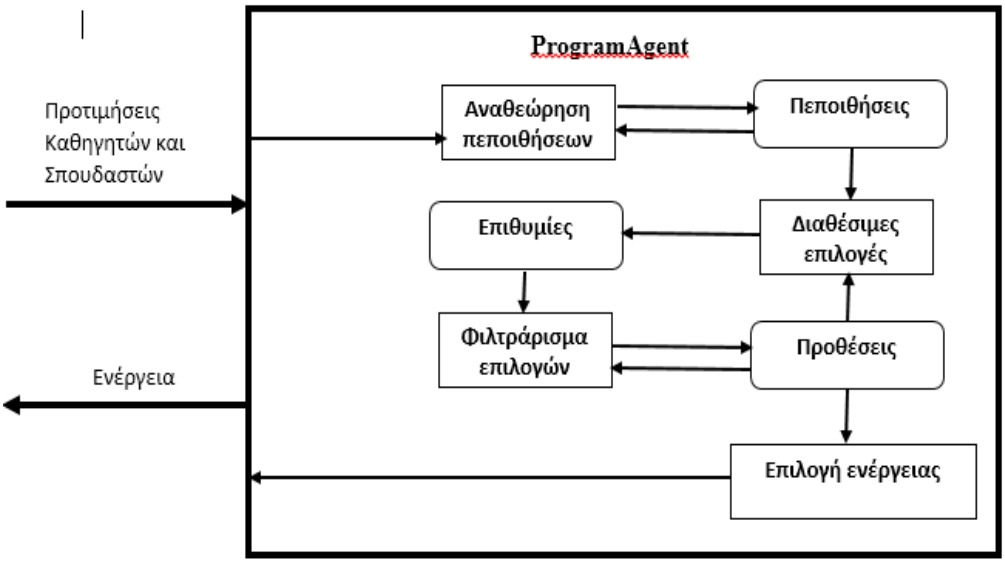
- Το σύνολο των προτιμήσεων των σπουδαστών
- Οι επιθυμίες του καθηγητή, οι οποίες έχουν την μορφή Επιθυμώ/Δεν επιθυμώ.
- Σε μαθήματα που περιέχουν και θεωρητικό και εργαστηριακό μέρος, το θεωρητικό μέρος τοποθετείται πρώτο μέσα στην εβδομάδα.
- Ο ισχυρός περιορισμός «Δεν μπορώ» των καθηγητών, ο οποίος ενδέχεται να μετατραπεί σε μεταγενέστερο στάδιο της διαπραγμάτευσης σε χαλαρό περιορισμό.

6.2 *Περιγραφή των πρακτόρων*

Οι τρεις κλάσεις πρακτόρων, οι οποίες υλοποιήθηκαν είναι:

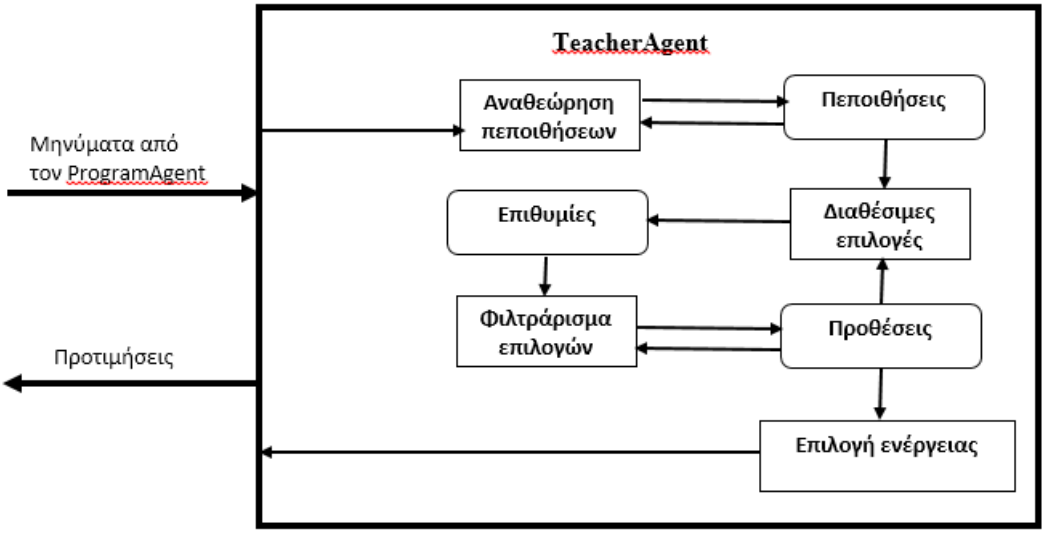
1. **“ProgramAgent”,** δηλαδή ο πράκτορας ο οποίος είναι υπεύθυνος για τη δημιουργία του προγράμματος. Πρόκειται για τον κεντρικό πράκτορα του συστήματος, ο οποίος είναι υπεύθυνος για την επικοινωνία με τους υπόλοιπους πράκτορες και το συντονισμό της δημιουργίας του προγράμματος. Η λειτουργία του περιγράφεται αναλυτικότερα στη συνέχεια.
2. **“TeacherAgent”,** δηλαδή ο προσωπικός πράκτορας του κάθε καθηγητή που διδάσκει στο τμήμα. Αυτός μέσα από ένα γραφικό περιβάλλον επικοινωνίας ενημερώνει τον καθηγητή για τη διαδικασία, λαμβάνει τις προτιμήσεις του και τον ειδοποιεί για τυχόν συγκρούσεις και για τα πρόγραμμα του, όπως αυτό εξελίσσεται.
3. **“StudentAgent”**, δηλαδή ο προσωπικός πράκτορας των σπουδαστών του τμήματος. Όπως προέκυψε από την ανάλυση απαιτήσεων, απαιτείται ένας πράκτορας για κάθε εξάμηνο. Αυτός θα εκλέγεται από τους υπόλοιπους σπουδαστές του τμήματος, θα αποστέλλει τις προτιμήσεις τους και θα ενημερώνεται για το τελικό πρόγραμμα του εξαμήνου, όπως αυτό θα δημιουργηθεί.

Ο εσωτερικός κόσμος του *ProgramAgent* θα ακολουθεί τη λογική BDI στον οποίο οι πεποιθήσεις του πράκτορα είναι η γνώση του για το περιβάλλον και των πρακτόρων που το αποτελούν, οι επιθυμίες του θα είναι σωστή και αποτελεσματική επικοινωνία του με τους πράκτορες των καθηγητών και των σπουδαστών και πρόθεση του είναι η σωστή και πλήρης δημιουργία του προγράμματος μαθημάτων της σχολής. Ο εσωτερικός κόσμος του πράκτορα περιγράφεται στο ακόλουθο σχήμα (Εικόνα 15):



Εικόνα 15.Εσωτερικός κόσμος ProgramAgent

Ο εσωτερικός κόσμος του *TeacherAgent* θα ακολουθεί και αυτός τη λογική BDI, στον οποίο οι πεποιθήσεις του πράκτορα είναι η γνώση του για το περιβάλλον και τον *ProgramAgent* με τον οποίο θα επικοινωνεί, οι επιθυμίες του είναι η σωστή και αποτελεσματική επικοινωνία του με τον *ProgramAgent*, με σκοπό τη δημιουργία του καλύτερου προγράμματος σύμφωνα με τις ιδιαίτερες προτιμήσεις του και πρόθεση του είναι η ικανοποίηση όσο το δυνατόν περισσότερων προτιμήσεων του και κατ' επέκταση η δημιουργία του καλύτερου προσωπικού του προγράμματος μαθημάτων. Ο εσωτερικός κόσμος του πράκτορα περιγράφεται στο ακόλουθο σχήμα (Εικόνα 16):



Εικόνα 16.Εσωτερικός κόσμος TeacherAgent

Ο *StudentAgent* είναι ένας πράκτορας, ο οποίος αποστέλλει απλά τις προτιμήσεις των σπουδαστών και λαμβάνει το πρόγραμμα του εξαμήνου. Δεν θα έχει περισσότερες επιλογές κατά τη διαδικασία τη διαπραγμάτευσης.

6.3 Μοντέλο διαπραγμάτευσης

Στη διαδικασία της διαπραγμάτευσης ανάμεσα στον *ProgramAgent* και τους υπόλοιπους πράκτορες του συστήματος διακρίνονται τα τρία ακόλουθα στάδια:

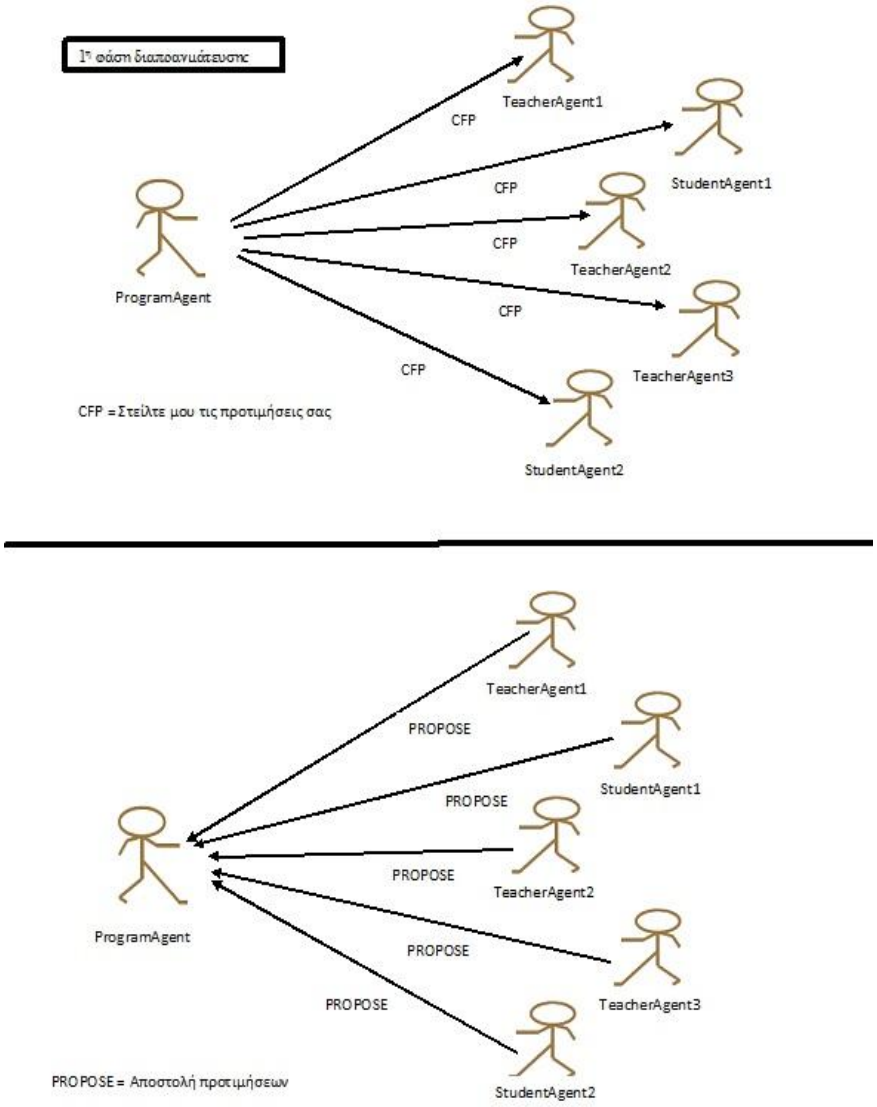
- **1ο στάδιο**: Ο ProgramAgent λαμβάνει τις προτιμήσεις από τους TeacherAgents και StudentAgents και προσπαθεί να δημιουργήσει το πρόγραμμα. Αποστέλλει στους TeacherAgents τις προτιμήσεις που δεν κατάφερε να ικανοποιήσει και το πρόγραμμα του καθηγητή, όπως αυτό έχει δημιουργηθεί μέχρι στιγμής.
- **2ο στάδιο**: Ο ProgramAgent λαμβάνει από τους καθηγητές για τους οποίους δεν ικανοποιήθηκε το σύνολο των προτιμήσεων τους, τις νέες τους προτιμήσεις. Μόλις λάβει όλες τις προτιμήσεις, προσπαθεί και πάλι να δημιουργήσει το πρόγραμμα. Αν δεν καταφέρει να τις ικανοποιήσει στο σύνολο τους, τότε αποστέλλει και πάλι πίσω τις

προτιμήσεις που δεν κατάφερε να ικανοποιήσει, το πρόγραμμα όπως έχει δημιουργηθεί μέχρι στιγμής, καθώς επίσης και προτάσεις για τα μαθήματα στα οποία παρατηρούνται οι συγκρούσεις.

- **3ο στάδιο**: Ο ProgramAgent λαμβάνει τις προτάσεις των καθηγητών για τα εναπομείναντα μαθήματα και τις τοποθετεί στο πρόγραμμα. Στη συνέχεια αποστέλλει σε αυτούς, καθώς και στους StudentAgents το τελικό πρόγραμμα και τερματίζει τη λειτουργία του.

Αν σε οποιοδήποτε από τρία αυτά στάδια ολοκληρωθεί το πρόγραμμα χωρίς συγκρούσεις, τότε ο ProgramAgent αποστέλλει το τελικό πρόγραμμα και τερματίζει τη λειτουργία του.

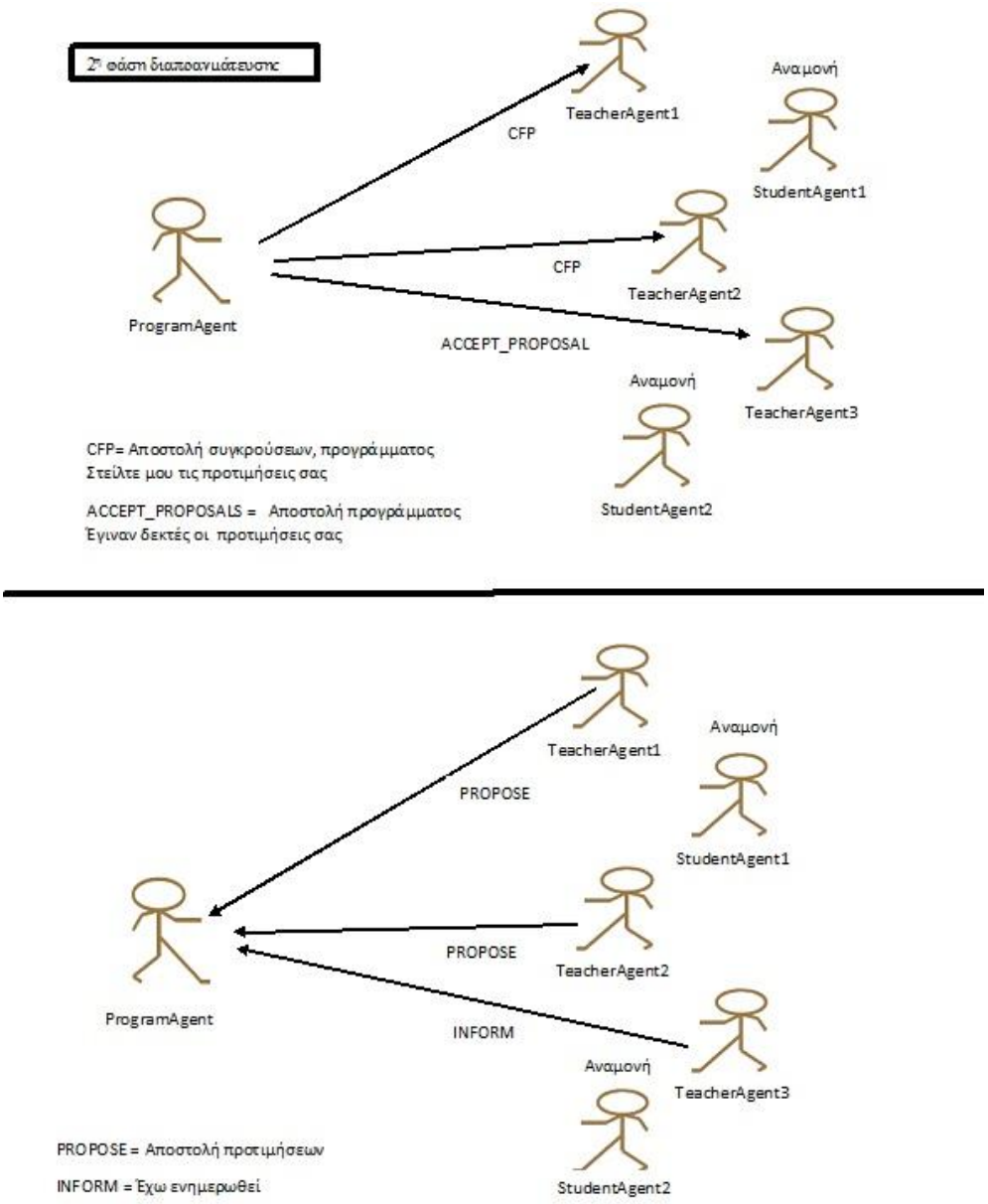
Στο 1ο στάδιο της διαπραγμάτευσης ο ProgramAgent εντοπίζει τους διαθέσιμους TeacherAgents και StudentAgents και τους ενημερώνει ότι ξεκίνησε η διαδικασία της δημιουργίας του προγράμματος. Αυτοί με τη σειρά τους του αποστέλλουν τις προτιμήσεις τους. Αυτή η φάση διαπραγμάτευσης περιγράφεται από το ακόλουθο σχήμα (Εικόνα 17):



Εικόνα 17. 1ο στάδιο διαπραγμάτευσης

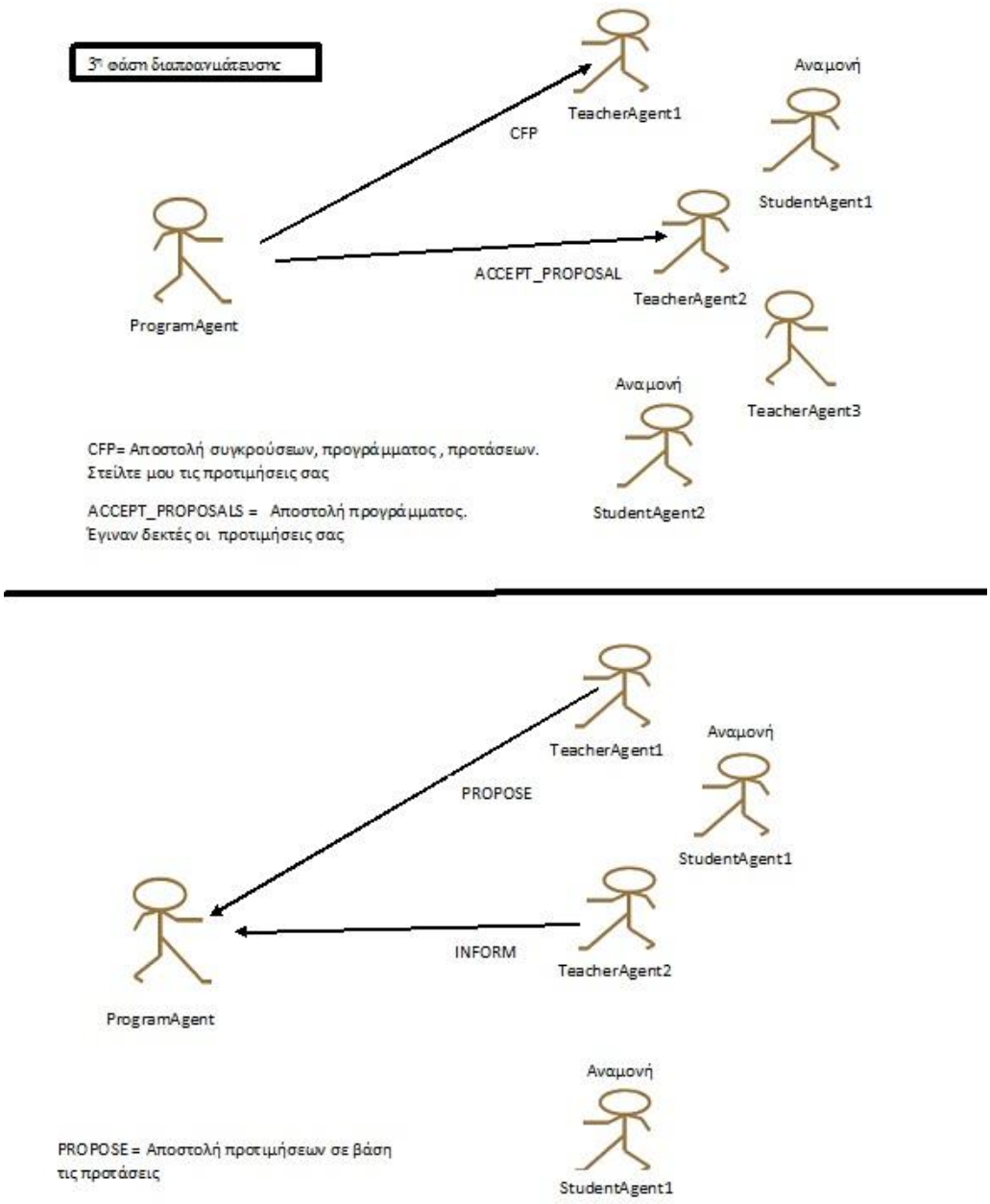
Οι TeacherAgents και StudentAgents αφού λάβουν την πρόσκληση αυτή, δηλώνουν τις προτιμήσεις τους σε ημέρες και ώρες και τις αποστέλλουν στον ProgramAgent για να τις επεξεργαστεί.

Στη συνέχεια ο ProgramAgent αφού λάβει τις προτάσεις από τους υπόλοιπους πράκτορες, προσπαθεί να δημιουργήσει το ωρολόγιο πρόγραμμα σύμφωνα με αυτές. Εντοπίζει τυχόν συγκρούσεις που θα παρουσιαστούν και τις αποστέλλει μαζί με το προσωρινό πρόγραμμα που δημιουργεί στους TeacherAgents, προκειμένου αυτοί να του αποστείλουν τις νέες προτιμήσεις τους. Στους TeacherAgents που ολοκλήρωσαν χωρίς συγκρούσεις αποστέλλεται ενημερωτικό μήνυμα, μαζί με το πρόγραμμα τους. Οι StudentAgents μπαίνουν σε κατάσταση αναμονής και περιμένουν την ολοκλήρωση του προγράμματος. Αυτή η δεύτερη φάση διαπραγμάτευσης περιγράφεται στο επόμενο σχήμα (Εικόνα 18):



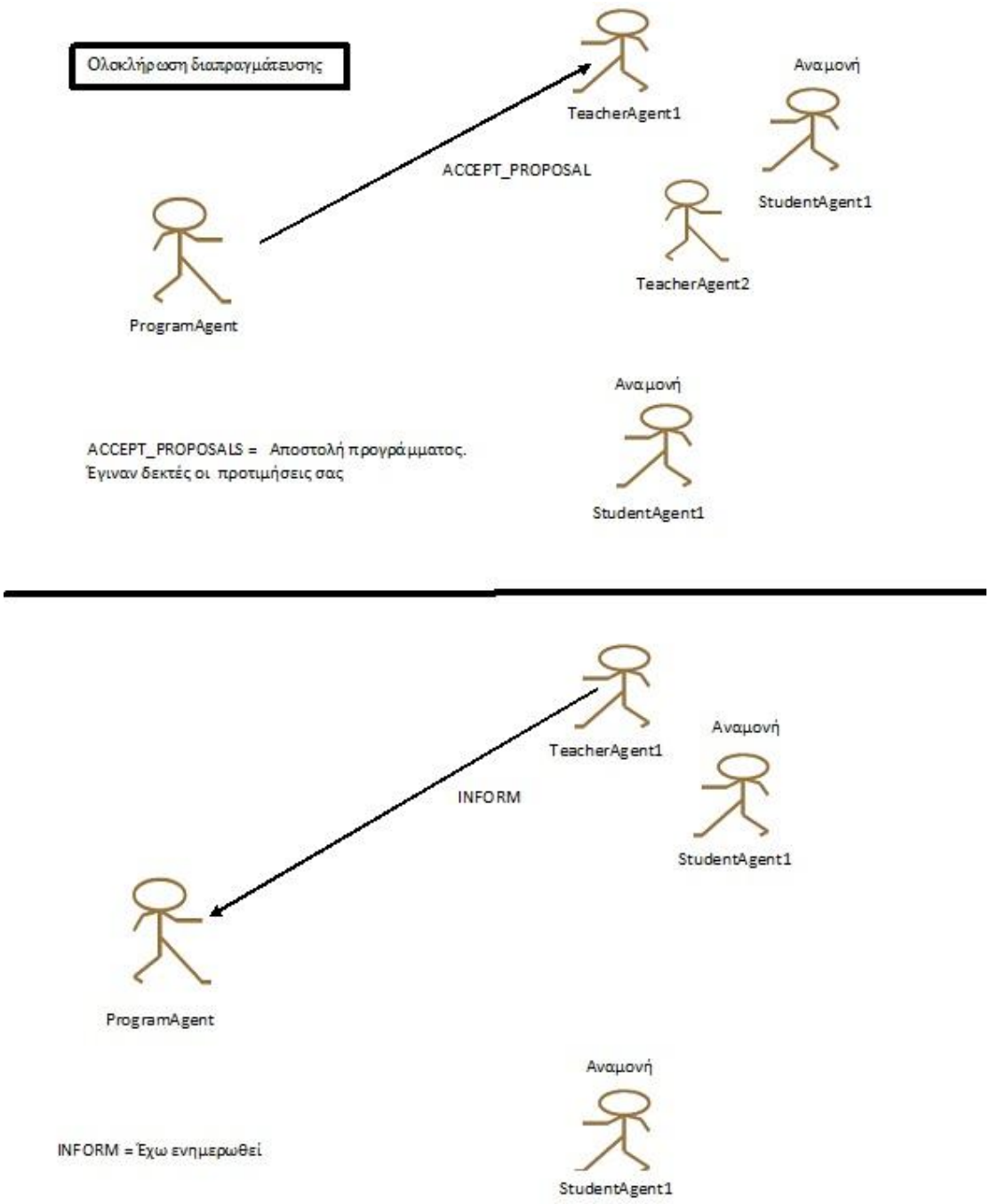
Εικόνα 18.2η φάση διαπραγμάτευσης

Μετά την εκ νέου αποστολή των προτιμήσεων από τους πράκτορες που δεν έχει ολοκληρωθεί το πρόγραμμα τους γιατί υπάρχουν συγκρούσεις, ο ProgramAgent αφού λάβει το σύνολο των προτιμήσεων από τους TeacherAgents, προσπαθεί εκ νέου να δημιουργήσει το πρόγραμμα. Αν εντοπίσει συγκρούσεις, τότε τις αποστέλλει μαζί με το προσωρινό πρόγραμμα που δημιουργεί στους TeacherAgents. Αυτή τη φορά αποστέλλει επίσης προτάσεις, για την τοποθέτηση των μαθημάτων που παρουσιάζεται το πρόβλημα. Ζητάει από τους TeacherAgents να του αποστείλουν τις τελικές προτάσεις τους. Στους TeacherAgents που ολοκλήρωσαν χωρίς συγκρούσεις αποστέλλεται ενημερωτικό μήνυμα, μαζί με το πρόγραμμα τους. Η τρίτη αυτή φάση της διαπραγμάτευσης παρουσιάζεται στο επόμενο σχήμα (Εικόνα 19):



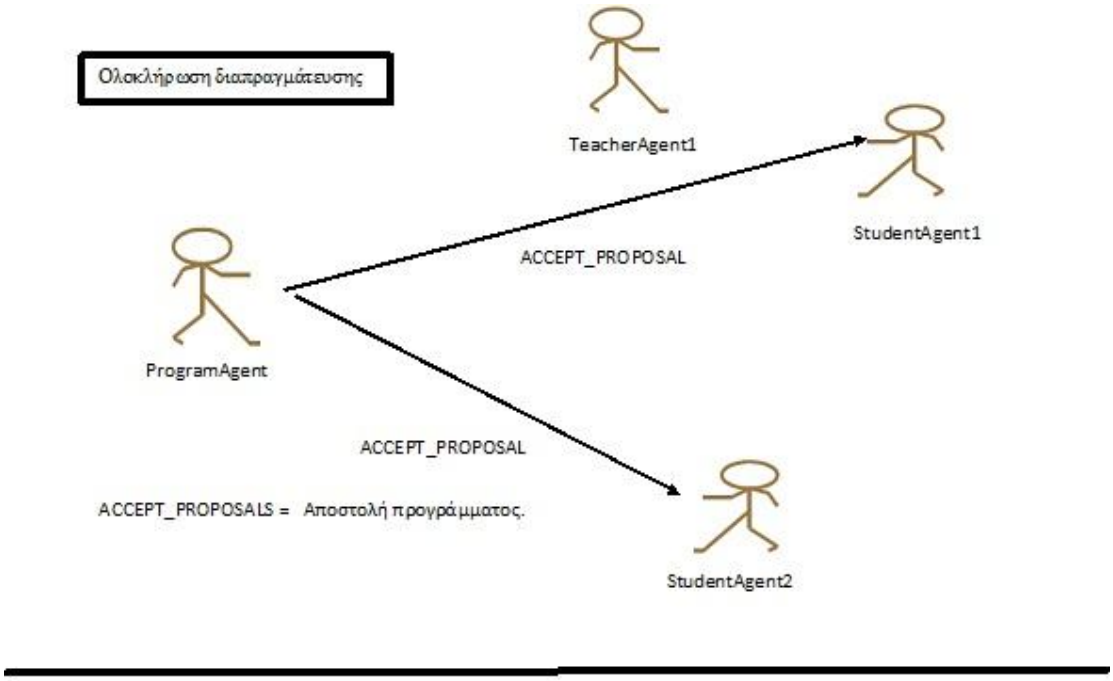
Εικόνα 19.3η φάση διαπραγμάτευσης

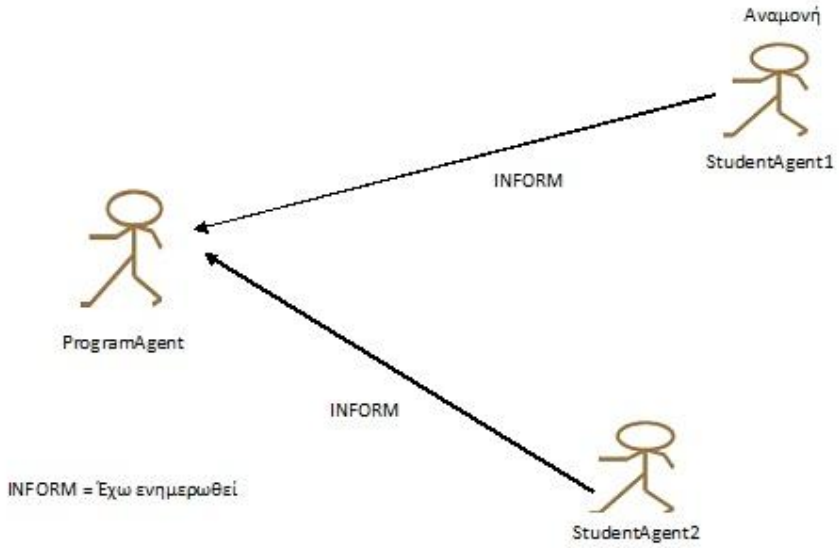
Αφού λάβει ο ProgramAgent τις τελικές προτάσεις από τους TeacherAgents που υπήρχε ακόμη σύγκρουση, τότε τις τοποθετεί στο πρόγραμμα. Αν και πάλι δεν τα καταφέρει με αυτές τις προτάσεις, τότε αποφασίζει τυχαία που θα τοποθετήσει τα εναπομείναντα μαθήματα, προκειμένου να ολοκληρώσει τη διαδικασία. Αποστέλλει σε αυτούς τους TeacherAgents το πρόγραμμα τους μαζί με ενημερωτικό μήνυμα, όπως φαίνεται στο ακόλουθο σχήμα (Εικόνα 20):



Εικόνα 20. Ολοκλήρωση διαπραγμάτευσης 1

Μόλις ενημερωθεί ότι οι TeacherAgents έχουν λάβει το τελικό πρόγραμμα τους, τότε ξεκινάει μια επικοινωνία με τους StudentAgents που είναι σε αναμονή, προκειμένου να αποστείλει και σε αυτούς το τελικό πρόγραμμα των εξαμήνων και να ολοκληρώσει τη διαδικασία. Αποστέλλει λοιπόν το πρόγραμμα και μόλις ενημερωθεί ότι το έλαβαν τερματίζει τη λειτουργία του. Αυτό περιγράφεται στο επόμενο σχήμα (Εικόνα 21):





Εικόνα 21. Ολοκλήρωση διαπραγμάτευσης 2

6.4 Υλοποίηση

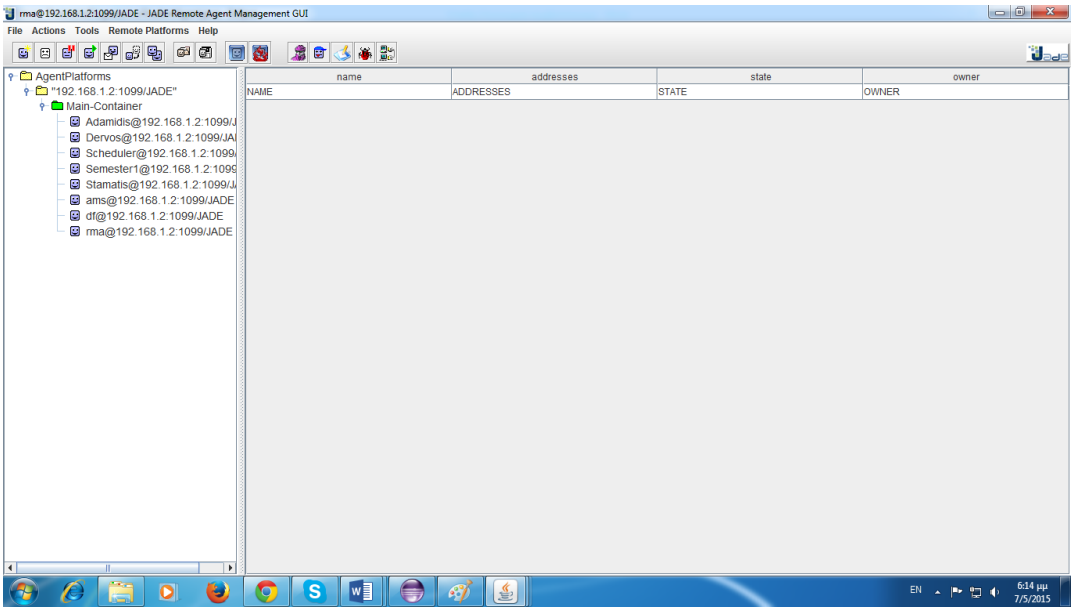
Ο ProgramAgent, ο οποίος ονομάζεται **Scheduler**, μαζί με τους υπόλοιπους TeacherAgents και StudentAgents ξεκινούν τη λειτουργία τους.

Οι TeacherAgents το πρώτο πράγμα το οποίο κάνουν είναι να δηλώσουν στον DF την ύπαρξη τους και τις υπηρεσίες που προσφέρουν. Αυτό γίνεται με τον ακόλουθο κώδικα:

```
// Δήλωση των Teacher service στις yellow pages
DFAgentDescription dfd = new DFAgentDescription();
dfd.setName(getAID());
ServiceDescription sd = new ServiceDescription();
sd.setType("Teacher-Agent");
sd.setName("JADE-timetable-program");
dfd.addServices(sd);
try {
DFService.register(this, dfd);
}
catch (FIPAException fe) {
fe.printStackTrace();
}

// Εκκίνηση Behaviours για τον TeacherAgent
addBehaviour(new IntitialNegotiation());
addBehaviour(new program_finished());
```

Σε όλη τη διαδικασία λειτουργίας του Scheduler και κατά τη διαδικασία διαπραγμάτευσης στο gui της jade, το οποίο εκτελείται καθ' όλη τη διάρκεια της εφαρμογής, έχουμε τη δυνατότητα να ελέγξουμε τον τρόπο λειτουργίας της εφαρμογής μας. Το gui εμφανίζεται στην επόμενη εικόνα:

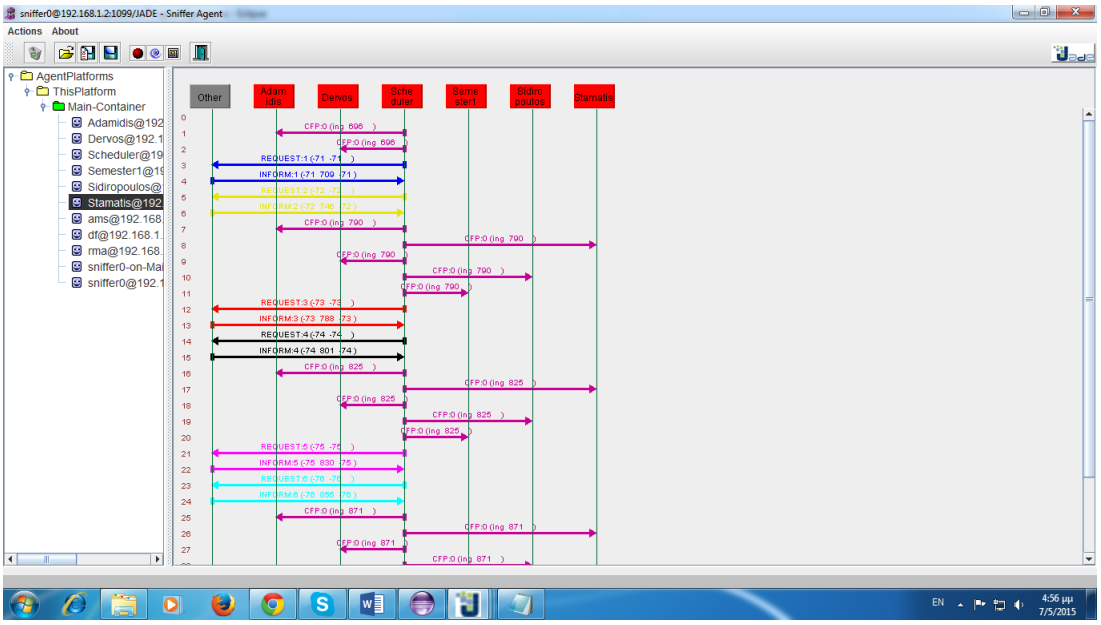


Εικόνα 22. Το gui της jade.

Μέσω αυτού μπορούμε να εκκινήσουμε ή να τερματίσουμε κάποιον πράκτορα της εφαρμογής, να αποστείλουμε μηνύματα ή να παρακολουθήσουμε τη διαδικασία ανταλλαγής μηνυμάτων μεταξύ των πρακτόρων.

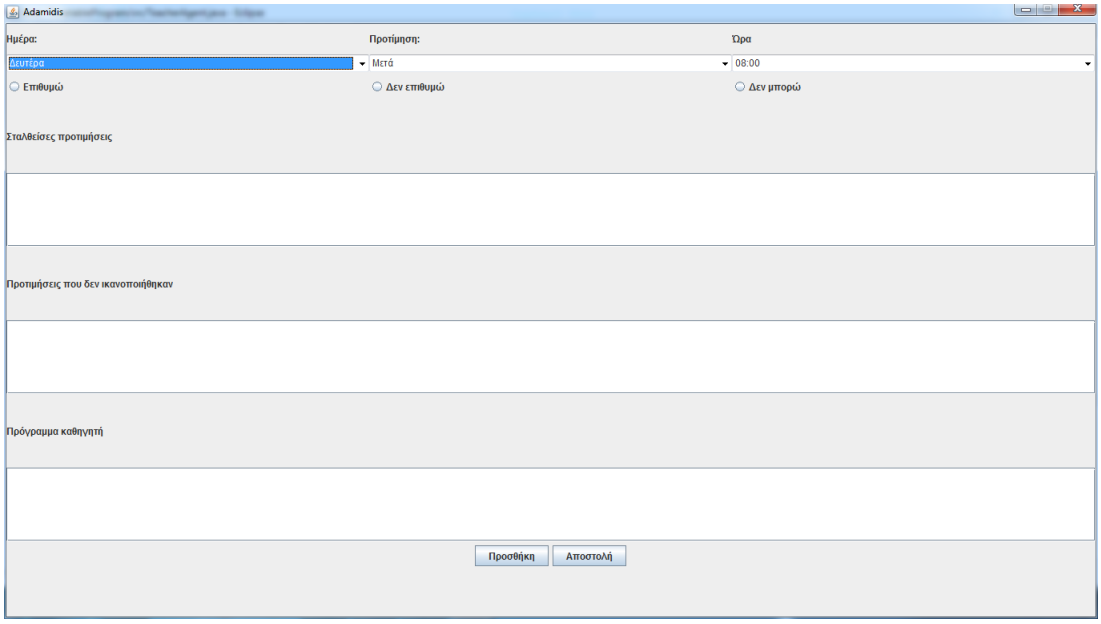
Τη διαδικασία ανταλλαγής μηνυμάτων μπορούμε να την παρακολουθήσουμε ξεκινώντας τον Sniffer Agent και επιλέγοντας «Do sniff this Agent» για κάθε πράκτορα που επιθυμούμε. Ένα

χαρακτηριστικό παράδειγμα ανταλλαγής μηνυμάτων που παρακολουθείται μέσω τουSniffer Agent παρουσιάζεται στην ακόλουθη εικόνα:



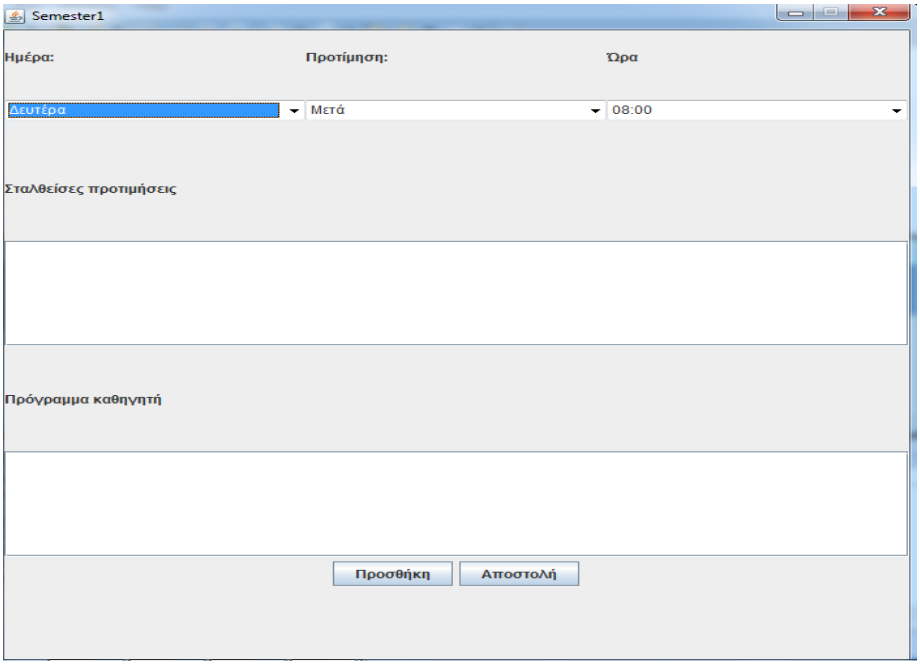
Εικόνα 23. Ο Sniffer Agent.

Στους TeacherAgents εμφανίζεται αρχικά ένα ενημερωτικό μήνυμα, το οποίο τους ενημερώνει για τη διαδικασία που θα ακολουθηθεί. Αφού το διαβάσουν προσεκτικά και πατήσουν το κουμπί «Επόμενο», εμφανίζεται το γραφικό μέσω του οποίου θα στέλνουν τις προτιμήσεις τους στον Scheduler και θα ενημερώνονται για τυχόν συγκρούσεις και για την πορεία δημιουργίας του προγράμματος τους. Η εικόνα που αντικρύζουν είναι η επόμενη:



Εικόνα 24. Το γραφικό του TeacherAgent

Αντίστοιχη διαδικασία ακολουθούν και οι StudentAgents. Το γραφικό, το οποίο αντικρύζουν και περιέχει λιγότερες επιλογές από τους TeacherAgents είναι:



Εικόνα 25. Το γραφικό των StudentAgents

Ο ProgramAgent αφού δηλώσει και αυτός στον DF την ύπαρξη του και τις υπηρεσίες που προσφέρει, ξεκινάει μια διαδικασία αναζήτησης των TeacherAgents και StudentAgents με τους οποίους πρέπει να διαπραγματευτεί. Στη συνέχει ξεκινάει τις Cyclic Behaviour Negotiation και FinishProgram προκειμένου να εκτελέσει τις ενέργειες για τις οποίες έχει προγραμματιστεί. Ο κώδικας με τον οποίο πραγματοποιείται αυτό είναι:

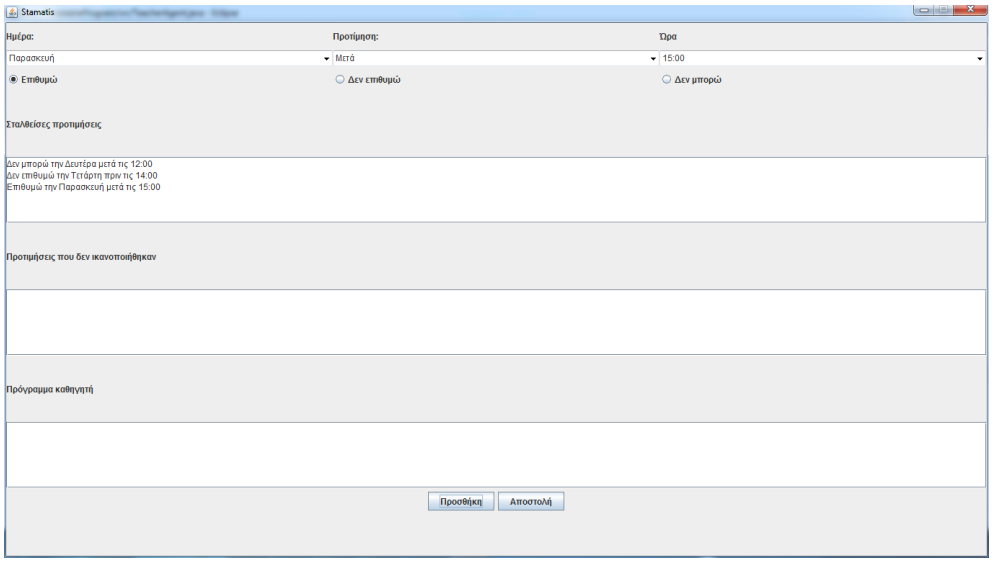
```
// Δημιουργία TickerBehaviour με την οποία δηλώνεται ο ProgramAgent και αναζητά τους TeacherAgents και StudentAgents
addBehaviour(new TickerBehaviour(this, 5000) {
	protected void onTick() {
	// Ενημέρωσε τη λίστα των teacher agents
	DFAgentDescription template = new DFAgentDescription();
	ServiceDescription sd = new ServiceDescription();
	sd.setType("Program-Agent");
	template.addServices(sd);
	try {
		DFAgentDescription [] result = DFService.search(myAgent,template);
		teacherAgents = new AID[result.length];
		for (int i = 0; i < result.length; ++i) {
			teacherAgents[i] = result[i].getName();
		}
	}
	catch (FIPAException fe) {
		fe.printStackTrace();
	}
	// Ενημέρωσε τη λίστα των student agents
	DFAgentDescription templateS = new DFAgentDescription();
	ServiceDescription sdS = new ServiceDescription();
	sdS.setType("Student-Agent");
	templateS.addServices(sdS);
	try {
```

```
            DFAgentDescription[]resultS=DFService.search(myAgent, templateS);
studentAgents = new AID[resultS.length];
            for (int i = 0; i < resultS.length; ++i) {
                  studentAgents[i] = resultS[i].getName();
            }
          }
          catch (FIPAException fe) {
                  fe.printStackTrace();
          }
// Εκκίνηση των Behaviour
          myAgent.addBehaviour(new Negotiation());
          myAgent.addBehaviour(new finishProgram());
}
} );
```

Ο Scheduler αποστέλλει στους TeacherAgents και StudentsAgents ένα μήνυμα Call For Proposal (CFP) με το οποίο τους ενημερώνει ότι ξεκίνησε τη λειτουργία του και τους ζητάει να του αποστείλουν τις προτάσεις τους.

Μόλις οι TeacherAgents και οι StudentAgents λάβουν το μήνυμα CFP ξεκινούν να τοποθετούν τις προτιμήσεις τους.

Οι TeacherAgents, μέσω των αντίστοιχων χρηστών/καθηγητών που εκπροσωπούν, επιλέγουν Ημέρα, Ώρα και τρεις διαφορετικούς τύπους περιορισμών προκριμένου να δηλώσουν την προτίμηση τους για την Ημέρα αυτή. Αυτοί είναι «Επιθυμώ», «Δεν επιθυμώ» και «Δεν μπορώ», επιλέγοντας κατάλληλα το ανάλογο check Button. Τοποθετούνε μια προς μια τις προτιμήσεις τους επιλέγοντας κάθε φορά «Προσθήκη». Μόλις τις τοποθετήσουν όλες, τότε επιλέγουν «Αποστολή», προκειμένου αυτές να αποσταλούν στον Scheduler.



Εικόνα 26. Τοποθέτηση προτιμήσεων απο τον TeacherAgent

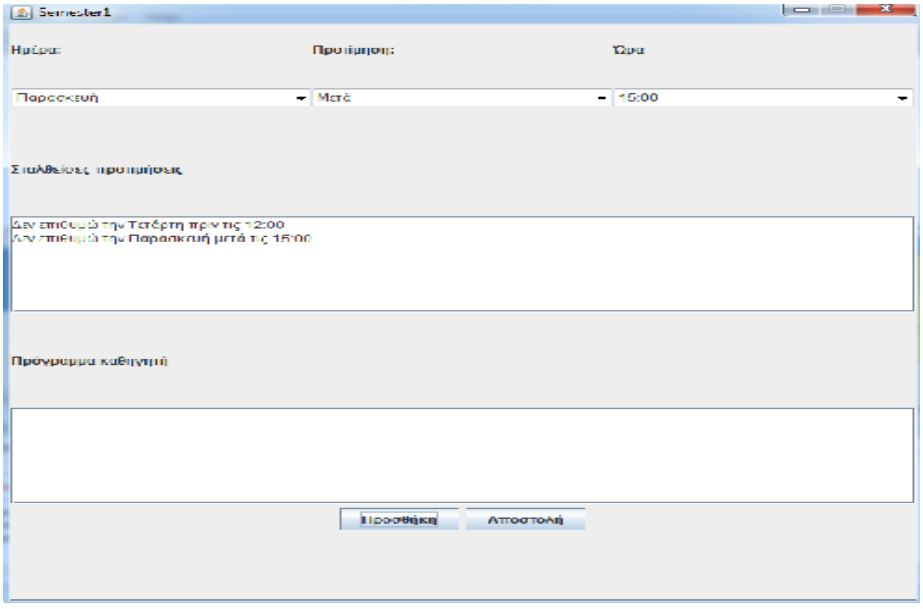
Η συλλογή των προτιμήσεων τους προκειμένου στη συνέχεια να γίνει η αποστολή τους στον Scheduler γίνετε μέσω της μεθόδου updatePreferences.

```
public void updatePreferences(final String day, final String pref, final String hour, final
String check1, final String check2, final String check3) {
        if (check1.equals("true") || check2.equals("true") || check3.equals("true")){
                if(preferences == " "){
                        preferences = getAID().getName() +"#"+day+"#"+pref+"#"+hour+
                        "#"+check1+"#"+check2+"#"+check3;
                }else {
                          String temp = getAID().getName()+ "#"+day+"#"+pref+"#"+hour+
                          "#"+check1+"#"+check2+"#"+check3;
                         if (!preferences.toLowerCase().contains(temp.toLowerCase())) {
                          preferences = preferences + "#" + getAID().getName()+"#"+day+"#"
                         +pref+"#"+hour+"#"+check1+"#"+check2+"#"+check3;
                         }
                }
        }
}
```

Η αποστολή τους στον Scheduler γίνετε με τον ακόλουθο τρόπο:

```
String returnMsg = msg.getContent();
ACLMessage reply = msg.createReply();
reply.setPerformative(ACLMessage.PROPOSE);
reply.setContent(preferences);
myAgent.send(reply);
```

Οι StudentAgentsμε τη σειρά τους, μέσω των αντίστοιχων χρηστών/φοιτητών που εκπροσωπούν, επιλέγουν μόνο ημέρες και ώρες στις οποίες δεν επιθυμούν να τοποθετηθούν μαθήματα του εξαμήνου τους. Στη συνέχεια αποστέλλουν τις προτιμήσεις τους με παρόμοιο τρόπο.



Εικόνα 27.Τοποθέτηση προτιμήσεων από τον StudentAgent

Η λειτουργία του Scheduler ελέγχεται μέσω μιας μεταβλητής step, η οποία μας υποδεικνύει σε ποια φάση βρίσκεται ο Scheduler.

Όταν το step έχει την τιμή 1, τότε ο Scheduler λαμβάνει τις προτιμήσεις των TeacherAgents και StudentAgents και τις κωδικοποιεί κατάλληλα. Μόλις λάβει όλες τις προτιμήσεις από όλους τους πράκτορες που υπάρχουν, τότε καλεί τη μέθοδο set_priorities, η οποία βρίσκεται μέσα στην κλάση PreferencesHandler, μέσω της οποίας δημιουργείται ένα HaspMap με τις προτεραιότητες για τους καθηγητές. Για τη δημιουργία των προτεραιοτήτων λαμβάνονται υπόψιν η βαθμίδα που ανήκουν οι καθηγητές, ο αριθμός των προτιμήσεων που έστειλαν και η σειρά άφιξης των προτιμήσεων τους. Αυτό γίνετε τοποθετώντας αντίστοιχα βάρη σε κάθε κατηγορία. Τα βάρη που επιλέχθηκαν είναι:

- Για τη βαθμίδα του καθηγητή (Επίκουρος Καθηγητής/Καθηγητής Εφαρμογών = 5, Καθηγητής = 4 και Αναπληρωτής Καθηγητής = 3).
- Για τον αριθμό των προτιμήσεων βάρος = 3.
- Για τη σειρά άφιξης βάρος = 2.

Ο τύπος μέσω του οποίου προκύπτουν οι προτεραιότητες είναι:

Προτεραιότητα Καθηγητή = *(βαθμίδα*βάρος βαθμίδας) + 3 *(αρ. προτιμήσεων -1) + 2*(σειρά άφιξης -1)*

Αφού δημιουργηθούν οι προτεραιότητες των καθηγητών, τότε μέσω της μεθόδου sort_priorities, η οποία βρίσκεται και αυτή στην PreferencesHandler, ταξινομούνται σε φθίνουσα σειρά και καλείται η μέθοδος CreateProgram, η οποία βρίσκεται στην κλάση Program για να δημιουργηθεί το πρόγραμμα. Μέσω των μεθόδων NoInProgram και teacherProgram, οι οποίες βρίσκονται στην κλάση Program, λαμβάνονται οι προτιμήσεις των καθηγητών που εμφανίζουν συγκρούσεις, καθώς και το πρόγραμμα τους. Η κατάσταση λειτουργίας του Scheduler μεταβαίνει στο step 2. Αυτό γίνεται με τον ακόλουθο τρόπο:

```
if (repliesCnt >= teacherAgents.length && repliesCntS >= studentAgents.length) {
        try {
                flag = true;
                preferencesHandler.setPriorities(arrivals);
                preferencesHandler.sort_priorities();
        program.createProgram(preferencesHandler,preferencesHandler,
preferencesHandler);
                conflicts = program.NoInProgram();
                teacherProgram = program.teacherProgram();
        } catch (IOException e) {
                System.out.println("Cant load program");
                        e.printStackTrace();
        }
                step = 2;
}
```

Στην κλάση Program υπάρχει ένα πίνακας program 3 διαστάσεων. Η πρώτη διάσταση είναι 5 θέσεων για κάθε ημέρα, η δεύτερη διάσταση 13 θέσεων για κάθε ώρα της ημέρας που γίνονται μαθήματα και η τρίτη διάσταση 7 θέσεων που αντιπροσωπεύουν τα εξάμηνα σπουδών.

Επίσης υπάρχουν τα HashMap conflicts, teacherProgram και semesterProgram, στα οποία κρατούνται οι συγκρούσεις που εμφανίζονται, το πρόγραμμα του κάθε καθηγητή και το πρόγραμμα του κάθε εξαμήνου αντίστοιχα.

Στη μέθοδο createProgram πραγματοποιούνται τα ακόλουθα βήματα:

1. Δεσμεύονται θέσεις για το διάστημα που έχει επιλεγεί να πραγματοποιούνται οι αναπληρώσεις του προγράμματος. Επιλέξαμε να γίνονται αναπληρώσεις την Πέμπτη από τις 18:00 μέχρι τις 21:00.
2. Γίνεται η τοποθέτηση των προτιμήσεων των σπουδαστών.
3. Για κάθε καθηγητή σύμφωνα με τις προτεραιότητες που έχουμε δημιουργήσει προηγουμένως, λαμβάνονται τα μαθήματα του από το αρχείο lessons.txt και ξεκινάει η διαδικασία δημιουργίας του προγράμματος του.
4. Αρχικά γίνεται προσπάθεια να ικανοποιήσουμε όλες τις προτιμήσεις που μας έχει αποστείλει, ελέγχοντας ένα πίνακα preference που έχουμε δημιουργήσει.
5. Αφού ολοκληρωθεί η διαδικασία με την ικανοποίηση των προτιμήσεων, μέσω της μεθόδου putRestLessons, γίνετε προσπάθεια να τοποθετηθούν τα εναπομείναντα μαθήματα.
6. Αν δεν τοποθετηθούν όλα τα μαθήματα στο πρόγραμμα, τότε καλείται η μέθοδος putLessonsOnStudentPrefs μέσω της οποία τοποθετούνται τα μαθήματα στις προτιμήσεις των σπουδαστών.
7. Στη συνέχεια καλείται η μέθοδος η putDenEpithumwLessons, μέσω της οποίας τοποθετούνται τα εναπομείναντα μαθήματα εκεί που ο καθηγητής δήλωσε ότι δεν επιθυμεί να τοποθετηθεί μάθημα.
8. Αν και τώρα δεν τοποθετηθούν μαθήματα, τότε μέσω της CheckPreferencedenMporw τοποθετούνται στο conflicts οι προτιμήσεις του καθηγητή που περιέχουν τον περιορισμό «Δεν μπορώ» και οι όποιες μπλοκάρουν το πρόγραμμα.
9. Τοποθετείται στο teacherProgram το πρόγραμμα του καθηγητή.
10. Η διαδικασία συνεχίζεται με τον ίδιο τρόπο και για τους υπόλοιπους καθηγητές και τα βήματα 3 εώς 9 επαναλαμβάνονται.

Εφόσον έχει ολοκληρωθεί η πρώτη φάση της δημιουργίας του προγράμματος, στο HashMap conflicts υπάρχουν οι συγκρούσεις που προέκυψαν από τις προτιμήσεις των καθηγητών και στο HashMap teacherProgram το πρόγραμμα του κάθε καθηγητή.

Στη συνέχεια, ο Scheduler αναλαμβάνει δράση, ελέγχοντας εάν ο καθηγητής υπάρχει στο conflicts. Ένα υπάρχει, τότε του αποστέλλεται και πάλι μήνυμα CFP, το οποίο περιέχει τις συγκρούσεις που παρατηρήθηκαν, καθώς επίσης και το προσωρινό του πρόγραμμα. Το μήνυμα αυτό έχει id = "conflicts-negotiation". Αυτό πραγματοποιείται με τις ακόλουθες γραμμές κώδικα:

```
for (i = 0; i < teacherAgents.length; i++) {
        ACLMessage sendback = new ACLMessage(ACLMessage.CFP);
        String aux = teacherAgents[i].toString();
        String teacher = teacherAddresses.get(aux);
        boolean check = false;
        conflictsBack = "";
        for ( String reciever : conflicts.keySet() ) {
                String reciever1 = reciever.split("#")[0];
                if (aux.contains(reciever1)) {
                        check = true;
                        conflictsBack = conflictsBack + conflicts.get(reciever) + "#";
                }
        }
        if (!check) {
                conflictsBack = "ok";
        }else{
                sendback.addReceiver(teacherAgents[i]);
        }
        teacher = "";
        for ( String key : teacherAddresses.keySet()) {
                if (aux.contains(key)) {
                        teacher = teacherAddresses.get(key);
                }
        }
        tProgram = teacherProgram.get(teacher);
        if(tProgram.equals("")){
                tProgram = "#";
        }

        String sendToTeacher = conflictsBack + "%" + tProgram;
        sendback.setContent(sendToTeacher);
        sendback.setConversationId("conflicts-negotiation");
        sendback.setReplyWith("negotiation"+System.currentTimeMillis());
        myAgent.send(sendback);
        // Prepare the template to get the program negotiation
        mt = MessageTemplate.and(MessageTemplate.
          MatchConversationId("conflicts-negotiation"),
MessageTemplate.MatchInReplyTo(proposeback.getReplyWith()));
}
```

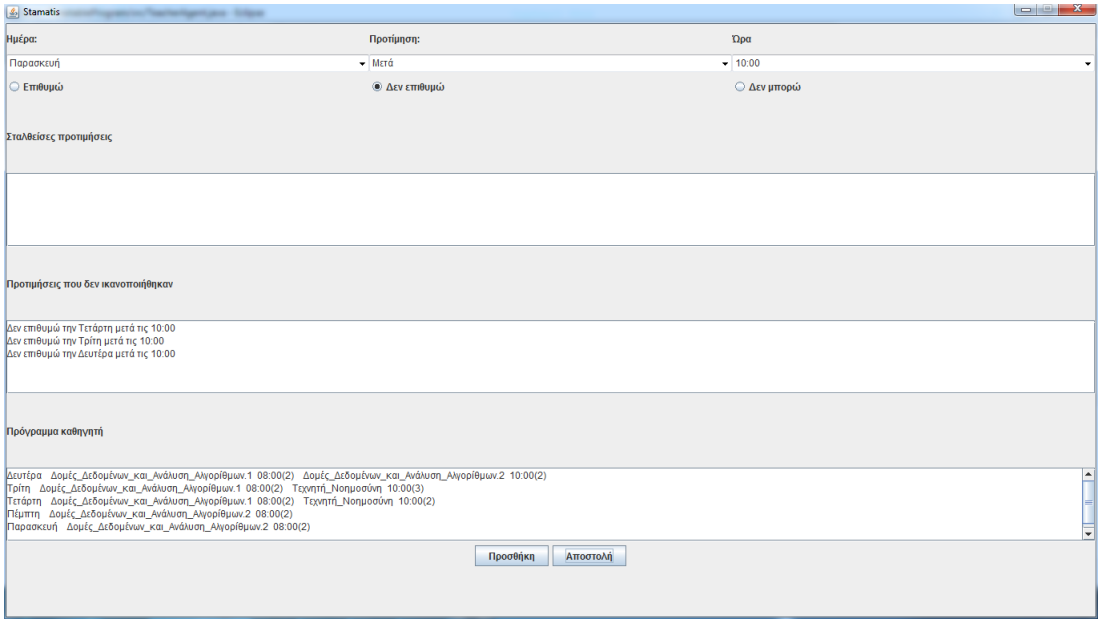
Στους καθηγητές, στους οποίους δεν εμφανίζονται conflicts, αποστέλλεται ένα μήνυμα ACCEPT_PROPOSALS, προκειμένου να ενημερωθούν αυτοί ότι δεν παρουσιάστηκε κάποιο πρόβλημα και το πρόγραμμα τους ολοκληρώθηκε επιτυχώς. Αυτό πραγματοποιείται με τις ακόλουθες γραμμές κώδικα:

```
for (i = 0; i < teacherAgents.length; i++) {
   ACLMessage accept_Proposal = new ACLMessage(ACLMessage.ACCEPT_PROPOSAL);
   String aux = teacherAgents[i].toString();
   boolean check = false;
   for ( String reciever : conflicts.keySet() ) {
         String reciever1 = reciever.split("#")[0];
         if (aux.contains(reciever1)) {
               check = true;
         }
  }
  if (!check) {
         accept_Proposal.addReceiver(teacherAgents[i]);
         String teacher = "";
         for ( String key : teacherAddresses.keySet()) {
               if (aux.contains(key)) {
                     teacher = teacherAddresses.get(key);
               }
         }
         tProgram = teacherProgram.get(teacher);
         agentsFinished.add(teacher);
         if(tProgram.equals("")){
               tProgram = "#";
         }
         String sendToTeacher = tProgram;
         accept_Proposal.setContent(sendToTeacher);
         accept_Proposal.setConversationId("finishing-program");
         accept_Proposal.setReplyWith("accept_Proposal"+System.currentTimeMillis());
         myAgent.send(accept_Proposal);
         // Prepare the template to get proposals
         mt = MessageTemplate.and(MessageTemplate.MatchConversationId("finishing-
         program"),
         MessageTemplate.MatchInReplyTo(accept_Proposal.getReplyWith()));
         }
}
```

Οι TeacherAgents λαμβάνουν το ανάλογο μήνυμα από τον Scheduler (είτε CFP, είτε ACCEPT_PROPOSAL) και αποφασίζουν τι πρέπει να κάνουν. Εάν λάβουν CFP, τότε ενημερώνουν μέσω κατάλληλων μεθόδων το γραφικό του χρήστη και είναι έτοιμες να λάβουν τις καινούργιες προτιμήσεις του καθηγητή. Αυτό γίνεται με τον ακόλουθο κώδικα:

```
conflicts = CreateMessage(parts[0]);
teacherPr = CreateProgram(parts[1]);
myGui.userProgram(teacherPr);
```

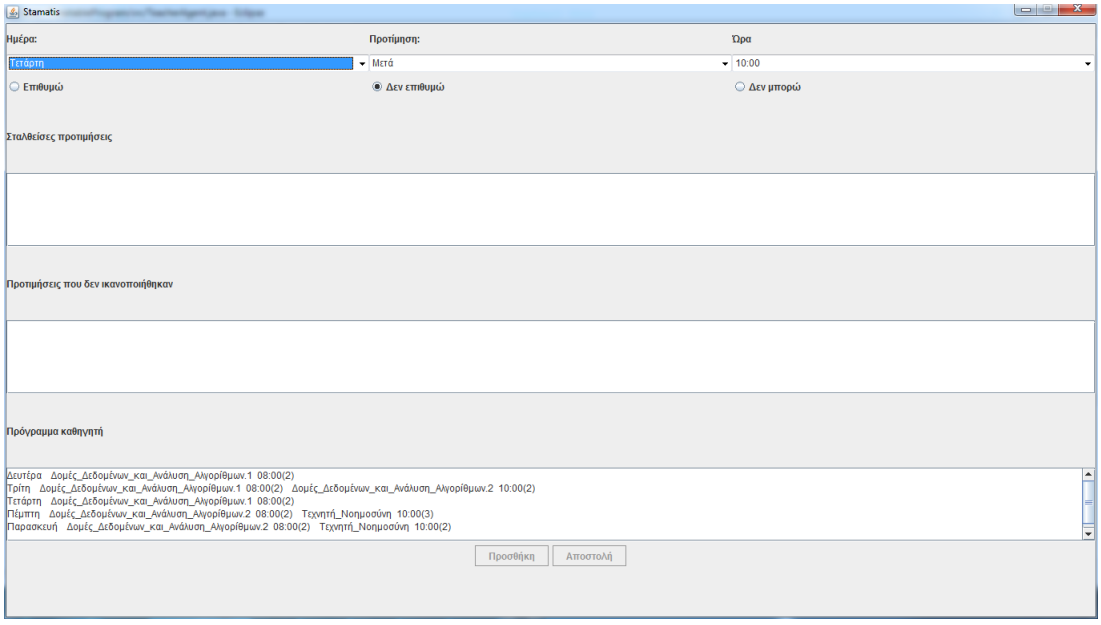
Ο καθηγητής κοιτάει το πρόγραμμα, όπως αυτό έχει δημιουργηθεί μέχρι στιγμής και τις προτιμήσεις του οι οποίες απορρίφθηκαν και τοποθετεί εκ νέου τις καινούργιες του προτιμήσεις.



Εικόνα 28. Τοποθέτηση νέων προτιμήσεων από τον TeacherAgent

Συλλέγονται οι νέες προτιμήσεις μέσω της μεθόδου updatePreferences και αποστέλλονται με παρόμοιο τρόπο, όπως και στο προηγούμενο βήμα στον Scheduler.

Εάν δεν υπάρχουν συγκρούσεις και το πρόγραμμα του καθηγητή έχει ολοκληρωθεί, τότε μέσω της behaviour program_Finished, λαμβάνεται το μήνυμα ACCEPT_PROPOSAL, ενημερώνεται το γραφικό του καθηγητή με το πρόγραμμα του και αποστέλλεται στον Scheduler ένα μήνυμα INFORM, προκειμένου να τον ενημερώσει ότι έλαβε το πρόγραμμα και έχει ενημερωθεί ότι η διαδικασία για αυτόν ολοκληρώθηκε.



Εικόνα 29. Ολοκλήρωση προγράμματος TeacherAgent

Αυτό υλοποιείται με τις ακόλουθες γραμμές κώδικα:

```
private class program_finished extends CyclicBehaviour {
        public void action() {
            MessageTemplate mt = MessageTemplate.
            MatchPerformative(ACLMessage.ACCEPT_PROPOSAL);
             ACLMessage msg = myAgent.receive(mt);
            if (msg != null) {
            // ACCEPT_PROPOSAL Message received. Process it
                String returnMsg = msg.getContent();
                proposeGui.dispose();
                myGui.disableButtons();
                myGui.clearPrefs();
                myGui.clearConflicts();
                myGui.clearProgram();
                teacherPr = CreateProgram(returnMsg);
                myGui.userProgram(teacherPr);
                ACLMessage reply = msg.createReply();
                reply.setPerformative(ACLMessage.INFORM);
                myAgent.send(reply);
            }else {
                block();
            }
        }
} // End of inner class
```

Ο Scheduler θα λάβει το μήνυμα INFORM από του καθηγητές για τους οποίους έχει ολοκληρωθεί το πρόγραμμα και μέσω της behaviour finishProgram, η οποία είναι υπεύθυνη να παραλάβει όλα τα μηνύματα INFORM είτε από τους TeacherAgents, είτε από τους StudentAgents προκειμένου να ελέγξει πόσοι από αυτούς έχουν ολοκληρώσει τη διαδικασία του προγράμματος επιτυχώς.

Μόλις το πρόγραμμα έχει ολοκληρωθεί σωστά και χωρίς συγκρούσεις για όλους τους TeacherAgents της εφαρμογής, τότε ο Scheduler, μέσω αυτής της behaviour, αποστέλλει το τελικό πρόγραμμα με τα μαθήματα του εξαμήνου τους, σε όλους StudentAgents της εφαρμογής, μαζί με ένα μήνυμα ACCEPT_PROPOSAL, προκειμένου να τους ενημερώσει ότι ολοκληρώθηκε η διαδικασία. Στη συνέχεια ο Scheduler τερματίζει τη λειτουργία του. Η διαδικασία που ακολουθείται στην finishProgram, παρουσιάζεται στις ακόλουθες γραμμές κώδικα:

```
private class finishProgram extends CyclicBehaviour {
        public void action() {
            MessageTemplate mt=
            MessageTemplate.MatchPerformative(ACLMessage.INFORM);
            ACLMessage msg = myAgent.receive(mt);
            if (msg != null) {
                if (msg.getPerformative() == ACLMessage.INFORM) {
                        finishedAgents++;
                        System.out.println("finishedAgents="+finishedAgents);
                }
```

```
            }else {
                block();
            }
          // Program finished with all Teacher Agents
          if (finishedAgents == teacherAgents.length) {
                semesterProgram = program.getSemesterProgram();
                String semesterP = "";
                for ( String reciever : semesterProgram.keySet() ) {
                    ACLMessage accept_Proposal = new ACLMessage
                    (ACLMessage.ACCEPT_PROPOSAL);
                        for (i = 0; i < studentAgents.length; i++) {
                                String aux = studentAgents[i].toString();
                                if (aux.contains(reciever)) {
                                accept_Proposal.addReceiver(studentAgents[i]);
                                semesterP = reciever;
                            }
                    }
                    String tProgram = semesterProgram.get(semesterP);
                    String sendToStudent = tProgram;
                    accept_Proposal.setContent(sendToStudent);
                    accept_Proposal.setConversationId("finishing-program");
              accept_Proposal.setReplyWith("accept_Proposal"+ System.currentTimeMillis());
                    myAgent.send(accept_Proposal);
                    // Prepare the template to get proposals
                            mt = MessageTemplate.and(MessageTemplate.MatchConversationId
                     ("finishing-program"),MessageTemplate.MatchInReplyTo
                     (accept_Proposal.getReplyWith()));
             }
             // Purchase successful. We can terminate
                    System.out.println(" Το πρόγραμμα ολοκληρώθηκε με επιτυχία ");
                    myAgent.doDelete();
          }
    }
}
```

Εάν ο Scheduler λάβει μέσω της behaviour Negotiation τις προτιμήσεις από τους καθηγητές που παρουσιάστηκε η σύγκρουση, τότε αφού και πάλι τις κωδικοποιήσει κατάλληλα, καλεί τη μέθοδο updateProgram που βρίσκεται στην κλάση program.

Ο αλγόριθμος που χρησιμοποιείται στη μέθοδο updateProgram είναι ο εξής:

1. Δημιουργείται ο πίνακας lessons, ο οποίος περιέχει τα μαθήματα των καθηγητών και ο οποίος προκύπτει από το αρχείο lessons.txt.
2. Οι καθηγητές, των οποίων οι προτιμήσεις παρουσιάζουν συγκρούσεις, λαμβάνονται με τη σειρά με την οποία έχουν ταξινομηθεί κατά το προηγούμενο στάδιο, σύμφωνα με το HashMap sorted_priorities.
3. Για κάθε έναν από αυτούς καλείται η μέθοδος updateLessons, η οποία σύμφωνα με το πρόγραμμα του καθηγητή το οποίο βρίσκεται στο teacherProgram και τις προτιμήσεις του καθηγητή που υπήρχαν στην 1η φάση της διαπραγμάτευσης, ανανεώνουν τον

κατάλογο με τα μαθήματα του καθηγητή. Οι τρεις κατάλογοι για κάθε καθηγητή είναι στα ArrayLists με όνομα teacherLessons, theory και lab, οι οποίοι περιέχουν αντίστοιχα το όνομα του μαθήματος, τις ώρες θεωρίας και τις ώρες εργαστηρίου που το αποτελούν.

4. Στη μέθοδο updateLessons, ελέγχεται αρχικά εάν κάποιο μάθημα, το οποίο έχει τοποθετηθεί στο πρόγραμμα του καθηγητή, παραβιάζει κάποια από τις προτιμήσεις του, όπως αυτές είχαν τοποθετηθεί κατά την 1η φάση της διαπραγμάτευσης. Εάν το παραβιάζει, τότε το μάθημα αυτό αφαιρείται από το πρόγραμμα που βρίσκεται στον πίνακα program. Εάν δεν παραβιάζει καμία από τις προτιμήσεις του καθηγητή, τότε ενημερώνονται κατάλληλα τα ArrayLists theory και lab, μειώνοντας κατάλληλα τις αντίστοιχες ώρες.
5. Γίνεται προσπάθεια να ικανοποιήσουμε όλες τις προτιμήσεις που μας έχει αποστείλει, ελέγχοντας έναν πίνακα preference που έχουμε δημιουργήσει.
6. Αφού ολοκληρωθεί η διαδικασία με την ικανοποίηση των προτιμήσεων, μέσω της μεθόδου putRestLessons, γίνεται προσπάθεια να τοποθετηθούν τα εναπομείναντα μαθήματα.
7. Αν δεν τοποθετηθούν όλα τα μαθήματα στο πρόγραμμα, τότε καλείται η μέθοδος putLessonsOnStudentPrefs μέσω της οποία τοποθετούνται τα μαθήματα στις προτιμήσεις των σπουδαστών.
8. Στη συνέχεια καλείται η μέθοδος putDenEpithumwLessons, μέσω της οποίας τοποθετούνται τα εναπομείναντα μαθήματα εκεί που ο καθηγητής δήλωσε ότι δεν επιθυμεί να τοποθετηθεί μάθημα.
9. Αν και τώρα δεν τοποθετηθούν μαθήματα, τότε μέσω της CheckPreferencedenMporw τοποθετούνται στο conflicts οι προτιμήσεις του καθηγητή που περιέχουν τον περιορισμό «Δεν μπορώ» και οι όποιες μπλοκάρουν το πρόγραμμα.
10. Τοποθετείται στο teacherProgram το πρόγραμμα του καθηγητή.
11. Εάν δεν έχουν τοποθετηθεί όλα τα μαθήματα του καθηγητή στο πρόγραμμα, τότε χρησιμοποιείται ένα πρωτόκολλό πειθούς (persuasion protocol). Αυτό πραγματοποιείται καλώντας τη μέθοδο proposeProgram, η οποία δημιουργεί ένα HashMap με όνομα programProposals, το οποίο περιέχει προτάσεις για τα εναπομείναντα μαθήματα, ή για τα μαθήματα που συνεχίζουν να εμφανίζουν συγκρούσεις με τις προτιμήσεις του καθηγητή.
12. Στη μέθοδο proposeProgram αρχικά δημιουργούνται προτάσεις για τα μαθήματα που δεν έχουν τοποθετηθεί ακόμα στο πρόγραμμα και στη συνέχεια για αυτά που τοποθετήθηκαν, αλλά παρουσιάζουν συγκρούσεις. Για κάθε μάθημα η πρόταση περιλαμβάνει το όνομα του μαθήματος και προτεινόμενη ημέρα και ώρα.

13. Η διαδικασία συνεχίζεται με τον ίδιο τρόπο και για τους υπόλοιπους καθηγητές και τα βήματα 3 εώς 13 επαναλαμβάνονται.

Εφόσον έχει ολοκληρωθεί η πρώτη φάση της δημιουργίας του προγράμματος, στο HashMap conflicts υπάρχουν οι συγκρούσεις που προέκυψαν από τις προτιμήσεις των καθηγητών, στο teacherProgram το πρόγραμμα του κάθε καθηγητή και στο programProposals οι προτάσεις για τα μαθήματα που παρουσιάζουν συγκρούσεις. Επίσης στο ArrayList lessonsAfterProposals, τοποθετούνται τα μαθήματα, μαζί με τις ημέρες και ώρες, τα οποία ενώ τοποθετήθηκαν στο πρόγραμμα, ζητείται από τους καθηγητές να αποστείλουν προτάσεις για αυτά, γιατί εξακολουθούν να παρουσιάζουν συγκρούσεις.

Στη συνέχεια ο Scheduler ελέγχει και πάλι για ποιους καθηγητές υπάρχουν συγκρούσεις, ελέγχοντας το HashMap conflicts και σε αυτούς αποστέλλει μήνυμα CFP, το οποίο περιέχει τις συγκρούσεις που παρατηρήθηκαν, το προσωρινό του πρόγραμμα, καθώς επίσης και τις προτάσεις για τα μαθήματα που εμφανίστηκαν συγκρούσεις και τα οποία βρίσκονται στο HashMap programProposals. Το μήνυμα αυτό έχει id = "conflicts-negotiation". Ο κώδικάς με τον οποίο υλοποιείται η παραπάνω διαδικασία είναι:

```
for (i = 0; i < teacherAgents.length; i++) {
        ACLMessage proposeback = new ACLMessage(ACLMessage.CFP);
        String aux = teacherAgents[i].toString();
        String teacher = teacherAddresses.get(aux);
        boolean check = false;
        for ( String reciever : conflicts.keySet() ) {
                String reciever1 = reciever.split("#")[0];
                if (aux.contains(reciever1)) {
                        check = true;
                        conflictsBack = conflictsBack + conflicts.get(reciever) + "#";
                }
        }
        if (!check) {
                conflictsBack = "ok";
        }else{
                proposeback.addReceiver(teacherAgents[i]);
        }
        String proposeBack = "";
        for ( String reciever : programProposals.keySet() ) {
                if (aux.contains(reciever)) {
                        check = true;
                        proposeBack = proposeBack + programProposals.get(reciever);
                        proposalsSend++;
                }
        }
        if (!check) {
                proposeBack = proposeBack + "none";
        }
        teacher = "";
        for ( String key : teacherAddresses.keySet()) {
                if (aux.contains(key)) {
```

```
                        teacher = teacherAddresses.get(key);
                }
        }
        tProgram = teacherProgram.get(teacher);
        if(tProgram.equals("")){
                tProgram = "#";
        }
        String sendToTeacher = conflictsBack + "%" + tProgram + "%" + proposeBack;
        proposeback.setContent(sendToTeacher);
        proposeback.setConversationId("propose-negotiation");
        proposeback.setReplyWith("negotiation"+System.currentTimeMillis());
        myAgent.send(proposeback);
        check = true;
        // Prepare the template to get the program negotiation
        mt = MessageTemplate.and(MessageTemplate.MatchConversationId("conflicts-
        negotiation"),MessageTemplate.MatchInReplyTo (proposeback.getReplyWith()));
```

Στους καθηγητές που δεν εμφανίζονται συγκρούσεις αποστέλλεται μήνυμα ACCEPT_PROPOSAL, προκειμένου αυτοί να ενημερωθούν ότι έγιναν δεκτές οι προτιμήσεις τους και το πρόγραμμα τους ολοκληρώθηκε με επιτυχία.

Οι TeacherAgents λαμβάνουν το ανάλογο μήνυμα από τον Scheduler (είτε CFP, είτε ACCEPT_PROPOSAL) και αποφασίζουν τι πρέπει να κάνουν. Εάν λάβουν CFP, τότε ενημερώνουν μέσω κατάλληλων μεθόδων το γραφικό του χρήστη σχετικά με τις αλλαγές που υπάρχουν στο πρόγραμμα του, καθώς επίσης και με τις προτιμήσεις του οι οποίες δεν ικανοποιήθηκαν. Επίσης αυτή τη φορά δημιουργείται ακόμη ένα γραφικό, το οποίο τον ενημερώνει για τις προτάσεις του Scheduler για τα μαθήματα που δεν έχουν τοποθετηθεί στο πρόγραμμα, ή για αυτά στα οποία παρουσιάζονται συγκρούσεις. Αυτό γίνεται με τον ακόλουθο κώδικα:

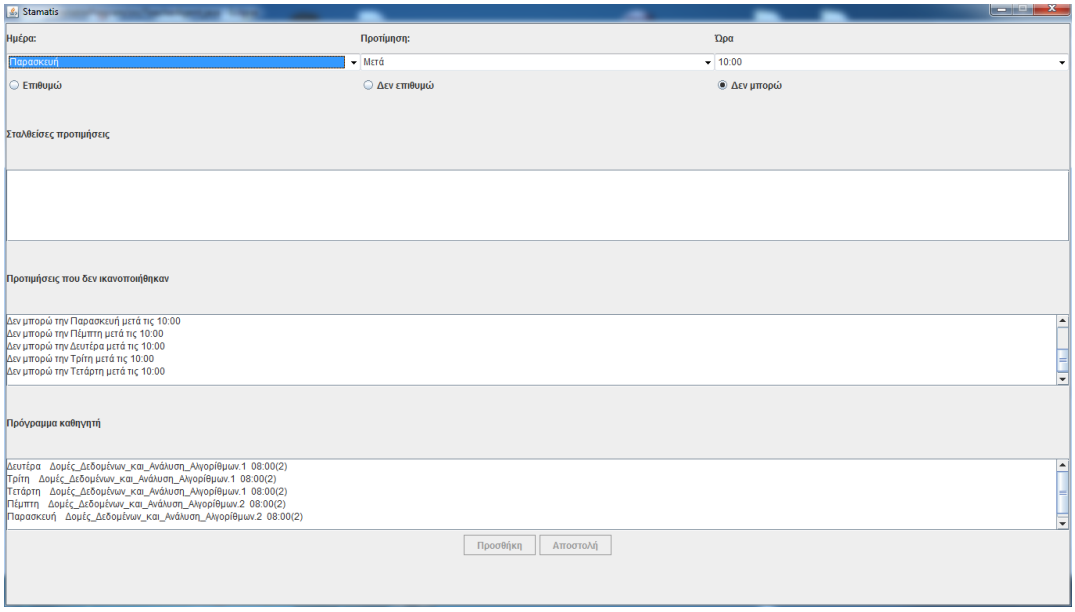
```
myGui.disableButtons();
myGui.clearPrefs();
myGui.clearConflicts();
conflicts = CreateMessage(parts[0]);
myGui.notifyUser(conflicts);
myGui.clearProgram();
teacherPr = CreateProgram(parts[1]);
myGui.userProgram(teacherPr);
if (!parts[2].equals("none")) {
        proposeGui.showGui();
        lessonsToPropose(parts[2]);
}
```

Στον πίνακα parts υπάρχουν αποθηκεμένα τα κομμάτια του μηνύματος που έχει φτάσει από τον Scheduler. Στο parts[0] υπάρχουν τα conflicts, στο parts[1] το πρόγραμμα του καθηγητή και στο parts[2] οι προτάσεις του Scheduler.

Στη μέθοδο lessonsToPropose αποκωδικοποιούνται οι προτάσεις του Scheduler προκειμένου να μπορέσουν να εμφανιστούν στο γραφικό του χρήστη. Επιλέχθηκε να εμφανίζονται μόνο οι τέσσερις πρώτες προτάσεις, προκειμένου ο χρήστης να μπορεί να επιλέξει γρήγορα μια από αυτές.

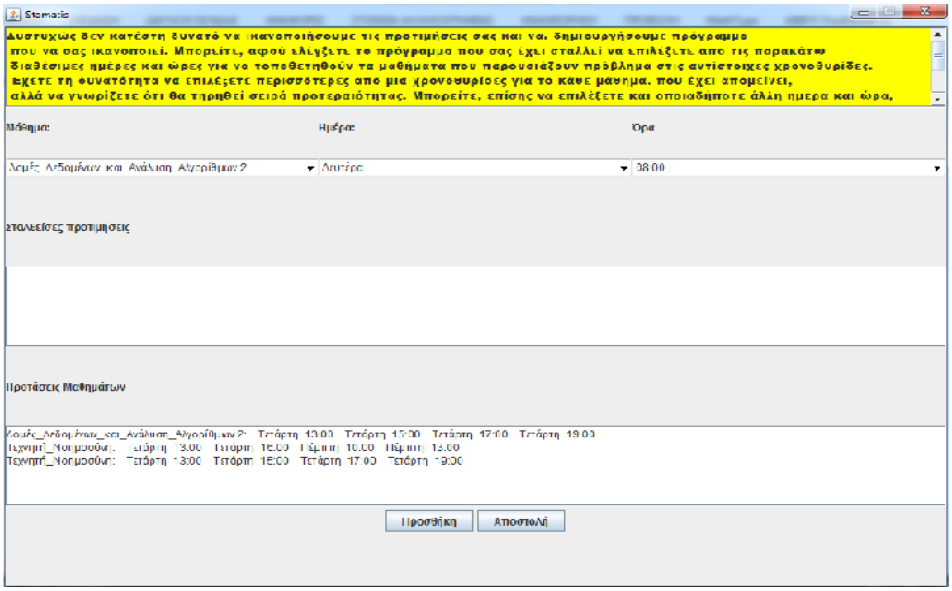
Στο χρήστη λοιπόν πλέον εμφανίζονται δυο γραφικά: Το TeacherGui, όπως και προηγουμένως, προκειμένου να δει το πρόγραμμα του και τις προτιμήσεις του που δεν ικανοποιήθηκαν και το proposalsGui, προκειμένου να στείλει τις προτιμήσεις του για τα εναπομείναντα μαθήματα. Στο TeacherGui τα κουμπιά «Προσθήκη» και «Αποστολή» είναι πλέον απενεργοποιημένα, έτσι ώστε να μην μπορεί να στείλει και πάλι προτιμήσεις.

Το TeacherGui, όπως εμφανίζεται στην οθόνη του καθηγητή είναι:



Εικόνα 30. TeacherGui στο 2ο στάδιο διαπραγμάτευησς

Το proposalsGui όπως εμφανίζεται στην οθόνη του καθηγητή είναι:



Εικόνα 31. proposalsGui. Οι προτάσεις του καθηγητή

Στο proposalsGui ο καθηγητής κάνει τις επιλογές του για τα εναπομείναντα μαθήματα πατώντας «Προσθήκη» για την κάθε επιλογή και στο τέλος πατάει «Αποστολή». Μπορεί να επιλέξει μια από τις διαθέσιμες επιλογές για κάθε μάθημα. Έχει τη δυνατότητα όμως να επιλέξει και οποιαδήποτε άλλη ημέρα και ώρα εκτός από τις προτεινόμενες, αλλά τότε δεν μπορεί να γνωρίζει αν θα ικανοποιηθεί ή αν θα απορριφθούν οι επιλογές του και θα τοποθετηθεί το μάθημα σε οποιαδήποτε άλλη κενή ημέρα και ώρα. Έχει τη δυνατότητα επίσης να επιλέξει και περισσότερες από μια επιλογές για κάθε μάθημα, τότε όμως του γνωστοποιείται ότι θα τηρηθεί σειρά προτεραιότητας. Δηλαδή αν ικανοποιηθεί κάποια από αυτές, τότε οι επόμενες επιλογές του για το ίδιο μάθημα θα αγνοηθούν. Τέλος αν δεν στείλει καμία επιλογή και πατήσει απλά «Αποστολή», τότε τα εναπομείναντα μαθήματα θα τοποθετηθούν σε τυχαίες ημέρες και ώρες προκειμένου να ολοκληρωθεί η διαδικασία.

Ο Scheduler λαμβάνει τις προτάσεις από τους TeacherAgents στους οποίους ζήτησε να του στείλουν τις προτάσεις τους, τις κωδικοποιεί κατάλληλα με τη μέθοδο proposals, η οποία βρίσκεται μέσα στην κλάση preferencesHandler. Μόλις λάβει όλες τις προτάσεις, τότε καλεί τη μέθοδο satisfyProposals, προκειμένου να τοποθετήσει στο πρόγραμμα τα υπόλοιπα μαθήματα και να ολοκληρώσει τη διαδικασία. Η παραπάνω διαδικασία υλοποιείται με τις ακόλουθες γραμμές κώδικα:

```
 // Receive the new program preferences
ACLMessage reply = myAgent.receive(mt);
if (reply != null) {
// Reply received
          if (reply.getPerformative() == ACLMessage.PROPOSE) {
               String preference = reply.getContent();
               String delimiter = "#";
                    String[] pref;
                    pref = preference.split(delimiter);
                    if (!preference.isEmpty()) {
```

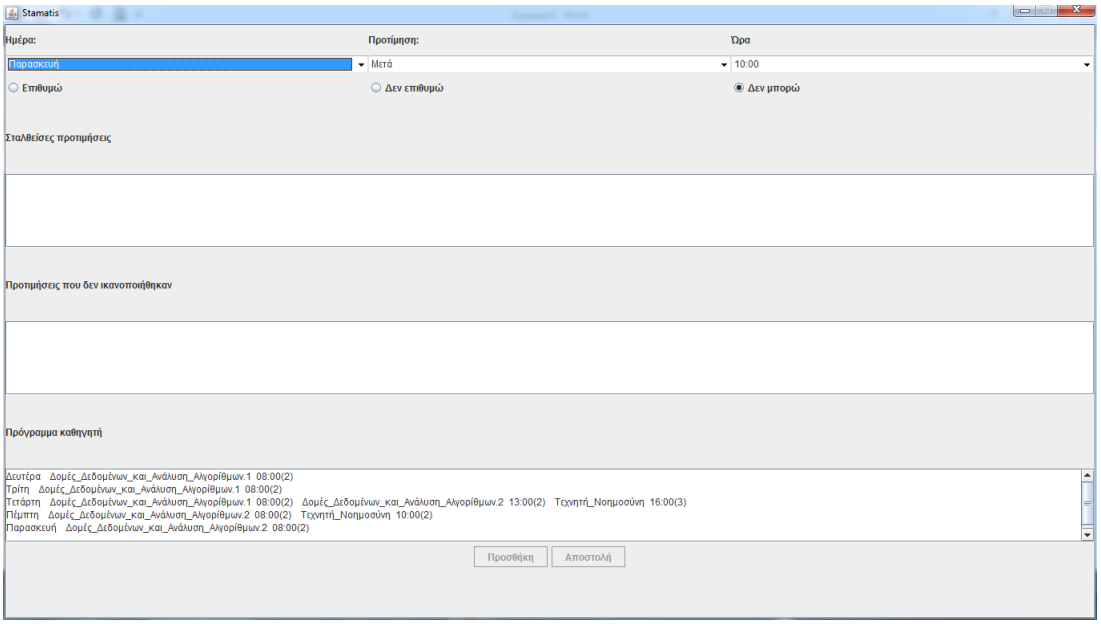
```
                        if (pref.length > 1) {
                                preferencesHandler.proposals(pref);
                        }
                        proposeBackNum++;
                }
        }
}
if (proposeBackNum == proposalsSend){
        flag4 = true;
        try {
      program.satisfyProposals(preferencesHandler,preferencesHandler);
        } catch (IOException e) {
          // TODO Auto-generated catch block
          e.printStackTrace();
        }
        teacherProgram = program.teacherProgram();
}
```

Μέσα στην μέθοδο satisfyProposals ο αλγόριθμος που ακολουθείται, προκειμένου να τοποθετηθούν στο πρόγραμμα οι προτάσεις των καθηγητών περιγράφεται από τα ακόλουθα βήματα:

1. Λαμβάνεται με τη σειρά όλοι οι καθηγητές οι οποίοι έχουν στείλει προτάσεις, πάντα σύμφωνα με την ταξινόμηση που έχει γίνει σε αυτούς σε προηγούμενο στάδιο.
2. Για το κάθε μάθημα που έχει αποστείλει πρόταση, αναζητείται στο lessonsAfterProposals να βρεθεί η ημέρα και ώρα που έχει τοποθετηθεί το συγκεκριμένο μάθημα στο προηγούμενο στάδιο, προκειμένου αυτό να αφαιρεθεί και να τοποθετηθεί η νέα πρόταση του καθηγητή.
3. Αν βρεθεί (δηλαδή το μάθημα τοποθετήθηκε στο πρόγραμμα, αλλά εξακολουθεί να παρουσιάζει σύγκρουση), τότε αυτό αφαιρείται από το πρόγραμμα και γίνεται προσπάθεια να τοποθετηθεί στην προτεινόμενη ημέρα και ώρα.
4. Αν αυτή είναι κενή, τότε τοποθετείται στο πρόγραμμα και διαγράφεται από το lessonsAfterProposals. Αν είναι κατειλημμένη, τότε προχωράει στην επόμενη πρόταση.
5. Αν η επόμενη πρόταση αφορά το ίδιο μάθημα, τότε αγνοείται, ενώ αν αφορά διαφορετικό μάθημα, τότε επαναλαμβάνονται τα βήματα 2-4.
6. Καλείται η μέθοδος finishProgram, η οποία ελέγχει αν έχουν τοποθετηθεί όλα τα μαθήματα του καθηγητή στο πρόγραμμα. Αν υπάρχουν ακόμη μαθήματα που δεν τοποθετήθηκαν, τότε αυτά τοποθετούνται στα πρώτα κενά διαστήματα που θα βρεθούν στο πρόγραμμα.
7. Μέσω της μεθόδου getProgram τοποθετείται στο HashMap teacherProgram το πρόγραμμα του καθηγητή.
8. Τα βήματα 2–5 επαναλαμβάνονται για όλους τους υπόλοιπους καθηγητές που έχουν αποστείλει προτάσεις.

Μόλις ολοκληρωθεί η τοποθέτηση των προτάσεων των καθηγητών στο πρόγραμμα, ο Scheduler λαμβάνει μέσω της μεθόδου returnProg, η οποία βρίσκεται στην κλάση program το πρόγραμμα του καθηγητή και το αποστέλλει στον καθηγητή, συνοδευόμενο με ένα μήνυμα ACCEPT_PROPOSAL, προκειμένου να τον ενημερώσει για την ολοκλήρωση του προγράμματος.

Ο TeacherAgent λαμβάνει το μήνυμα ACCEPT_PROPOSAL, μαζί με το πρόγραμμα που του έχει αποστείλει ο Scheduler και μέσω της μεθόδου program_finished ενημερώνει το γραφικό του καθηγητή με το οριστικό του πρόγραμμα μαθημάτων. Το γραφικό του καθηγητή παρουσιάζεται στην ακόλουθη εικόνα:



Εικόνα 32. Ολοκλήρωση προγράμματος καθηγητή.

Ο TeacherAgent αποστέλλει ένα μήνυμα INFORM στον Scheduler για να τον ενημερώσει ότι έχει λάβει το οριστικό του πρόγραμμα.

Ο Scheduler μέσω της behaviour finishProgram λαμβάνει το μήνυμα INFORM και μετράει πόσοι TeacherAgents έχουν ολοκληρώσει τη διαδικασία. Όταν ο αριθμός αυτών που έχουν αποστείλει μήνυμα INFORM, είναι ίσος με τον αριθμό των ενεργών πρακτόρων καθηγητών στην πλατφόρμα, τότε αποστέλλει σε όλους τους StudentAgents ένα μήνυμα ACCEPT_PROPOSAL για να τους ενημερώσει για την ολοκλήρωση του προγράμματος, συνοδευόμενο από το πρόγραμμα του εξαμήνου τους και ολοκληρώνει τη λειτουργία του εμφανίζοντας κατάλληλο μήνυμα. Ο κώδικας που υλοποιεί την finishProgram περιγράφεται παρακάτω:

```
private class finishProgram extends CyclicBehaviour {
        public void action() {
                MessageTemplate mt = MessageTemplate.MatchPerformative
                (ACLMessage.INFORM);
               ACLMessage msg = myAgent.receive(mt);
```

```
                    if (msg != null) {
                            if (msg.getPerformative() == ACLMessage.INFORM) {
                                    finishedAgents++;                                    }
                            }else {
                                    block();
                            }
                            // Program finished with all Teacher Agents
                    if (finishedAgents == teacherAgents.length) {
                      double evaluation = program.evaluate(preferencesHandler);
                      System.out.println("Το πρόγραμμα αξιολογήθηκε με άριστα το
            0 με:"+evaluation);
                      semesterProgram = program.getSemesterProgram();
                      String semesterP = "";
                      for ( String reciever : semesterProgram.keySet() ) {
                          ACLMessage accept_Proposal = new
              ACLMessage(ACLMessage.ACCEPT_PROPOSAL);
                              for (i = 0; i < studentAgents.length; i++) {
                                String aux = studentAgents[i].toString();
                                if (aux.contains(reciever)) {
                                        accept_Proposal.addReceiver(studentAgents[i]);
                                        semesterP = reciever;
                                        }
                              }
                              String tProgram = semesterProgram.get(semesterP);
                              String sendToStudent = tProgram;
                              accept_Proposal.setContent(sendToStudent);
                            accept_Proposal.setConversationId("finishing-program");
                            accept_Proposal.setReplyWith("accept_Proposal"
                            +System.currentTimeMillis()); // Unique value
                                            myAgent.send(accept_Proposal);
                            // Prepare the template to get proposals
                            mt = MessageTemplate.and(MessageTemplate.
                            MatchConversationId("finishing-program"),
          MessageTemplate.MatchInReplyTo(accept_Proposal.
                            getReplyWith()));
                    }
                    // Purchase successful. We can terminate
                    System.out.println(" Το πρόγραμμα ολοκληρώθηκε με επιτυχία ");
                                    myAgent.doDelete();
              }
    }
}  // End of inner class finishProgram
```

Μόλις οι StudentAgents λάβουν ένα μήνυμα ACCEPT_PROPOSAL από τον Scheduler, τότε εμφανίζουν στο γραφικό του φοιτητή το πρόγραμμα του εξαμήνου του και αποστέλλουν στον Scheduler ένα μήνυμα INFORM ολοκληρώνοντας τη διαδικασία.

Τα UML διαγράμματα, που αναπαριστούν τις κλάσεις που έχουν αναπτυχθεί, παρατίθενται στο Παράρτημα Β.

7. *Αξιολόγηση*

Προκειμένου να αξιολογήσουμε την εφαρμογή μας, επιλέξαμε να χρησιμοποιήσουμε την τιμή μιας συνάρτησης αξιολόγησης, η οποία μετράει το βαθμό ικανοποίησης των περιορισμών των χρηστών. Επίσης χρησιμοποιήθηκε η τεχνική του Τεστ Ευχρηστίας (Usability Testing) και συγκεκριμένα η μέθοδος «Think Aloud Protocol». Δόθηκε σε ένα πλήθος συμμετεχόντων ένας αριθμός σεναρίων, προκειμένου αυτοί να χρησιμοποιήσουν την εφαρμογή μας. Καταγράψαμε τη δραστηριότητα τους και παρακολουθήσαμε τις παρατηρήσεις που θέλησαν αυτοί να μας κάνουν. Στη συνέχεια με τη χρήση ερωτηματολόγιου καταγράφηκε ο βαθμός ικανοποίησης τους

7.1 Παράμετροι αξιολόγησης

Θέλοντας να μετρήσουμε το βαθμό ικανοποίησης των περιορισμών που τίθενται από τη μεριά των καθηγητών, λαμβάνουμε υπόψιν μας την τιμή μιας συνάρτησης αξιολόγησης η οποία εκτελείται αυτόματα με την ολοκλήρωση του προγράμματος. Η συνάρτηση αυτή λαμβάνει υπόψιν της, τον αριθμό των περιορισμών που τέθηκαν, καθώς επίσης και τον αριθμό και το είδος των περιορισμών που δεν κατέστη δυνατό να ικανοποιηθούν μετά την ολοκλήρωση της διαδικασίας. Όσο πιο κοντά στο 0 είναι η τιμή της συνάρτησης, τόσο καλύτερο είναι το πρόγραμμα που δημιουργείται.

Με τη χρήση ερωτηματολόγιων μετράμε το αισθητικό κομμάτι της εφαρμογής (εντυπώσεις του χρήστη από τη διεπαφή της εφαρμογής), η ευχρηστία της εφαρμογής (πόσο εύκολα κατάφερε να την χρησιμοποιήσει, τυχόν απορίες που του δημιουργήθηκαν), ικανοποίηση από το τελικό αποτέλεσμα (εάν το αποτέλεσμα ανταποκρίνονταν στις προσδοκίες του).

7.2 Συνάρτηση αξιολόγησης

Μετά την ολοκλήρωση της διαδικασίας διαπραγμάτευσης ανάμεσα στον προγραμματιστή και στα υπόλοιπα συμμετέχοντα μέρη (καθηγητές, σπουδαστές) και την τελική δημιουργία του προγράμματος, ο ProgramAgent καλεί μια συνάρτηση αξιολόγησης, η οποία υλοποιείται με τη μέθοδο evaluate, η οποία βρίσκεται μέσα στην κλάση program. Η συνάρτηση λαμβάνει υπόψιν της τις προτιμήσεις των καθηγητών και των αριθμό των συγκρούσεων ανάμεσα στο

τελικό πρόγραμμα και τις προτιμήσεις αυτές και οι οποίες δημιουργήθηκαν μετά την ολοκλήρωση της διαδικασίας.

Η μέθοδος αρχικά μετράει τον αριθμό των προτιμήσεων που έχει λάβει ο ProgramAgent. Στη συνέχεια λαμβάνονται οι συγκρούσεις που προέκυψαν και οι οποίες βρίσκονται στο HashMap conflicts. Προκειμένου να υπολογιστεί με δίκαιο τρόπο η τιμή της συνάρτησης, έχουν δοθεί κάποιες τιμές βάρους στο είδος των συγκρούσεων. Έτσι αποφασίστηκε, αν αυτές είναι της κατηγορίας «Δεν μπορώ», το οποίο αποτελεί και Ισχυρό Περιορισμό, τότε το βάρος τους είναι 1,5. Σε διαφορετική περίπτωση («Επιθυμώ»/ «Δεν Επιθυμώ»), τα οποία αποτελούν και Χαλαρούς Περιορισμούς, τότε το βάρος τους είναι 0,5. Η τιμή της συνάρτησης υπολογίζεται από τον ακόλουθο τύπο:

$$\text{Τιμή Συνάρτησης} = \frac{\text{Σύγκρουση1*βάρος} + \text{Σύγκρουση2*βάρος} + \text{.....} + \text{ΣύγκρουσηΝ*βάρος}}{\text{Πλήθος Προτιμήσεων}}$$

Η τιμή της συνάρτησης θα είναι προφανώς από 0 μέχρι 1. Εάν δεν υπάρχουν καθόλου συγκρούσεις και έχουν ικανοποιηθεί όλες οι προτιμήσεις των καθηγητών, τότε η τιμή της θα είναι 0. Όσο πιο κοντά στο 0 βρίσκεται, τόσο καλύτερο είναι το πρόγραμμα που δημιουργήθηκε. Ο κώδικας με τον οποίο υλοποιείται η παραπάνω αξιολόγηση είναι:

```
public double evaluate(PreferencesHandler preferences){
        double count = 0;
        int prefNum = preferences.size();
        String delimiter = "#";
        for (String key : conflicts.keySet() ) {
                String conf[] = conflicts.get(key).split(delimiter);
                 if (conf[5].equals("true")) {
                        count = count + 1.5;
                }else if(conf[3].equals("true") || conf[4].equals("true")){
                        count = count + 0.5;
                }
        }
        double evaluate = count / prefNum;
        return evaluate;
}
```

7.3 Τεστ Ευχρηστίας (Usability Testing)

Οι *δοκιμές ευχρηστίας* είναι μια τεχνική που χρησιμοποιείται με επίκεντρο την αλληλεπίδραση του χρήστη, ο οποίος αξιολογεί ένα προϊόν δοκιμάζοντας το. Αποτελεί μια αναντικατάστατη τεχνική χρήσης, αφού μας δίνει άμεση συμβολή για το πώς πραγματικά χρησιμοποιούν οι χρήστες το σύστημα μας. Πρόκειται στην ουσία για μια τεχνική η οποία χρησιμοποιείται προκειμένου να αξιολογηθεί ένα προϊόν. Αυτό μπορεί να θεωρηθεί ως μια αναντικατάστατη πρακτική χρησιμότητας, δεδομένου ότι δίνει άμεση ενημέρωση για το πώς πραγματικά οι χρήστες χρησιμοποιούν το σύστημα. Επικεντρώνεται στη μέτρηση της

ικανότητας χρησιμοποίησης ενός προϊόντος και την ικανοποίηση των στόχων που έχουν τεθεί (Usability testing, 2015).

Τα τεστ ευχρηστίας μετράνε ουσιαστικά τη χρηστικότητα του προϊόντος (της ιστοσελίδας ή της εφαρμογής, στη δική μας περίπτωση). Αποτελούν ουσιαστικά μια τεχνική "Μαύρου-κουτιού". Ο στόχος είναι να παρατηρήσουμε τους ανθρώπους που χρησιμοποιούν την εφαρμογή, ώστε να ανακαλύψουμε λάθη και περιοχές που χρειάζονται βελτίωση.

Η διαδικασία του τεστ ευχρηστίας ολοκληρώνεται σε 3 φάσεις:

1. Οργάνωση πειραμάτων.

   Στη φάση αυτή, αναγνωρίζουμε τις μεθόδους και τα εργαλεία που θα χρησιμοποιηθούν. Επιλέξαμε να χρησιμοποιήσουμε τη μέθοδο **«Think Aloud Protocol».**
2. Υλοποίηση Τεστ.
3. Αξιολόγηση αποτελεσμάτων.

7.3.1 Οργάνωση πειραμάτων

Προκειμένου να προχωρήσουμε στο Τεστ Ευχρηστίας επιλέχθηκαν 7 χρήστες. Οι 2 από αυτούς είναι μεταπτυχιακοί φοιτητές του τμήματος και επόμενοι 5 είναι καθηγητές σε Σχολείο της Δευτεροβάθμιας Εκπαίδευσης. Δύο από τους καθηγητές ταυτόχρονα διδάσκουν και ως Επιστήμονες/Διδάσκοντες στο Μαθηματικό τμήμα του Αριστοτέλειο Πανεπιστήμιου Θεσσαλονίκης, σύμφωνα με το Π.Δ. 407/80. Η συγκεκριμένη μέθοδος επιλέχθηκε επειδή είναι αποτελεσματική από τα πρώτα στάδια εφαρμογής της.). Οι τεχνικές που επιλέχθηκαν είναι «Σενάρια», «Ερωτηματολογίου» και “Think aloud protocols”. Οι «Παρατηρητές του Τεστ» εκπαιδεύονται στον τύπο των ερωτήσεων που θα κάνουν στους χρήστες κατά τη διάρκεια του τεστ. Δημιουργούνται Ερωτηματολόγια Αξιολόγησης, προκειμένου να μετρηθεί η ικανοποίηση των χρηστών από το σύστημα μας..

Το «*Σκέψου-δυνατά»* πρωτόκολλο (think-aloud protocol) είναι μια μέθοδος που χρησιμοποιείται για τη συλλογή δεδομένων σε δοκιμές ευχρηστίας για το σχεδιασμό και την ανάπτυξη προϊόντων, στην τομέα της ψυχολογίας και σε ένα ευρύ φάσμα των κοινωνικών επιστημών. Αφορά συμμετέχοντες με «μεγαλόφωνη» σκέψη, οι οποίοι επιτελούν μια σειρά από συγκεκριμένες εργασίες (σενάρια). Οι χρήστες καλούνται να λένε ό, τι κοιτάνε, σκέφτονται, κάνουν, και το αίσθημα που τους δημιουργεί η ενέργεια που κάνουν. Αυτό επιτρέπει στους παρατηρητές να δουν από πρώτο χέρι τη διαδικασία της διεκπεραίωσης της εργασίας (και όχι μόνο το τελικό προϊόν της). Οι παρατηρητές σε μια τέτοια δοκιμασία καλούνται να λάβουν αντικειμενικές σημειώσεις για όλα όσα λένε οι χρήστες, χωρίς να προσπαθήσουν να ερμηνεύσουν τις πράξεις και τα λόγια τους. Οι συνεδρίες δοκιμής είναι συνήθως ακουστικές και οπτικές και καταγράφονται έτσι ώστε οι προγραμματιστές να

μπορούν να πάνε πίσω και να ελέγξουν τι έκαναν οι συμμετέχοντες και πώς αντέδρασαν (Think aloud protocol, 2014).

7.3.2 Υλοποίηση Τεστ Ευχρηστίας

Μετά την επιλογή των συμμετεχόντων, τους καλέσαμε σε ένα ήσυχο περιβάλλον και ξεκινήσαμε το *Τεστ Ευχρηστίας*. Το σενάριο που ακολουθήσαμε και η συζήτηση εξελίχθηκε ως εξής:

- **Η εφαρμογή λειτουργεί σε ηλεκτρονικό υπολογιστή (το οποίο σας παρέχουμε) με λειτουργικό σύστημα Windows.**

Γεια σας, <όνομα συμμετέχοντα>. Το όνομα μου είναι Σωκράτης και θα πραγματοποιήσουμε μαζί την σημερινή συνεδρία.

Πριν ξεκινήσουμε έχουμε κάποιες πληροφορίες για εσάς και θα θέλαμε να σας τις διαβάσω ώστε να είμαστε σίγουροι ότι έχουμε καλύψει τα πάντα.

Πιθανόν, έχετε ήδη μια ιδέα για το λόγο που σας καλέσαμε, αλλά επιτρέψτε μας να σας τα ξανάπούμε εν συντομία. Ζητάμε από τους ανθρώπους να ελέγξουν την χρήση της εφαρμογής μας, ώστε να διαπιστώσουμε εάν λειτουργεί όπως πρέπει. Η συνεδρία θα διαρκέσει περίπου 20 λεπτά.

Αρχικά, θα θέλαμε να σας ενημερώσουμε ότι δοκιμάζουμε την Εφαρμογή και όχι εσάς. Δεν θεωρείτε λάθος ότι κάνετε. Στην πραγματικότητα, αυτό είναι το μόνο μέρος που δεν θα πρέπει να ανησυχείτε ότι θα κάνετε κάποιο λάθος. Καθώς χρησιμοποιείτε την εφαρμογή, θα σας θέτουμε κάποιες ερωτήσεις και θα θέλαμε να μας απαντάτε φωναχτά. Να μας πείτε αυτό που ψάχνετε, τι προσπαθείτε να κάνετε και τι σκέφτεστε. Αυτό θα είναι μια μεγάλη βοήθεια για εμάς. Επίσης, θα θέλαμε οτιδήποτε σκέφτεστε σε σχέση με την εφαρμογή, οτιδήποτε σας απασχολεί, σας δυσκολεύει ή σας αρέσει να το αναφέρετε φωναχτά.

Αυτό το κάνουμε για να βελτιωθεί η εφαρμογή μας, οπότε πρέπει αυτά που θα πείτε να είναι αληθή και ειλικρινά..

Εάν έχετε οποιαδήποτε ερώτηση μπορείτε να ρωτήσετε τον "Παρατηρητή" σας. Μπορεί να μην είναι σε θέση να σας απαντήσει ευθέως, αφού ενδιαφερόμαστε για το πώς οι άνθρωποι πράττουν όταν δεν έχουν κάποιον να κάθεται δίπλα τους για να τους βοηθά. Αλλά ωστόσο εάν έχετε οποιαδήποτε ερώτηση, θα

κάνουμε ότι μπορούμε για να σας απαντήσουμε. Και αν χρειάζεστε να κάνετε κάποιο διάλλειμα, απλώς ενημερώστε μας.

Θα πρέπει να γνωρίζετε ότι καθ΄ όλη τη διάρκεια της συνεδρίας, καταγράφουμε τη δραστηριότητα που κάνετε στον υπολογιστή. Αυτό γίνεται με σκοπό μόνο να παρατηρήσουμε πόσο εύκολα και γρήγορα μπορείτε να χρησιμοποιήσετε την εφαρμογή.

Έχετε κάποια ερώτηση έως εδώ;

Ωραία Μπορούμε λοιπόν να ξεκινήσουμε το τεστ.

- ***Εκκινήστε την εφαρμογή.***

Εκκινήστε την εφαρμογή επιλέγοντας Run  *στο περιβάλλον του Eclipse. Διαβάστε προσεκτικά το κείμενο που εμφανίζεται και κατόπιν πατήστε «Επόμενο». Πρώτα, θα σας ζητήσω να δείτε την εφαρμογή και να μου πείτε τι μπορείτε να κάνετε με αυτή: τι σας κάνει εντύπωση σε αυτή, με τι πιστεύετε ότι σχετίζεται η εφαρμογή, και για ποιο λόγο υπάρχει. Μπορείτε να μετακινηθείτε, εάν θέλετε, αλλά μην κάνετε κλικ σε τίποτα ακόμα.*

- ***Συνεχίστε το αυτό για περίπου δύο λεπτά.***

Ευχαριστώ.

«Εξηγούμε στον συμμετέχοντα με λίγα λόγια τη λειτουργία του προγράμματος και το λόγο ύπαρξης του.»

Τώρα θα σας ζητήσω να κάνετε κάποιες συγκεκριμένες εργασίες. Θα σας τις διαβάσω φωναχτά αλλά σας έχει δοθεί και ένα εκτυπωμένο αντίγραφο.

Έτσι, θα μάθουμε αρκετά για το πώς λειτουργεί η εφαρμογή. Και πάλι, θα πρέπει όσο είναι δυνατό, να προσπαθείτε να σκέφτεστε φωναχτά κατά τη διάρκεια συνέχισης της εργασίας.

- ***Δώστε στον συμμετέχοντα μια λίστα με τα μαθήματα και τις ώρες που αυτός έχει να διδάξει κατά τη διάρκεια του εξαμήνου.***
- ***Δώστε στον συμμετέχοντα το πρώτο σενάριο και διαβάστε το φωναχτά.***
- ***Επιτρέψτε στον χρήστη να προχωρήσει έως το σημείο που πιστεύετε ότι δεν παράγει κάτι ή αν έχει απογοητευτεί.***
- ***Για κάθε εργασία ξεκινάμε εκ νέου την εφαρμογή και δίνουμε το σενάριο στον χρήστη.***

- ***Επαναλαμβάνουμε την ίδια διαδικασία μέχρι να ολοκληρωθούν τα σενάρια, ή μέχρι να τελειώσει ο χρόνος της συνεδρίας.***

Ευχαριστώ, αυτό ήταν πολύ χρήσιμο.

Έχετε οποιαδήποτε απορία για εμάς, τώρα που έχουμε τελειώσει;

Τώρα θα σας δώσουμε να συμπληρώσετε ένα ερωτηματολόγιο προκειμένου να κρίνετε την εφαρμογή μας. Μην ανησυχείτε, το ερωτηματολόγιο αυτό είναι ανώνυμο.

- ***Ευχαριστήστε τους συμμετέχοντες και συνοδεύστε τους εκτός του δωματίου.***

Τα σενάρια που δόθηκαν στους καθηγητές είναι τα ακόλουθα:

Σενάριο 1ο:

Έστω ότι είσαστε ο καθηγητής "Stamatis". Τοποθετήστε ελεύθερα τις προτιμήσεις σας για το ωρολόγιο πρόγραμμα μαθημάτων στο Τμήμα Μηχανικών Πληροφορικής του ΤΕΙ Θεσσαλονίκης.

Διαπραγματευτείτε με τον πράκτορα του προγράμματος μέχρι την ολοκλήρωση του προγράμματος μαθημάτων.

Σενάριο 2ο:

Έστω ότι είσαστε ο καθηγητής "Stamatis". Τοποθετήστε τις ακόλουθες προτιμήσεις:

- ***Δεν μπορώ** την Δευτέρα μετά τις 10:00.*
- ***Δεν μπορώ** την Τρίτη μετά τις 10:00.*
- ***Δεν μπορώ** την Τετάρτη μετά τις 10:00.*
- ***Δεν μπορώ** την Πέμπτη μετά τις 10:00.*
- ***Δεν μπορώ** την Παρασκευή μετά τις 10:00.*

Διαπραγματευτείτε με τον υπεύθυνο του προγράμματος. Όταν σας ζητηθούν για δεύτερη φορά προτιμήσεις, τοποθετήστε τις ίδιες.

Συνεχίστε μέχρι να ολοκληρωθεί η διαδικασία.

Σενάριο 3ο:

Έστω ότι είσαστε ο καθηγητής "Stamatis". Τοποθετήστε τις ακόλουθες προτιμήσεις:

- ***Δεν επιθυμώ** την Δευτέρα πριν τις 18:00.*
- ***Δεν επιθυμώ** την Τρίτη πριν τις 18:00.*
- ***Δεν επιθυμώ** την Τετάρτη πριν τις 18:00.*
- ***Δεν επιθυμώ** την Πέμπτη πριν τις 18:00.*

- ***Δεν επιθυμώ*** *την Παρασκευή πριν τις 18:00.*

Διαπραγματευτείτε με τον υπεύθυνο του προγράμματος. Όταν σας ζητηθούν και δεύτερη φορά προτιμήσεις, τοποθετήστε τις όποιες προτιμήσεις επιθυμείτε.

Συνεχίστε μέχρι να ολοκληρωθεί η διαδικασία.

Τα σενάρια που δόθηκαν στους φοιτητές είναι τα ακόλουθα:

Σενάριο 1ο:

Έστω ότι είσαστε ο εκπρόσωπος των φοιτητών του 1ου εξαμήνου σπουδών. Τοποθετήστε τις ακόλουθες προτιμήσεις και περιμένετε να λάβετε το πρόγραμμα μαθημάτων σας.

- ***Δεν επιθυμώ*** *την Τετάρτη πριν τις 12:00.*
- ***Δεν επιθυμώ*** *την Παρασκευή μετά τις 15:00.*

Σενάριο 2ο:

Έστω ότι είσαστε ο εκπρόσωπος των φοιτητών του 1ου εξαμήνου σπουδών. Τοποθετήστε τις ακόλουθες προτιμήσεις και περιμένετε να λάβετε το πρόγραμμα μαθημάτων σας.

- ***Δεν επιθυμώ*** *την Δευτέρα μετά τις 16:00.*
- ***Δεν επιθυμώ*** *την Πέμπτη μετά τις 13:00.*

Το ερωτηματολόγιο που έχει δοθεί στους χρήστες για την αξιολόγηση της εφαρμογής μας, παρατίθεται στο Παράρτημα Γ.

Κατά την υλοποίηση του τεστ, κάθε φορά εξετάζεται ένας χρήστης. Η διενέργεια του τεστ βιντεοσκοπείται και καταγράφεται η δραστηριότητα των χρηστών με τη βοήθεια του λογισμικού "Camtasia Studio".

7.4 Αποτελέσματα

Μέσω της συνάρτησης αξιολόγησης που υλοποιήθηκε εντός του προγράμματος μπορούμε να μετρήσουμε την ποιότητα του προγράμματος όσον αφορά την ικανοποίηση των προτιμήσεων που τέθηκαν, σε σχέση με τους συνολικούς περιορισμούς που δόθηκαν στο πρόγραμμα από τους καθηγητές. Συμπερασματικά μπορούμε να πούμε ότι εάν δοθούν φυσιολογικοί περιορισμοί από τους καθηγητές, τότε η τιμή της συνάρτησης πλησιάζει πάρα πολύ στο 0. Αυτό σημαίνει ότι η εφαρμογή τα καταφέρνει αρκετά ικανοποιητικά να υλοποιήσει το στόχο της. Όταν δεν προκύπτουν καθόλου συγκρούσεις, τότε η τιμή της συνάρτησης είναι 0, γεγονός που σημαίνει ότι πετύχαμε το ιδανικό πρόγραμμα σύμφωνα με τις προτιμήσεις των καθηγητών. Αυτό το γεγονός βέβαια είναι αρκετά σπάνιο να συμβεί, λόγω του πλήθους των

περιορισμών που συνήθως τίθενται από τους καθηγητές. Άλλα θεωρούμε ότι με μια τιμή κάτω από 0,15, το πρόγραμμα πέτυχε το σκοπό του.

Στον πίνακα που ακολουθεί φαίνεται η τιμή της συνάρτησης αξιολόγησης για διαφορετικές εκτελέσεις του προγράμματος:

Πίνακας 13. Μεταβολή τιμής της συνάρτησης αξιολόγησης.

Συνολικές Προτιμήσεις	Συγκρούσεις "Δεν μπορώ"	Συγκρούσεις "Επιθυμώ/Δεν επιθυμώ"	Τιμή Συνάρτησης Αξιολόγησης
30	0	0	0,00
30	0	1	0,02
30	0	2	0,03
30	0	3	0,05
30	0	4	0,07
30	0	5	0,08
30	1	0	0,05
30	1	1	0,07
30	1	2	0,08
30	1	3	0,10
30	1	4	0,12
30	1	5	0,13
30	2	0	0,10
30	2	1	0,12
30	2	2	0,13
30	2	3	0,15
30	2	4	0,17
30	2	5	0,18
30	3	0	0,15
30	3	1	0,17
30	3	2	0,18
30	3	3	0,20
30	3	4	0,22
30	3	5	0,23
30	4	0	0,20
30	4	1	0,22
30	4	2	0,23
30	4	3	0,25
30	4	4	0,27
30	4	5	0,28
30	5	0	0,25
30	5	1	0,27
30	5	2	0,28
30	5	3	0,30
30	5	4	0,32
30	5	5	0,33

Από την υλοποίηση του τεστ ευχρηστίας προέκυψαν τα ακόλουθα συμπεράσματα:

- Σε σχέση με την αισθητική της εφαρμογής (ορθογραφικά λάθη, μέγεθος γραμμάτων, στοίχιση, μορφοποίηση, μπάρες κύλισης), οι χρήστες έμειναν αρκετά ευχαριστημένοι. Τους άρεσε ότι η εφαρμογή ήταν αρκετά λιτή σε εμφάνιση, με αποτέλεσμα ότι χρειαζόταν να είναι σε θέση να το εντοπίσουν εύκολα. Ένα θέμα το οποίο μας επισήμαναν και βελτιώσαμε ήταν σε σχέση με το χρωματικό υπόβαθρο κάποιων πεδίων, το οποίο τους φάνηκε πολύ έντονο σε σχέση με την υπόλοιπη εφαρμογή.
- Σε ότι έχει να κάνει με την εμφάνιση ενημερωτικών μηνυμάτων για τη χρήση της εφαρμογής και σε τυχόν απορίες που ενδέχεται να δημιουργηθούν στους χρήστες, τα αποτελέσματα του τεστ ήταν αντικρουόμενα. Ορισμένοι έμειναν ικανοποιημένοι από τις πληροφορίες που αυτά προσέφεραν και άλλοι ήθελαν να είναι περισσότερο αναλυτικά. Αποφασίσαμε λοιπόν να προσθέσουμε περισσότερο ενημερωτικό κείμενο, έτσι ώστε να παρέχουμε καλύτερη ενημέρωση στους χρήστες,
- Τέλος σε ότι έχει σχέση με την ουσία της εφαρμογής, δηλαδή την δημιουργία του τελικού ωρολόγιού προγράμματος, τη μέθοδο αποστολής των προτιμήσεων των χρηστών και τη διαπραγμάτευση με τον πράκτορα του προγραμματιστή, το σύνολο των χρηστών, μας χάρισε πολύ κολακευτικά σχόλια και έμεινε πολύ ικανοποιημένο από τη διαδικασία.

7.5 Σύνοψη συμπερασμάτων αξιολόγησης

Συμπερασματικά είμαστε σε θέση να αναφέρουμε ότι οι χρήστες παρότι μας επισήμαναν κάποια προβλήματα σε σχέση με την ενημέρωση που τους παρείχαμε μέσω ενημερωτικών μηνυμάτων, φαίνεται πως έμειναν πολύ ικανοποιημένοι από τον τρόπο χρήσης της εφαρμογής και την ποιότητα του ωρολογίου προγράμματος που δημιουργήθηκε σε σχέση πάντα με τις προτιμήσεις που αυτοί μας απέστειλαν. Επίσης η τιμή της συνάρτησης αξιολόγησης για ένα πλήθος εκτελέσεων της εφαρμογής, με διαφορετικές παραμέτρους κάθε φορά, εμφανίζει πολύ καλά αποτελέσματα και αποδεικνύει ότι η ποιότητα του τελικού ωρολογίου προγράμματος βρίσκεται σε αρκετά υψηλό επίπεδο.

Η χρήση της τεχνολογίας πρακτόρων και της διαδικασίας διαπραγμάτευσης που αυτοί μας παρέχουν φαίνεται να είναι ένα σημαντικό εργαλείο για όλα τα εμπλεκόμενα μέρη της διαδικασίας δημιουργίας του προγράμματος.

8. *Επίλογος*

Στα πλαίσια αυτής της διπλωματικής μελετήθηκε το πρόβλημα της δημιουργίας ωρολογίου προγράμματος, επικεντρωθήκαμε όμως στη δημιουργία του για Ανώτατα Εκπαιδευτικά Ιδρύματα. Μελετήθηκε επίσης, η τεχνολογία πρακτόρων λογισμικού και ο τρόπος με τον οποίο αυτοί συνεργάζονται και αλληλοεπιδρούν σε ένα πολυπρακτορικό σύστημα. Τέλος, αναπτύχθηκε ένα πολυπρακτορικό σύστημα για την δημιουργία του ωρολόγιού προγράμματος του τμήματος Μηχανικών Πληροφορικής του Αλεξάνδρειου Τεχνολογικού Εκπαιδευτικού Ιδρύματος Θεσσαλονίκης. Προκειμένου να δημιουργηθεί ένα όσο το δυνατόν πιο δίκαιο και αποδοτικό πρόγραμμα, διενεργήθηκε στατιστική έρευνα ανάμεσα σε καθηγητές και σπουδαστές του τμήματος. Το όλο σύστημα, έχει αναπτυχθεί σε γλώσσα προγραμματισμού Java, χρησιμοποιώντας την πλατφόρμα της JADE. Το τελικό σύστημα αξιολογήθηκε με τη βοήθεια μιας συνάρτησης αξιολόγησης και τη διενέργεια ενός τεστ ευχρηστίας.

8.1 Σύνοψη και συμπεράσματα

Το πρόβλημα της δημιουργίας ωρολογίου προγράμματος, αποτελεί ένα από τα πλέον σύνθετα προβλήματα που έχουν αντιμετωπιστεί τα τελευταία χρόνια, για την μελέτη και επίλυση του οποίου έχουν γραφτεί ένας σημαντικός αριθμός εργασιών. Η δυσκολία του πηγάζει από το γεγονός ότι θα πρέπει να ληφθεί υπόψιν ένας μεγάλος αριθμός περιορισμών. Είναι αποφασιστικής σημασίας λοιπόν, να γίνει προσπάθεια να προσομοιωθεί η διαδικασία δημιουργίας του με όσο το δυνατόν πιο πραγματικές συνθήκες. Στο κομμάτι αυτό έρχεται να δώσει λύση η τεχνολογία των ευφυών πρακτόρων λογισμικού. Η ανάπτυξη μιας ομάδας πρακτόρων σε ένα πολυπρακτορικό σύστημα, δημιουργεί τις συνθήκες διαπραγμάτευσης μεταξύ αυτών, προκειμένου αυτοί να συνεργαστούν για την επίλυση του προβλήματος. Για το σκοπό αυτό μελετήθηκε ένας αριθμός εργασιών για την επίλυση του προβλήματος με τη χρήση της τεχνολογίας των πρακτόρων. Αυτές όμως, κατά τη γνώμη μας παρουσίασαν κάποιες αδυναμίες. Τέτοιες αδυναμίες είναι το γεγονός ότι δεν λάμβαναν καθόλου υπόψιν τους τις προτιμήσεις των σπουδαστών και ότι δεν παρουσίαζαν κάποιο τρόπο για την επίσπευση της διαδικασίας σε περίπτωση που δεν προέκυπτε γρήγορα κάποιο συμπέρασμα κατά τη διαδικασία της διαπραγμάτευσης. Αυτά τα μειονεκτήματα επιχειρήθηκε να λύσουμε στα πλαίσια αυτής της διπλωματικής.

Από στατιστική έρευνα που δημιουργήθηκε, προέκυψε, πως η πλειοψηφία των συμμετεχόντων πλευρών επιθυμεί να λαμβάνει μέρος στη διαδικασία δημιουργίας του προγράμματος με έναν εκπρόσωπο φοιτητών από κάθε εξάμηνο. Επίσης όλα τα εμπλεκόμενα μέρη επιθυμούν η διαδικασία να ολοκληρώνεται έπειτα από δυο με τρεις φάσεις διαπραγμάτευσης.

Η διαδικασία αυτή προσομοιώθηκε με τη δημιουργία ενός πολυπρακτορικού συστήματος στο οποίο συμμετέχουν τρεις κλάσεις πρακτόρων, ο προγραμματιστής, οι καθηγητές και οι φοιτητές. Ανάμεσα τους ανταλλάχθηκαν FIPA-ACL μηνύματα και έπειτα από τις δυο πρώτες φάσεις διαπραγμάτευσης, χρησιμοποιήθηκε ένα πρωτόκολλο πειθούς, προκειμένου να προτείνει λύσεις και να ολοκληρωθεί η διαδικασία.

Από τη διαδικασία της αξιολόγησης που διενεργήθηκε, προκύπτει ξεκάθαρα κατά τη γνώμη μας, η υπεροχή της χρήσης της τεχνολογίας πρακτόρων για την δημιουργία του ωρολογίου προγράμματος έναντι των υπολοίπων τεχνικών, λόγω του δυνατότητας διαπραγμάτευσης και επικοινωνίας μεταξύ των συμμετεχόντων μερών. Επίσης, η πλατφόρμα της JADE, αποτελεί ένα ιδιαίτερα εύχρηστο και αποδοτικό εργαλείο για την επίτευξη της επικοινωνίας και της διαπραγμάτευσης μεταξύ των πρακτόρων και μπορεί να χρησιμοποιηθεί πολύ αποτελεσματικά για την επίλυση και άλλων σύνθετων προβλημάτων.

8.2 Μελλοντικές επεκτάσεις

Η εφαρμογή που αναπτύξαμε, παρότι κατά τη γνώμη μας επιλύει αρκετά ικανοποιητικά το πρόβλημα της δημιουργίας ωρολογίου προγράμματος, μπορεί να βελτιωθεί ακόμη περισσότερο.

Αρχικά θα πρέπει να τρέξει ένα βαθύ και εκτεταμένο τεστ αξιολόγησης με έναν μεγάλο αριθμό χρηστών (καθηγητών και φοιτητών), προκειμένου να είμαστε σε θέση να αξιολογήσουμε πραγματικά την εφαρμογή με μεγαλύτερη ακρίβεια.

Στο θέμα της δημιουργίας του προγράμματος, ένας αλγόριθμος round-robin θα βοηθούσε ώστε να γίνεται περισσότερη δίκαιη κατανομή των μαθημάτων ανάμεσα στους καθηγητές. Επίσης, θα ήταν περισσότερο ολοκληρωμένο, εάν λάμβανε υπόψιν του και τις διαθέσιμες αίθουσες του τμήματος, δηλαδή τον αριθμό των αιθουσών και εάν αυτές διαθέτουν εργαστηριακό εξοπλισμό, ή όχι. Ένας ακόμη παράγοντας, ο οποίος θα μπορούσε να βελτιωθεί, είναι να υπάρχει πρόβλεψη για μαθήματα επιλογής, τα οποία θα κατανέμονται ανάμεσα στα διαθέσιμα εξάμηνα.

Στο θέμα των πρακτόρων, αρχικά θα μπορούσε να τοποθετηθεί το πρόγραμμα σε έναν server και να αποστέλλεται ενημερωτικό μήνυμα ηλεκτρονικής αλληλογραφίας σε καθηγητές και σπουδαστές, προκειμένου να ξεκινήσει η διαδικασία της δημιουργίας του προγράμματος.

Τότε ένας πράκτορας για κάθε συμμετέχουσα πλευρά θα μεταναστεύσει στον υπολογιστή του αντίστοιχου εκπροσώπου και διαπραγματευτεί με τον προγραμματιστή.

Προκειμένου να υπάρξει μια περισσότερο ολοκληρωμένη προσέγγιση, θα ήταν εφικτό να επεκταθεί το σύστημα με τρόπο ώστε όλες οι λειτουργίες του τμήματος να πραγματοποιούνταν μέσω πρακτόρων. Τέτοιες λειτουργίες είναι η ανάθεση των μαθημάτων σε καθηγητές και η δήλωση των μαθημάτων από φοιτητές. Επίσης θα μπορούσε να δημιουργηθεί ανάλογο πολυπρακτορικό σύστημα και για τα υπόλοιπα τμήματα του Αλεξάνδρειου Τεχνολογικού Ιδρύματος και οι πράκτορες αυτοί να συνεργάζονται και να επικοινωνούν μεταξύ τους για τη δημιουργία ενός ολοκληρωμένου προγράμματος για ολόκληρο το Ίδρυμα.

Βιβλιογραφία

Aderemi O. Adewumi, Babatunde A. Sawyerr, & M. Montaz Ali. (2009). A heuristic solution to the university timetabling problem. *Engineering Computations*, *26*(8), 972–984. http://doi.org/10.1108/02644400910996853

Ahmad, A. M., Chian, C. Y., & Mohamad, F. (2002). Application of multi-agents for the development of university timetabling using JATLite. In *Student Conference on Research and Development, 2002. SCOReD 2002* (pp. 521–522). http://doi.org/10.1109/SCORED.2002.1033173

A SURVEY OF APPROACHES FOR UNIVERSITY COURSE TIMETABLING PROBLEM. (2014.). Retrieved May 16, 2015, from http://www.academia.edu/1842547/A_SURVEY_OF_APPROACHES_FOR_UNIVERSITY_COURSE_TIMETABLING_PROBLEM

Babkin, E., Abdulrab, H., & Babkina, T. (2009). AgentTime: A Distributed Multi-agent Software System for University's Timetabling. In M. A. Aziz-Alaoui & C. Bertelle (Eds.), *From System Complexity to Emergent Properties* (pp. 341–354). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-02199-2_17.

Bakir, M.A. & C. Aksop, (2008). " A 0-1 INTEGER PROGRAMMING APPROACH TO A UNIVERSITY TIMETABLING PROBLEM" in Hacettepe Journal of Mathematics and Statistics, Volume 37,(pp. 41 – 55).

Barraclough, E. D. (1965). The application of a digital computer to the construction of timetables. *The Computer Journal - CJ*, *8*(2), 136–146. http://doi.org/10.1093/comjnl/8.2.136

Center for History and New Media. (n.d.). Quick Start Guide. Retrieved from http://zotero.org/support/quick_start_guide

Ceschia, S., Di Gaspero, L., & Schaerf, A. (2012). Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. *Computers & Operations Research*, *39*(7), 1615–1624. http://doi.org/10.1016/j.cor.2011.09.014

Chatzara, K., Karagiannidis, C. & Stamatis, D. (2011). "Computers Can Feel Too: Intelligent Emotional Agents in E-Learning Systems", Chapter 10, (pp. 188-200).

Chen, Y. (2014). Design and implementation of intellective class scheduling system based on negotiation. In *2014 9th International Conference on Computer Science Education (ICCSE)* (pp. 1078–1082). http://doi.org/10.1109/ICCSE.2014.6926629

Chiu, D. K. W., Cheung, S. C., Hung, P. C. K., & Leung, H. (2004). Constraint-based negotiation in a multi-agent information system with multiple platform support. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences, 2004* (p. 10 pp.–). http://doi.org/10.1109/HICSS.2004.1265099

Coen, M. (1997). "Building Brains for Rooms: Designing Distributed Software Agents", In Proceedings of the Ninth Annual Conference on Innovative Applications of Artificial Intelligence (IAAI'97), (pp. 971-977).

Craenen, B. G. W., & Theodoropoulos, G. K. (2010). Interfacing multi-agent models to distributed simulation platforms: The case of PDES-MAS. In *Simulation Conference (WSC), Proceedings of the 2010 Winter* (pp. 587–594). http://doi.org/10.1109/WSC.2010.5679127

D'Inverno, M., Luck, M., Georgeff, M., Kinny, D., & Wooldridge, M. (2004). The dMARS Architecture: A Specification of the Distributed Multi-Agent Reasoning System. *Autonomous Agents and Multi-Agent Systems*, *9*(1-2), 5–53. http://doi.org/10.1023/B:AGNT.0000019688.11109.19

Eclipse (software). (2015, May 7). In *Wikipedia, the free encyclopedia*. Retrieved from http://en.wikipedia.org/w/index.php?title=Eclipse_(software)&oldid=661209889

Genetic algorithm. (2015, May 12). In *Wikipedia, the free encyclopedia*. Retrieved from http://en.wikipedia.org/w/index.php?title=Genetic_algorithm&oldid=661971980

Graph coloring. (2015, April 1). In *Wikipedia, the free encyclopedia*. Retrieved from http://en.wikipedia.org/w/index.php?title=Graph_coloring&oldid=654456359

Graph theory. (2015, May 10). In *Wikipedia, the free encyclopedia*. Retrieved from http://en.wikipedia.org/w/index.php?title=Graph_theory&oldid=661735004

Integer programming. (2015, May 15). In *Wikipedia, the free encyclopedia*. Retrieved from http://en.wikipedia.org/w/index.php?title=Integer_programming&oldid=662472855

Irene, H. S. F., Deris, S., & Mohd Hashim, S. Z. (2008). Investigating constraint-based reasoning for university timetabling problem. In *Imecs 2009: International Multi-Conference Of Engineers And Computer Scientists, Vols I and II* (pp. 139–143). China: Int Assoc Engineers-Iaeng. Retrieved from http://eprints.utm.my/12906/

Ito, T., & SHINTANI, T. (1997). *An Agenda-scheduling System Based on Persuasion Among Agents*. In Proceedings of IPSJ International Symposium on Information Systems and Technologies for Network Society, pp.287-294, World Scientific.

Jade Site | Java Agent DEvelopment Framework. (n.d.). Retrieved from http://jade.tilab.com/

Javeau, C. (1996). "Η έρευνα με ερωτηματολόγιο – Το εγχειρίδιο του καλού ερευνητή" Ποσοτικές Προσεγγίσεις-Μετάφραση: Κ. Τζαννόνε-Τζώρτζη,

Έκδοση/Διάθεση: Εκδόσεις Τυπωθήτω, Γεώργιου Δάρδανου, Αθήνα, 1996

Kambi, M., & Gilbert, D. (1996). Timetabling in Constraint Logic Programming. In *In Proc. 9th Symp. on Industrial Applications of PROLOG (INAP'96* (pp. 79–88).

Kautz, H. & M. Coen, (1994). "An Experiment in the Design of Software Agents", In Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), (pp. 438-443).

Kraus, S., Wilkenfeld, J., & Zlotkin, G. (1995). Multiagent negotiation under time constraints. *Artificial Intelligence*, *75*(2), 297–345. http://doi.org/10.1016/0004-3702(94)00021-R

Linear programming. (2015, April 22). In *Wikipedia, the free encyclopedia*. Retrieved from http://en.wikipedia.org/w/index.php?title=Linear_programming&oldid=658133098

Lü, Z., & Hao, J.-K. (2010). Adaptive Tabu Search for course timetabling. *European Journal of Operational Research*, *200*(1), 235–244. http://doi.org/10.1016/j.ejor.2008.12.007

Malim, M. R., Khader, A. T., & Mustafa, A. (2006). An immune-based approach to university course timetabling: Immune network algorithm. In *International Conference on Computing Informatics, 2006. ICOCI '06* (pp. 1–6). http://doi.org/10.1109/ICOCI.2006.5276571

Murray, K., Müller, T., & Rudová, H. (2007). Modeling and Solution of a Complex University Course Timetabling Problem. In E. K. Burke & H. Rudová (Eds.), *Practice and Theory of Automated Timetabling VI* (pp. 189–209). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-77345-0_13

Nandhini, M., & Kanmani, S. (2009). Implementation of class timetabling using multi agents. In *International Conference on Intelligent Agent Multi-Agent Systems, 2009. IAMA 2009* (pp. 1–2). http://doi.org/10.1109/IAMA.2009.5228065

Neruda, R., & Šlapák, M. (2012). Evolving Decision Strategies for Computational Intelligence Agents. In D.-S. Huang, J. Ma, K.-H. Jo, & M. M. Gromiha (Eds.), *Intelligent Computing Theories and Applications* (pp. 213–220). Springer Berlin Heidelberg. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-31576-3_28

Nouri, H. E., & Belkahla Driss, O. (2013). Distributed model for university course timetabling problem. In *2013 International Conference on Computer Applications Technology (ICCAT)* (pp. 1–6). http://doi.org/10.1109/ICCAT.2013.6521991

Obit, J. H., Landa-Silva, D., Ouelhadj, D., Vun, T. K., & Alfred, R. (2011). Designing a multi-agent approach system for distributed course timetabling. In *2011 11th International Conference on Hybrid Intelligent Systems (HIS)* (pp. 103–108). http://doi.org/10.1109/HIS.2011.6122088

Oprea, M. (2006.). MAS_UP-UCT: A Multi-Agent System for University Course Timetable Scheduling.

Qu, R., Burke, E. K., Mccollum, B., Merlot, L. T., & Lee, S. Y. (2009a). A Survey of Search Methodologies and Automated System Development for Examination Timetabling. *J. of Scheduling*, *12*(1), 55–89. http://doi.org/10.1007/s10951-008-0077-5

Qu, R., Burke, E. K., Mccollum, B., Merlot, L. T., & Lee, S. Y. (2009b). A Survey of Search Methodologies and Automated System Development for Examination Timetabling. *J. of Scheduling*, *12*(1), 55–89. http://doi.org/10.1007/s10951-008-0077-5

Redl, T. A. (n.d.). *University Timetabling via Graph Coloring: An Alternative Approach*.

Ren, Z., Anumba, C. J., & Ugwu, O. O. (2003). The development of a multi-agent system for construction claims negotiation. *Advances in Engineering Software*, *34*(11–12), 683–696. http://doi.org/10.1016/S0965-9978(03)00107-8

Suman, B., & Kumar, P. (2005). A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the Operational Research Society*, *57*(10), 1143–1160. http://doi.org/10.1057/palgrave.jors.2602068

Tan, T., Tan, T., West, G., & Low, S. Y. (2012). Learning and Cooperating Multi-agent Scheduling Repair Using a Provenance-Centred Approach. In *2012 5th International Conference on Human System Interactions (HSI)* (pp. 152–159). http://doi.org/10.1109/HSI.2012.30

Think aloud protocol. (2014, May 6). In *Wikipedia, the free encyclopedia*. Retrieved from http://en.wikipedia.org/w/index.php?title=Think_aloud_protocol&oldid=607394887

Tkaczyk, R., Ganzha, M., & Paprzycki, M. (2013). AgentPlanner - agent-based timetabling system - preliminary design and evaluation. In *System Theory, Control and Computing (ICSTCC), 2013 17th International Conference* (pp. 795–800). http://doi.org/10.1109/ICSTCC.2013.6689059

Usability testing. (2015, April 21). In *Wikipedia, the free encyclopedia*. Retrieved from http://en.wikipedia.org/w/index.php?title=Usability_testing&oldid=657639531

WANGMAETEEKUL, P. (2011). *Using Distributed Agents to Create University Course Timetables Addressing Essential & Desirable Constraints and Fair Allocation of Resources* (Doctoral). Durham University. Retrieved from http://etheses.dur.ac.uk/3602/

Bellifemine F.L, Caire G., Greenwood D., (2004). Developing Multi-Agent Systems with JADE -. Wiley Ed.: Retrieved May 16, 2015, from http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470057475.html

Wooldridge, M., & Jennings, N. R. (1995). Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, *10*, 115–152.

Yang, S., & Jat, S. N. (2011). Genetic Algorithms With Guided and Local Search Strategies for University Course Timetabling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *41*(1), 93–106. http://doi.org/10.1109/TSMCC.2010.2049200

Zhang, L., & Lau, S. K. (2005). Constructing university timetable using constraint satisfaction programming approach. In *International Conference on Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce* (Vol. 2, pp. 55–60). http://doi.org/10.1109/CIMCA.2005.1631445

Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας και Η. Σακελλαρίου. *Τεχνητή Νοημοσύνη - Γ' Έκδοση, ISBN: 978-960-8396-64-7, Έκδοση/Διάθεση: Εκδόσεις Πανεπιστημίου Μακεδονίας, 2011*

http://www.it.teithe.gr/wp-content/UserFiles/newprog/examinologio.pdf

http://www.it.teithe.gr/

www.fipa.org

Παράρτημα Α

Α1. Ερωτηματολόγιο για καθηγητής

1. Φύλο
   Άνδρας ☐ Γυναίκα ☐
2. Βαθμίδα:
   Κ. Εφαρμογών – Επίκουρος Καθηγητής ☐
   Αναπληρωτής καθηγητής ☐ Καθηγητής ☐

3. Πιστεύετε ότι θα έπαιζε σημαντικό ρόλο η άποψη σας στη διαδικασία δημιουργίας του ωρολογίου προγράμματος;

   ΝΑΙ ☐ ΟΧΙ ☐

4. Πόσα μαθήματα διδάσκετε στο εξάμηνο;

   1-2 ☐ 3-4 ☐ 4-6 ☐ Περισσότερα ☐

5. Διδάσκετε και σε άλλο Πανεπιστημιακό Ίδρυμα;

   ΝΑΙ ☐ ΟΧΙ ☐

6. Προτιμάτε πρωινά μαθήματα, ή απογευματινά;

   Πρωινά ☐ Απογευματινά ☐ Δεν έχω πρόβλημα ☐

7. Σας ενδιαφέρει το πρόγραμμα να είναι συμπαγές (χωρίς κενά ανάμεσα στα μαθήματα);

   ΝΑΙ ☐ ΟΧΙ ☐ Δεν με απασχολεί ☐

8. Υπάρχουν ώρες στις οποίες είναι αδύνατον να αναλάβετε κάποιο μάθημα;

ΝΑΙ ☐ ΟΧΙ ☐

9. Συνήθως πόσους περιορισμούς δίνετε κατά τη δημιουργία του ωρολογίου προγράμματος;

Κανέναν ☐ 1-3 ☐ 4-6 ☐ Περισσότερους από 4 ☐

10. Αν κατά τη δημιουργία του ωρολογίου προγράμματος, υπάρχει πρόβλημα με τους περιορισμούς σας, πόσες φορές πιστεύετε ότι θα έπρεπε να σας ζητηθεί εναλλακτική πρόταση, χωρίς να δημιουργείται μεγάλη καθυστέρηση;

Καμία ☐ 1-2 ☐ 3-4 ☐ Περισσότερους από 4 ☐

11. Πιστεύετε ότι θα πρέπει να υπάρχουν κατά τη διάρκεια της εβδομάδας κάποια κενά διαστήματα, με σκοπό να τοποθετούνται εκεί οι πιθανές αναπληρώσεις;

ΝΑΙ ☐ ΟΧΙ ☐

12. Πιστεύετε ότι οι φοιτητές θα πρέπει να έχουν άποψη κατά τη διαδικασία δημιουργίας του ωρολογίου προγράμματος;

ΝΑΙ ☐ ΟΧΙ ☐

13. Τι άλλο θεωρείτε βασικό στη διαδικασία δημιουργίας του ωρολογίου προγράμματος και το οποίο δεν αναφέρεται παραπάνω?

Ευχαριστώ για το χρόνο σας

Α2. Ερωτηματολόγιο για φοιτητές

1. Φύλο

   Άνδρας ☐ Γυναίκα ☐

2. Εξάμηνο σπουδών:

   1ο ☐ 2ο ☐ 3ο ☐ 4ο ☐ 5ο ☐ 6ο ☐ 7ο ☐

   Μεγαλύτερο εξάμηνο ☐ Μεταπτυχιακός φοιτητής ☐

3. Εργάζεστε παράλληλα με τις σπουδές;

   ΝΑΙ ☐ ΟΧΙ ☐

4. Πιστεύετε ότι θα έπαιζε σημαντικό ρόλο η άποψη σας στη διαδικασία δημιουργίας του ωρολογίου προγράμματος;

   ΝΑΙ ☐ ΟΧΙ ☐

   Αν όχι, δικαιολογήστε την απάντηση σας:

   ☐

5. Πως πιστεύετε ότι θα πρέπει να διατυπώνετε τις προτιμήσεις σας, χωρίς να δημιουργείται μεγάλη καθυστέρηση;

   Ο καθένας ξεχωριστά ☐

   Μέσω ενός ξεχωριστού εκπροσώπου για το κάθε εξάμηνο ☐

   Μέσω ενός εκπροσώπου για ολόκληρο το τμήμα ☐

6. Προτιμάτε πρωινά μαθήματα, ή απογευματινά;

   Πρωινά ☐ Απογευματινά ☐ Δεν έχω πρόβλημα ☐

7. Σας ενδιαφέρει το πρόγραμμα να είναι συμπαγές (χωρίς κενά ανάμεσα στα μαθήματα);

ΝΑΙ ☐ ΟΧΙ ☐ Δεν με απασχολεί ☐

8. Πιστεύετε ότι θα πρέπει να υπάρχουν κατά τη διάρκεια της εβδομάδας κάποια κενά διαστήματα, με σκοπό να τοποθετούνται εκεί οι πιθανές αναπληρώσεις;

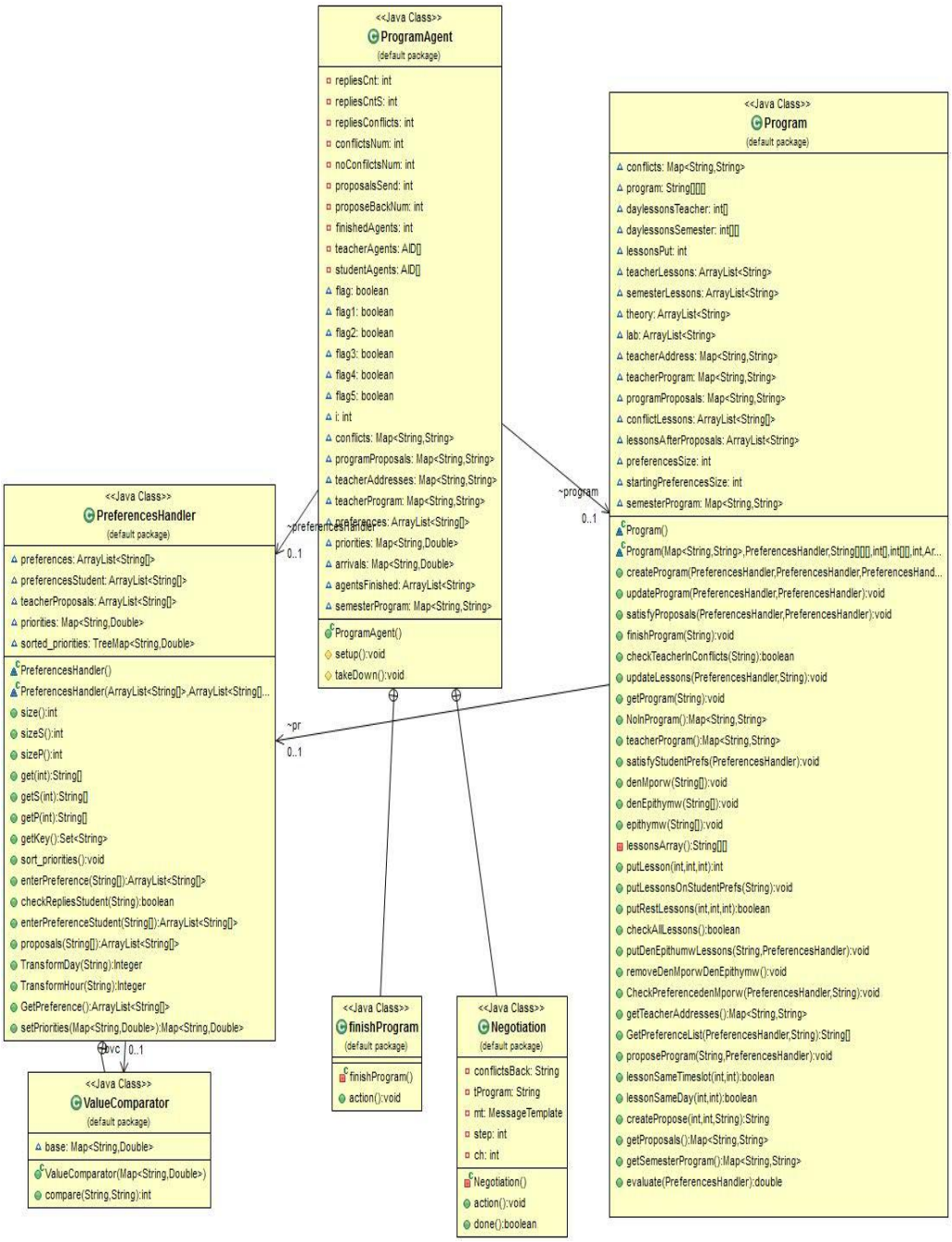
ΝΑΙ ☐ ΟΧΙ ☐

9. Τι άλλο θεωρείτε βασικό στη διαδικασία δημιουργίας του ωρολογίου προγράμματος και το οποίο δεν αναφέρεται παραπάνω?

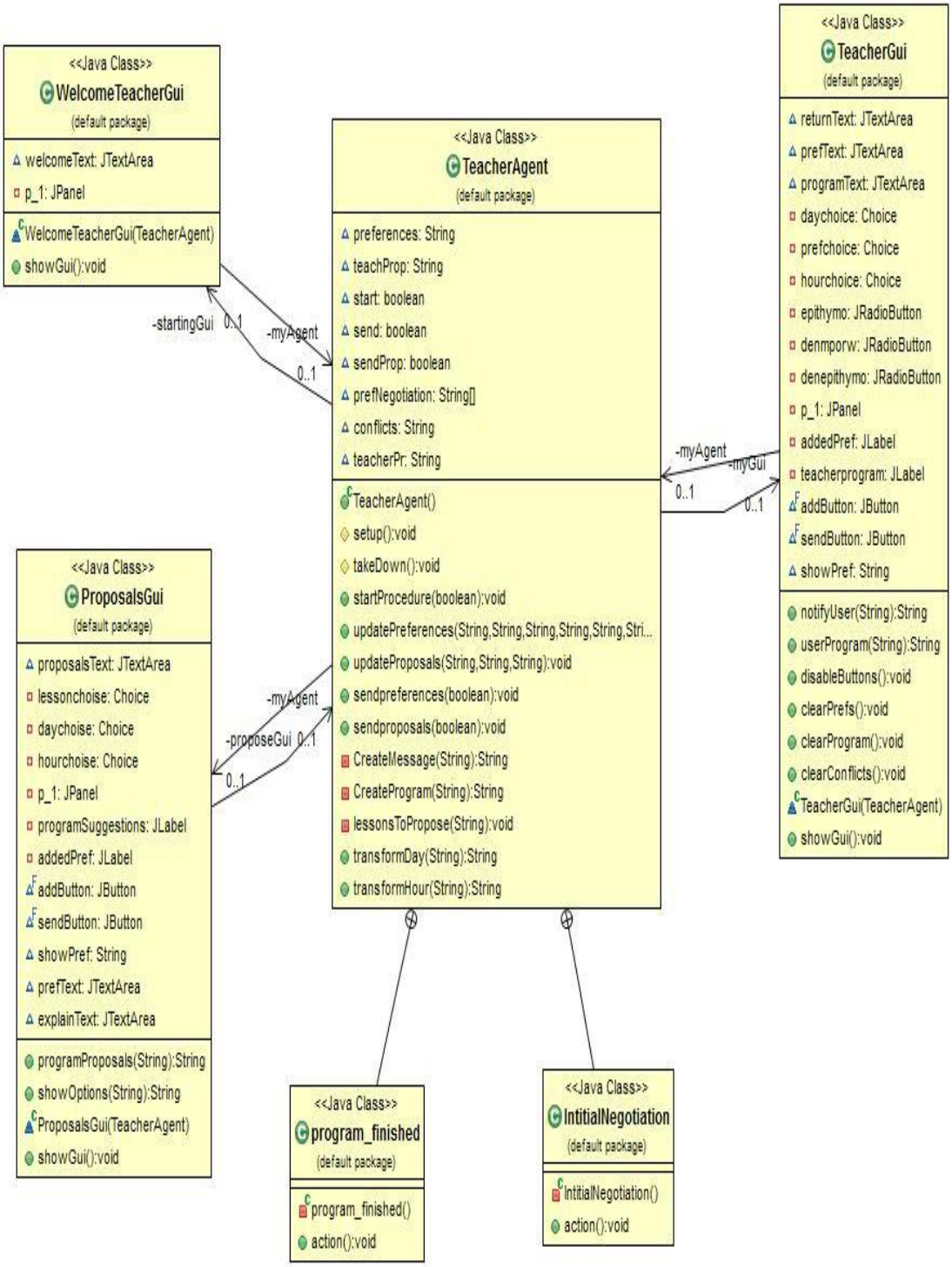
Ευχαριστώ για το χρόνο σας

Παράρτημα Β

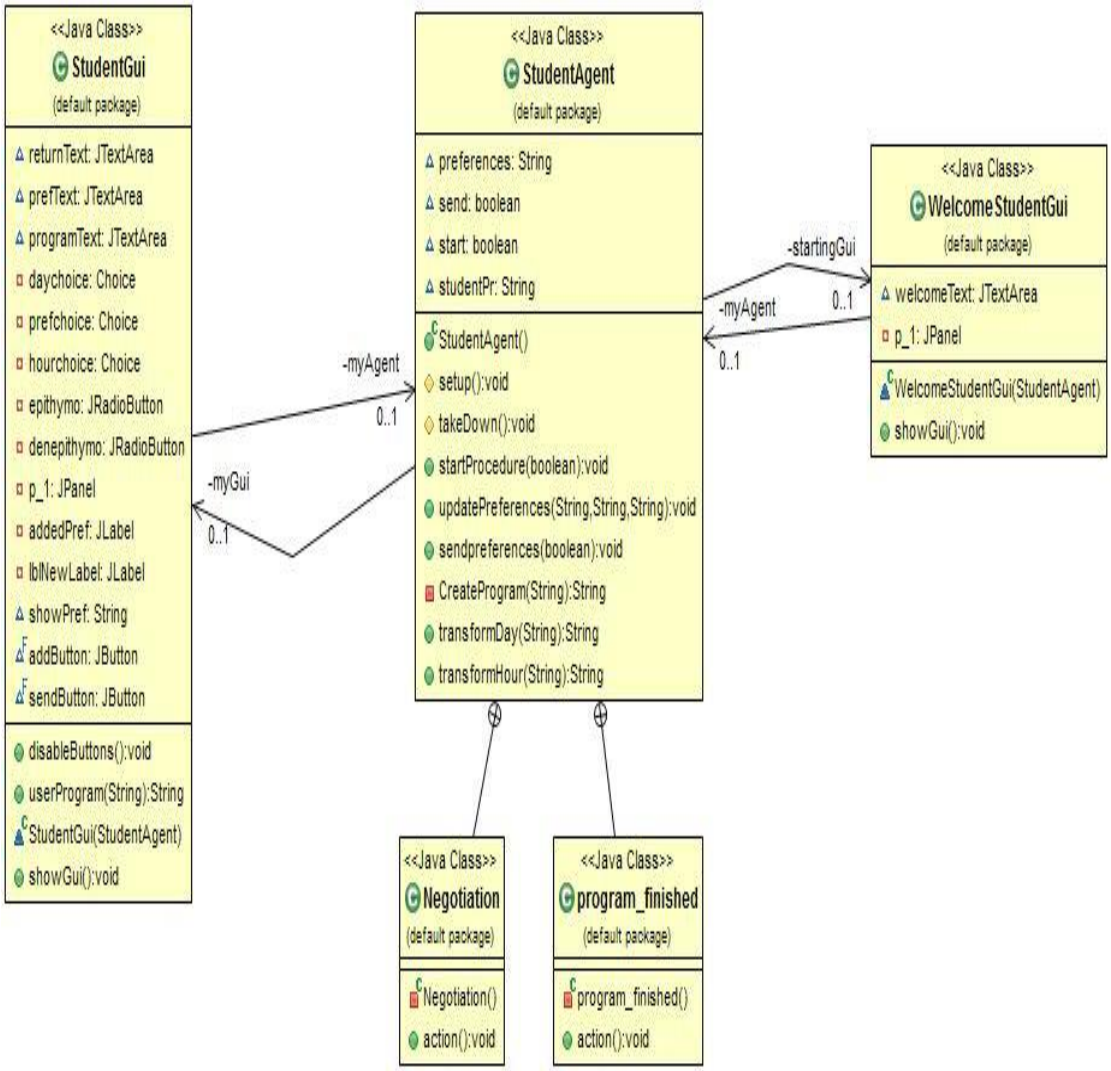
Β1. UML διάγραμμα κλάσεων ***ProgramAgent, Program, PreferencesHandler***



B2. UML διάγραμμα κλάσεων TeacherAgent, TeacherGui, WelcomeTeacherGui, ProposalsGui



B3. UML διάγραμμα κλάσεων StudentAgent, StudentGui, WelcomeStudentGui



Παράρτημα Γ

Ερωτηματολόγιο χρηστών για το Τεστ Ευχρηστίας

Παρακαλούμε συμπληρώστε με Χ το ακόλουθο ερωτηματολόγιο:

Υπήρχαν στο περιεχόμενο της εφαρμογής γραμματικά λάθη;	Καθόλου	Λίγα	Αρκετά	Πολλά
Ήταν όλα τα μεγέθη της γραμματοσειράς ίδια;	Ναι		Όχι	
Ήταν το κείμενο σωστά στοιχισμένο;	Ναι		Όχι	
Ήταν όλα τα κουμπιά με ίδια μορφοποίηση και μέγεθος;	Ναι		Όχι	
Εμφανίζονταν ενημερωτικό μήνυμα για οποιαδήποτε λειτουργία πρόκειται να εκτελέσετε;	Ναι		Όχι	
Σας δημιουργήθηκαν απορίες για τον τρόπο χρήσης της εφαρμογής	Καθόλου	Λίγες	Αρκετές	Πολλές
Μπορεί ο χρήστης να λειτουργήσει την εφαρμογή χωρίς να την ματαιώσει;	Ναι		Όχι	
Όταν χρειαζόταν εμφανιζόταν μπάρα κύλισης;	Ναι		Όχι	
Μπορέσατε να αποστείλετε τις προτιμήσεις που επιθυμούσατε;	Καθόλου	Λίγο	Αρκετά	Πολύ
Το ωρολόγιο πρόγραμμα που δημιουργήθηκε ανταποκρίνονταν στις προσδοκίες σας;	Καθόλου	Λίγο	Αρκετά	Πολύ
Ήταν εύχρηστη η εφαρμογή	Καθόλου	Λίγο	Αρκετά	Πολύ