ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ
ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
ΚΑΤΕΥΘΥΝΣΗ: «ΟΡΓΑΝΩΣΗ & ΔΙΟΙΚΗΣΗ»

Ανάπτυξη ενός Συστήματος Πολλαπλών Πρακτόρων Μοντελοποίησης και Ομαδοποίησης Χρηστών και Αντικειμένων με χρήση Πολυκριτήριων Μεθοδολογιών

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

Υπό τον

ΓΕΩΡΓΙΟ ΚΟΤΟΠΟΥΛΟ

Χανιά, Ιούλιος 2007

Η διατριβή του Γεωργίου Κοτόπουλου εγκρίνεται

Ματσατσίνης Νικόλαος	
Ζοπουνίδης Κωνσταντίνος	
Γρηγορούδης Ευάγγελος	

Ο Άπειρος κύκλος της ιδέας και της πράξης,

Της άπειρης επινόησης, του άπειρου πειράματος,

Φέρνει τη γνώση της κίνησης, μα όχι της ηρεμίας·

Τη γνώση του λόγου, μα όχι της σιωπής·

Τη γνώση των λέξεων και την άγνοια της Λέξης.

T.S. Eliot Χωρικά από τον «Βράχο», 1934

ΠΕΡΙΛΗΨΗ

Τα πληροφοριακά συστήματα χρησιμοποιούν σήμερα σε μεγάλο βαθμό το προφίλ των χρηστών (User Profiles) για την παροχή προσωποποιημένων υπηρεσιών. Τα προφίλ πρέπει να αντικατοπτρίζουν σε κάθε χρονική στιγμή τις τρέχουσες προτιμήσεις των χρηστών. Επειδή όμως οι προτιμήσεις αυτές αλλάζουν στη διάρκεια του χρόνου λόγω ψυχολογικών, κοινωνιολογικών, προτεραιοτήτων και άλλων παραγόντων, τα προφίλ πρέπει να μπορούν να προσαρμόζονται σε συνεχή βάση στις καινούργιες ανάγκες-απαιτήσεις των χρηστών. Εάν, λοιπόν, ένας πράκτορας (agent) παρακολουθεί την συμπεριφορά των χρηστών τότε θα είναι σε θέση να προσαρμόσει το προφίλ τους σύμφωνα με τις τρέχουσες προτιμήσεις τους. Επιπλέον, τα προφίλ μπορούν να χρησιμοποιηθούν από πράκτορες για να διαμορφώνουν ομάδες χρηστών που μοιράζονται κοινά ενδιαφέροντα, οι οποίες ομάδες μπορούν στη συνέχεια να χρησιμοποιηθούν για εξελιγμένες υπηρεσίες πληροφόρησης, διαφήμισης και άλλα. Για την υποστήριξη αυτών των διαδικασιών έχουν προταθεί διάφοροι αλγόριθμοι στην διεθνή βιβλιογραφία.

Στην εργασία αυτή αναζητήθηκε η λύση μέσω της εφαρμογής πολυκριτήριων μεθοδολογιών, εισάγοντας τη λογική της χρησιμότητας στη διαμόρφωση των προφίλ τόσο των χρηστών-πρακτόρων όσο και των αντικειμένων-πρακτόρων. Έτσι, έχει αναπτυχθεί ένα σύστημα πολλαπλών πρακτόρων το οποίο μοντελοποιεί τις προτιμήσεις των χρηστών, δημιουργώντας έναν ευφυή πράκτορα-χρήστη, το προφίλ του οποίου θα προσαρμόζεται ανάλογα με τη συμπεριφορά του. Τα προφίλ τους θα μεταβάλλονται ανάλογα με τις προτιμήσεις, την εμπειρία (τι άρθρα διαβάζει, πόσο συχνά, …). Παράλληλα μοντελοποιήθηκαν και τα αντικείμενα, αποκτώντας το δικό τους προφίλ, και συγκεκριμένα κείμενα και άρθρα τα οποία θα εκπροσωπούνται από έναν πράκτορα-έντυπο. Τα προφίλ τους θα μεταβάλλονται ανάλογα με τη χρήση τους και το ποιοι τα χρησιμοποιούν (ποιοι τα διαβάζουν, πόσο συχνά, …). Τέλος, μοντελοποιήθηκαν και υλοποιήθηκαν διαδικασίες χρησιμοποιώντας των προφίλ των χρηστών όπως: αξιολόγηση, αναζήτηση και διαφήμιση αντικειμένων, ομαδοποίηση πρακτόρων και άλλες.

Για ανάγκες επίδειξης και αξιολόγησης θα υλοποιηθεί μια συγκεκριμένη εφαρμογή στο πεδίο της ακαδημαϊκής κοινότητας στο πεδίο των επιστημονικών άρθρων.

ΕΥΧΑΡΙΣΤΙΕΣ

Ο συγγραφέας θα ήθελε να ευχαριστήσει τον καθηγητή κ. Νικόλαο Ματσατσίνη για την επίβλεψη και καθοδήγησή του κατά τη διάρκεια της εκπόνησης αυτής της μεταπτυχιακής διατριβής.

Θα ήθελε, επίσης, να ευχαριστήσει τους αναγνώστες της διατριβής καθηγητή κ. Κωνσταντίνο Ζοπουνίδη και λέκτορα κ. Ευάγγελο Γρηγορούδη για τον χρόνο που αφιέρωσαν στην ανάγνωσή της και για τις παρατηρήσεις τους.

Τέλος, ευχαριστεί τους φίλους, συνοδοιπόρους και συμφοιτητές Γιάννη Κοτόπουλο, Νατάσα Καραναστάση και Μαρία Φραντζή για την αμέριστη συμπαράστασή τους στη διαδικασία της εκπόνησής της.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΙΝΑΚΕΣ

ΣΧΗΜΑΤΑ

ΚΕΦΑΛΑΙΟ Α' - ΕΙΣΑΓΩΓΗ

ΑΝΑΓΚΑΙΟΤΗΤΑ

Παρόλο που η χρήση της γνώσης δεν είναι κάτι νέο, η αυξανόμενη ένταση της στην οικονομική διαδικασία είναι πλέον ασυγκράτητη. Η οικονομία μεταβάλλεται σταδιακά από αγροτική, τον 18ο αιώνα, σε βιομηχανική, τον 19ο και 20ο, και τελικά, σήμερα, σε γνωσιακή με κύρια χαρακτηριστικά, πλέον των άλλων, την γνώση και την καινοτομία, καθιστώντας, έτσι, την επιτυχία στο σύγχρονο αυτό περιβάλλον να συναρτάται απαραιτήτως από την γνώση, την δια βίου μάθηση και την ανθρώπινη δημιουργικότητα. Κάποια από τα παράδοξα της γνώσης, όπως τα αναφέρει ο καθ. Ron Johnston, είναι ότι η χρησιμοποίηση της γνώσης δεν σημαίνει απαραίτητα ότι αυτή αναλώνεται, το να την μεταφέρεις δεν σημαίνει ότι την χάνεις, παρόλο που είναι άφθονη, η δυνατότητα χρήσης της είναι ανεπαρκής, η παραγωγή της αντιστέκεται στην μαζικοποίηση, και τέλος το μεγαλύτερο μέρος από αυτήν ξεχνιέται στο τέλος της μέρας.

Επιπλέον χαρακτηριστικό της σύγχρονης κοινωνίας είναι η μεταβολή των καναλιών διάχυσης της γνώσης. Η κοινωνία της πληροφορίας, λοιπόν, είναι πραγματικότητα με όλο και περισσότερους ανθρώπους από όλο τον κόσμο να διαμοιράζουν πληροφορία μέσω ηλεκτρονικών μορφών. Παρόλο, λοιπόν, που η πρόσβαση στην πληροφορία είναι σε μεγάλο βαθμό ελεύθερη και η δυνατές επιλογές άφθονες, ουσιαστικά οδηγούμαστε σε μία έλλειψη πληροφόρησης λόγω ανικανότητας επιλογής του σωστού μέσου πληροφόρησης. Ελεύθερη επιλογή του μέσου δεν μπορεί να υπάρξει εκεί που ο όγκος των επιλογών είναι χαοτικός, ενώ διαφαίνεται οι επιλογές να αυξάνονται και να αυξάνονται με μεγαλύτερη ακόμα ένταση στο μέλλον.

Στη διαδικασία της δια βίου μάθησης όλοι οι άνθρωποι πρέπει να πληροφορούνται και να ενημερώνεται περισσότερο από επαρκώς για να μπορέσουν να παραμείνουν ανταγωνιστικοί στην οικονομία της γνώσης. Χρειάζονται δηλαδή τα μέσα εκείνα που θα διακρίνουν και θα διατάσσουν ένα σύνολο από εναλλακτικές σε αυτές που είναι δυνατόν να τους φανούν περισσότερο χρήσιμες. Ο κύριος στόχος αυτής της διατριβής είναι να προσφέρει αυτό το μέσο στο πεδίο των επιστημονικών άρθρων.

Τα επιστημονικά άρθρα σε περιοδικά, βιβλία ή διαδίκτυο αυξάνονται με φρενήρεις ρυθμούς καθιστώντας την επιλογή του κατά περίπτωση καταλληλότερου για ενημέρωση τυχαία διαδικασία. Σκοπός μας είναι να μειώσουμε την τυχαιότητα εισάγοντας προφίλ και ιστορικό

χρήσης των αναγνωστών. Με την αξιοποίηση αυτών των δύο μπορούμε να βελτιώσουμε την ικανοποίηση των αναγνωστών ενεργώντας για την ουσιαστικότερη απόκτηση γνώσης.

Επιπλέον, το πεδίο των επιστημονικών άρθρων μπορεί πολύ εύκολα να διευρυνθεί σε κάθε αντικείμενο το μόνο που χρειάζεται να αλλάζει κάθε φορά είναι τα χαρακτηριστικά των αντικειμένων και το προφίλ του χρήστη, αλλά τίποτα από την μεθοδολογία. Οπότε κατά την ανάλυση, τον σχεδιασμό και την υλοποίηση του συστήματος θα ληφθεί υπόψη η διάκριση ανάμεσα στην συγκεκριμένη εφαρμογή για τα επιστημονικά άρθρα και το γενικό μοντέλο του συστήματος.

Πιο συγκεκριμένα θα μελετήσουμε την εφαρμογή της πολυκριτήριας μεθοδολογίας προστιθέμενης χρησιμότητας UTA* σε συνδυασμό με την μεθοδολογία από την περιοχή της ανάκτησης πληροφορίας (Information Retrieval -IR) p-norm στο περιβάλλον πολλαπλών ευφυών πρακτόρων οι οποίοι συνεργάζονται και αποφασίζουν αυτόνομα για τις καλύτερες επιλογές προς τους αναγνώστες.

ΣΤΟΧΟΙ ΤΗΣ ΕΡΓΑΣΙΑΣ

Κύριος στόχος της εργασίας, όπως αναφέρθηκε και στην προηγούμενη παράγραφό, είναι η βελτίωση της ικανοποίησης αναγνωστών για δημοσιευμένα άρθρα με χρήση προφίλ και ιστορικό χρήσης (ανάγνωσης) των αναγνωστών. Η μεθοδολογία για την αξιοποίηση των δύο αυτών πυλώνων μπορεί να συμβάλει ουσιαστικά στην απόδοση του συστήματός μας.

Επιμέρους, μπορούμε να διακρίνουμε τους ακόλουθους στόχους:

- Τον καθορισμό της πληροφορίας που θα διαχειριζόμαστε για ένα επιστημονικό άρθρο, αναφέρεται και ως μετα-πληροφορία του άρθρου. Εδώ το αποκαλούμε προφίλ ενός άρθρου.

- Τον καθορισμό των στοιχείων που απαρτίζουν το προφίλ (τις προτιμήσεις) ενός αναγνώστη.

- Τον καθορισμό των στοιχείων που απαρτίζουν το ιστορικό χρήσης. Μπορεί να μην είναι απλά τα άρθρα που διάβασε αλλά και σε τι βαθμό τον ικανοποίησαν.

- Τον καθορισμό των κριτηρίων αναζήτησης άρθρων και των κριτηρίων κατάταξής τους.

- Τον καθορισμό της λειτουργικότητας του συστήματος.

Όσον αφορά στη μεθοδολογία για την αξιοποίηση της πληροφορίας του συστήματος διακρίνουμε τους παρακάτω στόχους:

- Τον σχεδιασμό και την υλοποίηση ενός μηχανισμού αποκοπής των άρθρων με βάση κάποια κριτήρια που συνιστάτε σε μεθοδολογίες από την περιοχή της Ανάκτησης Πληροφορίας (Information Retrieval - IR) και συγκεκριμένα στην μεθοδολογία ασαφούς λογικής (fuzzy logic) p-norm (πι νόρμ).

- Τον σχεδιασμό και την υλοποίηση ενός μηχανισμού κατάταξης των άρθρων με βάση το προφίλ του και το ιστορικό χρήσης. Αυτός ο μηχανισμός βασίζεται σε μεθοδολογίες Υποστήριξης Αποφάσεων (Decision Making/Support) και συγκεκριμένα στην πολυκριτήρια μεθοδολογία προστιθέμενης χρησιμότητας UTA*.

- Τον σχεδιασμό και την υλοποίηση ενός συστήματος και μιας γραφικής διεπαφής χρηστών (graphical user interface) που θα προσφέρει την απαιτούμενη λειτουργικότητα. Συγκεκριμένα, το σύστημα θα βασίζεται σε περιβάλλον πολλαπλών πρακτόρων οι οποίοι θα ενεργούν αυτόνομα για να προσφέρουν υπηρεσία στους χρήστες. Το γραφικό εργαλείο βασίζετε σε τεχνολογίες διαδικτύου ώστε το συνολικό σύστημα να είναι προσβάσιμο από παντού.

ΔΟΜΗ ΤΗΣ ΕΡΓΑΣΙΑΣ

Στο δεύτερο κεφάλαιο θα αναφερθούμε στις τεχνολογίες που χρησιμοποιήθηκαν στην διατριβή αυτή και σε σχετικές εργασίες που έχουν αναπτυχθεί αλλού.

Στο τρίτο κεφάλαιο θα γίνει ανάλυση των απαιτήσεων του συστήματός μας με έμφαση στα δεδομένα που διατηρούνται από το σύστημα, στους μηχανισμούς αξιοποίησής τους και στην λειτουργικότητα του συστήματος.

Στο τέταρτο κεφάλαιο παρουσιάζεται ο σχεδιασμός τόσο της μοντελοποίησης του προβλήματος όσο και του συστήματος.

Στο πέμπτο κεφάλαιο υπάρχει η υλοποίηση του συνολικού συστήματος που περιλαμβάνει την αρχιτεκτονική, το σύστημα πολλαπλών πρακτόρων, την γραφική εφαρμογή και τη μονάδα που επιλύει UTA* προβλήματα.

Στο έκτο και τελευταίο κεφάλαιο γίνετε μία ανακεφαλαίωση, αναφέρονται κάποια συμπεράσματα όπως επίσης και κάποιες επισημάνσεις για μελλοντικές επεκτάσεις.

ΚΕΦΑΛΑΙΟ Β' – ΤΕΧΝΟΛΟΓΙΕΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ - ΣΧΕΤΙΚΕΣ ΕΡΓΑΣΙΕΣ

Η ΜΕΘΟΔΟΣ UTA*

Η μέθοδος UTA* είναι ένα μοντέλο για κατάταξη επιλογών βασισμένο σε ένα σύνολο κριτηρίων και κάποιων διαταγμένων εναλλακτικών με βάση τις προτιμήσεις των χρηστών. Ο χρήστης ορίζει ένα σύνολο κριτηρίων και κατατάσσει κάποιες εναλλακτικές δίνοντας για κάθε μία από αυτές μία τιμή σε κάθε κριτήριο. Η μέθοδος επιλύοντας ένα γραμμικό πρόβλημα υπολογίζει πόσο σημαντικό είναι το κάθε κριτήριο για τον χρήστη καθώς και πώς επηρεάζει η τιμή κάθε κριτηρίου την συνολική κατάταξη. Το σύνολο αυτών των τιμών ονομάζονται μερικές χρησιμότητες, η διατήρηση των οποίων μας επιτρέπει να κατατάσσουμε τις καινούργιες επιλογές του χρήστη. Η μέθοδος αυτή προτάθηκε από τους Σίσκο και Γιαννακόπουλο στο [3] ενώ βασίστηκε στην προγενέστερη μέθοδο UTA [7]. Στις προηγούμενες αναφορές υπάρχει αναλυτική περιγραφή των μεθόδων. Τέλος ο χρήστης μπορεί να βρει επιπλέον πληροφορία για την υλοποίηση αυτής της μεθόδου στο πρώτο παράρτημα.

ΤΟ P-NORM ΜΟΝΤΕΛΟ

Τα κλασικά συστήματα ανάκτησης πληροφοριών (information retrieval systems) μοντελοποιούν τόσο αντικείμενα πληροφορίας όσο και τις ερωτήσεις ως πίνακες αποτελούμενους από στοιχεία (βλέπε [4] και [14][14] για λεπτομέρειες). Το σύνολο των αντικειμένων της πληροφορίας που ικανοποιούν μία ερώτηση περιλαμβάνει εκείνα τα αντικείμενα που περιέχουν οποιοδήποτε στοιχείο περιέχετε στον πίνακα της ερώτησης.

Αντίθετα, στην παραδοσιακή Δυαδική λογική (Boolean logic), μία ερώτηση αναπαριστάται ως μία δυαδική έκφραση που αποτελείται από αναπόσπαστα στοιχεία και τους τρεις δυαδικούς τελεστές (AND, OR, NOT) ως συνδέσμους. Το σύνολο των αντικειμένων πληροφορίας που συνιστούν την απάντηση σε μία ερώτηση ικανοποιούν την δυαδική έκφραση. Το ελάττωμα αυτής της προσέγγισης είναι ότι ένα αντικείμενο πληροφορίας θεωρείται είτε σχετικό (relevant) είτε μη-σχετικό (non-relevant) ως προς μία δεδομένη ερώτηση. Δεν υφίσταται, ως εκ τούτου, μέθοδος κατάταξης των αντικειμένων αναφορικά προς τον βαθμό που ταιριάζουν στην ερώτηση.

Για να ξεπεραστεί το παραπάνω μειονέκτημα, το Εκτεταμένο Δυαδικό Μοντέλο (Extended Boolean Model) εισήχθη στην βιβλιογραφία της ανάκτησης πληροφορίας (information retrieval). Αυτό το μοντέλο είναι γενίκευση της δυαδικής λογικής και στηρίζεται στη θεωρία των ασαφούς συνόλων (fuzzy set theory). Μας παρέχει με τύπους για τον υπολογισμό περίπλοκων δυαδικών εκφράσεων ούτως ώστε στα αντικείμενα πληροφορίας να δίδετε μια βαθμολογία στο διάστημα [0,1] αντί μίας απλής δυαδικής απάντησης (true/false). Ποικίλες μελέτες (όπως [1] και [2]) αποδεικνύουν την ανωτερότητα της απόδοσής του σε σύγκριση με τις παραδοσιακές μεθόδους ανάκτησης πληροφοριών. Παρακάτω περιγράφεται ένας ορισμός του μοντέλου, έτσι όπως παρουσιάστηκε στην [1], κατάλληλα διασκευασμένο για τις ανάγκες της εργασίας:

Ορισμός: Weighted Fuzzy Information Retrieval System (WFIRS):

Ένα WFIRS C ορίζεται σαν μία τετράδα $C = \langle F, I, Q, E \rangle$ όπου:

F είναι ένα σύνολο χαρακτηριστικών (features) που χρησιμοποιούνται για να περιγράφουν αντικείμενα πληροφορίας και να σχηματίζουν ερωτήσεις.

I είναι ένα σύνολο αντικειμένων πληροφορίας (information items). Κάθε αντικείμενο πληροφορίας αντιστοιχεί σε μία συνάρτηση από τα F στο διάστημα [0,1]. Αυτό ισοδυναμεί με την έκφραση $I \subseteq [0,1]^F$.

Q είναι ένα σύνολο ερωτήσεων (queries) που ταυτοποιούνται από το σύστημα. Ανταποκρίνονται σε δυαδικές εκφράσεις που χρησιμοποιούν χαρακτηριστικά (F) σαν άτομα και τους τελεστές AND, OR, NOT ως συνδέσμους. Επιπρόσθετα κάθε υπό-ερώτηση συσχετίζεται με ένα σχετικό βάρος από το διάστημα [0,1] που αντιπροσωπεύει τη σχετική "βαρύτητά" του αναφορικά προς τις υπόλοιπα υπό-ερωτήσεις. Παραδείγματα έγκυρων ερωτήσεων είναι: $\langle q_1, w_1 \rangle AND \langle q_2, w_2 \rangle$ και $\langle q_1, w_1 \rangle OR \langle q_2, w_2 \rangle$.

E είναι μία συνάρτηση αξιολόγησης (evaluation) $E : Q \times I \to [0,1]$ που δίνει μία τιμή στο [0,1] σε κάθε έγκυρη ερώτηση από το Q όσον αναφορά κάθε αντικείμενο πληροφορίας. Έτσι, $E(q,i)$ είναι η κατάταξη ομοιότητας του αντικειμένου πληροφορίας i όσον αναφορά την

ερώτηση q. Ε ορίζεται έτσι αναδρομικά στηριγμένο σε συναρτήσεις αξιολόγησης των λογικών (logical) τελεστών:

$$E\left(\langle q_1, w_1\rangle AND\langle q_2, w_2\rangle, i\right) = f_{AND}\left(\langle E(q_1, i), w_1\rangle, \langle E(q_2, i), w_2\rangle\right)$$

$$E\left(\langle q_1, w_1\rangle OR\langle q_2, w_2\rangle, i\right) = f_{OR}\left(\langle E(q_1, i), w_1\rangle, \langle E(q_2, i), w_2\rangle\right)$$

$$E\left(NOTq_1, i\right) = f_{NOT}\left(E(q_1, i)\right)$$

$E(f, i) = i(f)$ (παρατηρήστε ότι i είναι μία συνάρτηση από το F στο [0,1]).

Για μία ερώτηση $q \in Q$, το υποσύνολο ANS(q) του I που περιέχει όλα τα στοιχεία του I για τα οποία η συνάρτηση αξιολόγησης F έχει θετική (>0) τιμή, είναι απάντηση στο q. Αυτό το υποσύνολο ορίζεται ως:

$$ANS(q) = \left\{\langle i, E(q, i)\rangle \in I \times (0,1] \,\middle|\, E(q, i) > 0\right\}.$$

Παρατηρήστε ότι στην πράξη ενδέχεται να πάρουμε πάρα πολλές πιθανές απαντήσεις, τις περισσότερες με πολύ μικρές τιμές. Σ' αυτήν την περίπτωση μπορούμε να θεωρήσουμε μόνο τις τιμές με E(q,I)>Emin, για κάποιο κατάλληλο για την εφαρμογή Emin, ή να πάρουμε τα N πρώτα αντικείμενα με τις υψηλότερες καταχωρήσεις, πάλι για κάποιο κατάλληλο N.

Για να αξιολογήσει κανείς τις ερωτήσεις που αναγνωρίζονται από ένα WFIRS, πρέπει να του δοθούν οι συναρτήσεις αξιολόγησης fNOT, fAND, και fOR. Υπάρχει μία πληθώρα ορισμών τέτοιων συναρτήσεων. Ο παρακάτω πίνακας παρουσιάζει τους ορισμούς που αντιστοιχούν στο p-norm Extended Boolean Model, το οποίο χρησιμοποιείται από το σύστημα. Παρατηρήστε ότι οι συναρτήσεις fAND, και fOR είναι n-αδικές αντί δυαδικές. Αυτό οφείλεται στο γεγονός ότι δεν είναι μεταβατικές.

$f_{AND}\left((a_1,w_1),\ldots,(a_n,w_n)\right)$	$f_{OR}\left((a_1,w_1),\ldots,(a_n,w_n)\right)$	$f_{NOT}(a)$
$1-\left(\frac{\sum_{i=1}^{n}(1-a_i)^p \cdot w_i^p}{\sum_{i=1}^{n} w_i^p}\right)^{1/p} \quad 1 \le p \le \infty$	$\left(\frac{\sum_{i=1}^{n} a_i^p \cdot w_i^p}{\sum_{i=1}^{n} w_i^p}\right)^{1/p} \quad 1 \le p \le \infty$	$1-a$

Πίνακας 1: Σταθμητές (Weighted) n-αδικές βασικές συναρτήσεις αξιολόγησης του p-Norm Extended Boolean Model.

ΤΟ ΠΟΛΥ-ΠΡΑΚΤΟΡΙΚΟ ΠΕΡΙΒΑΛΛΟΝ

Ένας πράκτορας είναι ένα υπολογιστικό σύστημα που είναι τοποθετημένο σε κάποιο περιβάλλον και που είναι ικανό για αυτόνομη δράση σε αυτό το περιβάλλον ώστε να επιτευχθούν οι προσχεδιασμένοι στόχοι του [20]. Ο όρος "πράκτορας" χρησιμοποιείται για να περιγράψει ένα εξαιρετικά ευρύ φάσμα συστημάτων. Αυτά τα συστήματα κυμαίνονται από απλά συστήματα (π.χ. Hewlett-Packard's New Wave) ως περίπλοκης, τεχνητής νοημοσύνης συστήματα που χρησιμοποιούν την συμπερασματική διαδικασία και την οργάνωση-προγραμματισμό [21].

Αν και υπάρχει μεγάλη συζήτηση για ακριβώς τι ένας πράκτορας είναι, θεωρούνται οι εξής απαραίτητοι όροι για το περιβάλλον πρακτόρων [22]:

(i) *αυτονομία* - μπορεί να δράσει χωρίς την άμεση επέμβαση άλλων και έχει τον έλεγχο των δράσεων του και του εσωτερικής κατάστασής του

(ii) *ανταπόκριση* - μπορεί να αντιδράσει σε ένα έγκαιρο χρονικό διάστημα στις περιβαλλοντικές αλλαγές

(iii) *ενεργητικότητα και προνοητικότητα* - μπορεί να πάρει την πρωτοβουλία όπου απαιτείται και

(iv) *κοινωνική δυνατότητα* - μπορεί να αλληλεπιδράσει για να ολοκληρώσει την επίλυση του προβλήματός του και για να βοηθήσει άλλους [22].

Τα τελευταία χρόνια η έννοια του πράκτορα και ειδικότερα του ευφυή πράκτορα αποδεικνύεται όλο και πιο σημαντική για διάφορα ερευνητικά πεδία. Οι ευφυείς πράκτορες είναι σε θέση να αντιληφθούν το περιβάλλον τους και να ανταποκριθούν (έγκαιρα) στις αλλαγές που εμφανίζονται σε αυτό. Εμφανίζουν συμπεριφορά κατευθυνόμενη από τον στόχο, και είναι σε θέση να συνεργάζονται με άλλους πράκτορες (και ενδεχομένως και ανθρώπους) [20].

Οι ευφυείς πράκτορες υποτίθεται ότι συνήθως εκθέτουν αυτόνομη συμπεριφορά που καθορίζεται από [23]:

• την προνοητικότητα, σημαίνει να πάρουν την πρωτοβουλία να ικανοποιηθούν οι δεδομένοι στόχοι σχεδίου και να εκτεθεί η κατευθυνόμενη από τον στόχο συμπεριφορά,

• οι αντιδραστικές δράσεις, σημαίνει αντίληψη του περιβάλλοντος και έγκαιρη διαχείριση αλλαγής για να επιτύχουν τους δεδομένους στόχους σχεδίου, και

• κοινωνική συνεργασία σε ομάδες με άλλους πράκτορες ή/και ανθρώπινους χρήστες όταν απαιτείται [23].

Οι ευφυείς πράκτορες πληροφοριών είναι αυτόνομες υπολογιστικές οντότητες λογισμικού που έχουν πρόσβαση σε πολλαπλές, ετερογενείς πηγές στοιχείων και πληροφοριών, και προνοούν, μεσολαβούν, και διατηρούν τις σχετικές πληροφορίες εξ ονόματος των χρηστών τους, ή άλλων πρακτόρων. Οι πράκτορες πληροφοριών προορίζονται ειδικά για (1) να παρέχουν μια δυναμική ανακάλυψη των πόρων, (2) για να επιλύσουν τη σύνθετη αντίσταση πληροφοριών των καταναλωτών και των προμηθευτών πληροφοριών, και (3) για να προσφέρουν προστιθεμένης αξίας υπηρεσίες πληροφοριών και προϊόντα. Αυτό περιλαμβάνει, ειδικότερα, ανάκτηση, αγορά, φιλτράρισμα, σύντηξη, και παρουσίαση σχετικών πληροφοριών στους αποφασίζοντες που τις ζητάνε και κατά προτίμηση στον συντομότερο χρόνο. Η κατάσταση γίνεται ακόμα πιο σύνθετη από τη στιγμή που οι πελάτες καθώς επίσης και τα προϊόντα, οι υπηρεσίες και η ποιότητα των εμπόρων μπορεί να αλλάξει γρήγορα κατά τη διάρκεια του χρόνου [23].

Οι περισσότεροι αυτόνομοι πράκτορες είναι τοποθετημένοι σε ένα κοινωνικό πλαίσιο και πρέπει να αλληλεπιδράσουν με άλλους πράκτορες (και ανθρώπινους και τεχνητούς) για να ολοκληρώσουν τους στόχους του προβλήματός που επιλύουν. Τέτοιοι πράκτορες είναι συνήθως σε θέση να εφαρμόσουν ένα ευρύ φάσμα δράσεων και να εμφανίσουν ποικίλες κοινωνικές αλληλεπιδράσεις. Αντιμέτωπος με αυτήν την ποικιλία των επιλογών, ένας πράκτορας πρέπει να αποφασίσει τι να κάνει. Υπάρχουν πολλές πιθανές συναρτήσεις λήψης απόφασης που θα μπορούσαν να υιοθετηθούν για να κάνουν την επιλογή. Κάθε τέτοια συνάρτηση θα έχει μια διαφορετική επίδραση στην επιτυχία του μεμονωμένου πράκτορα και του γενικού συστήματος στο οποίο είναι τοποθετημένη [22].

ΣΧΕΤΙΚΕΣ ΕΡΓΑΣΙΕΣ

BIBSTER

Το Bibster [9] είναι ένα βασισμένο σε Java σύστημα για να βοηθάει ερευνητές να διαχειριστούν, να αναζητούν, και να διαμοιράζονται βιβλιογραφικά μεταδεδομένα (όπως του BibTex) σε ένα peer-to-peer δίκτυο. Το πλεονέκτημα του συστήματος είναι ότι παρέχει την δυνατότητα αναζήτησης σε κατανεμημένα peer-to-peer δίκτυα χρησιμοποιώντας τεχνολογίες από το Semantic Web, ενώ επιτρέπει τον εύκολο διαμοιρασμό δεδομένων μεταξύ ερευνητών.

Η διαφορά του Bibster από το σύστημα που αναπτύχθηκε εδώ είναι ότι αυτό επικεντρώνετε στον συσχετισμό μεταδεδομένων χωρίς να χειρίζεται ανάδραση από τους χρήστες ενώ εμείς ενδιαφερόμαστε για την σωστή επεξεργασία της ανάδρασης με στόχο την διαμόρφωση γνώσης για τον χρήστη ώστε να βελτιώνουμε με το προφίλ του τα αποτελέσματα.

Η ΕΥΡΩΠΑΪΚΗ ΨΗΦΙΑΚΗ ΒΙΒΛΙΟΘΗΚΗ

Ο στόχος της ευρωπαϊκής ψηφιακής βιβλιοθήκης (The European Library – TEL [10]) είναι να δημιουργήσει μία συνεργατική πλατφόρμα και να ορίσει ένα σύστημα για ενοποιημένη πρόσβαση στις μεγαλύτερες συλλογές από τις ευρωπαϊκές βιβλιοθήκες. Αυτό πραγματοποιήθηκε με την εφαρμογή μιας νέας τεχνικής για αναζήτηση και ανάκτηση μέσω URL μαζί με ένα νέο πρότυπο μεταδεδομένων.

Το μοντέλο μεταδεδομένων που χρησιμοποιείται από το TEL είναι ένα Dublin Core Application Profile, ενώ έχει ληφθεί υπόψη ότι οι λειτουργικές ανάγκες θα αλλάζουν με τον χρόνο όποτε το μοντέλο θα πρέπει να μπορεί να εξελίσσετε με ελεγχόμενο τρόπο. Αυτό το παρέχουν μέσω μιας κεντρικής βάσης μεταδεδομένων που περιέχει όλα τα χαρακτηριστικά των μεταδεδομένων στο TEL.

Εμείς εδώ αντιμετωπίζουμε το ίδιο πρόβλημα με διαφορετική προσέγγιση εισάγοντας την έννοια του προφίλ και της ανάδρασης με το σύστημα, ενώ μία επέκταση για την χρήση ενός πρότυπου μοντέλου μεταδεδομένων με δυνατότητες επέκτασης θα είχε ενδιαφέρον.

WEBMATE

Το WebMate [11] είναι ένας πράκτορας που βοηθάει τους χρήστες του διαδικτύου να αναζητούν και να πλοηγούνται σε ιστοσελίδες. Συγκεκριμένα, μπορεί να κάνει παράλληλες

αναζητήσεις σε πολλές μηχανές αναζήτησης, να βελτιώνει τις λέξεις κλειδιά με ποιο σχετικές. Χρησιμοποίει ανάδραση από τους χρήστες, ενώ μαθαίνει για τα ενδιαφέροντά τους, προτείνει νέα URLs με βάση το προφίλ τους και επιλεγμένους πόρους και, τέλος, στέλνει σε φίλους τις ιστοσελίδες που πλοηγούνται οι χρήστες. Επίσης, κάνει διάφορα ακόμα που δεν θα μας απασχολήσουν εδώ.

Το WebMate κάνει πάρα πολλά από αυτά που κάνουμε κι εμείς αλλά για το πεδίο των ιστοσελίδων χωρίς όμως να χρησιμοποίει καθόλου μεταδεδομένα αλλά μόνο τις ίδιες τις ιστοσελίδες.

ΈΝΑ ΠΟΛΥ-ΠΡΑΚΤΟΡΙΚΟ ΣΥΣΤΗΜΑ ΣΥΣΤΑΣΕΩΝ ΓΙΑ ΤΑΙΡΙΑΣΜΑΤΑ

Ο Leonard Foner στο [12] περιγράφει ένα σύστημα ταιριάσματος σχεδιασμένο να βρίσκει ανθρώπους με κοινά ενδιαφέροντα και να τους γνωρίζει μεταξύ τους. Ο μηχανισμός έχει σχεδιαστεί για να γνωρίζει τους πάντες μεταξύ τους, σε αντίθεση με τα συμβατικά μέσα διαδικτύου που γνωστοί είναι μόνο όσοι αφιερώνουν χρόνο να μιλάνε. Οι πράκτορες που αποτελούν το σύστημα ταιριάσματος λειτουργούν σε αποκεντρωμένο τρόπο, ενώ μπορούν να ομαδοποιούνται σε συστοιχίες που αντικατοπτρίζουν τα ενδιαφέροντα των χρηστών. Αυτές οι συστοιχίες αργότερα χρησιμοποιούνται για να κάνουν νέες γνωριμίες ή να επιτρέψουν στους χρήστες να στείλουν μηνύματα σε άλλους που έχουν κοινά ενδιαφέροντα . Ο αλγόριθμος χρησιμοποιεί συστάσεις μεταξύ των πρακτόρων με παρόμοιο τρόπο όπως χρησιμοποιείται ο προφορικός λόγος για την αναζήτηση κάποιου ειδικού.

Το σύστημα παρουσιάζει πολλά κοινά με το δικό μας αλλά ενώ αυτό χρησιμοποιεί απλό κείμενο για να κατασκευάσει ομάδες, εμείς χρησιμοποιούμε μία δομή κριτηρίων, προφίλ , για να γνωρίσουμε μεταξύ τους πράκτορες και προσφέρουμε και μία σειρά άλλες λειτουργίες , όπως η διαχείριση αντικειμένων, η συσχέτισή τους με το προφίλ των χρηστών, η αναζήτηση με χρήση του προφίλ, αξιοποίηση της ανάδρασης του χρήστη, κλπ.

ΠΡΟΣΩΠΟΠΟΙΗΜΕΝΗ ΑΝΑΖΗΤΗΣΗ GOOGLE

Η προσωποποιημένη αναζήτηση είναι ένα μέρος της προσπάθειας της εταιρίας Google να βελτιώσουν την αναζήτηση των χρηστών. Αυτό που γίνεται είναι να συγκεντρώνονται όλες οι αναζητήσεις που κάνει κάποιος καθώς και τα αποτελέσματα που τελικά προτίμησε ο χρήστης. Ο χρήστης έχει την δυνατότητα να διαχειρίζεται αυτό τις αναζητήσεις του παρελθόντος. Επίσης, η Google προσφέρει το Google Scholar όπου τα επιστημονικά άρθρα από διάφορες

πηγές έχουν καταχωρηθεί χρησιμοποιώντας μεταδεδομένα . Ο συνδυασμός των δύο παραπάνω μαζί με τον κλασσικό αλγόριθμο κατάταξης της Google (page rank) μπορούν να έχουν πολύ καλά αποτελέσματα αν και το σύστημα είναι ακόμα υπό δοκιμή.

Η διαφορά είναι ότι, ενώ το Google για να κατατάξει κάποιο άρθρο βασίζεται μόνο στο πόσο σχετικό είναι με τα άρθρα που έχει ήδη αναζητήσει και δει ο χρήστης όσο αναφορά το περιεχόμενο, σε εμάς γίνετε η προσπάθεια, μέσω του μοντέλου UTA* και της αξιολόγησης, να υπολογίσουμε πόσο χρήσιμο θα είναι ένα άρθρο στον χρήστη.

ΚΕΦΑΛΑΙΟ Γ' - ΑΝΑΛΥΣΗ ΑΠΑΙΤΗΣΕΩΝ

Για να μπορέσει το σύστημα να προσφέρει προσωποποιημένες υπηρεσίες προς τους χρήστες για την χρήση οποιονδήποτε αντικειμένων χρειάζεται να ικανοποιούνται κάποιες συγκεκριμένες προϋποθέσεις. Αυτές τις προϋποθέσεις θα αναλύσουμε εδώ ενώ θα θέσουμε και την λειτουργικότητα που πρέπει να ικανοποιείται από την εφαρμογή όσο αναφορά της βασικές ιδιότητές του.

Συγκεκριμένα πρέπει να αναλυθούν και να περιγραφούν είδος της πληροφορίας που θα διατηρείται από την εφαρμογή μας και πώς αυτή θα αξιοποιείται για να προσφέρει προστιθέμενη αξία στους χρήστες. Τέλος, θα περιγραφεί σε γενικές γραμμές τι θα μπορεί να κάνει ο χρήστης με την εφαρμογή.

ΤΑ ΑΝΤΙΚΕΙΜΕΝΑ

Τα αντικείμενα μπορεί να ανήκουν σε διαφορετικές περιοχές σε κάθε ένα σύστημα (π.χ. ένα σύστημα αναφέρεται σε αρχεία μουσικής, ένα άλλο σε ιστοσελίδες, ένα τρίτο σε επιστημονικά άρθρα, κ.λπ.). Στην εργασία αυτή θα μελετήσουμε ένα σύστημα επιστημονικών άρθρων, με προδιαγραφή, όμως, την εύκολη επέκτασή του σε κάποιο άλλο. Αυτό που διαφοροποιεί τα συστήματα μεταξύ τους είναι, αφενός, τα χαρακτηριστικά που περιγράφουν ένα αντικείμενο (π.χ. άλλες πληροφορίες μας ενδιαφέρουν στα αρχεία μουσικής και άλλες στα επιστημονικά άρθρα). Αφετέρου δε, η διεπαφή (user interface) με την όποια αλληλεπιδρά ο χρήστης με το σύστημα, ώστε το σύστημα να συλλέγει τις πληροφορίες που χρειάζεται κ.λπ.

Κάθε αντικείμενο, λοιπόν, περιγράφεται από ένα σύνολο χαρακτηριστικών. Στην βιβλιογραφία αυτή η πληροφορία απαντάται άλλες φορές ως μετα-δεδομένα, άλλες φορές ως προφίλ, κ.λπ. Εδώ εμείς θα την ονομάσουμε προφίλ του αντικειμένου τα χαρακτηριστικά του όποιου δεν θα είναι ιεραρχικά αλλά θα είναι ενός επιπέδου και θα ονομάζονται κριτήρια. Αυτό γίνεται για να μπορούν, όπως θα φανεί παρακάτω, να εφαρμοστούν πάνω σε αυτά πολυκριτήριες μεθοδολογίες. Για κάθε κριτήριο θα υπάρχει μία βαθμολόγια που θα αναφέρετε στο πόσο σχετικό είναι το αντικείμενο προς αυτό το κριτήριο. Αυτή η βαθμολογία θα προκύπτει από με το αλγόριθμο UTA* [3] και ανάλογα με την χρήση που γίνετε στο αντικείμενο. Μία πρώτη ανάθεση τιμών μπορεί να γίνετε είτε από κάποιον χρήστη, είτε αυτοματοποιημένα με αλγορίθμους ανάλυσης του περιεχομένου τους (content analysis). Το πώς θα συλλέγεται αυτή η πληροφορία χρήσης και το ποιος και πως θα την διαχειρίζεται θα φανεί παρακάτω.

Τα αντικείμενα μπορούν να ομαδοποιούνται ανάλογα με τα χαρακτηριστικά τους.

ΟΙ ΧΡΗΣΤΕΣ

Κάθε χρήστης πρέπει να έχει ένα προφίλ το οποίο να διαμορφώνεται ανάλογα με την χρήση του συστήματος που κάνει ο χρήστης καθώς επίσης και απευθείας από τον ίδιο το χρήστη. Όμοια με τα αντικείμενα ένα προφίλ περιλαμβάνει ένα σύνολο από κριτήρια. Το σύνολο των κριτήριων ορίζει το πλήρες φάσμα και τα χαρακτηριστικά, κατηγοριοποίηση κλπ. της περιοχής (domain) που θέλουμε να καλύψουμε και ταυτίζεται με τα κριτήρια τον αντικείμενων. Όπως και με τα αντικείμενα έτσι και με τους χρήστες οι τιμές σε κάθε κριτήριο θα προσδιορίζονται με τον αλγόριθμο UTA* με βάση το ποια αντικείμενα κατανάλωσε και πόσο συχνά τα καταναλώνει. Η πληροφορία δηλαδή που έχουμε στο προφίλ αναφέρεται στο τι χαρακτηριστικά πρέπει να έχει ένα αντικείμενο για να αρέσει στον χρήστη.

Ένας χρήστης, αρχικά, θα πρέπει να εγγραφεί στο σύστημα. Κατά την διαδικασία αυτή ο χρήστης μπορεί να αρχικοποιεί το προφίλ του με κάποιες τιμές. Επίσης, ένας χρήστης μπορεί να «μπαίνει» στο σύστημα (logon) και να «βγαίνει» (logoff), καθώς επίσης, και να αλλάζει το προφίλ του και τα δεδομένα που έχουν καταχωρηθεί για αυτόν (βλέπε και παρακάτω).Τέλος, οι χρήστες μπορούν να ομαδοποιούνται με βάση τα προφίλ τους.

ΔΙΑΧΕΙΡΙΣΗ ΠΛΗΡΟΦΟΡΙΑΣ ΧΡΗΣΗΣ

Όπως έγινε φανερό από τα παραπάνω, η κύρια πληροφορία που μας χρειάζεται αναφέρεται στο ιστορικό χρήσης των αντικειμένων από τους χρήστες. Αυτή η πληροφορία μπορεί να κρατείτε είτε κατανεμημένη τόσο στους χρήστες όσο και στα αντικείμενα, είτε κεντρικά από το από το σύστημα. Η πληροφορία αυτή μπορεί να αλλάζει δυναμικά, π.χ. ένα το αντικείμενο Α το έχουν καταναλώσει οι χρήστες Χ1 και Χ2 με τα συγκεκριμένα προφίλ τους, με βάση τα οποία κατασκευάζεται το προφίλ του αντικείμενου. Στην περίπτωση που η πληροφορία χρήσης κρατάτε κατανεμημένα, όταν αλλάξουν τα προφίλ των χρηστών αυτών θα πρέπει να ενημερωθεί η πληροφορία του αντικείμενου Α ώστε να αλλάξει τις καταχωρήσεις για τα προφίλ των χρηστών που το έχουν καταναλώσει. Διαφορετικά θα έχει παλιά πληροφορία. Ένα άλλο χαρακτηριστικό της πληροφορίας αυτής είναι ότι πρέπει να αντέχει σε κατάρρευση του συστήματος (system crash), κ.λπ. κοντολογίς πρέπει να πληροί τις προϋποθέσεις που πληρούν οι βάσεις δεδομένων για ACID (atomicity, consistency, isolation and durability).

Για να ικανοποιούνται οι παραπάνω προϋποθέσεις και το σύστημα να λειτουργεί με ικανοποιητική απόδοση ελαχιστοποιώντας τις ενημερώσεις (updates) έχοντας την δυνατότητα να διατηρεί τα δεδομένα η πληροφορία θα κρατείτε κεντρικά σε μία βάση δεδομένων.

ΣΥΛΛΟΓΗ ΠΛΗΡΟΦΟΡΙΑΣ ΧΡΗΣΗΣ

Η συλλογή του ιστορικού χρήσης θα γίνεται από την διεπαφή (user interface) με τον χρήστη. Όπως είπαμε παραπάνω αυτή η διεπαφή εξαρτάται από το είδος των αντικειμένων στο όποιο αναφέρετε το σύστημα. Είναι σαφές ότι πρέπει η συλλογή και ανάλυση των δεδομένων να είναι όσο το δυνατόν πιο αδιαφανής στο χρήστη.

Στην εφαρμογή που εξετάζεται εδώ όπου τα αντικείμενα είναι επιστημονικά άρθρα και δεν υπάρχουν μηχανισμοί για αυτόματη επεξεργασία άρθρων και παραγωγή των μεταδεδομένων τους, ενώ η ανάπτυξη τέτοιων μηχανισμών ξεπερνάει τους στόχους αυτής της εργασίας, την εισαγωγή των δεδομένων των άρθρων θα αναλαμβάνουν οι ίδιοι οι χρήστες του συστήματος. Επίσης, το ιστορικό χρήσης δημιουργείται με την αξιολόγηση που παρέχει ο χρήστης για κάποιο αντικείμενο που έχει καταναλώσει.

ΚΑΤΑΣΚΕΥΗ ΚΑΙ ΠΡΟΣΑΡΜΟΓΗ ΠΡΟΦΙΛ

Όλη η παραπάνω πληροφορία συλλέγεται και διαχειρίζεται για να αξιοποιηθεί με δύο τρόπους. Αφενός, για την κατασκευή και προσαρμογή των προφίλ τόσο των χρηστών όσο και των αντικειμένων και, αφετέρου, για την προσφορά προσωποποιημένων υπηρεσιών προς τους χρήστες (σχετικά αντικείμενα, κ.λπ.) και για την προβολή (διαφήμιση) των αντικειμένων.

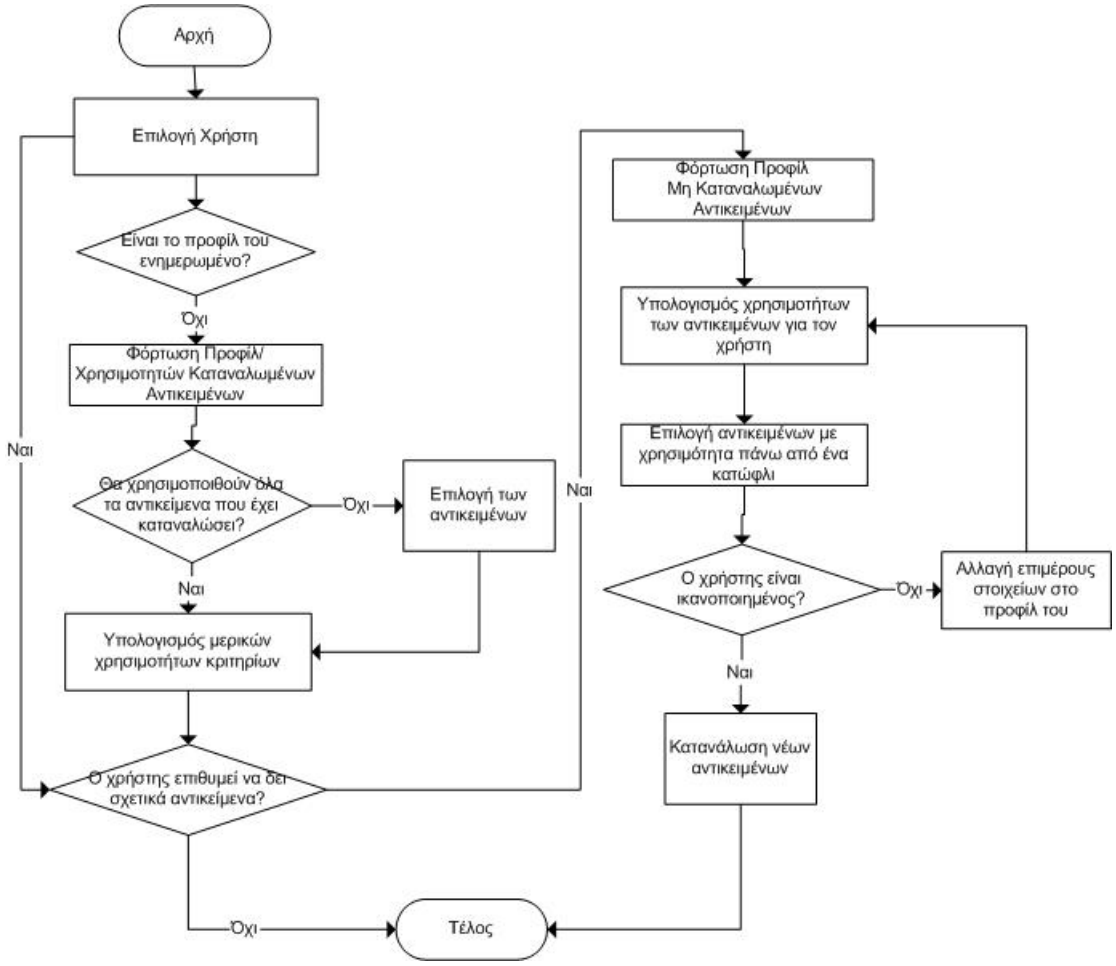
Η κατασκευή και προσαρμογή (creation and adaptation) των προφίλ γίνετε με την αξιοποίηση της γνώσης που υπάρχει από την συλλογή της πληροφορίας χρήσης. Σημειωτέον, δε, ότι παράλληλα με την πληροφορία αυτή, υπάρχει και γνώση αναφορικά στο πόσο συχνά ένα αντικείμενο χρησιμοποιείται από κάποιον χρήστη ή/και πόσο σχετικό θεωρεί ένα αντικείμενο ο χρήστης. Η μεθοδολογία προσαρμογής του προφίλ, όπως έχει ήδη αναφερθεί, θα χρησιμοποιεί των αλγόριθμο UTA*.

Η κατασκευή και προσαρμογή των προφίλ θα μπορούσε να γίνετε κάθε φορά που καταναλώνεται ένα αντικείμενο, αλλά αυτό θα δημιουργούσε προβλήματα απόδοσης. Έτσι, είναι προτιμότερο να υπάρχει μια μονάδα λογισμικού που θα αναλαμβάνει να ενημερώνει τα προφίλ ανάλογα με την παλαιότητα τους και τον φόρτο του συστήματος.

ΑΞΙΟΠΟΙΗΣΗ ΠΡΟΦΙΛ

Τόσο για την παροχή προσωποποιημένων υπηρεσιών προς τους χρήστες, όσο και για την δυνατότητα προβολής αντικειμένων προς τους ενδιαφερόμενους χρήστες θα γίνεται αξιοποίηση των προφίλ των χρηστών και των αντικειμένων. Ο αλγόριθμος έχει να κάνει με την εύρεση της ολικής χρησιμότητας των αντικειμένων που δεν έχει καταναλώσει ο χρήστης ως προς το προφίλ του χρήστη αυτού. Αν αυτή η ολική χρησιμότητα ξεπερνάει ένα κατώφλι τότε το αντικείμενο ενδιαφέρει τον χρήστη. Αυτό, από αρχική σκοπιά ακούγεται απλό, αλλά αν αναλογιστεί κανείς ότι πρέπει να υπολογίσει τις χρησιμότητες όλων των χρηστών (ή όλων των αντικειμένων) και με δεδομένο ότι ο αριθμός των χρηστών (πόσο μάλλον των αντικειμένων) δεν είναι αμελητέος τότε τα προβλήματα απόδοσης γίνονται καίρια. Αυτά τα προβλήματα μπορεί να αντιμετωπιστούν με δύο τρόπους. Ο πρώτος αναφέρεται στην δημιουργία ομάδων συναφών αντικειμένων (αντίστοιχα χρηστών) (classification).

Οι λειτουργίες για την κατασκευή και αξιοποίηση του προφίλ των χρηστών απεικονίζονται στο σχήμα 1. Είναι προφανές ότι κατασκευή και η αξιοποίηση του προφίλ μπορεί να γίνονται σε άσχετο χρόνο και όχι απαραίτητα συνεχόμενα.



Σχήμα 1: Βήματα για την κατασκευή του προφίλ χρήστη καθώς και για την πρόταση σχετικών αντικειμένων.

ΛΕΙΤΟΥΡΓΙΚΟΤΗΤΑ ΕΦΑΡΜΟΓΗΣ

Το σύστημα πρέπει να προσφέρει ένα σύνολο λειτουργιών ώστε να μπορεί να χρησιμοποιηθεί από γραφικά εργαλεία και χρήστες. Κάποιες από τις λειτουργίες αυτές μπορεί να είναι και σημαντικές εσωτερικές του συστήματος διεργασίες. Τελικά προσδιορίζουν την συνολική λειτουργικότητα του συστήματος.

Δημιουργία Χρηστών

Με την λειτουργία αυτή μπορούν να δημιουργηθούν νέοι χρήστες και να εισαχθούν στο υπάρχον σύστημα. Κατά τη δημιουργία ενός χρήστη δίνετε τόσο πληροφορία που σχετίζετε με δημογραφικές πληροφορίες όσο και πληροφορία σχετιζόμενη με τα ενδιαφέροντα των χρηστών.

Διαπίστευση (Login) Χρηστών

Για να μπορέσει ένας χρήστης να αλληλεπιδράσει με το σύστημα πρέπει να πρώτα να διαπιστευθεί με βάση κάποιου συνθηματικού. Από το σημείο αυτό και μετά ο χρήστης έχει πρόσβαση στην πληροφορία που τον αφορά ενώ κατά την αναζήτηση νέων αντικειμένων γίνετε χρήση του προφίλ του.

Αποσύνδεση Χρηστών

Με την λειτουργία αυτή τερματίζει η σύνδεση των χρηστών με το σύστημα.

Ενημέρωση προφίλ χρηστών

Ο χρήστης έχει την δυνατότητα να αλλάξει το προφίλ του ανά πάσα στιγμή. Παράλληλα το προφίλ ενημερώνετε αυτόματα σε σχέση με τα αντικείμενα που τον ενδιαφέρουν.

Δημιουργία αντικειμένων

Όλοι οι χρήστες μπορούν να εισαγάγουν στο σύστημα καινούργια αντικείμενα. Κατά την εισαγωγή των αντικείμενων δίνετε πληροφορία που χαρακτηρίζει το αντικείμενο (προφίλ αντικειμένου).

Ενημέρωση αντικειμένων

Η πληροφορία των αντικειμένων μπορεί να αλλάξει. Επίσης, ανάλογα με την αξιολόγηση που τους γίνετε το προφίλ του ενημερώνετε δυναμικά.

Αξιολόγηση αντικειμένων

Οι χρήστες μετά που καταναλώσουν ένα αντικείμενο μπορούν να το αξιολογήσουν με βάση ενός συνόλου κριτηρίων. Κατά την διαδικασία αυτή τόσο το προφίλ των χρηστών ενημερώνετε όσο και το προφίλ των αντικειμένων δυναμικά.

Αναζήτηση αντικείμενων

Οι χρήστες μπορούν να αναζητούν αντικείμενα που δεν έχουν καταναλώσει με βάση ενός συνόλου κριτηρίων. Τα αντικείμενα αυτά κατατάσσονται ανάλογα με το προφίλ του χρήστη.

Διαφήμιση αντικειμένων

Τα αντικείμενα μπορούν να διαφημίζονται προς τους χρήστες κάνοντας χρήση των προφίλ τους. Αυτή είναι μία αυτόματη λειτουργία που αφορά νέους χρήστες, νέα αντικείμενα καθώς

και χρήστες που ενημέρωσαν το προφίλ τους. Με αυτή τη λειτουργία το προφίλ λειτουργεί σαν μηχανισμός διαρκούς αναζήτησης.

Διαχείριση κατάστασης συστήματος

Το σύστημα χαρακτηρίζεται ανά πάσα χρονική στιγμή από τους χρήστες και τα αντικείμενα που υπάρχουν σε αυτό καθώς και από τα προφίλ τους. Επιπλέον, το σύστημα χαρακτηρίζεται και από τις διαδικασίες που εκκρεμούν προς εκτέλεση. Το σύνολο όλων αυτών ονομάζεται κατάσταση του συστήματος και πρέπει να μπορεί να αποθηκεύεται για την περίπτωση πτώσης του συστήματος από εξωτερικούς παράγοντες (system crash και system failure). Επίσης, πρέπει να είναι δυνατόν να φορτωθεί και να ενεργοποιηθεί στην κεντρική μνήμη.

ΑΝΑΚΕΦΑΛΑΙΩΣΗ

Σε αυτό το κεφάλαιο παρουσιάστηκαν οι γενικότερες κατευθυντήριες γραμμές για τον σχεδιασμό του μοντέλου των δεδομένων και του μοντέλου αξιοποίησης και δημιουργίας προφίλ, ενώ ορίστηκε η λειτουργικότητα του συστήματος

Στο επόμενο κεφάλαιο θα γίνει ο σχεδιασμός τόσο των μηχανισμών και των δεδομένων της εφαρμογής, όσο και της αρχιτεκτονικής του συστήματος.

ΚΕΦΑΛΑΙΟ Δ' – ΣΧΕΔΙΑΣΜΟΣ

Στο κεφάλαιο αυτό θα παρουσιαστεί διεξοδικά η σχεδίαση του συνολικού συστήματος. Δύο είναι οι κύριοι άξονες που θα μας απασχολήσουν με πρώτο τον σχεδιασμό τόσο των προφίλ των χρηστών και των αντικειμένων-άρθρων όσο και την περιγραφή και ανάλυση των επιμέρους κριτηρίων με βάση τα οποία θα γίνετε κατάταξη των συναφών αντικειμένων ή επιλογή χρηστών για αποδέκτες αντικειμένων. Ο δεύτερος άξονας είναι ο τεχνικός σχεδιασμός της εφαρμογής και αποτύπωση της λειτουργικότητας στην αρχιτεκτονική του συστήματος.

Όσο αφορά στα προφίλ των χρηστών και των αντικειμένων θα καθοριστούν τα επιμέρους πεδία που θα τα απαρτίζουν ούτως ώστε να αποσαφηνιστεί και το γενικότερο πλαίσιο της εφαρμογής και τι ακριβώς θα υποστηρίζει σε θέματα όπως αξιολόγηση αντικειμένων και μηχανισμούς προτιμήσεων. Επιπλέον, θα αναλυθούν τα κριτήρια και ο μηχανισμός προσδιορισμού των τιμών τους.

Το σύστημα σχεδιάστηκε με μία αρχιτεκτονική τεσσάρων επιπέδων. Ειδικά το τμήμα που περικλείει όλοι την λογική της εφαρμογής και υλοποιεί τις προδιαγραφές του συστήματος είναι ένα πολύ-πρακτορικό περιβάλλον όπου ευφυείς πράκτορες αλληλεπιδρούν με βάση ενός συγκεκριμένου πρωτοκόλλου που σχεδιαστικέ με οντολογία.

ΓΕΝΙΚΟ ΜΟΝΤΕΛΟ

ΤΟ ΠΡΟΦΙΛ ΤΟΥ ΧΡΗΣΤΗ

Σε αυτή την παράγραφο θα εξετάσουμε τα στοιχεία που αποτελούν το προφίλ ενός χρήστη. Βασικός γνώμονας κατά την σύνταξη αυτών αποτέλεσαν τρεις βασικοί πυλώνες: 1) Δημογραφικές πληροφορίες (στατικές), 2) ιστορικό χρήσης (δυναμικό) και 3) προτιμήσεις. Οι προτιμήσεις του διακρίνονται περαιτέρω σε δύο κατηγορίες ανάλογα με την χρήση τους. Αυτές που χρησιμοποιούνται για την αποκοπή άρθρων και αυτές που τα κατατάσσουν με βαθμό σχετικότητας.

Ο επόμενος πίνακας παρουσιάζει όλα τα στοιχεία που διατηρούνται στο προφίλ ενός χρήστη. Πολλές από τις προτιμήσεις κατάταξης υπάρχουν και σαν αποκοπής (φίλτρα) και δεν είναι παράδοξο γιατί ανάλογα με τις ανάγκες ενός χρήστη μπορούν να λειτουργούν είτε σαν αποκοπής είτε σαν κατάταξης.

	Στοιχείο
Δημογραφικά στοιχεία	Ονοματεπώνυμο χρήστη
	Τίτλος (Mr, Msc, Phd, Dr, Prof)
	Θέση
	Ίδρυμα/Επιχείρηση
	Email
Προτιμήσεις Κατάταξης	Κατηγορίες άρθρων
	Συγγραφείς
	Πηγές δημοσίευσης
	Λέξεις κλειδιά
Προτιμήσεις Αποκοπής	Λέξεις κλειδιά
	Κατηγορία άρθρων
	Συγγραφείς
	Έτος δημοσίευσης
	Μορφή (doc, pdf, txt, html)
	Πηγή δημοσίευσης
	Γλώσσα συγγραφής
	Επιστημονικά πεδία
Ιστορικό χρήσης	Λίστα συγγραφέων
	Περιεχόμενο άρθρων (από keywords, τίτλο, κλπ.) με στατιστικά
	Πηγές δημοσιεύσεων
	Βαθμός εμπειρίας

Πίνακας 2: Τα στοιχεία που αποτελούν ένα προφίλ χρήστη

ΤΟ ΠΡΟΦΙΛ ΤΟΥ ΆΡΘΡΟΥ ΚΑΙ ΑΞΙΟΛΟΓΗΣΗ

Για κάθε ένα άρθρο διατηρούμε ένα σύνολο πληροφοριών που επιτρέπουν στο σύστημα να τα προτείνει σε χρήστες και το ονομάζουμε προφίλ του άρθρου. Το προφίλ του άρθρου αποτελείται από δύο κατηγορίες στοιχείων: αυτά που αναφέρονται στο περιεχόμενό του και είναι στατικά (π.χ. τίτλος) και την αξιολόγηση που του κάνουν οι αναγνώστες του.

Ο επόμενος πίνακας παρουσιάζει όλα τα στοιχεία που διατηρούνται στο προφίλ ενός χρήστη. Η αξιολόγηση των άρθρων γίνετε από χρήστες ανάλογα και με την σχετικότητα των χρηστών με το επιστημονικό πεδίο.

	Στοιχείο
Πληροφορίες Περιεχομένου	Συγγραφείς
	Περίληψη
	Τίτλος
	Ίδρυμα/Επιχείρηση
	Έτος δημοσίευσης
	Μορφή (doc, pdf, txt, html)
	Λέξεις κλειδιά
	Επιστημονικά πεδία
	Πηγή δημοσίευσης
	Γλώσσα συγγραφής
	Κατηγορία άρθρου (με βαθμό σε κάθε κατηγορία)
Αξιολόγηση	Αξιολόγηση συγγραφέων (βαθμός σχετικότητας με το επιστημονικό πεδίο)
	Βαθμός γνησιότητας άρθρου
	Σημαντικότητα (συνεισφορά στην επιστήμη)
	Σημαντικότητα για τους χρήστες
	Ποιότητα
	Συνέπεια περίληψης με κείμενο
	Πληρότητα βιβλιογραφίας
	Συνολική Αξιολόγηση

Πίνακας 3: Τα στοιχεία που αποτελούν ένα προφίλ άρθρου.

Το σύνολο των στοιχείων με βάση τα οποία οι χρήστες αξιολογούν τα άρθρα είναι γενικά κριτήρια άλλα με βάση αυτά πρέπει να μπορούμε να κατατάξουμε τα άρθρα που έχει καταναλώσει ο χρήστης ώστε να μπορέσουμε να κατασκευάσουμε το UTA* πρόβλημα.

Ο μηχανισμός που αποφασίζουμε για την κατάταξη των άρθρων στο ιστορικό ενός χρήστη είναι αρχικά να τα κατατάσσουμε με βάση την συνολική αξιολόγηση και για αυτά που έχουν ίδια συνολική αξιολόγηση να συγκρίνουμε το μέσο όρο των επιμέρους κριτηρίων αξιολόγησης και αν έχουν και πάλι ίδια τιμή τότε βρίσκονται στην ίδια κατάταξη.

ΤΑ ΚΡΙΤΗΡΙΑ

Τέσσερα είναι τα κριτήρια για την κατασκευή των πολυκριτήριων πινάκων μερικών και ολικών χρησιμοτήτων. Με βάση, λοιπόν, αυτά τα τέσσερα θα μπορούν να κατατάσσονται αντικείμενα και προτείνονται σε χρήστες νέα αντικείμενα. Συγκεκριμένα τα κριτήρια είναι: το περιεχόμενο των άρθρων, οι συγγραφείς τους, οι κατηγορίες τους, και οι πηγή δημοσίευσής τους. Το πρόβλημα είναι πώς, λόγου χάρη, από ένα σύνολο λέξεων κλειδιών με βάρη του προφίλ του χρήστη και το περιεχόμενο ενός άρθρου μπορούμε να προσδιορίσουμε μία τιμή για αυτό το κριτήριο σε ένα συγκεκριμένο εύρος τιμών.

Στην επόμενη παράγραφο μοντελοποιείται διεξοδικά ο μηχανισμός απόδοσης τιμών στα επιμέρους κριτήρια.

ΤΟ ΠΛΑΙΣΙΟ ΑΣΑΦΟΥΣ ΛΟΓΙΚΗΣ ΓΙΑ ΤΙΣ ΕΚΤΙΜΗΣΕΙΣ ΠΡΟΤΙΜΗΣΕΩΝ

Όλες οι εκτιμήσεις προτιμήσεων που υπολογίζονται από τους απόδοσης τιμών στα κριτήρια μπορούν να θεωρηθούν σαν αιτήσεις ανάκτησης πληροφοριών βασισμένες σε ομοιότητα. Όλες αυτές οι διαφορετικών ειδών εκτιμήσεις προτιμήσεων μπορούν να μοντελοποιηθούν χρησιμοποιώντας το ίδιο γενικό μαθηματικό πλαίσιο βασισμένο στην θεωρία ανάκτησης πληροφοριών. Αυτό το πλαίσιο περιγράφεται εδώ.

Γενικά, κάθε αίτηση για ανάκτηση αντικειμένων που ανήκουν σε ένα σύμπαν I που περιγράφεται από ένα χώρο χαρακτηριστικών F, αντιστοιχεί σε μία ερώτηση q που συγκροτείται από ένα (πιθανώς) δομημένο σύνολο χαρακτηριστικών F', τα οποία, με την σειρά τους, είναι υποσύνολο του F. Αυτό το γενικό σχήμα μπορεί να χρησιμοποιηθεί για να περιγράψει τα παρακάτω είδη λειτουργικότητας (βλέπε την στήλη "Ερμηνεία"):

Σύμπαν I	**Χώρος Χαρακτηριστικών F**	**Ερωτήσεις Q**	**Ερμηνεία**

Άρθρα	Χαρακτηριστικά περιγραφής περιεχομένου (περίληψη λέξεις κλειδιά, και τίτλος).	Λέξεις κλειδιά με βάρη από το προφίλ του χρήστη.	Εκτίμηση βάρους για τον πολύ-κριτήριο πίνακα
Άρθρα	Συγγραφείς	Συγγραφεί ς με βάρη από το προφίλ του χρήστη.	Εκτίμηση βάρους για τον πολύ-κριτήριο πίνακα
Άρθρα	Κατηγορίες άρθρου	Κατηγορίες άρθρου με βάρη από το προφίλ του χρήστη.	Εκτίμηση βάρους για τον πολύ-κριτήριο πίνακα
Άρθρα	Πηγή δημοσίευσης	Πηγές δημοσίευσης με βάρη από το προφίλ του χρήστη.	Εκτιμήσεις ομοιότητας χρηστών με αντικείμενο
Άρθρα	Χαρακτηριστικά του άρθρου	Κριτήρια αναζήτησης (Προτιμήσεις αποκοπής) χρήστη	Εκτίμηση σχετικότητας άρθρου σε σχέση με την αναζήτηση του χρήστη.
Άρθρα	Χαρακτηριστικά του άρθρου	Χαρακτηριστικά του άρθρου	Εκτίμηση ομοιότητας δύο άρθρων
Χρήστες	Χαρακτηριστικά του προφίλ του χρήστη	Χαρακτηριστικά του προφίλ του χρήστη	Εκτίμηση ομοιότητας δύο προφίλ χρηστών

Πίνακας 4: Διαφορετικά είδη λειτουργικότητας εκφραζόμενα από ένα γενικό σύστημα ανάκτησης πληροφορίας.

Σημαντικό είναι να παρατηρήσουμε ότι το πλαίσιο αυτό δεν θα χρησιμοποιηθεί μόνο για την απόδοση τιμών στα κριτήρια με βάση τις προτιμήσεις του χρήστη, αλλά και για εύρεση σχετικότητας ενός άρθρου με βάση κάποια κριτήρια αναζήτησης που έδωσε ο χρήστης, για την εύρεση σχετικότητας δύο άρθρων και για την εύρεση σχετικότητας δύο προφίλ χρηστών.

Με βάση τώρα αυτήν την παραπάνω μοντελοποίηση και μοντέλο ασαφούς λογικής p-norm που περιγράφηκε στο δεύτερο κεφάλαιο εμείς θεωρούμε ότι μεταξύ των προτιμήσεων κατάταξης του προφίλ του χρήστη υπάρχει ασαφής διάζευξη (fuzzy OR) ενώ μεταξύ των προτιμήσεων αποκοπής ισχύει φόρμουλα κανονικής σύζευξης (conjunctive normal form). Οι εξισώσεις που υπολογίζουν τις τιμές αυτών των συναρτήσεων δίνονται στον πίνακα 1. Η εξίσωση της ασαφούς σύζευξης επαναλαμβάνετε εδώ για ευκολία:

$$\left(\frac{\sum_{i=1}^{n} a_i^p \cdot w_i^p}{\sum_{i=1}^{n} w_i^p} \right)^{1/p} \quad 1 \le p \le \infty$$

Όπου *a* είναι 1 αν υπάρχει το χαρακτηριστικό (feature) *i* και 0 διαφορετικά, ενώ *w* είναι το βάρος της προτίμησης του χρήστη για το ίδιο χαρακτηριστικό. Τέλος, *p* είναι μία σταθερά που αποτυπώνει το πόσο αυστηρή είναι η διάζευξη.

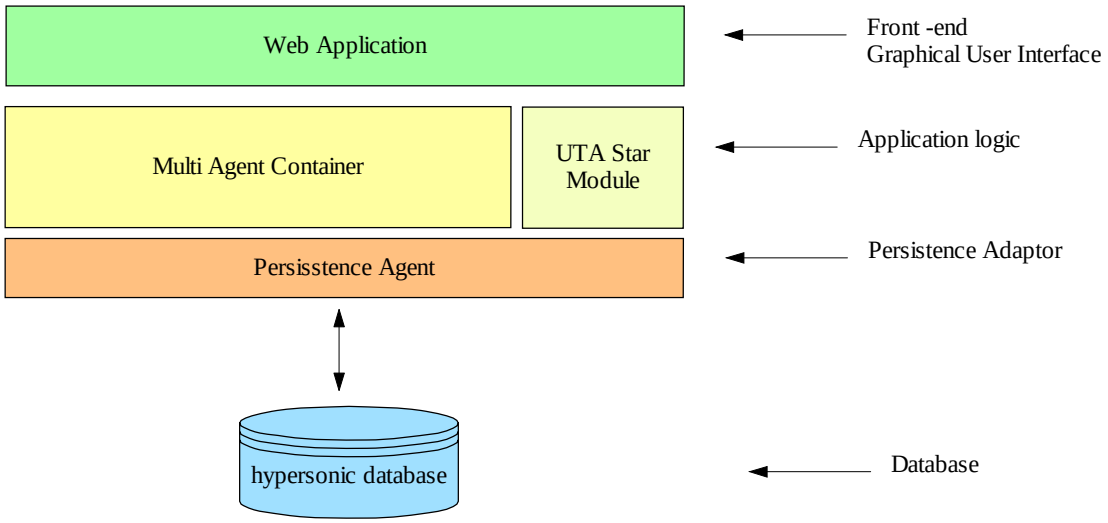
Η παραπάνω φόρμουλα μας επιστρέφει ένα αριθμό στο διάστημα [0, 1]. Εμείς αποτυπώνουμε αυτό το διάστημα σε ακέραιους αριθμούς με βάση την παρακάτω συνάρτηση:

$$\varphi(\chi) = \begin{cases} 0, & \chi \in [0, 0.3) \\ 1, & \chi \in [0.3, 0.5) \\ 2, & \chi \in [0.5, 0.7) \\ 3, & \chi \in [0.7, 1] \end{cases}$$

Κατά αυτόν τον τρόπο δοθέντος ενός άρθρου και ενός προφίλ μπορούμε να καταλήξουμε σε τέσσερις τιμές, μία για κάθε ένα κριτήριο.

ΓΕΝΙΚΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗ

Η γενική αρχιτεκτονική του συστήματος επιλέχθηκε να είναι πολυεπίπεδη (multi-tier). Το πλεονέκτημα αυτής της αρχιτεκτονικής είναι ότι επειδή τα επίπεδα είναι διακριτά μπορούν να αντικατασταθούν από αλλά ανά πάσα στιγμή χρησιμοποιώντας τα προηγούμενα ως έχουν. Στο σχήμα 2 παρουσιάζουμε την γενική αρχιτεκτονική και διακρίνουμε τέσσερα επίπεδα.



Σχήμα 2: Πολυεπίπεδη αρχιτεκτονική συστήματος.

Ξεκινώντας από πάνω προς τα κάτω ξεχωρίζουμε το επίπεδο διεπαφής με τον χρήστη που είναι μία εφαρμογή ιστού. Θα μπορούσε όμως να είναι και κάποιο άλλο γραφικό εργαλείο χρησιμοποιώντας ακριβώς την ίδια λογική από κάτω με το ίδιο API.

Στο δεύτερο επίπεδο οργανώνετε η λογική της εφαρμογής (application logic) που είναι ένα πολύ-πρακτορικό σύστημα μαζί με την αυτοτελή μονάδα που επιλύει UTA* προβλήματα μαζί με το γραμμικό επιλυτή Lindo. Η πολύ-πρακτορική μονάδα προσφέρει όλοι την λειτουργικότητα που απαιτείται από τις προδιαγραφές και αναλύεται σε επόμενη παράγραφο.

Ένα επίπεδο χαμηλότερα βρίσκετε ο προσαρμογέας (Adaptor) που διαχειρίζεται την κατάσταση του συστήματος και αναλαμβάνει να την αποθηκεύσει στη μόνιμη αποθήκη (persistence storage). Αυτό το τμήμα λογισμικού εξαρτάται απόλυτα από την βάση δεδομένων που θα χρησιμοποιηθεί για την τελική αποθήκευση της κατάστασης. Επίσης αναλαμβάνει να φορτώσει την κατάσταση προηγούμενων σωσμένων καταστάσεων στη κεντρική μνήμη και να την ενεργοποιήσει.

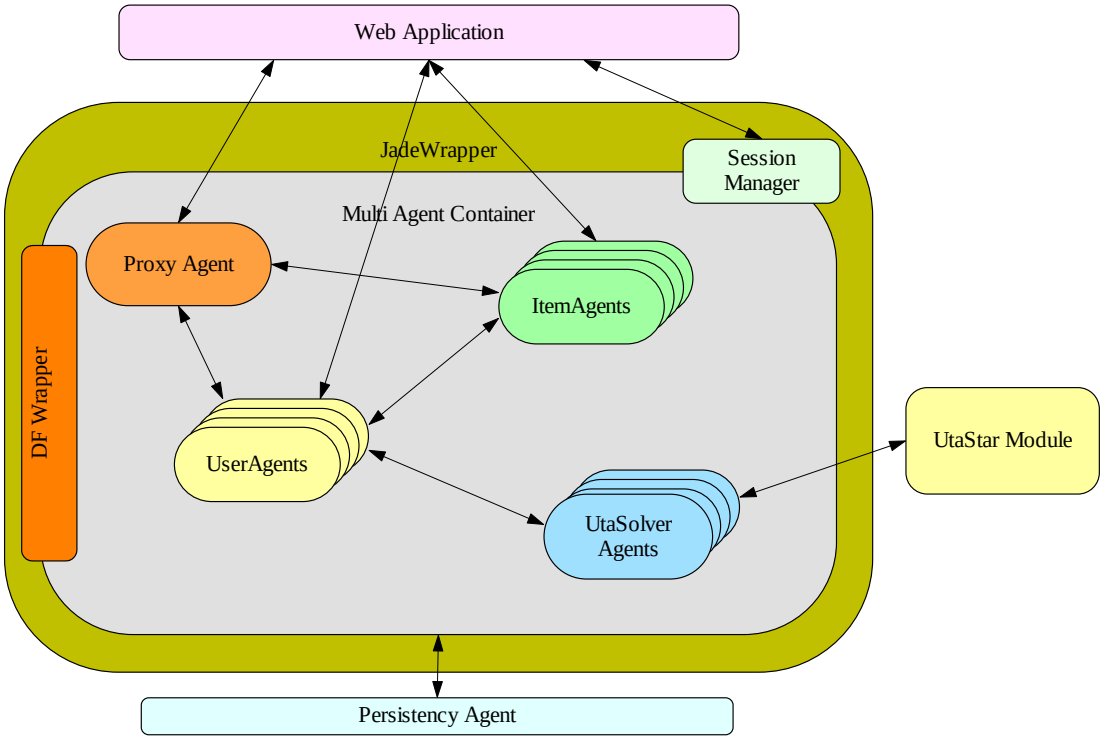
Τέλος, το τελευταίο επίπεδο είναι το σύστημα διαχείρισης βάσης δεδομένων όπου θα αποθηκεύονται οι καταστάσεις και χρησιμοποιήθηκε η hypersonicSQL database engine [16].

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΜΟΝΑΔΑΣ ΕΥΦΥΩΝ ΠΡΑΚΤΟΡΩΝ

Το σύστημα για να ικανοποιήσει τις πιο πάνω λειτουργίες έχει μοντελοποιήθει με χρήση ευφυών πρακτόρων των οποίων οι συμπεριφορές (behaviors) ανταποκρίνονται στις παραπάνω

λειτουργίες. Οι πράκτορες επικοινωνούν μεταξύ τους πάνω είτε από ένα κεντρικό σύστημα είτε κατανεμημένα. Στο παρακάτω σχήμα παρουσιάζεται η αρχιτεκτονική των πρακτόρων και του συστήματος. Φαίνεται ο σχεδιασμός να έχει γίνει για ένα κεντρικό σύστημα το οποίο θα περιλαμβάνει ένα proxy, μια βάση δεδομένων και ένα σύνολο διαφορετικών πρακτόρων, αλλά στην πραγματικότητα οι πράκτορες μπορεί να βρίσκονται σε διαφορετικά μηχανήματα και απλά να επικοινωνούν μεταξύ τους.

Παρακάτω ακολουθεί μια συνοπτική περιγραφή των πρακτόρων και των άλλων μονάδων λογισμικού ενώ αναλυτική περιγραφή μπορεί να βρει ο αναγνώστης στο επόμενο κεφάλαιο («υλοποίηση»).



Σχήμα 3: Αρχιτεκτονική του πολυ-πρακτορικής μονάδας.

User-Agent

Κάθε χρήστης (User) αντιστοιχίζεται σε έναν πράκτορα με όνομα User-Agent ο οποίος αναλαμβάνει να διατηρεί και να ενημερώνει το προφίλ του χρήστη (User-Profile) ανάλογα με τις κινήσεις του και τις προσωπικές επιλογές του. Ένας user-agent δημιουργείται όταν ένας χρήστης εγγράφετε στο σύστημα και έχει ένα σύνολο από φίλους (πράκτορες με κοινά ενδιαφέροντα). Γενικά, έχει τις εξής συμπεριφορές (behaviors):

- Κατανάλωση Αντικειμένου

- Βαθμολόγηση Αντικειμένου
- Υπολογισμός Προφίλ
- Αλλαγή Προφίλ
- Πρόταση Αντικειμένων
- Εύρεση Σχετικών Χρηστών (Φίλων)

Item Agent

Κάθε αντικείμενο-άρθρο (Item) αντιστοιχίζεται ένα πράκτορα με όνομα Item-Agent ο οποίος αναλαμβάνει να διατηρεί και να ενημερώνει το προφίλ του αντικειμένου (Item-Profile) ανάλογα με την χρήση του αντικειμένου. Τα αντικείμενα διατηρούν μία λίστα με σχετικά αντικείμενα. Έχει τις εξής συμπεριφορές (behaviors):

- Κατανάλωση
- Υπολογισμός Προφίλ
- Αλλαγή Προφίλ
- Εύρεση Σχετικών Αντικειμένων
- Διαφήμιση (προώθηση σε σχετικούς χρήστες)
- Πρόταση Σχετικών Αντικειμένων

Uta Solver Agent

Ο πράκτορας Uta Solver Agent είναι επιφορτισμένος να επιλύει προβλήματα με τον αλγόριθμο UTA*. Αυτός ο πράκτορας είναι κύριο λειτουργικό σημείο του συστήματος και είναι εντελώς ανεξάρτητος από την περιοχή μας. Θα μπορούσε δηλαδή να χρησιμοποιηθεί και σε άλλα πολυ-πρακτορικά συστήματα. Μπορούν να υπάρχουν περισσότεροι του ενός πράκτορες αυτού του είδους.

Proxy Manager

Ο πράκτορας proxy manager αναλαμβάνει να διαχειριστεί την επικοινωνία του πολύ-πρακτορικού συστήματος με το front-end επίπεδο. Περιλαμβάνει ένα σύνολο ιδιοτήτων που αντικατοπτρίζουν το σύνολο της προσφερόμενης λειτουργικότητας.

Persistence Manager

Ο Persistence Manager είναι ένας πράκτορας που αναλαμβάνει να διαχειριστεί για λογαριασμό όλων των άλλων πρακτόρων τα δεδομένα τους και την κατάστασή τους πάνω από

μία βάση δεδομένων. Στη βάση δεδομένων κρατούνται τόσο τα προφίλ των χρηστών και των αντικειμένων, όσο και η πληροφορία χρήσης του συστήματος. Γενικότερα κατάσταση του συστήματος αναφέρουμε την κατάσταση κάθε πράκτορα και το σύνολο των απεσταλμένων αλλά μη παραδομένων μηνυμάτων.

UtaSar Module

Η μονάδα αυτή είναι ανεξάρτητη από την εφαρμογή και μπορεί να χρησιμοποιηθεί και σε άλλες εφαρμογές με την χρήση του API της. Αναλαμβάνει να επιλύσει προβλήματα UTA* μετατρέποντάς τα σε γραμμικά προβλήματα και επιλύοντας μετα τα γραμμικά προβλήματα με χρήση του Lindo, ενός επιλυτή γραμμικών προβλημάτων. Τα προβλήματα μετα-βελτιστοποιούνται και ξαναλύνονται σε μια προσπάθεια να βρεθεί η καλύτερη λύση.

DF Wrapper

Αυτή η μονάδα προσφέρει ένα κοινό για όλους τους πράκτορες τρόπο να χρησιμοποιούν την υπηρεσία DF του Jade που λειτουργεί ως yellow-pages. Αναλαμβάνει να αρχικοποιήσει την υπηρεσία και στη συνέχεια οι πράκτορας εγγράφονται στην υπηρεσία μέσω αυτού που αναλαμβάνει να κατασκευάσει κατάλληλα το προφίλ του κάθε πράκτορα ανάλογα με το είδος του. Επίσης, οι πράκτορες που να διαγράφονται από την υπηρεσία κατά το δοκούν. Στην εφαρμογή μας υπάρχει συγκεκριμένη πολιτική που ακολουθεί κάθε πράκτορας αναφορικά με την υπηρεσία DF και θα φανεί στο επόμενο κεφάλαιο.

ΕΠΙΚΟΙΝΩΝΙΑ ΚΑΙ ΣΥΝΕΡΓΑΣΙΑ ΠΡΑΚΤΟΡΩΝ

Σε αυτήν την παράγραφο παρουσιάζεται ο σχεδιασμός όλου του περιβάλλοντος των πρακτόρων με χρήση οντολογίας. Με την οντολογία ουσιαστικά ορίστηκε μία κοινή γλώσσα επικοινωνίας ανάμεσα στους πράκτορες. Μέσω αυτής της γλώσσας οι πράκτορες ανταλλάσουν δεδομένα και πληροφορίες καθώς επίσης ζητούν από κάποιον άλλο πράκτορα να ενεργήσει κατά ένα τρόπο. Δύο λοιπόν είναι τα στοιχεία που μας ενδιαφέρουν με πρώτο την δυνατή πληροφορία που μπορούν να ανταλλάξουν μεταξύ τους οι πράκτορες και δεύτερον τις ενέργειες που μπορούν να πραγματοποιήσουν. Οι ενέργειες χωρίζονται σε δύο επιμέρους κατηγορίες· στις απλές ερωτήσεις και ενημερώσεις και στις ενέργειες που περιλαμβάνουν πολύπλοκη επικοινωνία. Η πρώτη κατηγορία που αναφέρεται ως predicates (κατηγορήματα) ή ενημερώνουν κάποιον πράκτορα ότι κάτι έγινε ή του ζητάνε μια πληροφορία και η απάντηση είναι το ίδιο κατηγόρημα με κάποια πληροφορία ή μήνυμα λάθους. Στην δεύτερη που αναφέρεται ως agent action (ενέργεια πράκτορα) ξεκινάει μία διαδικασία που μπορεί να

περιλαμβάνει και περισσότερους πράκτορες και η απάντηση μπορεί να είναι οτιδήποτε (από πληροφορία σε κάποιο αντικείμενο της οντολογίας μέχρι μία άλλη ενέργεια ή απλά ότι όλα πήγαν καλά) και εξαρτάτε από το είδος της ενέργειας και την λογική της διαδικασίας της.

Οι παρακάτω πίνακες δείχνουν το σύνολο των εννοιών που αποτελούν την οντολογία επικοινωνίας των πρακτόρων. Παρουσιάζονται γενικά και όχι συγκεκριμένα με τα πεδία τους πράγμα που θα μας απασχολήσει στο κεφάλαιο της υλοποίησης.

Στον παρακάτω πίνακα παρατίθενται τα αντικείμενα (concepts) που μπορούν να έχουν οι πράκτορες ως μεταβλητές καθώς και να ανταλλάζουν μεταξύ τους με μηνύματα. Στην πρώτη στήλη αναφέρεται το διακριτικό όνομα του αντικειμένου, ενώ στη δεύτερη η ερμηνεία και ο τρόπος χρήσης του.

Όνομα	Ερμηνεία
AuthorW	Συνδυασμός συγγραφέα με βάρος προτίμησης.
SourceW	Συνδυασμός πηγής δημοσίευσης με βάρος προτίμησης.
Author	Στοιχείο που περιγράφει ένα συγγραφέα.
Credit	Περιέχει τα κριτήρια αξιολόγησης
User Profile	Περιέχει προφίλ του χρήστη
Problem	Περιέχει ένα UTA* πρόβλημα
Item Sort	
Item	Το αντικείμενο
Usage Data	Περιέχει δεδομένα ιστορικού χρήσης
Suggestion	Μια πρόταση του συστήματος
KeywordW	Συνδυασμός λέξης κλειδιού και βάρους προτίμησης.
User Preferences	Προτιμήσεις Χρήστη
Scientific Field	Επιστημονικό πεδίο
Category	Συνδυασμός κατηγορίας και βάρους προτίμησης.
Solution	Περιέχει μία UTA* λύση
Criterion	Ένα γενικό κριτήριο
User	Ο χρήστης

Πίνακας 5: Τα αντικείμενα (concepts) και οι ερμηνείες τους της οντολογίας για την ανταλλαγή μηνυμάτων.

Στον παρακάτω πίνακα παρουσιάζονται οι ενέργειες στις οποίες μπορούν να προβούν οι πράκτορες για να διεκπεραιώσουν τους σκοπούς τους.

Όνομα	Ερμηνεία
Request Profile	Ζητάει να κατασκευαστεί ένα προφίλ
Read Usage	Ζητάει το ιστορικό χρήσης
Update Item	Ανανεώνει ένα αντικείμενο
Update Profile	Ανανεώνει ένα προφίλ
Advertise	Διαφημίζει ένα αντικείμενο
Consume	Καταναλώνει ένα αντικείμενο για έναν χρήστη
Suggest	Προτείνει ένα αντικείμενο
Proxy Consume	Όμοια με την Cosnume αλλά με λιγότερη πληροφορία
Search	Αναζητάει
Register User	Εγγράφει ένα καινούργια χρήστη
Send Relevance	Στέλνει σχετικότητα ενός αντικειμένου με κάποια κριτήρια
Profile Sort	Κατατάσσει ένα σύνολο αντικειμένων με βάση το προφίλ του χρήστη
Create Item	Δημιουργεί ένα αντικείμενο
Find Friend	Αναζητάει για φίλους πράκτορες με κοινά χαρακτηριστικά.
Update User	Ενημερώνει τα δεδομένα ενός χρήστη

Πίνακας 6: Οι ενέργειες των πρακτόρων και οι ερμηνείες τους της οντολογίας για την ανταλλαγή μηνυμάτων.

Στον παρακάτω πίνακα παρατίθενται τα κατηγορήματα (ερωτήσεις και δελτία ενημέρωσης) – predicates- στα όποια προβαίνουν οι πράκτορες.

Όνομα	Ερμηνεία
User Exists	Ρωτάει αν υπάρχει ένας χρήστης
Is Consumed	Ρωτάει αν έχει καταναλωθεί κάποιο αντικείμενο
Get User	Επιστρέφει πληροφορίες και το προφίλ του χρήστη

Sort Items	Ταξινομεί ένα σύνολο αντικείμενων
Get Relevance	Ρωτάει την σχετικότητα ενός αντικείμενου με ένα σύνολο προτιμήσεων
Get Suggestions	Επιστρέφει τις προτάσεις του συστήματος για τον χρήστη
Item Exists	Ρωτάει αν υπάρχει ένα αντικείμενο
Error	Γενικό μήνυμα λάθους
Is Suggested	Ρωτάει αν κάποιο αντικείμενο έχει προταθεί στο χρήστη
Get Item	Επιστρέφει το αντικείμενο.

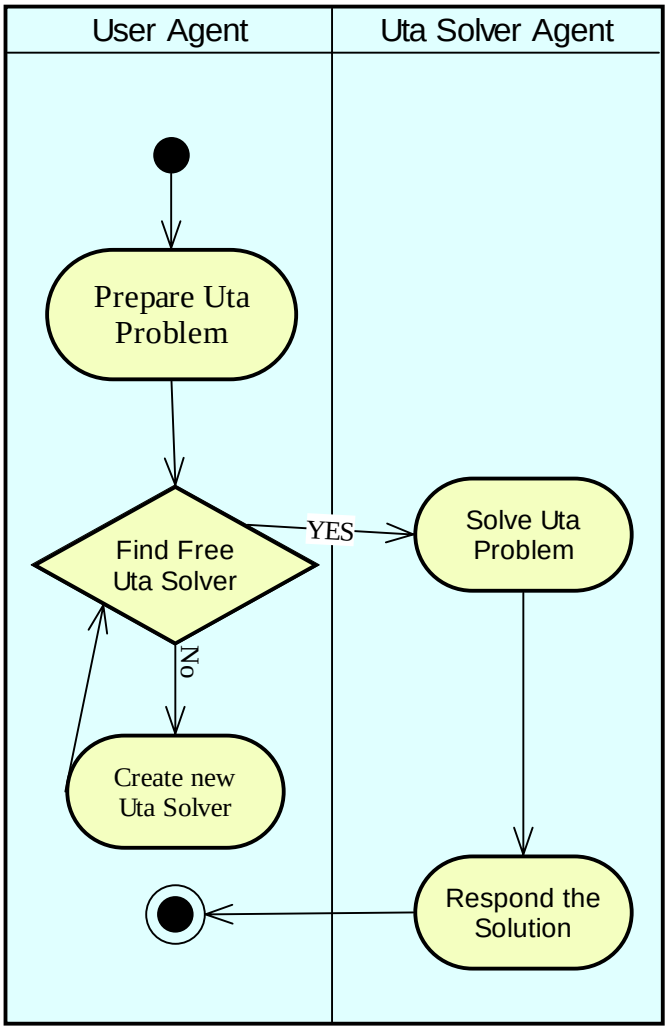
Πίνακας 7: Τα κατηγορήματα- ερωτήσεις και δελτία ενημέρωσης- (predicates) της οντολογίας για την ανταλλαγή μηνυμάτων.

ΔΙΑΔΙΚΑΣΙΕΣ

Σε αυτήν την παράγραφο παρουσιάζονται οι διαδικασίες που εκτελούνται από τους πράκτορες του συστήματος σαν συνεργαζόμενες συμπεριφορές με ανταλλαγή κατάλληλων μηνυμάτων, το οποίο αποτελεί ουσιαστικά ολόκληρη την λογική των πρακτόρων και τις εφαρμογής. Συγκεκριμένα δεν θα παρουσιαστούν διεξοδικά όλες άλλα μόνο πέντε που είναι και οι πιο περίπλοκες. Στην ανάλυση που ακολουθεί παρουσιάζουμε τις διαδικασίες που εμπλέκουν διάφορους πράκτορες και τι μηνύματα ανταλλάσουν για να επιτύχουν τον τελικό στόχο.

Ενημέρωση προφίλ χρήστη

Η ενημέρωση του προφίλ ενός χρήστη επιτυγχάνεται με αντίστοιχη συμπεριφορά του πράκτορα χρήστη και άλλες απαραίτητες για την ανταλλαγή μηνυμάτων και εκτέλεση εντολών του πράκτορα UtaSolve. Αυτή η συμπεριφορά ενεργοποιείται περιοδικά και εφόσον έχει υπάρξει αλλαγή στο ιστορικό χρήσης του χρήστη. Εφόσον ενεργοποιηθεί προσπαθεί να ενημερώσει τους πολύ-κριτήριους πίνακες του προφίλ. Το παρακάτω UML διάγραμμα ενεργειών καταδεικνύει τις ενέργειες που χρειάζονται για να ενημερωθεί το προφίλ.

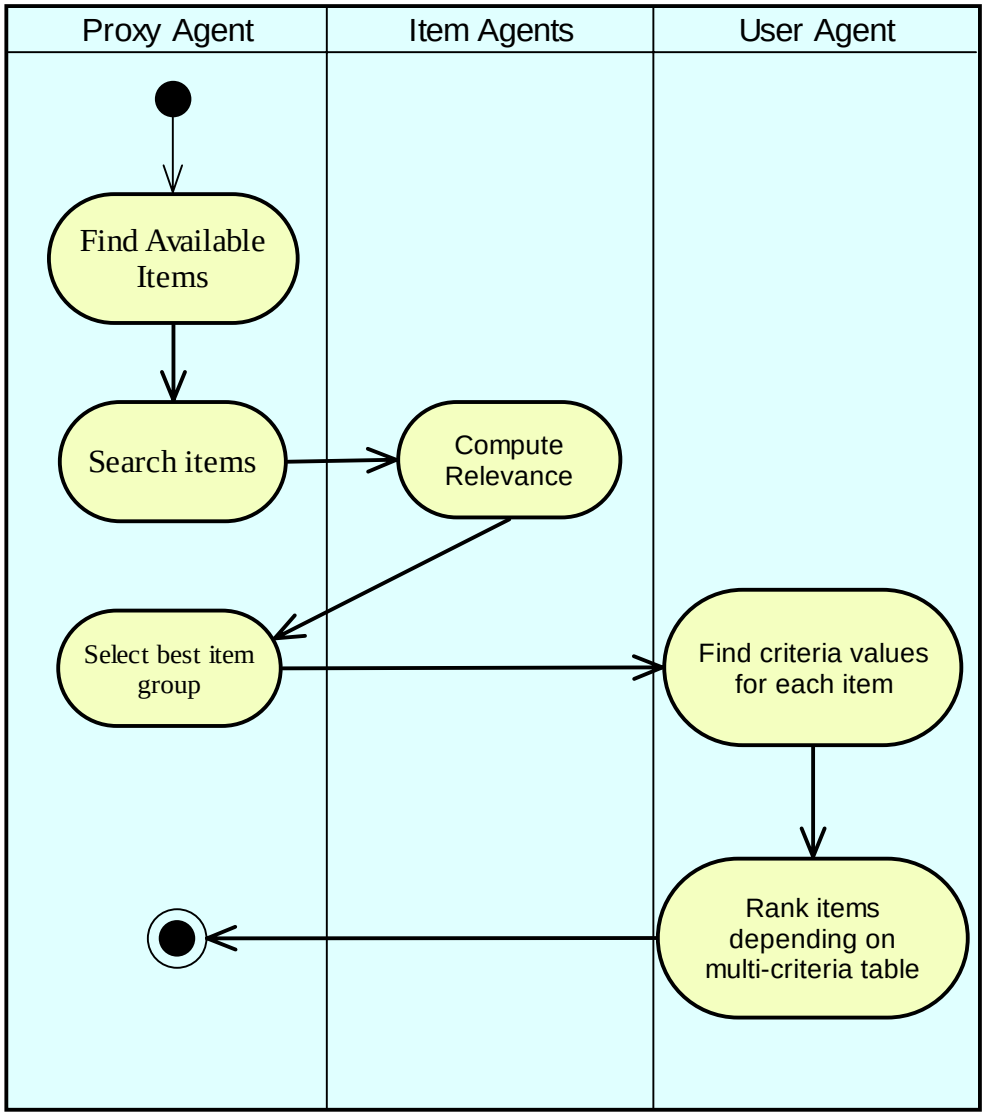


Σχήμα 4:UML διάγραμμα ενεργειών για την ενημέρωση του προφίλ ενός χρήστη.

Οι ενέργειες που εκτελούνται είναι αρχικά να συλλεχθούν τα αντικείμενα που έχουν καταναλωθεί και να βρεθούν οι τιμές για τα επιμέρους κριτήρια χρησιμοποιώντας το p-norm μοντέλο. Αφού έχει γίνει η προετοιμασία του πρβλήματος UTA* (prepare uta problem), αναζητάμε μέσω της υπηρεσίας DF ελεύθερους πράκτορες Uta Solver. Εάν δεν υπάρχουν ελεύθεροι τότε δημιουργούμε ένα νέο, διαφορετικά παίρνουμε τον πρώτο και του αποστέλλουμε ένα μήνυμα με το πρόβλημα. Μετά η συμπεριφορά αυτή τερματίζει ενώ η διαδικασία συνεχίζετε στον πράκτορα Uta Solve. Όταν το βρεθεί κάποια λύση τότε ο πράκτορας UTA Solve αποστέλλει μήνυμα με την λύση στον πράκτορα που αρχικά του ζήτησε την λύση. Όταν λάβει το μήνυμα τότε ενημερώνει το τοπικό πεδίο του με το καινούργιο πολύ-κριτήριο πίνακα.

Αναζήτηση Αντικειμένων

Η αναζήτηση αντικειμένων επιτυγχάνεται με την αντίστοιχη συμπεριφορά του πράκτορα Proxy που αναλαμβάνει ρόλο μεσολαβητή ανάμεσα στους πράκτορες και την εφαρμογή διεπαφής με τους χρήστες. Όταν ο πράκτορας Proxy λάβει αίτηση για αναζήτηση τότε ενεργοποιεί αυτήν την συμπεριφορά για κάποιον χρήστη και με κάποιες προτιμήσεις αναζήτησης. Όταν η συμπεριφορά ολοκληρωθεί ενημερώνει τον εντολέα μέσω του διαχειριστή συνόδων (session manager) για τα αποτελέσματα. Το παρακάτω UML διάγραμμα ενεργειών καταδεικνύει τις ενέργειες που χρειάζονται για να γίνει μια αναζήτηση.



Σχήμα 5: UML διάγραμμα ενεργειών για την αναζήτηση αντικειμένων με βάση κάποια κριτήρια και το προφίλ ενός χρήστη.

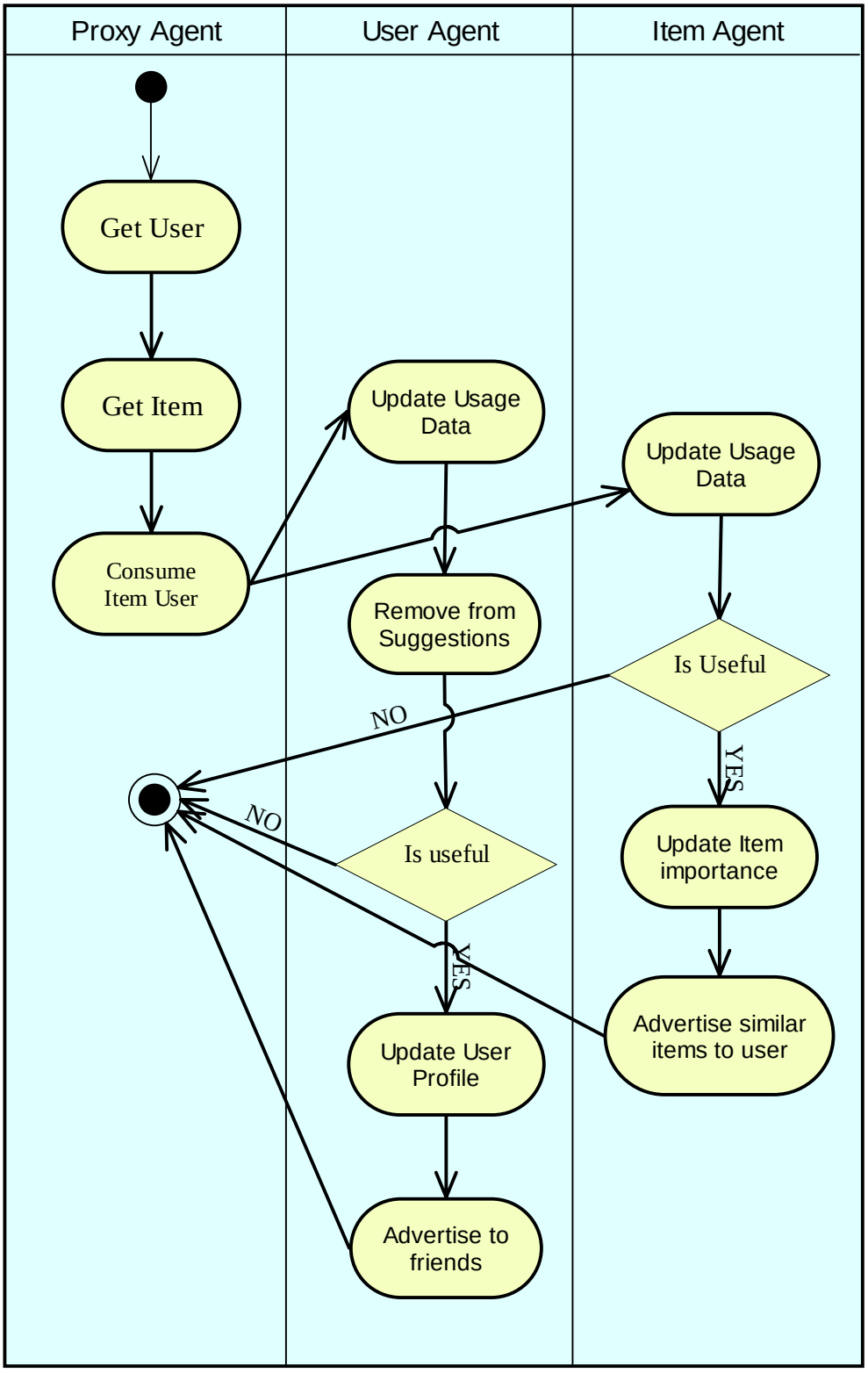
Αρχικά μέσω τις υπηρεσίας DF αναζητούνται αντικείμενα στα οποία αποστέλλονται μηνύματα με τα κριτήρια αναζήτησης και κατόπιν γίνετε συλλογή των απαντήσεων. Ο κάθε πράκτορας

αντικειμένου που λαμβάνει ένα τέτοιο μήνυμα υπολογίζει με βάση το μοντέλο p-norm πόσο σχετικό είναι με τα δοθέντα κριτήρια και στέλνουν πίσω την απάντησή τους. Αφού γίνει η συλλογή και κατηγοριοποίηση στον proxy σε τρεις κατηγορίες επιλέγονται τα αντικείμενα τις πιο σχετική κατηγορίας. Αυτά τα αντικείμενα αποστέλλονται στον χρήστη. Ο χρήστης υπολογίζει αρχικά τιμές κριτηρίων ανάλογα με το προφίλ του με βάση το p-norm μοντέλο και στη συνέχεια κατατάσσει τα αντικείμενα με βάση τις χρησιμότητες που υπολογίζει για κάθε ένα.

Αξιολόγηση Αντικειμένων

Η αξιολόγηση αντικειμένων επιτυγχάνεται με την αντίστοιχη συμπεριφορά του πράκτορα Proxy που αναλαμβάνει ρόλο μεσολαβητή ανάμεσα στους πράκτορες και την εφαρμογή διεπαφής με τους χρήστες. Όταν ο πράκτορας Proxy λάβει αίτηση για αξιολόγηση τότε ενεργοποιεί αυτήν την συμπεριφορά για κάποιον χρήστη και κάποιο αντικείμενο και με κάποιες τιμές αξιολόγησης.

Το παρακάτω UML διάγραμμα ενεργειών καταδεικνύει τις ενέργειες που χρειάζονται για να γίνει μια αξιολόγηση. Αρχικά, ζητείται από τους πράκτορες να δώσουν τα δεδομένα τους (τα δεδομένα του χρήστη και τα δεδομένα του αντικειμένου). Μόλις ολοκληρωθεί αυτή η διαδικασία στέλνονται ένα μήνυμα με την ενέργεια Consume με παραλήπτες τους πράκτορες χρήστη και αντικειμένου. Αυτοί οι πράκτορες ανάλογα με την κρίση που έχει γίνει στο αντικείμενο (αν είναι σημαντικό), ενημερώνει το προφίλ προσθέτοντας τα ανάλογα πεδία με μικρό βάρος ή ενημερώνει υπάρχοντα προσαρμόζοντας ανάλογα το βάρος. Επίσης, διαφημίζει αυτά τα αντικείμενα σε φίλους πράκτορες. Αντίστοιχά, ενημερώνετε και η γενική αξιολόγηση ενός άρθρου και διαφημίζονται σε αυτόν τον χρήστη σχετικά αντικείμενα.

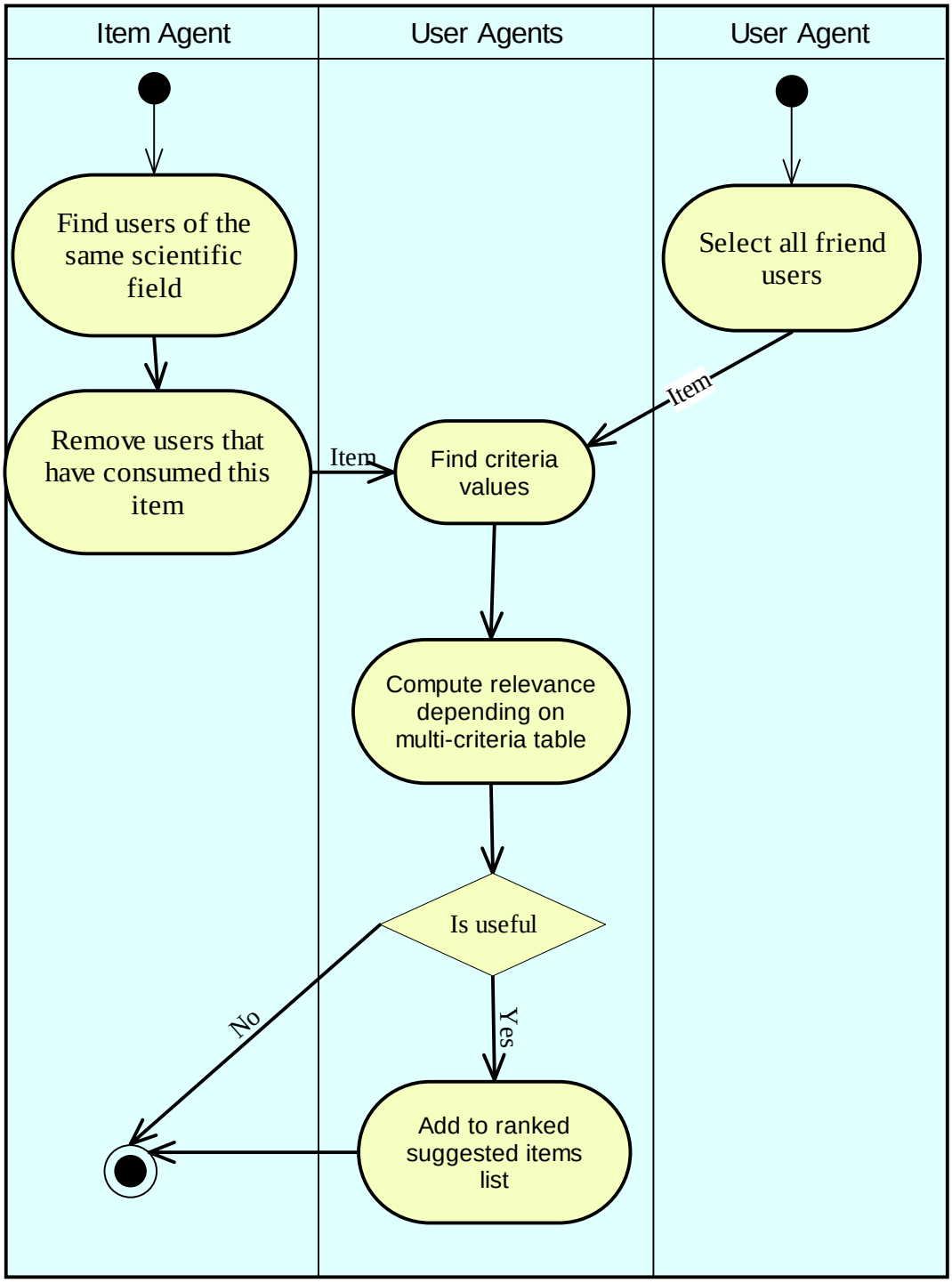


Σχήμα 6: UML διάγραμμα ενεργειών για την αναζήτηση αντικειμένων με βάση κάποια κριτήρια και το προφίλ ενός χρήστη.

Διαφήμιση Αντικειμένων

Η διαφήμιση ενός αντικείμενου σε κάποιους χρήστες που ενδιαφέρονται για αυτό επιτυγχάνετε με αντίστοιχη συμπεριφορά που έχουν οι πράκτορες αντικείμενο και χρήστης. Αυτή η συμπεριφορά ενεργοποιείται όταν δημιουργείται ένα αντικείμενο και όταν καταναλώνετε ένα χρήσιμο αντικείμενο.

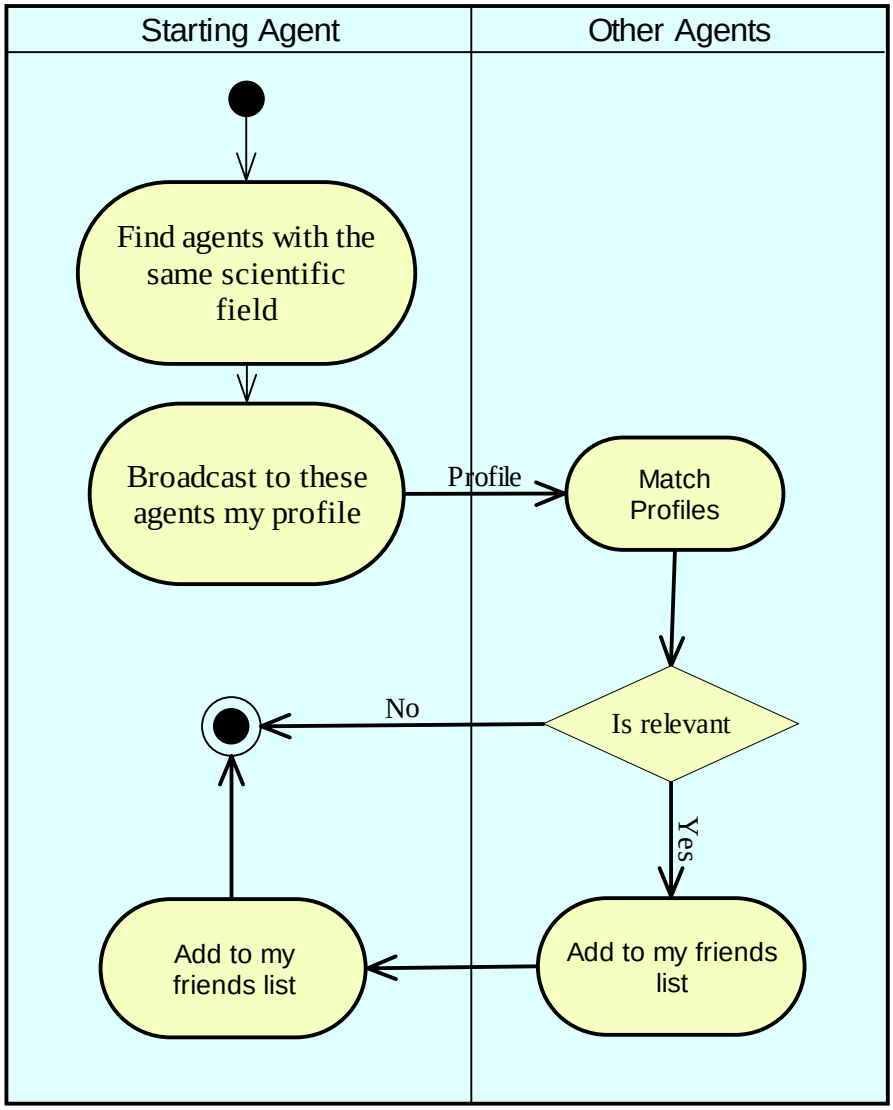
Το παρακάτω UML διάγραμμα ενεργειών καταδεικνύει τις ενέργειες που χρειάζονται για να διαφημιστεί ένα αντικείμενο. Η διαδικασία αυτή μπορεί να ξεκινήσει με διάφορους τρόπους ανάλογα με τον πράκτορα. Συγκεκριμένα μπορεί να ξεκινήσει είτε από ένα αντικείμενο στην αρχή του, είτε από ένα αντικείμενο όταν καταναλωθεί το ίδιο, είτε, τέλος, από έναν χρήστη όταν καταναλώσει ένα αντικείμενο και θέλει να το στείλει σε φίλους του. Επιλέγονται, λοιπόν, οι χρήστες που θα τους σταλεί το αντικείμενο, και το ή τα αντικείμενα που θα σταλούν. Η επιλογή των χρηστών γίνετε, ανάλογα με την περίπτωση, είτε από τα δεδομένα στην διαδικασία κατανάλωσης, είτε μέσω της υπηρεσίας DF για κάποια επιστημονικά πεδία. Ο πράκτορας χρήστης όταν λάβει ένα αντικείμενο ως πρόταση πρώτα το περνά από μια διαδικασία προσδιορισμού της χρησιμότητάς τους και εάν είναι χρήσιμο το προσθέτει στην ταξινομημένη λίστα με τις προτάσεις.



Σχήμα 7: UML διάγραμμα ενεργειών για την διαφήμιση αντικειμένων με βάση κάποια κριτήρια και το προφίλ ενός χρήστη.

Εύρεση Φίλου

Η διαδικασία εύρεσης ενός φίλου αφορά τόσο τους χρήστες όσο και τα αντικείμενα. Στα αντικείμενα αναφέρεται σε εύρεση σχετικών, ενώ στους χρήστες σε εύρεση χρηστών με παρόμοιο προφίλ. Αυτή η συμπεριφορά ενεργοποιείται όταν δημιουργείται ένα αντικείμενο ή ένας χρήστης. Το παρακάτω UML διάγραμμα ενεργειών καταδεικνύει τις ενέργειες που χρειάζονται για να βρεθεί ένας φίλος.



Σχήμα 8: UML διάγραμμα ενεργειών για την εύρεση φίλων με βάση κάποια κριτήρια και το προφίλ ενός χρήστη.

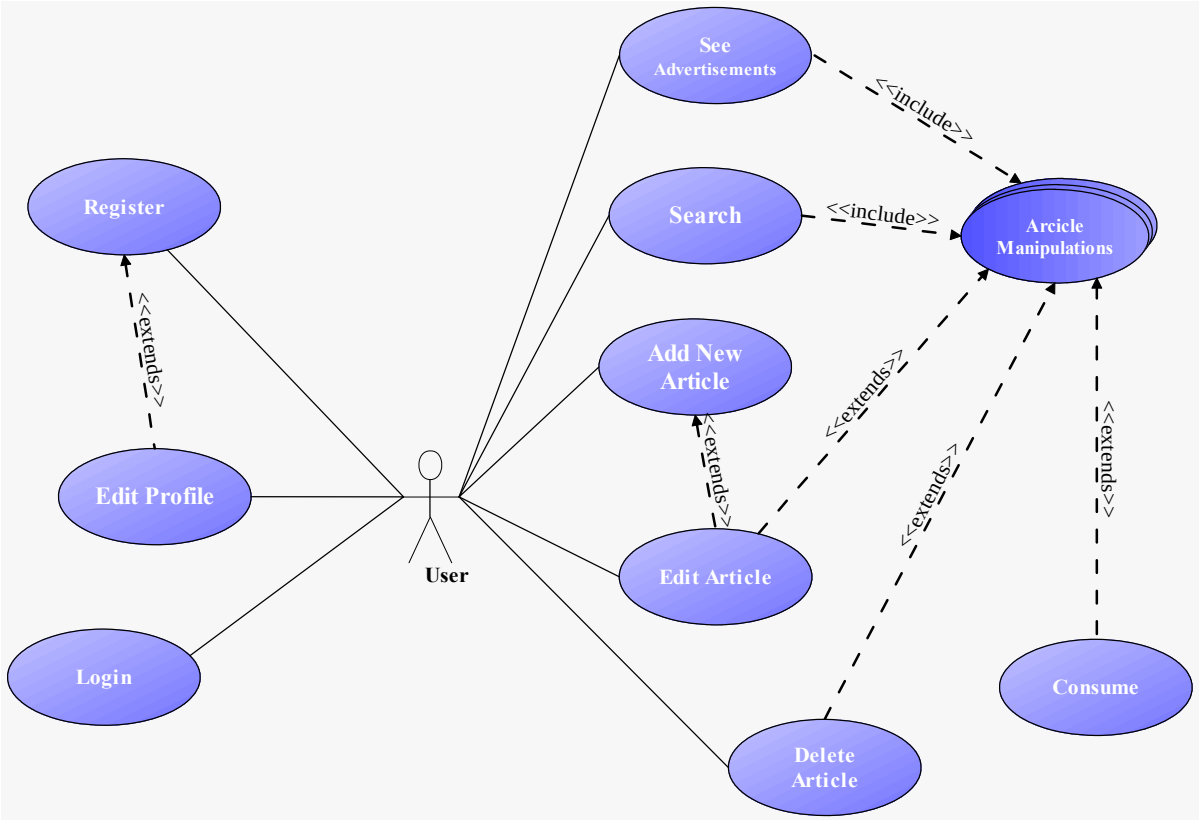
Αρχικά επιλέγονται οι παραλήπτες πράκτορες με κοινά επιστημονικά πεδία και αποστέλλετε σε όλους αυτούς το προφίλ του χρήστη ή το αντικείμενο (ανάλογα με τον τύπο του πράκτορα). Οι παραλήπτες μόλις λάβουν το μήνυμα υπολογίζουν με βάση το p-norm

μοντέλο πόσο κοινά είναι τα προφίλ μεταξύ τους και ανάλογα είτε το προσθέτουν στην λίστα με τους φίλους τους και ζητάνε και από τον αποστολέα να κάνει το ίδιο είτε δεν κάνουν τίποτα.

Η ΕΦΑΡΜΟΓΗ ΙΣΤΟΥ

Σε αυτήν την παράγραφο θα δούμε τον σχεδιασμό της εφαρμογής ιστού. Επειδή πρόκειται για μία εφαρμογή που αφόρα τον χρήστη μας ενδιαφέρει να την σχεδιάσουμε από την δική του σκοπιά. Δηλαδή, τι μπορεί να κάνει ο χρήστης και ποια είναι τα βήματα που πρέπει να ακολουθήσει για να επιτύχει αυτούς τους στόχους.

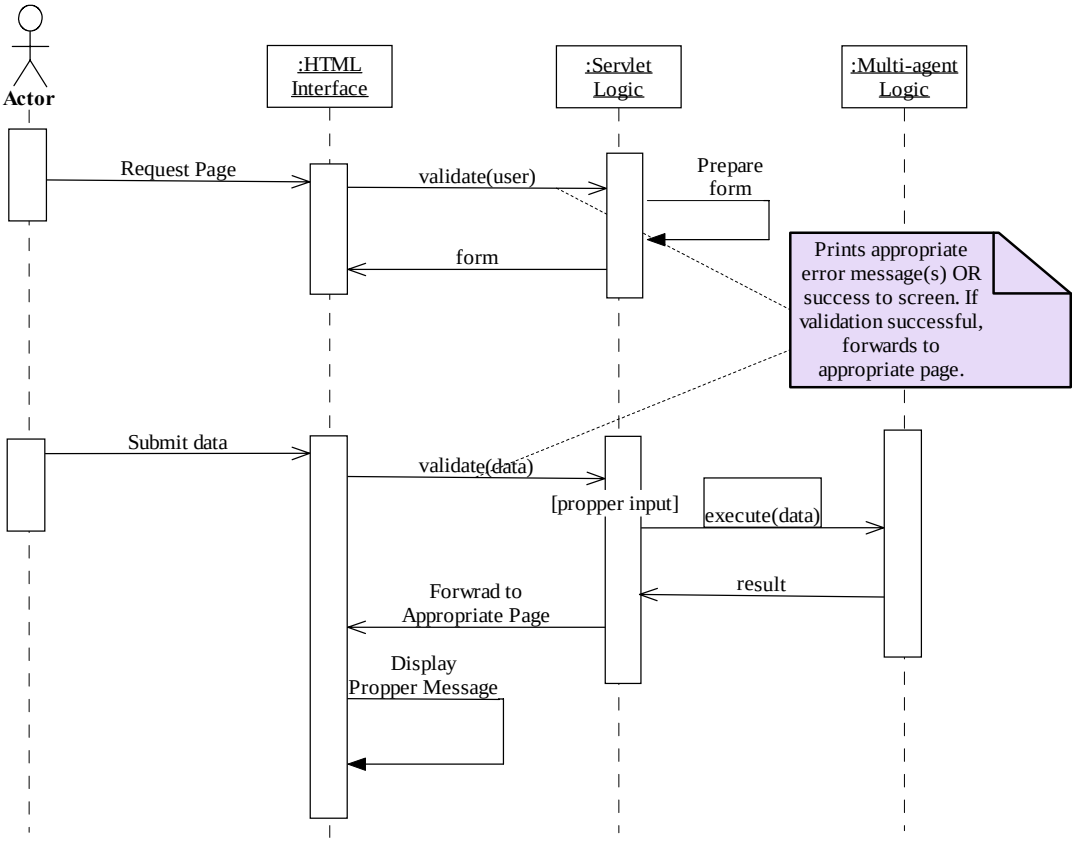
Το παρακάτω UML διάγραμμα περιπτώσεων χρήσης (use case) δείχνει όλες τις δυνατότητες που έχει ένας χρήστης πάνω στο σύστημα. Συγκεκριμένα ένας χρήστης μπορεί να διαπιστευτεί στο σύστημα (login), να δημιουργήσει ένα καινούργιο λογαριασμό (register), να αλλάξει το προφίλ του, να εισαγάγει ένα καινούργιο άρθρο, να διαγράψει ένα υπάρχον άρθρο, να αλλάξει επιμέρους πεδία κάποιου άρθρου, να αναζητήσει άρθρα, και, τέλος, να δει προτάσεις του συστήματος (advertisements). Επιπλέον μετά την αναζήτηση ή την επισκόπηση κάποιου άρθρου μπορεί να το καταναλώσει, να το διαγράψει, ή να αλλάξει επιμέρους πεδία.



Σχήμα 9: UML διάγραμμα περιπτώσεων χρήσης του συστήματος.

Ο χρήστης όμως δεν μπορεί να κάνει όλες αυτές τις ενέργειες με τυχαίο τρόπο, αλλά πρέπει πρώτα από όλα είτε να διαπιστευτεί στο σύστημα (login) είτε να δημιουργήσει καινούργιο λογαριασμό (register). Από το σημείο εκείνο και έπειτα μπορεί να εκτελέσει οποιαδήποτε άλλη ενέργεια με δικιά του επιλογή.

Η κάθε μία από τις παραπάνω ενέργειες εκτελείται με μία συγκεκριμένη σειρά βημάτων που εξασφαλίζουν ότι ο χρήστης είναι διαπιστευμένος στο σύστημα άλλα και ότι τα δεδομένα που υποβάλει είναι ορθά ώστε να εκτελεστεί τελικά από το πολύ-πρακτορικό περιβάλλον. Αυτή η διαδικασία παρουσιάζεται στο παρακάτω UML ακολουθιακό διάγραμμα (sequence diagram).



Σχήμα 10: UML ακολουθιακό διάγραμμα (sequence diagram) για την αλληλεπίδραση με τον χρήστη της εφαρμογής ιστού.

Παρατηρούμε, λοιπόν, ότι γίνετε έλεγχος σε κάθε σελίδα που ζητείται από τον χρήστη (εκτός από την αρχική σελίδα για να διαπιστευτεί) ότι γίνετε έλεγχος αν είναι έγκυρος (διαπιστευμένος) και αν ισχύει αυτό τότε ανάλογα με το αίτημα του χρήστη ετοιμάζουμε την σωστή φόρμα και την δείχνουμε στον χρήστη. Αυτός την γεμίζει ή την χρησιμοποιεί και μας στέλνει κάποια δεδομένα. Η λογική της εφαρμογής ελέγχει ως προς την ορθότητά τους αυτά τα δεδομένα (αν έχουν σωστές τιμές, κλπ.) και αν είναι έγκυρα τότε ζητάει από το πολύ-

πρακτορικό σύστημα να εκτελέσει την αντίστοιχη ενέργεια. Αν αυτή έχει αποτελέσματα τότε αυτά επεξεργάζονται από το servlet, αποφασίζεται πια θα είναι η επόμενη σελίδα, αποθηκεύονται κάποια μηνύματα για τον χρήστη στην σελίδα ανάλογα με την έκβαση της διαδικασίας και όλα αυτά στέλνονται πίσω στην ιστοσελίδα, η οποία τα δείχνει όλα αυτά κατάλληλα.

ΑΝΑΚΕΦΑΛΑΙΩΣΗ

Σε αυτό το κεφάλαιο παρουσιάστηκε ολόκληρος ο σχεδιασμός της εφαρμογής μας. Συγκεκριμένα, είδαμε τόσο την γενική αρχιτεκτονική ολόκληρου του συστήματος όσο και επιμέρους για το πολύ-πρακτορικό σύστημα και την εφαρμογή ιστού. Σημαντικό στοιχείο του σχεδιασμού ήταν ο καταμερισμός του συστήματος σε τρία επίπεδα και ο καθορισμός των δυνατών ενεργειών κάθε ενός επιπέδου ξεχωριστά με βάση την λειτουργικότητα που είχε οριστεί στο προηγούμενο κεφάλαιο.

Στο επόμενο κεφάλαιο θα περιγραφεί ο τρόπος που υλοποιήθηκαν τόσο η αρχιτεκτονική και τα επιμέρους επίπεδα και ενέργειες, όσο και επιμέρους αλγόριθμοι που αναπτύχθηκαν.

ΚΕΦΑΛΑΙΟ Ε' – ΥΛΟΠΟΙΗΣΗ

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο θα αναφερθώ στην υλοποίηση του σχεδιασμού που έγινε ώστε οι στόχοι τις εργασίας να επιτευχθούν. Διάφορα ζητήματα που δεν απασχόλησαν ιδιαίτερα τον σχεδιασμό έπρεπε να ληφθούν υπόψιν και σε αυτά θα δοθεί μεγαλύτερη έμφαση. Η αρχιτεκτονική θα αναλυθεί περεταίρω ώστε όλες οι διεπαφές μεταξύ των μονάδων να καταστούν σαφείς. Η υλοποίηση του πολύ-πρακτορικού περιβάλλοντος, η επικοινωνία μεταξύ των πρακτόρων καθώς ιδιαιτέρες μονάδες που χρειάστηκαν θα αναλυθεί, ενώ θα περιγραφεί και ο μηχανισμός που καταστεί το σύστημα σταθερό από πτώσεις (crashes). Τέλος, η εφαρμογή δικτύου που υλοποιήθηκε θα αναλυθεί ως προς την αρχιτεκτονική που χρησιμοποιεί.

Όπως είχε φανεί και κατά τον σχεδιασμό της αρχιτεκτονικής το σύστημα διακρίνετε σε διάφορα επίπεδα (tiers) που το καταστούν πιο ευέλικτο. Το πρωτόκολλο επικοινωνίας μεταξύ των επιπέδων μας επιτρέπει σε μεταγενέστερη φάση να αλλάξουμε τον σχεδιασμό και την υλοποίηση καθενός επιπέδου ξεχωριστά χωρίς να επηρεάζονται τα υπόλοιπα. Το ενδιαφέρον με την υλοποίηση αυτού του σχεδιασμού είναι ουσιαστικά ότι προσφέρετε πάνω από την πολύ-πρακτορική πλατφόρμα ένας wrapper που αναλαμβάνει όλη την βρώμικη δουλειά αγνοώντας το περιεχόμενο και την λογική τόσο της πολύ-πρακτορικής εφαρμογής, όσο και των υπολοίπων επιπέδων.

Οι βασικές αρχές που ορίστηκαν στο σχεδιασμό διατηρούνται και στην υλοποίηση ενώ έμφαση θα δοθεί στην υλοποίηση των αλγορίθμων UTA* και p-norm. Ακόμα το πρωτόκολλο επικοινωνίας με την μονάδα υπολογισμού UTA* θα περιγραφεί, καθώς και εναλλακτικοί μηχανισμοί χρήσης του.

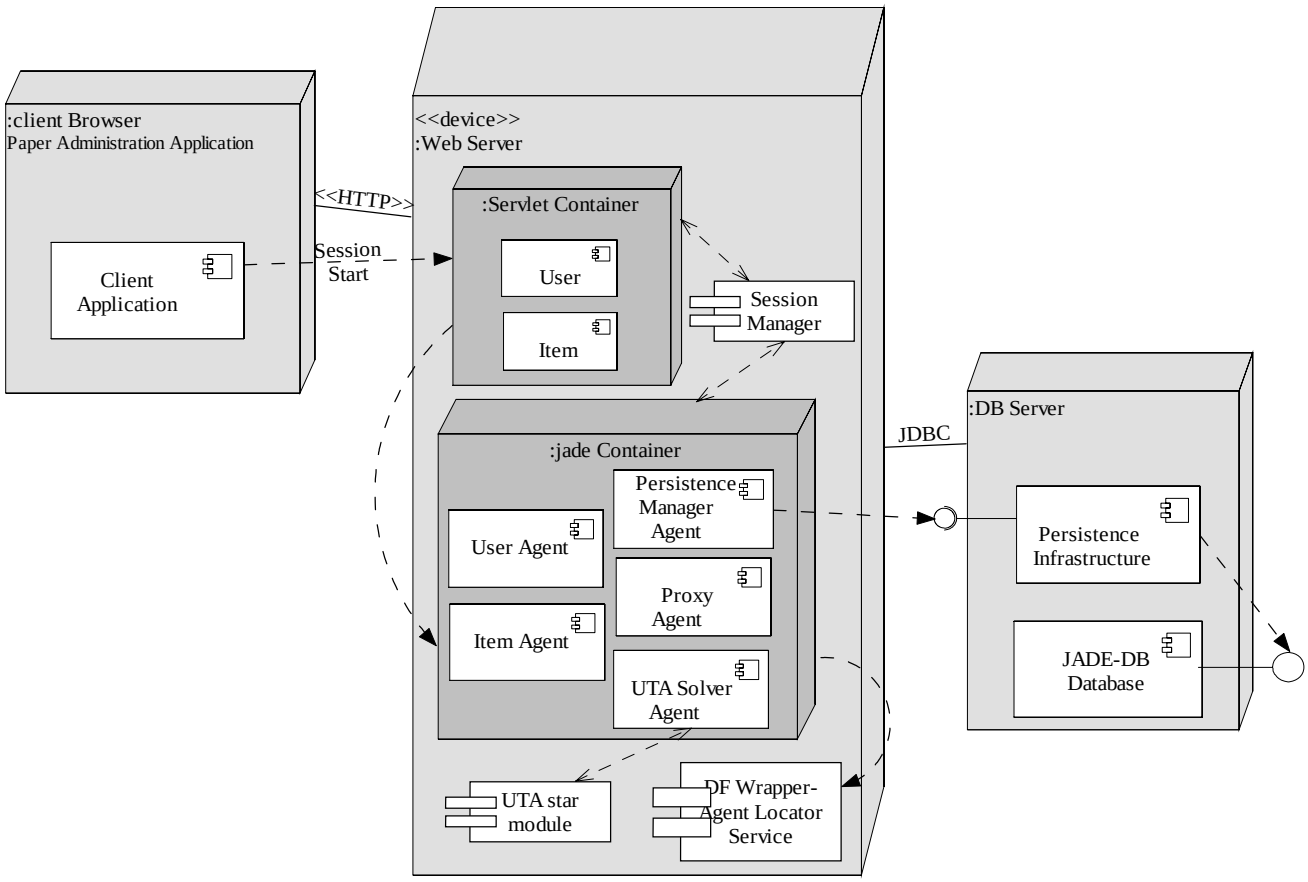
Όλη η υλοποίηση έγινε πάνω από Java 2 και χρησιμοποιήθηκαν η πλατφόρμα JADE [5] για το πολύ-πρακτορικό περιβάλλον μαζί με το πρότυπο FIPA [15] για την επικοινωνία μεταξύ των πρακτόρων. Για την αποθήκευση της κατάστασης του συστήματος χρησιμοποιήθηκε hypersonic SQL database engine [16] μαζί με το persistence add-in του Jade. Η εφαρμογή δικτύου υλοποιήθηκε με την τεχνολογία Java Servlet [19] σε συνδυασμό με την βιβλιοθήκη Jakarta Struts [6], όλα του Apache project [17]. Η οντολογία του πολύ-πρακτορικού περιβάλλοντος υλοποιήθηκε σε OWL, ενώ η παραγωγή των beans από την οντολογία έγινε με

χρήση του Bean Generator plugin του Protégé [8] και του Jade. Όλη η εφαρμογή εκτελείτε από τον Apache Tomcat [18].

Στις επόμενες παραγράφους θα περιγραφεί πρώτα η υλοποίηση της γενικής αρχιτεκτονικής, μετά η υλοποίηση αναφορικά με το πολύ-πρακτορικό περιβάλλον και το πρωτόκολλο επικοινωνίας μέσω του wrapper. Θα ακολουθήσει η περιγραφή της εφαρμογής δικτύου και της διαχείρισης της βάσης δεδομένων, ενώ τέλος, θα περιγραφεί το μοντέλο της UTA* και το πρωτόκολλό επικοινωνίας με την αυτοτελή αυτή μονάδα.

ΑΡΧΙΤΕΚΤΟΝΙΚΗ

Σε αντίθεση με την αρχιτεκτονική που παρουσιάσαμε στο προηγούμενο κεφάλαιο (σχεδιασμός), αυτή που θα παρουσιάσουμε εδώ είναι πιο ακριβής και τυπική στον καθορισμό των επιμέρους μονάδων και στην αλληλεπίδραση μεταξύ τους. Το παρακάτω σχήμα παρουσιάζει αυτήν την φορμαλισμένη περιγραφή.



Σχήμα 11: UML διάγραμμα με την γενική αρχιτεκτονική του συστήματος.

Το σύστημα χρησιμοποιείτε από έναν απλό browser όπως ο Internet Explorer της Microsoft ή ο Firefox της Mozilla. Ο τελικός χρήστης μπορεί να διαχειριστεί το σύστημα και να χειριστεί το προφίλ του απομακρυσμένα μέσα από δυναμικές ιστοσελίδες.

Η εφαρμογή βρίσκετε πάνω στο εξυπηρετητή ιστού Tomcat της Apache. Αυτός διαχειρίζεται έναν servlet container μέσα στο όποιον διατηρούνται sessions για τους συνδεδεμένους χρήστες και διάφορα αλλά απαραίτητα για την εφαρμογή αντικείμενα.

Μέσα από τον εξυπηρετητή φορτώνετε και τον περιβάλλον των πρακτόρων (JADE container) όπου ζουν οι πράκτορες και ενεργούν σύμφωνα με τη λογική τους εξυπηρετώντας της αιτήσεις της εφαρμογής ιστού καθώς και εσωτερικές διαδικασίες που άπτονται της λογικής κάθε μεμονωμένου πράκτορα. Εδώ ζει και ο πράκτορας που διαχειρίζεται την σταθερότητα του συστήματος σώζοντας την κατάστασή του με βάση τη λογική του.

Υπάρχει επίσης μία μονάδα λογισμικού που διαχειρίζεται την επικοινωνία μεταξύ της front-end εφαρμογής και των πρακτόρων με χρήση sessions. Σημειωτέον ότι αυτά τα sessions δεν σχετίζονται με τα sessions της εφαρμογής ιστού.

Η μονάδα που κατασκευάζει και επιλύει προβλήματα UTA* είναι επίσης ανεξάρτητη από το πολύ-πρακτορικό περιβάλλον και θα αναλυθεί στο πρώτο παράρτημα.

Η βάση δεδομένων στην οποία διατηρείται η κατάσταση του πολύ-πρακτορικού περιβάλλοντος ονομάζετε JADE-DB ενώ την διαχειρίζεται ένα Java DBMS (DataBase Management System) η Hypersonic database engine. Το σύστημα διαχείρισης καθώς και η βάση είναι εκτός του server και η επικοινωνία γίνετε με το πρωτόκολλο JDBC (java database connectivity).

Λεπτομέρειες για την υλοποίηση όλων αυτών μονάδων και της λογικής τους ακολουθούν στις επόμενες παραγράφους.

ΠΟΛΥ-ΠΡΑΚΤΟΡΙΚΟ ΣΥΣΤΗΜΑ

Το πολυ-πρακτορικό σύστημα είναι η καρδιά της εφαρμογής, εμπεριέχει, δηλαδή, όλο την λογική της. Στο σύστημα αυτό διάφορες μονάδες λογισμικού, πράκτορες, αυτενεργούν και συνεργάζονται για να προσφέρουν μία υπηρεσία στους τελικούς αποδέκτες της εφαρμογής. Η αυτενέργεια καθορίζετε μέσα από κάποιες συμπεριφορές που μεταβάλλουν ή αφήνουν άθικτη

την κατάσταση του συστήματος. Θυμίζω ότι η κατάσταση του συστήματος ορίζετε από το σύνολο των μη παροδικών δεδομένων που διατηρούν οι πράκτορες. Επίσης, στην κατάσταση του συστήματος ανήκει και το σύνολο των εκκρεμών μηνυμάτων. Η συνεργασία των πρακτόρων καθορίζετε από ένα συγκεκριμένο σύνολο μηνυμάτων που ανταλλάζουν για την επίτευξη κάποιου συνολικού σκοπού.

Και για τις δύο αυτές περιπτώσεις έχουν χρησιμοποιηθεί συγκεκριμένες πλατφόρμες και πρότυπα. Το γενικότερο δε πολυ-πρακτορικό σύστημα λειτουργεί πάνω από την πλατφόρμα JADE [5]. Το πρότυπο για το σύστημα είναι το FIPA (Foundation for Intelligent Agents) [15].

Το FIPA μας επιτρέπει να ορίσουμε μία «γλώσσα» επικοινωνίας μεταξύ των πρακτόρων χρησιμοποιώντας οντολογίες. Αυτή η γλώσσα ή οντολογία περιλαμβάνει ένα σύνολο αντικειμένων (concepts) που μπορούν να ανταλλάζονται με συγκεκριμένα πεδία ή ιδιότητες (properties), ένα σύνολο ενεργειών που μπορούν να εκτελεστούν (actions), καθώς και ένα σύνολο ερωτήσεων και κατηγορημάτων με τα όποια ένας πράκτορας είτε ερωτάτε για κάτι, είτε ενημερώνετε για κάτι άλλο (predicates).

Οι πράκτορες δημιουργούνται και ανήκουν κατά ομάδες σε ένα πλαίσιο που ονομάζετε κιβώτιο (container). Για να μπορέσει ένας πράκτορας να βρει (discover) κάποιον που να προσφέρει συγκεκριμένες υπηρεσίες χρειάζεται ένα σύστημα ανεξάρτητο από τον container όπου ο κάθε πράκτορας εγγράφετε σε αυτό μαζί με τις υπηρεσίες που προσφέρει (template) και λειτουργεί σαν yellow pages. Αυτό είναι ανάλογο των name services (NS) των κατανεμημένων συστημάτων. Η πλατφόρμα Jade προσφέρει ένα συγκεκριμένου τύπου υπηρεσίας που ονομάζετε DF. Η υπηρεσία αυτή είναι γνωστή σε όλους τους πράκτορες και μέσω αυτού του μηχανισμού οι πράκτορες μπορούν να ανακαλύπτουν αυτούς που χρειάζονται. Υλοποιήθηκε ένας wrapper της υπηρεσίας DF που να διαχειρίζεται τα templates των πρακτόρων και να τα κατασκευάζει με αδιάφανο τρόπο για αυτούς. Σημειωτέον ότι η κατάσταση της υπηρεσίας DF, δηλαδή οι εγγραφές των πρακτόρων σε αυτήν, δεν αποτελεί μέρος της γενικότερης κατάστασης του συστήματος μας, διότι αναπαράγετε αυτόματα.

Τέλος, το πολυ-πρακτορικό σύστημα είναι περιτυλιγμένο γύρω από έναν wrapper, JadeSystemWrapper, ο όποιος αναλαμβάνει να διαχειριστεί με αδιαφανεί και ενιαίο τρόπο για τους πράκτορες το σύστημα Jade. Με αυτόν τον τρόπο δίνονται συγκεκριμένες δυνατότητες στους πράκτορες περιορίζοντας τις ελευθέριες τους για συγκεκριμένες ενέργειες.

Στις επόμενες παραγράφους ακολουθεί λεπτομερείς παρουσίαση των ενεργειών (actions), των κατηγορημάτων (predicates) και των αντικειμένων (concepts) που ορίσθηκαν για την επικοινωνία των πρακτόρων. Ακολουθεί περιγραφή του μηχανισμού και των τύπων ανταλλαγής μηνυμάτων που υλοποιήθηκαν καθώς και λεπτομερής αναφορά στους πράκτορες και τις συμπεριφορές τους. Ο μηχανισμός ανεύρεσης και οι τύποι των templates περιγράφονται στη συνέχεια, ενώ τέλος δίνεται και το πλαίσιο αναφοράς των πρακτόρων μέσω του wrapper.

WRAPPER

Σε αυτήν την παράγραφο θα αναφερθούμε στην μονάδα λογισμικού που επιτρέπει στο σύστημα να διαχειρίζεται το περιβάλλον Jade μέσα στο οποίο ζουν οι πράκτορες.

Η μονάδα αυτή συνιστάτε από μέρη που καθορίζουν τον κύκλο ζωής των πρακτόρων, την διαχείριση των πακέτων που οργανώνονται οι πράκτορες (containers), τη διαχείριση των πρακτόρων που ζουν στο περιβάλλον και ποιοι μπορούν να δημιουργηθούν σε αυτό. Ουσιαστικά σε αυτό το κομμάτι «ζει» το περιβάλλον Jade με ένα τρόπο που είναι προσπελάσιμος από όλους τους πράκτορες και τις άλλες μονάδες λογισμικού, είναι ένα κοινό σημείο αναφοράς όλου του περιβάλλοντος που προσφέρει ένα σύνολο διευκολύνσεων σε όλους. Χρειαζόταν ένας μηχανισμός να δημιουργούμε, να σκοτώνουμε και γενικότερα να διαχειριζόμαστε τους πράκτορες χωρίς να μας απασχολούν θέματα που άπτονται το περιβάλλον Jade. Έτσι, με την χρήση του wrapper γίνετε διαχείριση του συστήματος Jade ώστε να λειτουργεί σωστά.

ΑΝΕΥΡΕΣΗ ΠΡΑΚΤΟΡΩΝ

Για να επικοινωνήσουν και τελικά για να συνεργαστούν δύο πράκτορες πρέπει ο ένας να γνωρίζει την ύπαρξη του άλλου καθώς επίσης και την συγκεκριμένη διεύθυνση του μέσα στο περιβάλλον Jade. Για την λύση αυτού του προβλήματος, το οποίο εμφανίστηκε πολύ παλιά σε άλλες περιοχές, κατά καιρούς έχουν προταθεί πολλές και διαφορετικές μέθοδοι. Σε εμάς και πάνω στο περιβάλλον Jade υπάρχει ένας μηχανισμός που λειτουργεί ανάλογα με τα naming services των λειτουργικών συστημάτων ή των application servers, η υπηρεσία DF. Πάνω από αυτήν την υπηρεσία έχει φτιαχτεί ένας wrapper που μοντελοποιεί τον τρόπο που διαχειρίζονται την υπηρεσία οι πράκτορες.

Αυτό που γίνετε ακριβώς είναι κάθε πράκτορας όταν ενδιαφέρετε να βρεθεί από άλλους εγγράφετε στην υπηρεσία (μέσω του wrapper) με κάποια επιπλέον πληροφορία που

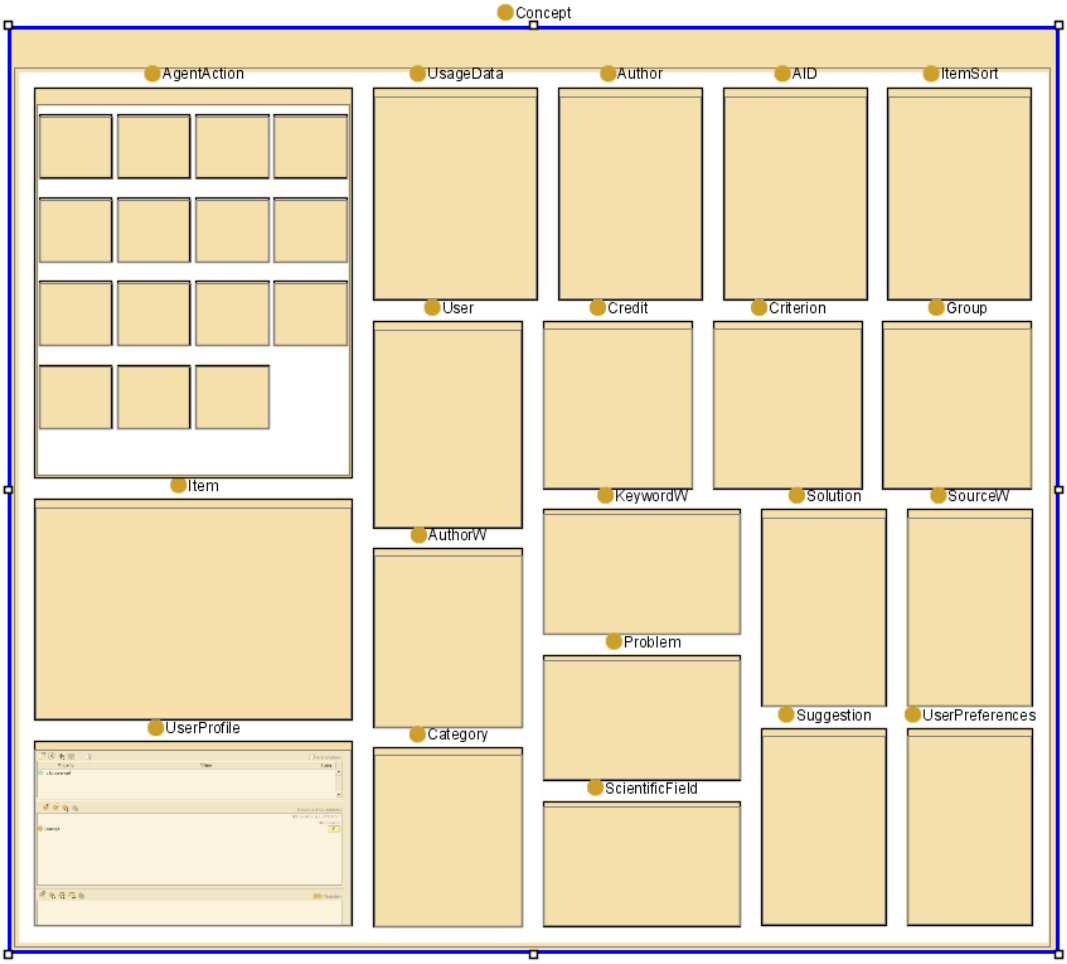
περιγράφει ποιος είναι, τι είναι και ενδεχομένως τι κάνει. Αυτό το δεύτερο κομμάτι το αναλαμβάνει αυτόματα ο wrapper. Μετά όταν κάποιος θέλει να βρει κάποιον ή κάποιους πράκτορες που να κάνουν κάτι συγκεκριμένο καλεί την αντίστοιχη μέθοδο του wrapper και παίρνει ένα σύνολο με τις διευθύνσεις των πρακτόρων που πληρούν τα κριτήρια που έθεσε.

ΓΛΩΣΣΑ ΕΠΙΚΟΙΝΩΝΙΑΣ - ΟΝΤΟΛΟΓΙΑ

Στην οντολογία, όπως έχει ήδη αναφερθεί, ορίζονται τα αντικείμενα που ανταλλάζονται μέσω μηνυμάτων, οι δυνατές ενέργειες των πρακτόρων καθώς και ερωτήσεις και δελτία ενημέρωσης των πρακτόρων. Η οντολογία λειτουργεί σαν την κοινή γλώσσα που καταλαβαίνουν οι πράκτορες. Σε αυτήν την παράγραφο θα παρουσιάσουμε μέσω πινάκων αυτήν την γλώσσα. Η οντολογία υλοποιήθηκε σε Protégé [8] ενώ χρησιμοποιήθηκε το bean generator επιπρόσθετο του Jade για να παραχθούν java beans και οντολογικά αντικείμενα κατανοητά στο περιβάλλον του Jade.

Στο κεφάλαιο του σχεδιασμού είχαμε αναφερθεί στις έννοιες που ορίστηκαν στην οντολογία και στον σκοπό που εξυπηρετούν, ενώ σε αυτήν την παράγραφο θα εξετάσουμε τα επιμέρους πεδία που έχει η κάθε έννοια και τί αλληλουχία ενεργειών γεννά η κάθε ενέργεια. Για λόγους ευκολίας του αναγνώστη επαναλαμβάνονται οι έννοιες μέσω τριών σχημάτων. Μετά από κάθε σχήμα ακολουθεί η επιμέρους ανάλυση κάθε έννοιας.

Τα αντικείμενα



Σχήμα 12: Τα αντικείμενα που μπορούν να ανταλλάξουν ως πληροφορία οι πράκτορες, όπως ορίστηκαν στην οντολογία.

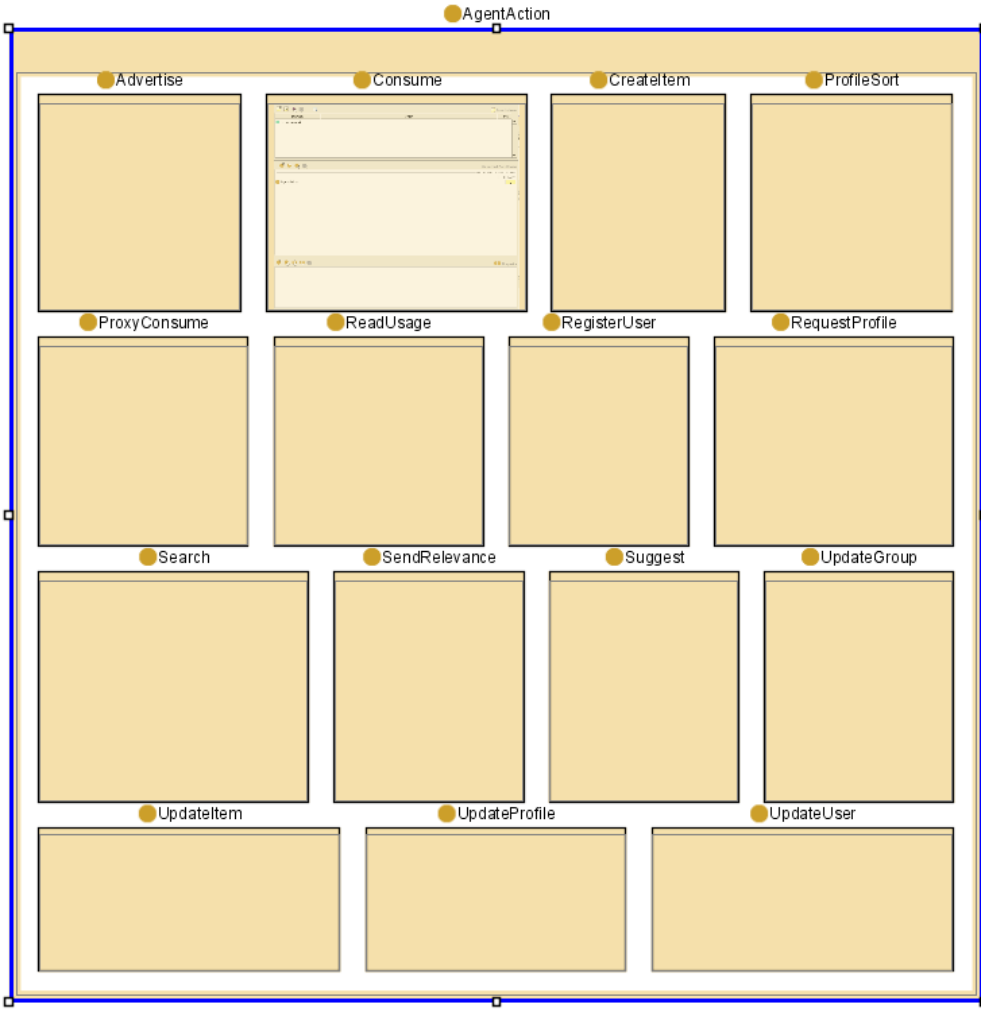
Το παραπάνω σχήμα εμφανίζει τις βασικές έννοιες που διατηρούν πληροφορία στο μοντέλο μας και χρησιμοποιούνται για ανταλλαγή πληροφορίας μεταξύ των πρακτόρων αλλά και του γενικότερου συστήματος. Συγκεκριμένα για κάθε ένα από αυτά θα παρατεθούν τα πεδία που τα απαρτίζουν ακολουθούν:

- AuthorW:
  - Weight, Το βάρος ενός συγγραφέα
  - Author, ο συγγραφέας
- SourceW
  - Weight, Το βάρος μίας πηγής
  - Source type, η πηγή
- Author
  - Affiliation, Οργανισμός
  - Name, Το όνομα του συγγραφέα
- Credit

    - Quality, Ποιότητα άρθρου
    - References, Πληρότητα βιβλιογραφίας
    - Total satisfaction, Συνολική ικανοποίηση
    - Genuine, Γνησιότητα
    - Originality, Πρωτοτυπία θέματος
    - User Importance, Προσωπικό ενδιαφέρον
    - Consistency, Συνέπεια
    - Size, Μέγεθος σε σχέση με την κάλυψη του θέματος
    - Science Importance, Συνεισφορά στην επιστήμη
- User Profile
    - Categories with weight, Λίστα κατηγοριών με βάρη
    - Sources with weight, Λίστα πηγών με βάρη
    - Keywords with weight, Λίστα λέξεων κλειδιών με βάρη
    - Authors with weight, Λίστα αρθρογράφων με βάρη
    - Last update, τελευταία ενημέρωση
- Item Sort
    - Weight, βάρος
    - Item, το αντικείμενο
- Item
    - Fields, Επιστημονικές περιοχές
    - File Type, τύπος αρχείου
    - Authors, Συγγραφείς
    - Affiliation, Οργανισμός
    - Credit, Αξιολόγηση
    - Synopsis, Περίληψη
    - Categories, Κατηγορίες με βάρη
    - Source type, Πηγές δημοσίευσης
    - URL, Η ηλεκτρονική διεύθυνση του άρθρου
    - Language, Γλώσσα
    - Keywords, Λέξεις κλειδιά
    - Item Id, Αναγνωριστικό του αντικειμένου
    - Publication Date, Ημερομηνία δημοσίευσης
    - Title, Τίτλος
- Usage Data
    - Accesses, Προσβάσεις
    - Item, Το αντικείμενο
    - Credit, Αξιολόγηση
- Group
    - Importance, Σημασία
    - Thresholds, Κατώφλια κριτηρίων
- Suggestion
    - Weight, βάρος
    - Item, αντικείμενο
- Keyword W
    - Weight, βάρος
    - Keyword, λέξη κλειδί
- Scientific Field
    - Name, όνομα
- Category
    - Name, κατηγορία

  - Weight, βάρος
- Criterion
  - Worst , χειρότερη τιμή
  - Best , καλύτερη
  - Name, όνομα
  - Steps, διαστήματα
- User
  - Email, Διεύθυνση αλληλογραφίας
  - Userid, Κωδικός χρήστη
  - User name, όνομα χρήστη
  - User profile, το προφίλ του χρήστη
  - User title, Τίτλος (Mr, Mrs, Prof)
  - Affiliation, οργανισμός
  - Real name, πραγματικό όνομα
  - Position, θέση
  - Password, συνθηματική λέξη

Οι ενέργειες των πρακτόρων



Σχήμα 13: Οι ενέργειες που μπορούν να ζητήσουν οι πράκτορες από άλλους πράκτορες, όπως ορίστηκαν στην οντολογία.

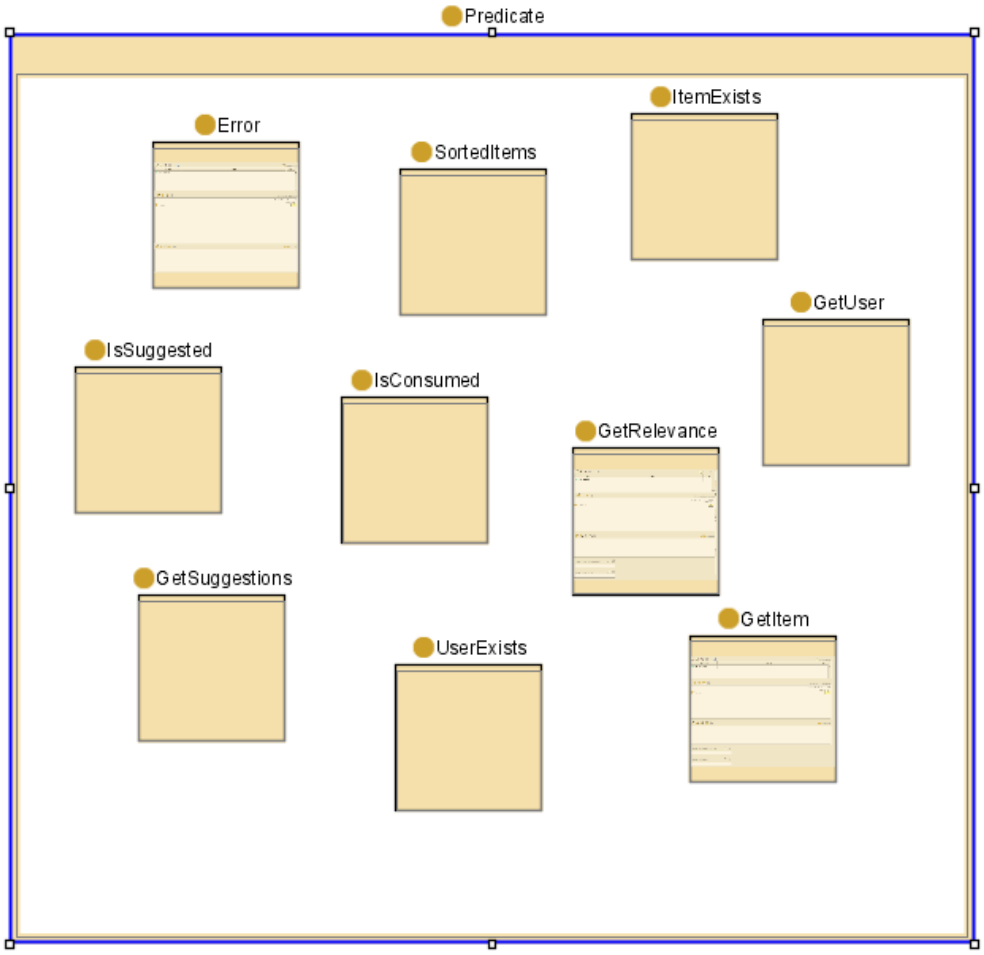
Οι ενέργειες που πραγματοποιούνται από τους πράκτορες και η αλληλουχία των ενεργειών που προκαλούν παρουσιάζονται παρακάτω:

- Request Profile, Ζητάει να ενημερώσει κατασκευάσει ένα UTA* πρόβλημα και να υπολογίσει την λύση.
    - Usage data Τα δεδομένα χρήσης
    - User profile, το προφίλ του χρήστη
- Read usage, Ζητάει το ιστορικό χρήσης για κάποιον χρήστη ή αντικείμενο
    - Owner ο χρήστης ή το αντικείμενο
- Update item, Ζητάει από ένα πράκτορα να ενημερώσει τα δεδομένα του.
    - Item, Το αντικείμενο με το οποίο θα ενημερωθεί.
- Update Profile, Ζητάει από έναν πράκτορα να ενημερώσει το προφίλ του
    - Session id, Η σύνοδος που αποθηκευτεί το προφίλ
- Advertise, Διαφημίζει ένα αντικείμενο
    - Item, το αντικείμενο
- Consume, Καταναλώνει ένα αντικείμενο
    - Item consumed, Το αντικείμενο
    - Consumer, Ο χρήστης
    - Credit, η αξιολόγηση
- Suggest, Ζητάει να επιστραφούν τα suggestions
    - User, ο χρήστης
- Proxy Consume Καταναλώνει ένα αντικείμενο από τον proxy
    - Item id, το διακριτικό του αντικειμένου
    - User id, το διακριτικό του χρήστη
    - Credit, η αξιολόγηση
- Search, Αναζητάει αντικείμενα
    - Categories with weight, Λίστα με κατηγορίες με βάρη
    - Sources with weight, Λίστα με πηγές με βάρη
    - Keywords with weight, Λίστα με λέξεις κλειδιά με βάρη
    - Authors with weight, Λίστα με συγγραφείς με βάρη
- Register user, Εγγράφει ένα χρήστη
    - Username, Το όνομα του χρήστη
    - Password, συνθηματικό
- Send relevance, Αποστέλλει την σχετικότητα ενός αντικειμένου

- Item , Το αντικείμενο
- Relevance, Η σχετικότητα

- Profile Sort, Κατατάσσει με βάση το προφίλ κάποια αντικείμενα
  - Items, τα αντικείμενα
- Create Item, Δημιουργεί ένα αντικείμενο
  - Item, Το αντικείμενο
- Find Friend , Αναζητάει ένα φίλο (χρήστη ή αντικείμενο) με βάση ένα προφίλ
  - Profile, Το προφίλ
- Update User, Ενημερώνει τα δεδομένα του χρήστη
  - User, Τα δεδομένα

Τα κατηγορήματα



Σχήμα 14: Τα κατηγορήματα που μπορούν να ζητήσουν οι πράκτορες από άλλους πράκτορες, όπως ορίστηκαν στην οντολογία.

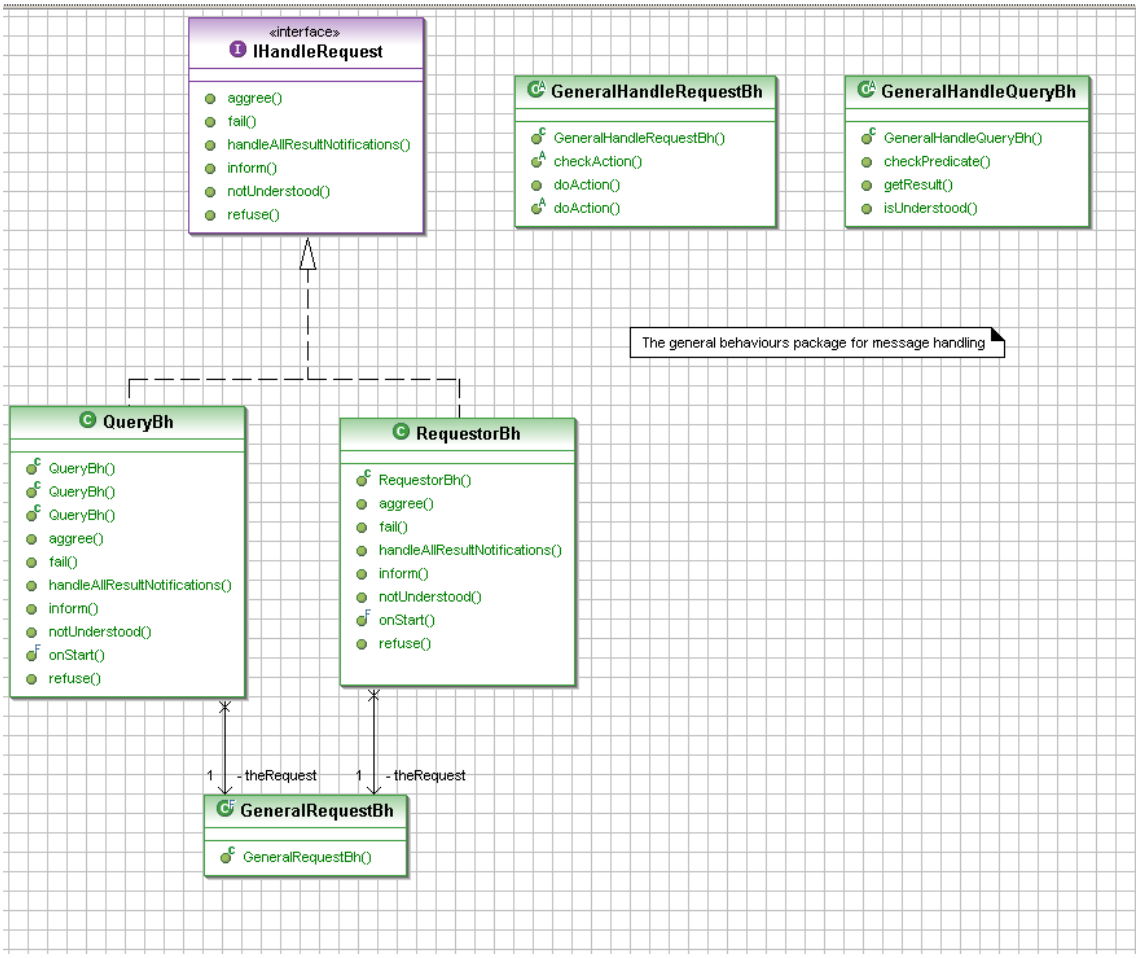
Τα κατηγορήματα είναι ερωτήσεις ή απλά μηνύματα κατάφασης, άρνησης, ή λάθους. Ακολουθεί μία αναλυτική περιγραφή τους:

- User Exists, Ρωτάει αν υπάρχει ένας χρήστης
    - User name, το διακριτικό του χρήστη
- Is Consumed, Ρωτάει αν έχει καταναλωθεί ένα αντικείμενο
    - User, Ο χρήστης
    - Item, Το αντικείμενο
- Get User, Ρωτάει τα δεδομένα ενός χρήστη
    - User, τα δεδομένα που επιστρέφονται
- Sorted Items Ζητάει τα καταταγμένα αντικείμενα
    - Sorted items, τα αντικείμενα
- Get Relevance Επιστρέφει την σχετικότητα για κάποιο αντικείμενο
    - Item, Το αντικείμενο
    - Relevance, Η σχετικότητα
- Get Suggestions, Επιστρέφει τις προτάσεις του συστήματος
    - Suggestions, οι προτάσεις
- Item exists, Ζητάει ναι ή όχι ανάλογα με το αν ένα αντικείμενο υπάρχει
    - Item id, Το διακριτικό του αντικείμενου
- Error, Επιστρέφει ένα μήνυμα
    - Message, Το μήνυμα του λάθους
- Is Suggested, Ρωτάει αν ένα αντικείμενο έχει προταθεί σε κάποιον χρήστη
    - User, Ο χρήστης
    - Suggestion, Η πρόταση
- Get Item, Επιστρέφει τα δεδομένα του πράκτορα
    - Item, Το αντικείμενο

ΠΡΩΤΟΚΟΛΛΟ ΜΗΝΥΜΑΤΩΝ

Στην πλατφόρμα Jade υπάρχουν διάφορα πρωτόκολλα και πρότυπα για ανταλλαγή μηνυμάτων μεταξύ των πρακτόρων. Το πιο ολοκληρωμένο και ισχυρό είναι το ACM της FIPA [15] που χρησιμοποιήθηκε και εδώ. Σε αυτό το πρωτόκολλο ορίζονται τόσο τα είδη των μηνυμάτων που μπορούν να σταλούν (INFORM κλπ), όπως και τα είδη των απαντήσεων (AGREE, FAIL, κλπ). Η υλοποίηση αυτού του πρωτοκόλλου γίνετε από συμπεριφορές που

μπορούν να ενσωματωθούν σε πράκτορες. Στην υλοποίηση αυτή κατασκευάσαμε συγκεκριμένων τύπων συμπεριφορών, αρκετά βέβαια γενικών ώστε να μπορούν να χρησιμοποιηθούν για όλα τα μηνύματα που να διαχειρίζονται όλη την δύστροπη γλώσσα του πρωτοκόλλου αφήνοντας στις επιμέρους συμπεριφορές των πρακτόρων να διαχειρίζονται μόνο τα οντολογικά αντικείμενα και όποιον τύπο απάντησης χρειάζονται. Το πακέτο αυτό των γενικών συμπεριφορών φαίνεται στο παρακάτω UML διάγραμμα. Σημαντικό είναι να επισημανθεί ότι ένα μήνυμα μπορεί να έχει είτε έναν είτε πολλούς αποδέκτες και αντίστοιχα είτε μία είτε πολλές απαντήσεις.



Σχήμα 15: UML διάγραμμα για το πακέτο των γενικών συμπεριφορών μηνυμάτων.

Τα επιμέρους μηνύματα που ανταλλάζονται μεταξύ των πρακτόρων παρουσιάζονται σε κάθε ένα πράκτορα ξεχωριστά, ενώ εδώ θα αναφερθούμε τις ιδιότητες που παρουσιάζουν οι γενικοί τύποι συμπεριφορών ανταλλαγής μηνυμάτων. Συγκεκριμένα οι συμπεριφορές για ερωτήσεις ή ενημερώσεις ή αιτήσεις (QueryBh, RequestBh και GeneralRequestBh), για χειρισμό

ερωτήσεων (GeneralHandleQueryBh), και, τέλος, για χειρισμό αιτήσεων (GeneralHandle RequestBh).

GeneralRequestBh

Η συμπεριφορά GeneralRequestBh υλοποιεί το από το πρωτόκολλο ανταλλαγής μηνυμάτων μεταξύ πρακτόρων ACM την συμπεριφορά που στέλνει ένα γενικό μήνυμα σε πολλαπλούς παραλήπτες και αναλαμβάνει να προωθήσει την απάντησή τους, ανάλογα με τον τύπο της (INFORM. AGREE, FAIL, NOT UNDERSTOOD, REFUSE, κλπ) στην συμπεριφορά που ζήτησε να σταλεί το συγκεκριμένο μήνυμα, η οποία οφείλει να υλοποιεί την διεπαφή (interface) IHandleRequest. Για την περίπτωση που οι παραλήπτες είναι πολλοί υπάρχει η δυνατότητα να προωθηθούν συγκεντρωμένες όλες οι απαντήσεις. Παρόλο που οι συμπεριφορές που χρησιμοποιούν αυτήν την συμπεριφορά μπορούν να είναι οτιδήποτε, με την προϋπόθεση ότι κληρονομούν την προαναφερθείσα διεπαφή, έχουν υλοποιηθεί δύο συγκεκριμένες, πλην αρκετά γενικές, συμπεριφορές για συγκεκριμένες και συνηθισμένες ενέργειες: η QueryBh και η RequestBh.

QueryBh

Η συμπεριφορά QueryBh είναι αρκετά γενική και παίρνει μία ερώτηση από την οντολογία (predicate), κατασκευάζει το κατάλληλο με την περίπτωση ACM μήνυμα και αν ο πράκτορας που την κάλεσε περιμένει απάντηση του προωθεί το περιεχόμενο του μηνύματος που είναι και αυτό predicate ορισμένο στην οντολογία.

RequestBh

Η συμπεριφορά RequestBh είναι αντίστοιχη της QueryBh αλλά για ενέργειες πρακτόρων (Agent Actions). Οι απαντήσεις μπορούν να είναι οποιοδήποτε αντικείμενο (concept) ή άλλη ενέργεια (action) ανάλογα με την αλληλουχία μηνυμάτων που ορίζετε σε ένα πρωτόκολλο επικοινωνίας.

GeneralHandleQueryBh και GeneralHandleRequestBh

Ενώ όλες οι προηγούμενες συμπεριφορές είναι για να ξεκινήσουν μία επικοινωνία μεταξύ δύο πρακτόρων οι συμπεριφορές GeneralHandleQueryBh και GeneralHandleRequestBh περιμένουν διαρκώς για μηνύματα. Όπως και πριν κάποιος πράκτορας πολύ εύκολα μπορεί να τις χρησιμοποιήσει κληρονομώντας τις και υλοποιώντας κάποιες μεθόδους. Σε αυτές τις μεθόδους πρέπει να ελέγχει αν μπορεί να καταλάβει το μήνυμα και να μετά να διαχειριστεί το μήνυμά και να δημιουργήσει μία απάντηση. Σε αυτές τις συμπεριφορές έχει σαν είσοδο

ανάλογα ποια από τις δύο κληρονομεί το agent action ή το predicate ενώ αντίστοιχα επιστρέφει το αντικείμενο που θα αναλάβουν οι γενικές αυτές συμπεριφορές να προωθήσουν πίσω στον αποστολέα.

ΠΡΑΚΤΟΡΕΣ

Κάθε ένας πράκτορας συνιστάτε από δύο επιμέρους πράγματα. Το πρώτο είναι οι μεταβλητές που περιέχει των οποίων οι τιμές καθορίζουν την κατάσταση τους, ενώ το δεύτερο αναφέρετε στις συμπεριφορές που έχει. Συγκεκριμένα για κάθε έναν πράκτορα ορίζουμε αυτά τα δύο χαρακτηριστικά του στις παρακάτω υποπαραγράφους.

Όπως έχει επισημανθεί όλη η λογική των πρακτόρων βρίσκεται κρυμμένη στις διάφορες συμπεριφορές που έχουν. Για να υλοποιηθεί κάθε διαδικασία που σχεδιάστηκε στο προηγούμενο κεφάλαιο χρειάζεται κάθε φορά ένας αριθμός από συμπεριφορές διαφόρων πρακτόρων. Οι συμπεριφορές αυτές (που παρουσιάστηκαν συνοπτικά παρακάτω ως μέρος των πρακτόρων) συνεργάζονται μεταξύ τους ανταλλάσοντας κατάλληλα μηνύματα για να επιτύχουν τον τελικό στόχο.

User-Agent

Κάθε χρήστης (User) αντιστοιχίζετε έναν πράκτορα με όνομα User-Agent ο οποίος αναλαμβάνει να διατηρεί και να ενημερώνει το προφίλ του χρήστη (User-Profile) ανάλογα με τις κινήσεις του και τις προσωπικές επιλογές του. Ένας user-agent δημιουργείται όταν ένας χρήστης εγγράφετε στο σύστημα. Διατηρεί, επίσης, ένα σύνολο με σχετικούς χρήστες-πράκτορες. Τέλος, ο πράκτορας είναι μόνιμα εγγεγραμμένος στην υπηρεσία DF με χαρακτηριστικά από το προφίλ του για να διευκολύνει την αναζήτηση με βάση αυτά. Ο πράκτορας έχει τα εξής αντικείμενα:

Αντικείμενο	Ερμηνεία
User	Εδώ διατηρούνται πληροφορίες για τον χρήστη και το προφίλ του.
UsageData	Το σύνολο των αντικείμενων (άρθρων) που έχει καταναλώσει ο χρήστης με την αξιολόγησή του.
Suggestions	Προτεινόμενα αντικείμενα για τον χρήστη. Προκύπτουν από διαφημίσεις.

IUtaSolution	Η λύση του προβλήματος UTA για το προφίλ με βάση την οποία ταξινομούνται τα αποτελέσματα των αναζητήσεων.
Friends	Οι φίλοι πράκτορες (πράκτορες με παρόμοιο προφίλ)

Πίνακας 8: Τα αντικείμενα του πράκτορα χρήστη.

Επίσης, έχει τις εξής συμπεριφορές (behaviors):

Συμπεριφορά	Ερμηνεία
GeneralHandleRequestBh	Συμπεριφορά που ακούει σε αιτήσεις άλλων πρακτόρων. Συγκεκριμένα ακούει στις παρακάτω αιτήσεις: Consume, UpdateProfile, Advertise, UpdateUser, και ProfileSort.
GeneralHandleQueryBh	Συμπεριφορά που ακούει σε ερωτήσεις άλλων πρακτόρων. Συγκεκριμένα ακούει στις παρακάτω ερωτήσεις: GetUser και Get Suggestions.
UpdateProfileBehavior	Συμπεριφορά που τρέχει ανά τακτά χρονικά διαστήματα και υπό την προϋπόθεση ότι έχει αλλάξει κάτι στα δεδομένα του χρήστη και υπολογίζει τις καινούργιες χρησιμότητες των κριτήριων.
FindBehavior	Βρίσκει άλλους πράκτορες-χρήστες με παρόμοιο προφίλ.

Πίνακας 9: Οι συμπεριφορές του πράκτορα χρήστη.

Item Agent

Κάθε αντικείμενο-άρθρο (Item) αντιστοιχίζετε ένα πράκτορα με όνομα Item-Agent ο οποίος αναλαμβάνει να διατηρεί και να ενημερώνει το προφίλ του αντικειμένου (Item-Profile) ανάλογα με την χρήση του αντικειμένου. Διατηρεί, επίσης, ένα σύνολο με σχετικά αντικείμενα. Τέλος, ο πράκτορας είναι μόνιμα εγγεγραμμένος στην υπηρεσία DF με χαρακτηριστικά από το προφίλ του για να διευκολύνει την αναζήτηση με βάση αυτά. Ο πράκτορας έχει τα εξής αντικείμενα:

Αντικείμενο	Ερμηνεία
Item	Εδώ διατηρούνται πληροφορίες για το αντικείμενο και το προφίλ του.

UsageData	Το σύνολο των χρηστών που έχουν καταναλώσει το αντικείμενο με την αξιολόγησή τους.
Friends	Οι φίλοι πράκτορες (πράκτορες με παρόμοιο περιεχόμενο) .

Πίνακας 10: Τα αντικείμενα του πράκτορα αντικείμενο.

Επίσης, έχει τις εξής συμπεριφορές (behaviors):

Συμπεριφορά	Ερμηνεία
GeneralHandleRequestBh	Συμπεριφορά που ακούει σε αιτήσεις άλλων πρακτόρων. Συγκεκριμένα ακούει στις παρακάτω αιτήσεις: Consume, UpdateProfile, UpdateItem, και Search.
GeneralHandleQueryBh	Συμπεριφορά που ακούει σε ερωτήσεις άλλων πρακτόρων. Συγκεκριμένα ακούει στην ερώτηση: GetItem.
AdvertiseBehavior	Συμπεριφορά που τρέχει ανά τακτά χρονικά διαστήματα και διαφημίζει το αντικείμενο.
FindBehavior	Βρίσκει άλλους πράκτορες-αντικείμενα με παρόμοιο προφίλ.

Πίνακας 11: Οι συμπεριφορές του πράκτορα αντικείμενο.

Uta Solver Agent

Ο πράκτορας Uta Solver Agent είναι επιφορτισμένος να επιλύει προβλήματα με τον αλγόριθμο UTA*. Αυτός ο πράκτορας είναι κύριο λειτουργικό σημείο του συστήματος και είναι εντελώς ανεξάρτητός από την περιοχή μας. Θα μπορούσε δηλαδή να χρησιμοποιηθεί και σε άλλα πολυ-πρακτορικά συστήματα. Μπορούν να υπάρχουν περισσότεροι του ενός πράκτορες αυτού του είδους. Υπάρχει μία λίμνη (pool) από τέτοιους πράκτορες έτοιμους προς χρήση ώστε να μην χρειάζεται να δημιουργούνται νέοι συνεχώς. Όταν ένας πράκτορας επιλύει ένα πρόβλημα τότε δεν μπορεί να βρεθεί μέσω της υπηρεσίας DF, ενώ μόλις ελευθερωθεί επανεγγράφετε στην υπηρεσία ως έτοιμο να δεχτεί νέες αιτήσεις. Ο πράκτορας αυτός δεν έχει αντικείμενα επειδή δεν έχει κατάσταση (stateless) ενώ με την συμπεριφορά GeneralHandleRequestBh ακούει σε νέες αιτήσεις του τύπου RequestProfile.

Persistency Manager

Ο Persistency Manager είναι ένας πράκτορας που αναλαμβάνει να διαχειριστεί για λογαριασμό όλων των άλλων πρακτόρων (user agent, κ.λπ.) τα δεδομένα τους και την κατάστασή τους πάνω από μία βάση δεδομένων. Στη βάση δεδομένων κρατείται η συνολική

κατάσταση του συστήματος συμπεριλαμβανομένων των εκκρεμούντων μηνυμάτων (μηνύματα που έχουν σταλεί και δεν έχουν παραλειφθεί ακόμα). Αυτό που κάνει ο persistency manager είναι ανά τακτά χρονικά διαστήματα να αποθηκεύει ένα στιγμιότυπο (snapshot) του συστήματος στην βάση δεδομένων και να την φορτώνει όταν το σύστημα επανέρχεται. Και αυτός δεν έχει κατάσταση ενώ δεν σώζετε πότε ο ίδιος στην βάση ούτε εγγράφετε στην υπηρεσία DF.

Proxy

Ο πράκτορας proxy αναλαμβάνει όλη την εξωτερική επικοινωνία με άλλες εφαρμογές μετατρέποντας τις αιτήσεις τους σε κατάλληλες συμπεριφορές αλληλεπιδρώντας με τους πράκτορες μέσω μηνυμάτων. Λειτουργεί ουσιαστικά σαν διεκπεραιωτής του πολύ-πρακτορικού συστήματος. Αυτός είναι ο μόνος που μπορεί να λαμβάνει μηνύματα από εξωτερικές εφαρμογές με την υπηρεσία Object-to-Agent (O2A) communication. Στην μοναδική κυκλική συμπεριφορά που έχει λαμβάνει αντικείμενα από τις εφαρμογές, τα οποία είναι συγκεκριμένου τύπου MessageObject και ανάλογα με τον τύπο τους κατασκευάζει και εκτελεί αντίστοιχες συμπεριφορές. Οι τύποι των μηνυμάτων που μπορεί να λάβει καθώς και οι ενέργειες που επιλαμβάνεται φαίνονται στον παρακάτω πίνακα.

Τύπος Μηνύματος	Ενέργεια	Συμπεριφορά
Logon	Διαπιστεύει κάποιον χρήστη.	Εσωτερικά
Load Items	Επιστρέφει όλα τα αντικείμενα και τους χρήστες που υπάρχουν στο σύστημα.	LoadBehaviour
Update Item	Ενημερώνει τα δεδομένα ενός αντικειμένου.	ItemUpdateBehaviour
Update User	Ενημερώνει τα δεδομένα ενός χρήστη.	UserUpdateBehaviour
Search	Αναζητάει με βάση κάποια κριτήρια αντικείμενα για κάποιον χρήστη.	ProxySearchBehaviour
Consume	Καταναλώνει ένα αντικείμενο για ένα χρήστη με βάση κάποια αξιολόγηση.	ProxyConsumeBehavior
Get Advertisments	Επιστρέφει τις διαφημίσεις ενός χρήστη.	QueryBh
Create User	Δημιουργεί έναν νέο χρήστη.	Εσωτερικά
Create Item	Δημιουργεί ένα νέο αντικείμενο.	Εσωτερικά

Πίνακας 12: Οι τύποι μηνύματος και οι αντίστοιχες ενέργειες που εκτελεί ο πράκτορας proxy.

ΕΦΑΡΜΟΓΗ ΔΙΚΤΥΟΥ

Η εφαρμογή δικτύου υλοποιήθηκε πάνω από τον apache tomcat με χρήση της βιβλιοθήκης struts [6]. Με την βιβλιοθήκη struts μπορούμε να ορίσουμε τις ενέργειες αλλά και την αλληλουχία τους που μπορούν να πραγματοποιηθούν στην εφαρμογή μας καθώς μία σειρά από ετικέτες (tags) που καλούνται από τις δυναμικές ιστοσελίδες και το servlet και κάποιες φόρμες που περιέχουν τα δεδομένα τις εφαρμογής. Σε αυτήν την παράγραφο περιγράφετε ο μηχανισμός λειτουργίας ολόκληρης της εφαρμογής ιστού σε συνδυασμό και με μία σειρά από δυναμικές σελίδες που υλοποιήθηκαν.

Έχει υλοποιηθεί ένας servlet που αρχικοποιεί την εφαρμογή ιστού διαβάζοντας αρχικά τα δεδομένα που υπάρχουν στους πράκτορες και κατασκευάζοντας τις κατάλληλες δομές που τα διαχειρίζονται για λογαριασμό των χρηστών. Παρέχετε η δυνατότητα να υπάρχουν καταστάσεις και σύνοδοι (sessions) μέσα στην εφαρμογή. Συγκεκριμένα κατασκευάζετε μία σύνοδος όταν ένας χρήστης διαπιστεύεται (login) στο σύστημα και αποθηκεύετε εκεί το διαπιστευτήριό του. Με αυτό το διαπιστευτήριο είναι δυνατόν να γνωρίζουμε ποιος χρήστης εκτελεί μια ενέργεια. Λεπτομέρειες της διαδικασίας αυτής θα φανούν στην αντίστοιχη παράγραφο των ενεργειών.

Γενικότερα για κάποιον χρήση του συστήματος διακρίνουμε μόνο δύο καταστάσεις: να είναι διαπιστευμένος και να μην είναι. Όταν δεν είναι διαπιστευμένος δεν μπορεί να αλληλεπιδράσει με το σύστημα. Επειδή όμως η πολιτική του συστήματος είναι να είναι ανοιχτό σε όλους του χρήστες, ο καθένας έχει την δυνατότητα (πρόσβαση) να κατασκευάσει το προφίλ του και μέσω αυτού να διαπιστευτεί.

Τέλος, η εφαρμογή λειτουργεί σε ελληνικά αλλά έχει την δυνατότητα να αλλάξει γλώσσες πολύ απλά με την χρήση ενός αρχείου για άλλη γλώσσα. Σε αυτό το αρχείο ορίζονται όλα τα μηνύματα και οι επεξηγήσεις προς τους χρήστες.

ΕΝΕΡΓΕΙΕΣ ΚΑΙ ΦΟΡΜΕΣ ΕΝΕΡΓΕΙΩΝ

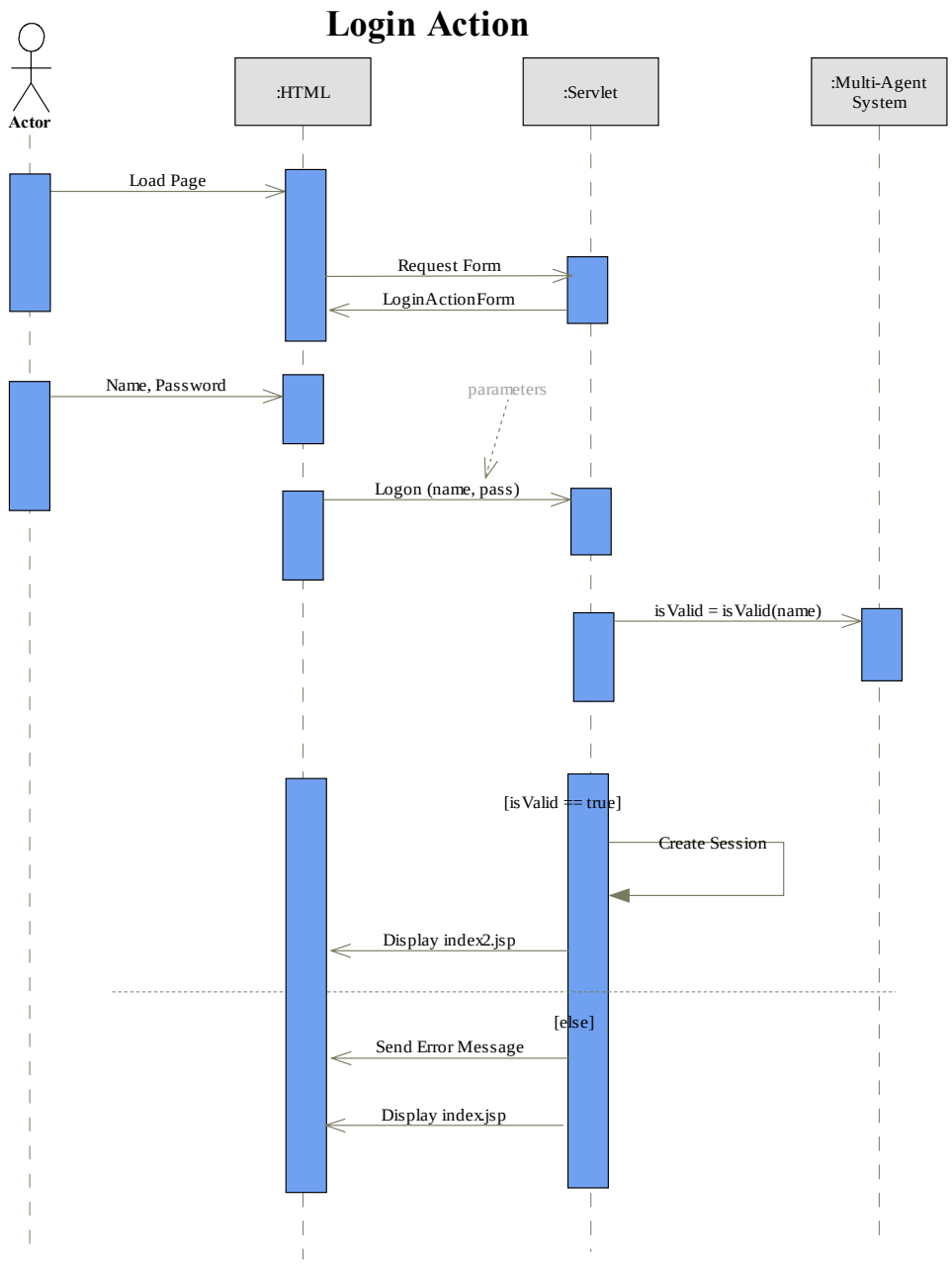
Οι χρήστες ανάλογα με την κατάστασή τους (διαπιστευμένοι ή μη) είναι δυνατόν να εκτελέσουν ένα σύνολο ενεργειών (actions). Σε κάθε μία ενέργεια ορίζουμε ένα σύνολο ιδιοτήτων που καθορίζουν σε ποια ιστοσελίδα αναφέρονται και που πρέπει να πλοηγηθεί ο

χρήστης ανάλογα με την έκβαση ή τον τύπο της ενέργειας και την λογική της εφαρμογής (π.χ. αποτυχία, επιτυχία, ενημέρωση ή διαγραφή, κλπ). Οι ενέργειες που έχουν οριστεί, οι ιδιότητές τους, καθώς και ροή τους ακολουθούν εδώ.

Οι φόρμες ή ορθότερα οι φόρμες ενεργειών είναι ειδικού τύπου φόρμες όπου μπορούν να οριστούν τα πεδία της φόρμας (π.χ. τίτλος κλπ), να γίνει έλεγχος ορθότητας κατά την αλλαγή ή εισαγωγή δεδομένων (validation), να αρχικοποιούνται και άλλα. Γενικά είναι ένας μηχανισμός για ευέλικτη χρήση φορμών σε συνδυασμό με ενέργειες.

Διαπίστευση (Login)

Η ενέργεια "logon" παρέχει την δυνατότητα στους χρήστες να διαπιστευτούν στο σύστημα. Χρησιμοποιεί την φόρμα LoginActionForm, και την ενέργεια LoginAction για να διαχειριστεί το όνομα και το συνθηματικό του χρήστη και να προβεί στην κατάλληλη διαδικασία αντίστοιχα. Εάν επιτύχει (success) δημιουργεί μία νέα σύνοδο με το διαπιστευτήριο του χρήστη και προωθεί τον χρήστη στην σελίδα "index2.jsp". Εάν αποτύχει προωθεί τον χρήστη στην ίδια σελίδα φορτώνοντας και κάποιο μήνυμα λάθους. Το παρακάτω UML διάγραμμα σχηματοποιεί αυτήν την ενέργεια.

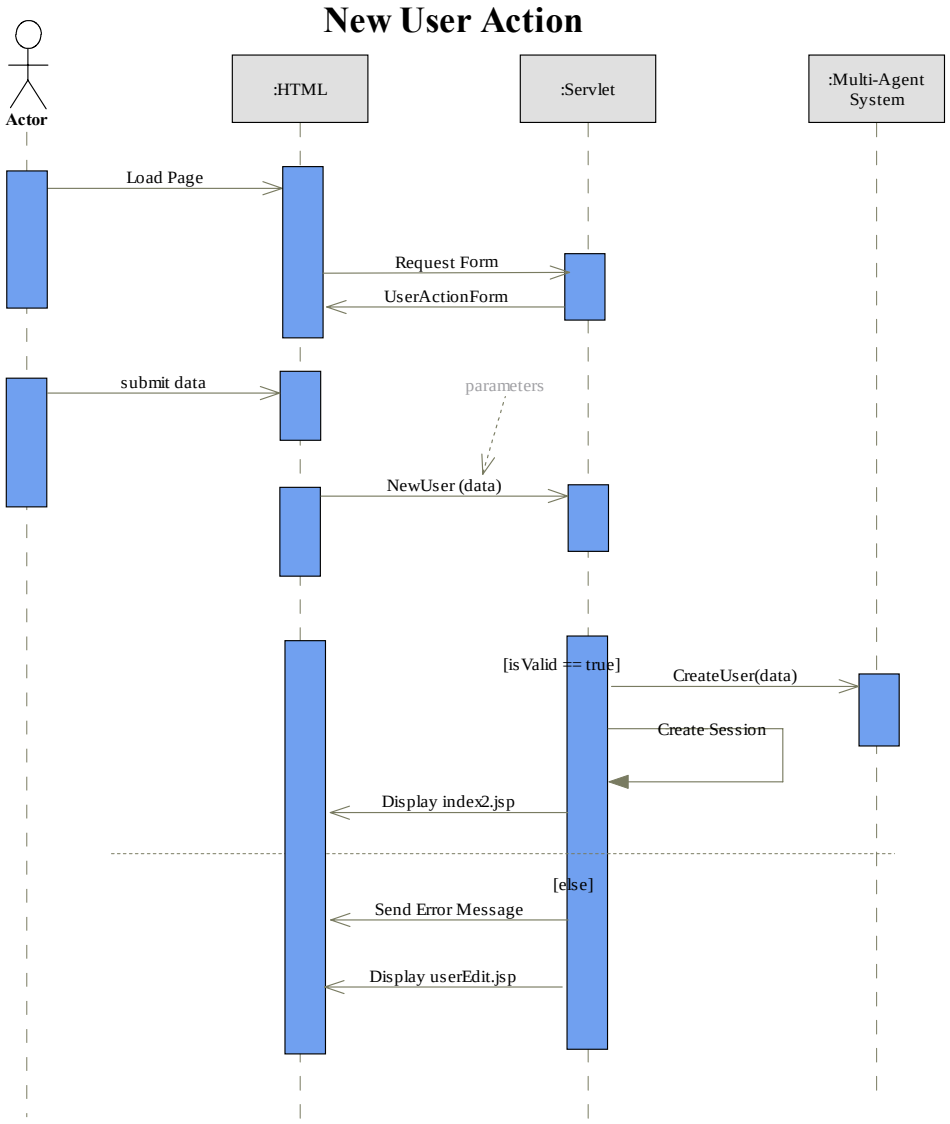


Σχήμα 16: Η ακολουθία ενεργειών κατά την ενέργεια διαπίστευσης.

Δημιουργία Χρήστη

Η ενέργεια "newuser" παρέχει την δυνατότητα στους χρήστες να δημιουργούν ένα νέο προφίλ και διαπιστευτούν από το σύστημα. Χρησιμοποιεί την φόρμα UserActionForm, και την ενέργεια StoreAction για να διαχειριστεί τα στοιχεία του προφίλ του χρήστη και να προβεί στην κατάλληλη διαδικασία αντίστοιχα. Εάν επιτύχει (success) δημιουργεί ένα νέο πράκτορα και μία νέα σύνοδο με το διαπιστευτήριο του χρήστη και προωθεί τον χρήστη στην σελίδα "index2.jsp". Μπορεί να αποτύχει εάν υπάρχει ήδη χρήστη με το ίδιο διακριτικό ή τα δεδομένα που εισήγαγε δεν είναι σωστά. Τα δεδομένα που περιέχονται στο προφίλ του είναι

αυτά που έχουν οριστεί στην οντολογία. Εάν αποτύχει προωθεί τον χρήστη στην ίδια σελίδα φορτώνοντας και κάποιο μήνυμα λάθους. Το παρακάτω UML διάγραμμα σχηματοποιεί αυτήν την ενέργεια.



Σχήμα 17: Η ακολουθία ενεργειών κατά την ενέργεια δημιουργίας χρήστη.

Αποσύνδεση (Logoff)

Η ενέργεια "logoff" παρέχει την δυνατότητα στους χρήστες να αποσυνδεθούν από το σύστημα. Δεν χρησιμοποιεί καμία φόρμα παρά μόνο την ενέργεια LogoffAction. Εάν επιτύχει (success) καταστρέφει την σύνοδο με το διαπιστευτήριο του χρήστη και προωθεί τον χρήστη στην σελίδα "index.jsp". Δεν αποτυγχάνει ποτέ. Αυτή η ενέργεια είναι προσβάσιμη μόνο από διαπιστευμένους χρήστες.

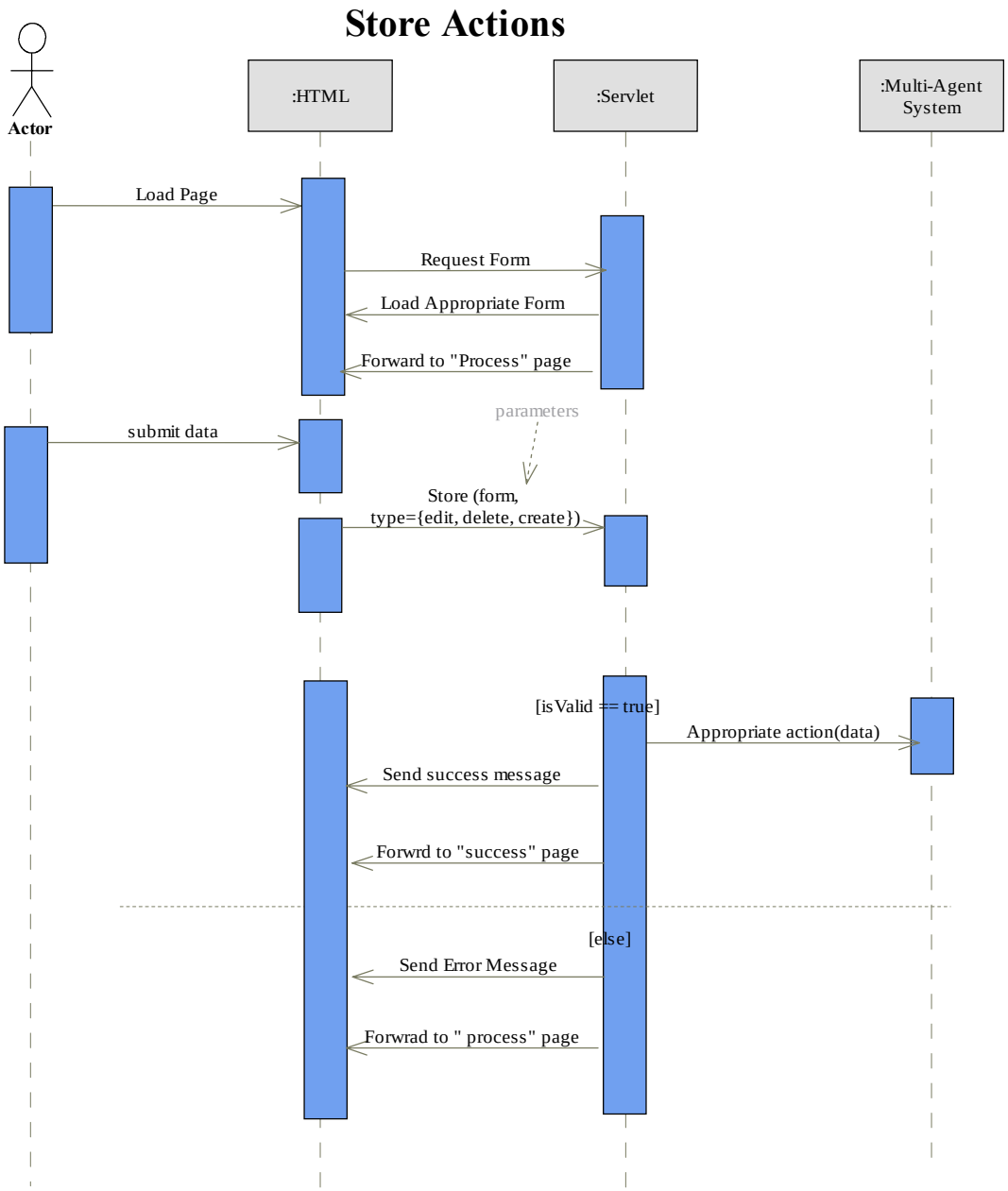
Ενημέρωση Δεδομένων

Σε αυτήν την παράγραφο περιγράφουμε μία σειρά από ενέργειες που αντιμετωπίζονται με κοινό και ενιαίο τρόπο και αφορούν στην δημιουργία, ενημέρωση και διαγραφή χρηστών, άρθρων και αρθρογράφων. Αυτές οι ενέργειες είναι τρείς οι "user", "item" και "author" και χρησιμοποιούν τις φόρμες UserActionForm, ItemActionForm, και AuthorActionForm αντίστοιχα. Τα στοιχεία που περιέχονται στις φόρμες έχουν οριστεί στην οντολογία. Όλες χρησιμοποιούν την ενέργεια StoreAction, όπως και η δημιουργία χρήστη, ενώ ανάλογα με τον τύπο της ενέργειας εκτελείτε η ανάλογη διαδικασία. Για να επιτευχτεί αυτό οι φόρμες έχουν κάποια επιπλέον πεδία και μεθόδους που κληρονομούν από μία κατάλληλη διεπαφή (interface) ώστε να τις αντιμετωπίζει με ενιαίο τρόπο η ενέργεια StoreAction. Οι σελίδες τις κάθε μίας είναι αντίστοιχα οι "user.jsp" και "userEdit.jsp", "item.jsp" και "itemEdit.jsp", και "author.jsp" και " authorEdit.jsp". Εάν επιτύχουν οδηγούνται στις πρώτες (ονομάζονται success) και εάν όχι στις δεύτερες (ονομάζονται process) με κατάλληλα μηνύματα και στις δύο περιπτώσεις. Αυτές οι ενέργειες είναι προσβάσιμες μόνο από διαπιστευμένους χρήστες. Ο παρακάτω πίνακας παραθέτει όλα τα παραπάνω για κάθε ενέργεια.

Ενέργεια	Φόρμα	Επιτυχής σελίδα (success)	Σελίδα επεξεργασίας (process)
user	UserActionForm	user.jsp	userEdit.jsp
item	ItemActionForm	item.jsp	itemEdit.jsp
author	AuthorActionForm	author.jsp	authorEdit.jsp

Πίνακας 13: Οι ενέργειες ενημέρωσης δεδομένων και τα βασικά χαρακτηριστικά τους.

Το παρακάτω UML διάγραμμα σχηματοποιεί αυτές τις ενέργειες.



Σχήμα 18: Η ακολουθία ενεργειών για την επεξεργασία δεδομένων.

Αναζήτηση Αντικείμενων

Η ενέργεια "suggest" παρέχει την δυνατότητα στους χρήστες να αναζητούν άρθρα με βάση κάποιες προτιμήσεις. Χρησιμοποιεί την φόρμα SuggestActionForm, και την ενέργεια SuggestAction για να εισαχθούν οι προτιμήσεις της αναζήτησης (που έχουν οριστεί στην οντολογία) και να προβεί στην κατάλληλη διαδικασία στο πολύ-πρακτορικό περιβάλλον αντίστοιχα. Εάν επιτύχει (success) προωθεί τον χρήστη στην σελίδα "item.jsp" φορτώνοντας τα αντικείμενα που έχουν επιστραφεί από την αναζήτηση. Εάν, όμως, αποτύχει προωθεί τον

χρήστη στην ίδια σελίδα φορτώνοντας και κάποιο μήνυμα λάθους. Αυτή η ενέργεια είναι προσβάσιμη μόνο από διαπιστευμένους χρήστες.

Αξιολόγηση Αντικείμενου

Η ενέργεια "consume" παρέχει την δυνατότητα στους χρήστες να αξιολογούν άρθρα. Χρησιμοποιεί την φόρμα ConsumeActionForm, και την ενέργεια ConsumeAction για να εισαχθούν τα δεδομένα της αξιολόγησης (που έχουν οριστεί στην οντολογία) και να προβεί στην κατάλληλη διαδικασία στο πολύ-πρακτορικό περιβάλλον αντίστοιχα. Εάν επιτύχει (success) προωθεί τον χρήστη στην σελίδα "item.jsp". Εάν, όμως, αποτύχει προωθεί τον χρήστη στην ίδια σελίδα φορτώνοντας και κάποιο μήνυμα λάθους. Αυτή η ενέργεια είναι προσβάσιμη μόνο από διαπιστευμένους χρήστες.

ΕΤΙΚΕΤΕΣ

Οι ετικέτες χρησιμεύουν στην βελτίωση και αυτόματη παραγωγή του HTML κώδικα αλλά και στην λογική της εφαρμογής. Έχουν οριστεί τέσσερις ετικέτες και συγκεκριμένα: η ετικέτα επιστροφής (BackTag), η ετικέτα φόρτωσής (LoadTag), η ετικέτα συνδέσμων (LinkTag) και η ετικέτα ελέγχου διαπίστευσης (CheckLogon).

Η ετικέτα ελέγχου διαπίστευσης όποτε χρησιμοποιηθεί σε μία ιστοσελίδα ελέγχει αν υπάρχει διαπιστευμένος χρήστης στη σύνοδο και εάν δεν υπάρχει προωθεί τον χρήστη στην αρχική σελίδα για να διαπιστευτεί. Επιπλέον, αποθηκεύει την σελίδα που ζήτησε ο χρήστης και όταν διαπιστευτεί τον οδηγεί σε αυτήν και όχι στην κανονική (index2.jsp).

Η ετικέτα φόρτωσης έχει δυο πεδία με τα όποια αυτός που την χρησιμοποίει καθορίζει τι ακριβώς δεδομένα θέλει να φορτωθούν στην σελίδα. Παράδειγμα είναι η επεξεργασία ενός άρθρου όπου εκτός από τα δεδομένα του άρθρου (φορτώνονται αυτόματα) χρειάζεται να φορτώσουμε και μία λίστα με όλους τους αρθρογράφους ώστε να μπορεί ο χρήστης να επιλέξει ανάμεσά τους ποιοι έχουν συγγράψει το άρθρο.

Η ετικέτα συνδέσμων κατασκευάζει ανάλογα με τα πεδία που της δύνονται ένα HTML σύνδεσμο κατάλληλο για πλοήγηση.

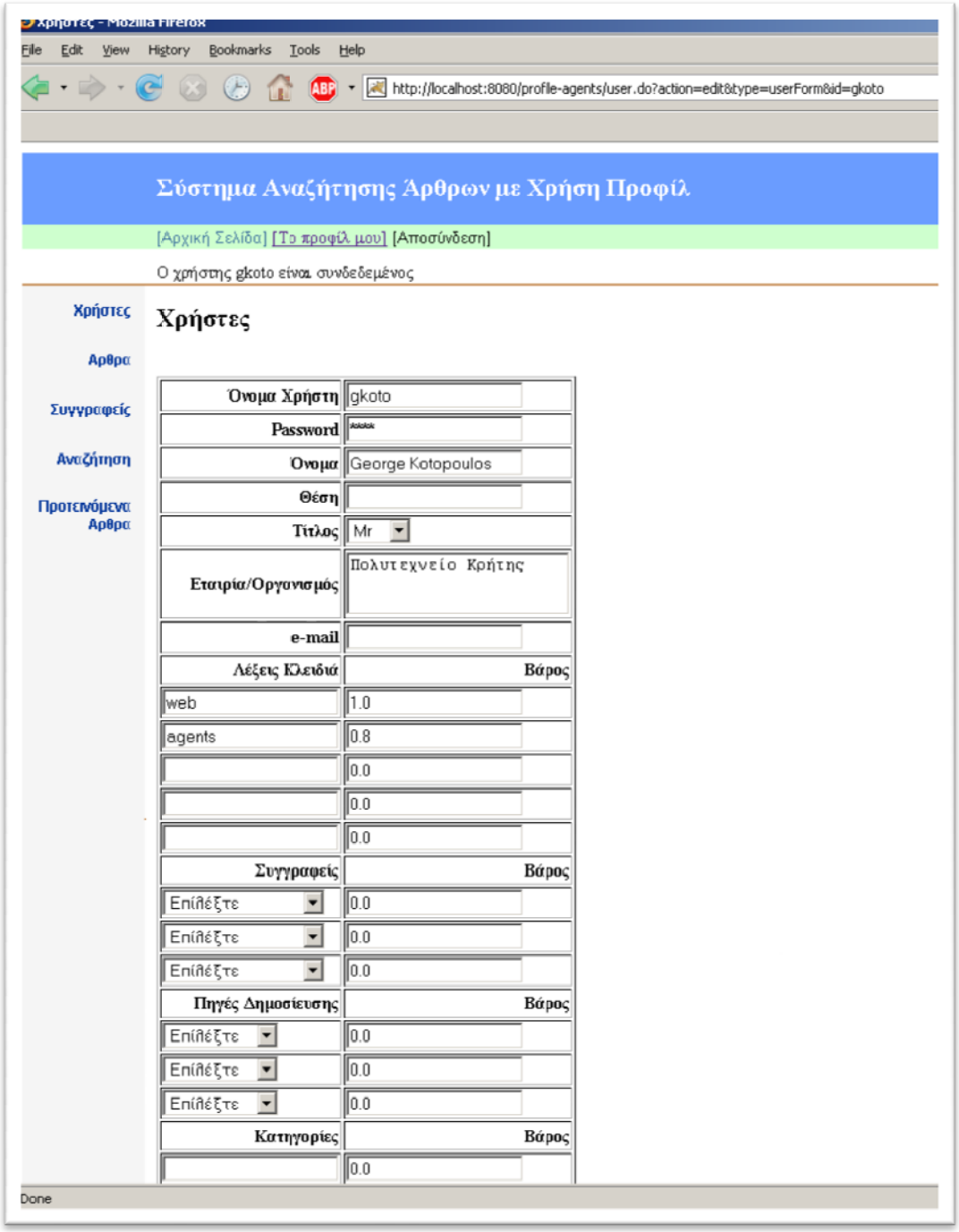
Η ετικέτα επιστροφής χρησιμοποιείται για να κατασκευάσουμε ένα κατάλληλο σύνδεσμο για επιστροφή σε μία σελίδα που μας κάλεσε. Με αυτήν ανατρέπετε η λογική προώθηση των ενεργειών και οδηγείται σε κάποια άλλη. Αν για παράδειγμα θέλουμε μετά που κάνουμε

επεξεργασία σε ένα συγγραφέα να μην προωθηθούμε στην ορισμένη σελίδα "author.jsp" αλλά σε κάποια άλλη αυτό το κάνουμε με χρήση αυτής της ετικέτας

ΣΕΛΙΔΕΣ

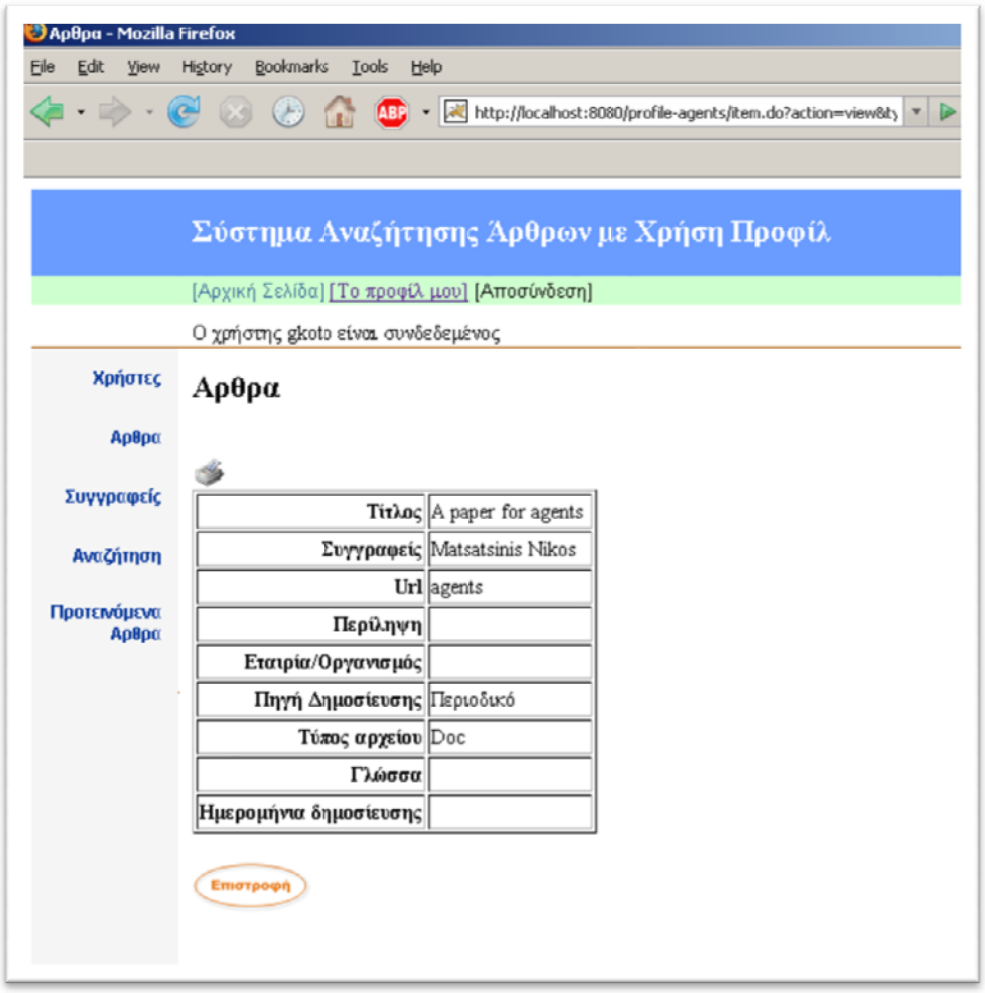
Σε αυτήν την παράγραφο φαίνονται οι περιπτώσεις χρήσης και πώς υλοποιήθηκαν σαν ιστοσελίδες μέσα από μία σειρά από στιγμιότυπα (snapshots) των ιστοσελίδων. Οι ιστοσελίδες ανάλογα με την κατάσταση τους (τα δεδομένα και το είδος τους, από πού καλέστηκαν, κλπ) συμπεριφέρονται με διαφορετικούς τρόπους με χρήση των κατάλληλων ενεργειών. Με τις εικόνες αυτές επιδεικνύονται τα κύρια και πιο σημαντικά σημεία που έχουν αναλυθεί παραπάνω, ενώ δεν γίνετε διεξοδική αναπαράσταση όλων των σελίδων σε όλες τις καταστάσεις τους διότι δεν θα πρόσθετε κάτι επιπλέον στην κατανόηση της υλοποίησης. Ο αναγνώστης έχει την δυνατότητα να πλοηγηθεί ο ίδιος στο σύστημα ένα το φορτώσει πάνω σε έναν εξυπηρετητή εφαρμογών ιστού tomcat.

Στο παρακάτω σχήμα φαίνετε η σελίδα για την επεξεργασία/δημιουργία του προφίλ ενός χρήστη. Σε αυτήν την σελίδα ο χρήστης δίνει δημογραφικά στοιχεία αλλά και τις προτιμήσεις του αναφορικά με λέξεις κλειδιά, συγγραφείς και άλλα. Επίσης διακρίνετε το μενού που δίνει δυνατότητα στους χρήστες να εισάγουν καινούργια και να επεξεργαστούν άρθρα και συγγραφείς, να αναζητήσουν άρθρα, να αποσυνδεθούν και άλλα.



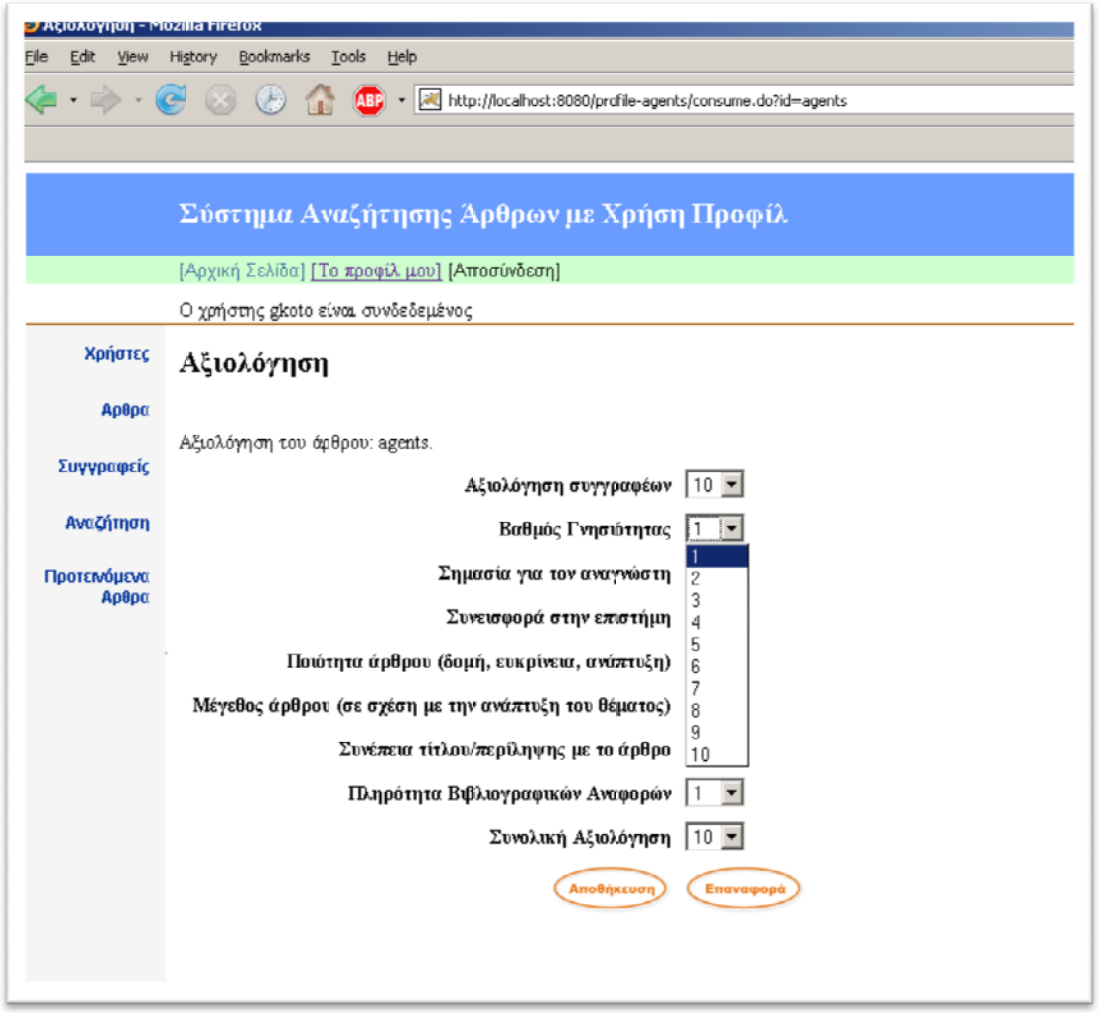
Σχήμα 19: Σελίδα για την επεξεργασία του προφίλ ενός χρήστη. Διακρίνονται τόσο δημογραφικές πληροφορίες όσο και προτιμήσεις σε λέξεις κλειδιά, συγγραφείς, κλπ.

Στο παρακάτω σχήμα φαίνονται τα χαρακτηριστικά του άρθρου "A paper for Agents". Ο χρήστης μπορεί να βλέπει τα χαρακτηριστικά των άρθρων που διαβάζει και μετά να τα αξιολογεί.



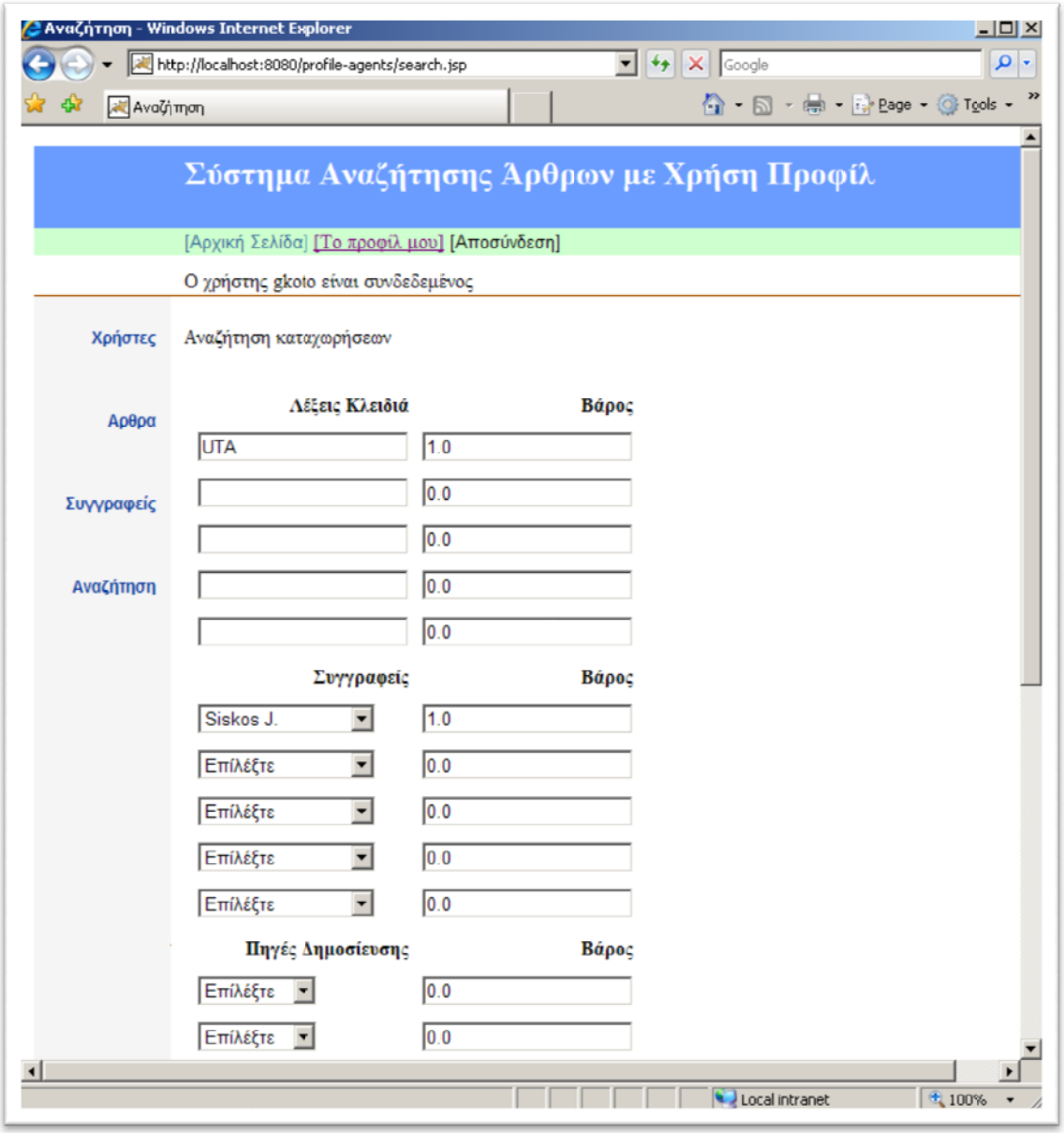
Σχήμα 20: Τα χαρακτηριστικά του άρθρου "A paper for Agents".

Στο επόμενο σχήμα φαίνετε η σελίδα όπου γίνετε η αξιολόγηση του άρθρου «A paper for Agents». Διακρίνονται όλα τα στοιχεία που χρειάζονται για την αξιολόγηση ενός άρθρου. Σε αυτήν την σελίδα επίσης φαίνετε ποιος χρήστης είναι συνδεδεμένος και πιο άρθρο αξιολογεί.



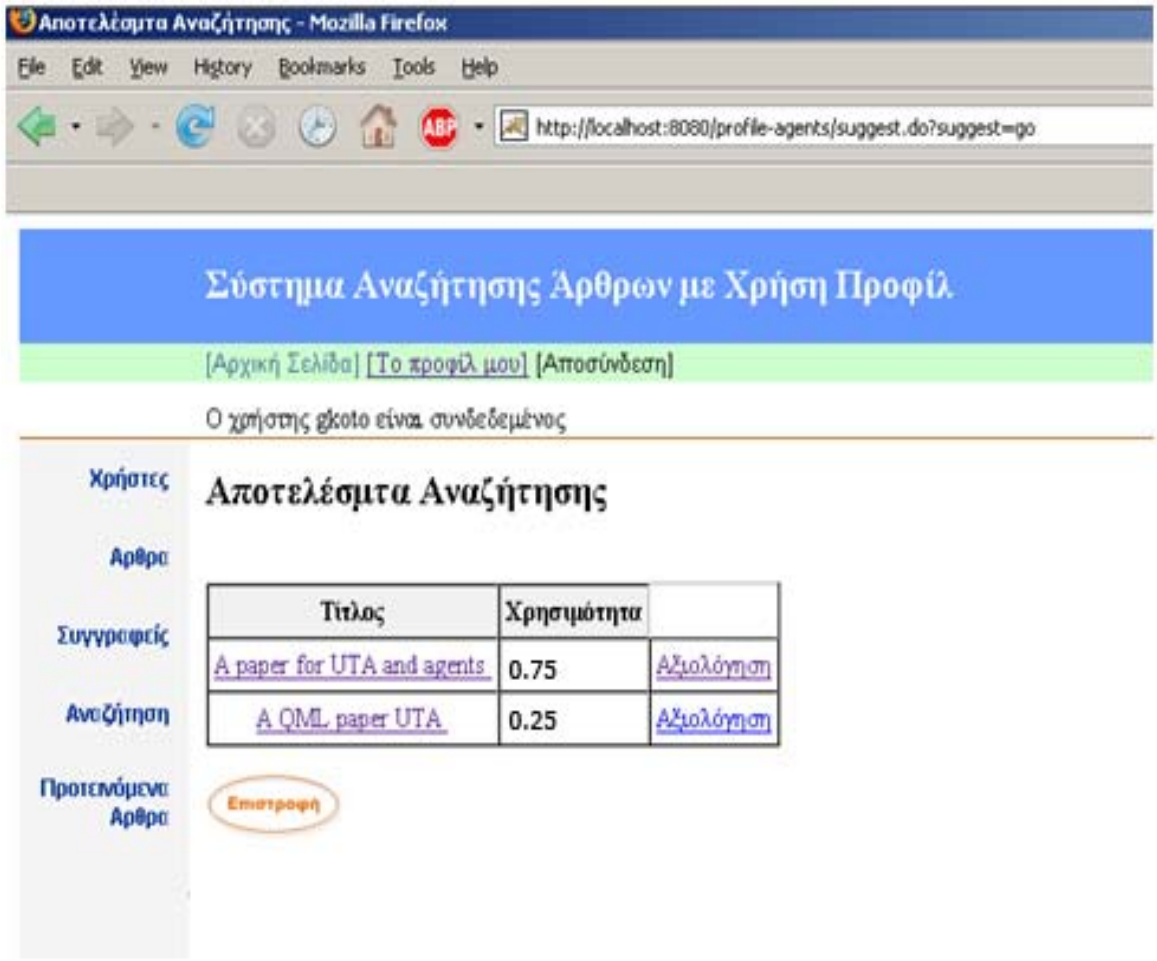
Σχήμα 21: Σελίδα αξιολόγησης άρθρου.

Στο παρακάτω σχήμα μπορούμε να διακρίνουμε την σελίδα αναζήτησης άρθρων με βάση ένα σύνολο κριτήριων. Αυτά είναι κριτήρια αποκοπής ενώ εσωτερικά το σύστημα θα αξιοποιήσει το προφίλ του χρήστη για να κατατάξει τα άρθρα που πληρούν αυτά τα κριτήρια.



Σχήμα 22: Η σελίδα για αναζήτηση άρθρου με βάση κάποια κριτήρια.

Στο παρακάτω σχήμα φαίνονται τα αποτελέσματα της αναζήτησης και κατάταξης των άρθρων με βάση την εκτιμώμενη χρησιμότητά τους. Παρατηρούμε ότι το από τα δύο άρθρα που αναφέρονται στη UTA το πρώτο έχει μεγαλύτερη χρησιμότητα γιατί είναι γραμμένο από τον ίδιο συγγραφέα που αξιολογήσαμε θετικά ενώ περιλαμβάνει και μία λέξη κλειδί που έχουμε βάλει στο προφίλ μας την "Agents". Ο χρήστης μπορεί να δει τα χαρακτηριστικά των άρθρων να επιλέξει πιο θέλει να διαβάσει και να το αξιολογήσει.



Σχήμα 23: Αποτελέσματα αναζήτησης καταταγμένα με βάση τα χαρακτηριστικά του προφίλ του χρήστη με την μέθοδο UTA.

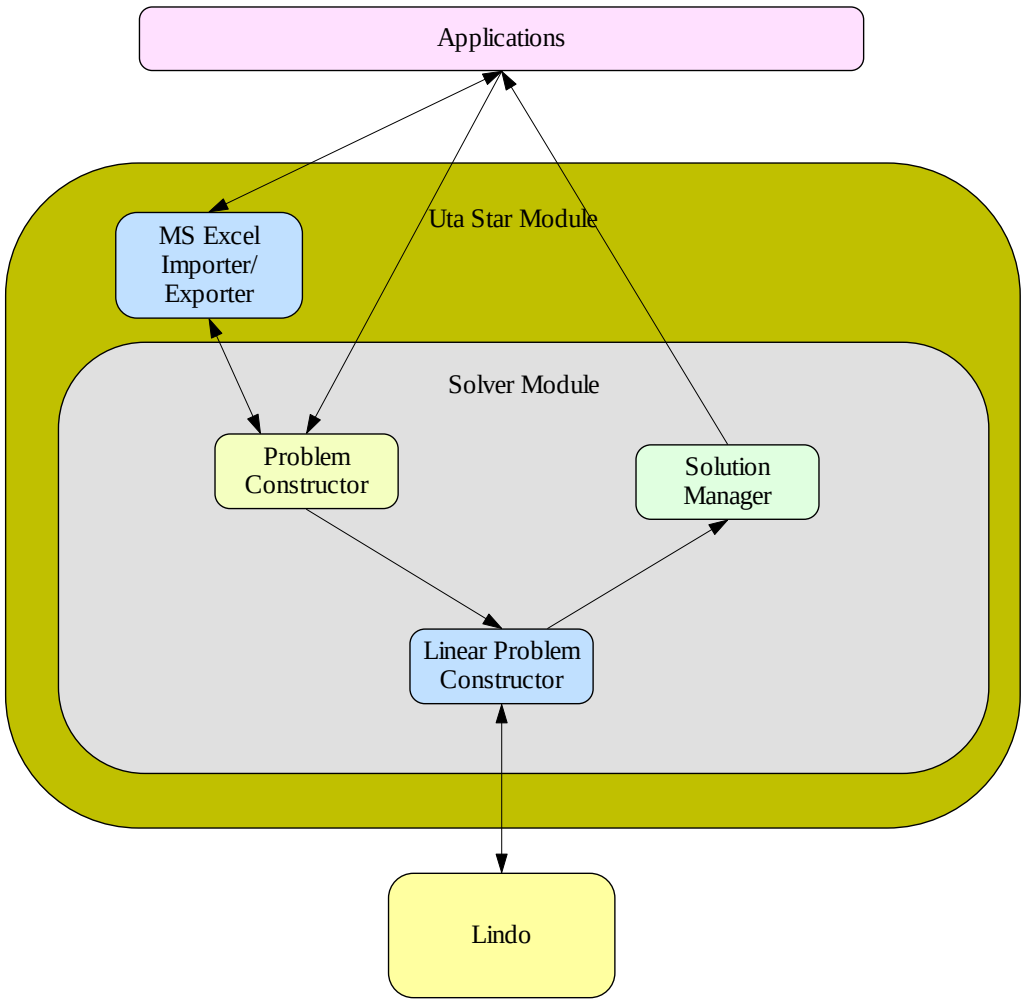
Η ΜΟΝΑΔΑ UTA*

Η μονάδα UTA* (γιούτα σταρ) είναι επιφορτισμένη με την επίλυση UTA* προβλημάτων. Ένα UTA* πρόβλημα αποτελείται από διάφορα τμήματα. Αρχικά έχει δομική πληροφορία για το πρόβλημα η οποία περιλαμβάνει κάποιες παραμέτρους του προβλήματος και συγκεκριμένα: μία παράμετρο δ που καθορίζει το περιθώριο λάθος του γραμμικού προβλήματος, ένα σύνολο κριτήριων και για κάθε ένα από αυτά την χειρότερη τιμή που μπορεί να πάρει, την καλύτερη, καθώς και τον αριθμό των διαστημάτων που θα πρέπει να διαχωριστεί. Τέλος, περιλαμβάνει και τα καθεαυτό δεδομένα του προβλήματος που είναι μία τιμή για κάθε κριτήριο της κάθε μίας εναλλακτικής καταταγμένα ανάλογα με τις προτιμήσεις του αποφασίζοντα.

Επιπλέον πληροφορίες όπως οδηγίες εγκατάστασης και παραδείγματα χρήσης υπάρχουν στο πρώτο παράρτημα.

ΑΡΧΙΤΕΚΤΟΝΙΚΗ

Η αρχιτεκτονική του συστήματος αποτελείται από δύο επιμέρους τμήματα. Το πρώτο αφορά την υλοποίηση του αλγορίθμου UTA* και το API με το οποίο άλλες εφαρμογές μπορούν να ζητάνε την λύση στο πρόβλημα. Για την λύση του γραμμικού προβλήματος χρησιμοποιείται το περιβάλλον Lindo. Το δεύτερο αφορά το γραφικό περιβάλλον μέσω του οποίου τελικοί χρήστες μπορούν να κατασκευάσουν ένα πρόβλημα και ζητήσουν την λύση. Επίσης είναι δυνατόν εφαρμογές να φορτώνουν κάποιο πρόβλημα από ένα αρχείο Excel και να το σώζουν επίσης εκεί. Δόθηκε έμφαση ώστε ο χρήστης να είναι σε θέση να ορίσει όλες τις παραμέτρους της UTA*. Το παρακάτω σχήμα παρουσιάζει την αρχιτεκτονική της μονάδας UTA*.



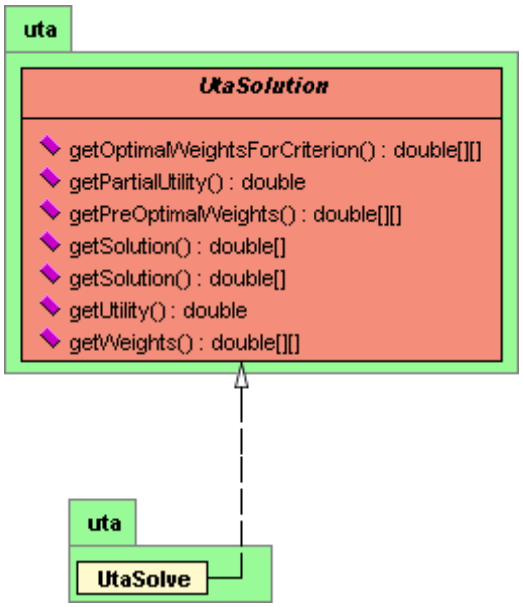
Σχήμα 24: Η αρχιτεκτονική της μονάδας UTA*.

Η εφαρμογή (ή το γραφικό εργαλείο) που χρησιμοποιεί τη μονάδα αρχικά χρησιμοποιεί την διεπαφή (API) της μονάδας Problem Constructor για να κατασκευάσει ένα πρόβλημα. Αυτό μπορεί να γίνει εναλλακτικά από κάποιο αρχείο MS Excel χρησιμοποιώντας την μονάδα Importer, ενώ το αρχείο πρέπει να ακολουθεί μία συγκεκριμένη δομή που θα παρουσιαστεί επόμενη παράγραφο. Μετά το πρόβλημα ανασκευάζετε σε γραμμικό πρόβλημα σύμφωνα με τα [3]και [7] και το επιλύουμε χρησιμοποιώντας το Lindo.

Αφού έχουμε πάρει την πρώτη λύση και στην περίπτωση που έχουμε πάνω από μία επιχειρούμε να την μετα-βελτιστοποιήσουμε. Κατασκευάζουμε γραμμικά προβλήματα ίσα με τον αριθμό των κριτηρίων και βελτιστοποιούμε την λύση για κάθε ένα από αυτά. Τέλος, παίρνουμε την μέση τιμή των επιμέρους λύσεων. Η εφαρμογή μπορεί να χρησιμοποιήσει την λύση αυτή όχι μόνο για να δει τους πίνακες χρησιμοτήτων που παρήχθησαν αλλά και για να παράγει ολικές χρησιμότητες για καινούργιες εναλλακτικές μέσω της διεπαφής (API) της λύσης που προσφέρετε στην εφαρμογή.

ΤΟ API ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ΚΑΙ ΤΗΣ ΛΥΣΗΣ

Η κλάση που χρησιμοποιείται για αλληλεπίδραση με άλλες εφαρμογές είναι το interface UtaSolution, το οποίο υλοποιεί η UtaSolve κλάση. Παρακάτω ακολουθεί το UML διάγραμμα που υποδεικνύει το σύνολο των προσφερόμενων συναρτήσεων.



Σχήμα 25: UML διάγραμμα του UtaSolution interface. Μέσω αυτού οι καλούσες εφαρμογές μπορούν να παίρνουν τις λύσεις ενός Uta star προβλήματος.

Όπως φαίνεται και στο παραπάνω διάγραμμα η εξωτερική εφαρμογή μπορεί να χρησιμοποιήσει τις παρακάτω συναρτήσεις της UtaSolution για να πάρει τις λύσεις που θέλει.

- double[] getSolution(). Ο καλών παίρνει έναν μονοδιάστατο πίνακα με την λύση για τις εναλλακτικές του αρχικού προβλήματος.
- double[] getSolution(double[][] values) Ο καλών παίρνει έναν μονοδιάστατο πίνακα με την λύση για τις εναλλακτικές των τιμών που πήρε ως είσοδο.
- double[][] getWeights() Ο καλών παίρνει έναν δισδιάστατο πίνακα με τα βάρη που υπολογίστηκαν από την λύση του αρχικού προβλήματος μετά-βελτιστοποιημένα.
- double[][] getPreOptimalWeights() Ο καλών παίρνει έναν δισδιάστατο πίνακα με τα βάρη που υπολογίστηκαν από την λύση του αρχικού προβλήματος πριν την μετά-βελτιστοποίηση.
- double[][] getOptimalWeightsForCriterion(int crid) Ο καλών παίρνει έναν δισδιάστατο πίνακα με τα βάρη που υπολογίστηκαν από την λύση του αρχικού προβλήματος βελτιστοποιημένα για το κριτήριο που περιγράφεται από την παράμετρο crid.
- double getUtility(double[] values) Ο καλών παίρνει έναν double με την λύση για την εναλλακτική που εισήγαγε.
- double getPartialUtility(double value, int crid) Ο καλών παίρνει έναν double με την λύση για την τιμή ενός συγκεκριμένου κριτηρίου που ορίζει στις παραμέτρους.

Η εφαρμογή για να επιλύσει ένα πρόβλημα, και να μπορεί να χρησιμοποιήσει τις παραπάνω συναρτήσεις, πρέπει να φτιάξει ένα αντικείμενο UtaSolve με το UTA star πρόβλημα χρησιμοποιώντας έναν από τους παρακάτω τρόπους.:

public UtaSolve(double[][] values, int[] intervals, double[] best, double[] worst, int[] rank, double d)

- Ένα δισδιάστατο πίνακα από double όπου περιέχονται οι τιμές κάθε εναλλακτικής (γραμμές) για κάθε κριτήριο (στήλες).
- Έναν μονοδιάστατο με int όπου περιέχονται οι αριθμοί των υποδιαστημάτων για κάθε ένα κριτήριο.
- Έναν μονοδιάστατο με double όπου περιέχονται οι βέλτιστες τιμές κάθε κριτηρίου.
- Έναν μονοδιάστατο με double όπου περιέχονται οι χειρότερες τιμές κάθε κριτηρίου.

- Έναν μονοδιάστατο πίνακα με int όπου κάθε τιμή είναι η κατάταξη της κάθε μίας εναλλακτικής.
- Έναν double που δηλώνει το δ (δέλτα).

Εναλλακτικά, το αντικείμενο UtaSolve μπορεί να δημιουργηθεί με την συνάρτηση public UtaSolve(uta.UtaProblem problem).Η UtaProblem είναι ένα interface το οποίο μία κλάση του καλούντος πρέπει να υλοποιεί.

Το interface UtaProblem έχει τις πιο κάτω μεθόδους που κάποια κλάση πρέπει να υλοποιεί:

- public double[][] getValues();
- public int[] getIntervals();
- public double[] getBest();
- public double[] getWorst();
- public int[] getRank();
- public double getD();

Επιπλέον πληροφορίες όπως οδηγίες εγκατάστασης και παραδείγματα χρήσης υπάρχουν στο πρώτο παράρτημα.

ΤΕΧΝΟΛΟΓΙΕΣ ΥΛΟΠΟΙΗΣΗΣ

Η υλοποίηση του συστήματος έγινε με χρήση της γλώσσας Java2. Για την ανάπτυξη του πολύ-πρακτορικού συστήματος χρησιμοποιήθηκε η πλατφόρμα Jade 3.3, η οποία προκρίθηκε μεταξύ πολλών άλλων που είναι διαθέσιμες, χάρη στην πληρότητα κάλυψης του προτύπου FIPA για πολυ-πρακτορικά συστήματα, στην ευχρηστία του κατά την ανάπτυξη, αλλά και απόδοση του. Για την λύση των γραμμικών προβλημάτων που απαιτεί η μέθοδος UTA* χρησιμοποιήθηκε το Lindo 2. Ως σύστημα διαχείρισης βάσης δεδομένων χρησιμοποιήθηκε η hypersonic 1.8.0.7, που τρέχει πάνω από περιβάλλον Java2. Για την επικοινωνία της βάσης με την εφαρμογή μας χρησιμοποιήθηκαν τα hibernate 3.1 και persistence add-in του Jade. Τέλος, για την ανάπτυξη της εφαρμογής ιστού με την οποία θα αλληλεπιδρούν οι χρήστες χρησιμοποιήθηκε ο Apache Tomcat 6.0.10 και η τεχνολογίες Servlet και JSP καθώς και Apache Struts 1. Η υλοποίηση έγινε στο Eclipse 3.2, ενώ η οντολογία υλοποιήθηκε σε Protégé 32 μαζί με τον bean-generator add-in του JADE.

Η εφαρμογή είναι συμβατή με τα λειτουργικά συστήματα Windows και Linux/Unix, όπου είναι συμβατή και η εφαρμογή Lindo 2. Επίσης, χρειάζεται να είναι εγκατεστημένη τουλάχιστον Java 1.5.

ΑΝΑΚΕΦΑΛΑΙΩΣΗ

Σε αυτό το κεφάλαιο εξηγήθηκε λεπτομερώς η υλοποίηση που έλαβε χώρα στα πλαίσια αυτής της εργασίας. Συγκεκριμένα παρουσιάστηκε η αρχιτεκτονική τριών επιπέδων όπου διαχωρίζετε η διεπαφή με τον χρήστη (εφαρμογή ιστού), η λογική της εφαρμογής (πολύ-πρακτορικό περιβάλλον με δυνατότητα επίλυσης προβλημάτων UTA*), και το σύστημα διαχείρισης βάσεων δεδομένων όπου φυλάσσονται στιγμιότυπα (snapshots) του συστήματος.

Συγκεκριμένα, έγινε λεπτομερής περιγραφή του πολύ-πρακτορικού συστήματος και πως διαχειρίζεται το περιβάλλον Jade με έμφαση στην ανάλυση της οντολογίας για τα μηνύματα, στον μηχανισμό ανταλλαγής μηνυμάτων ανάμεσα στους πράκτορες, και στην λειτουργικότητα των πρακτόρων και των συμπεριφορών τους. Τέλος, περιγράφηκε ο μηχανισμός που επικοινωνεί η λειτουργική αυτή μονάδα με το επίπεδο της διεπαφής αλλά και το επίπεδο της βάσης δεδομένων.

Στη συνέχεια αναλύθηκε η εφαρμογή δικτύου με έμφαση στη λειτουργικότητα που προσφέρει στους μηχανισμούς με τους οποίους το επιτυγχάνει (ενέργειες, ετικέτες, φόρμες, δυναμικές σελίδες).

Τέλος, περιγράφηκε η αρχιτεκτονική της μονάδας που επιλύει προβλήματα UTA* ενώ δόθηκε και η αναλυτική περιγραφή της διεπαφής με άλλες εφαρμογές (API).

ΚΕΦΑΛΑΙΟ ΣΤ' – ΣΥΜΠΕΡΑΣΜΑΤΑ - ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

Το σύστημα που σχεδιάστηκε και υλοποιήθηκε στην διατριβή αυτή προσφέρει προσωποποιημένες υπηρεσίες αναζήτησης και προτάσεις σε χρήστες αναφορικά με επιστημονικά άρθρα, ενώ δόθηκε έμφαση ώστε η μοντελοποίηση να είναι αρκετά γενική και να μπορεί να χρησιμοποιηθεί και για άλλους τύπους αντικειμένων.

Κύριες επιλογές για τον σχεδιασμό και την υλοποίηση στάθηκε η χρήση της πολύ-κριτήριας μεθοδολογίας UTA* για την κατάταξη των εναλλακτικών, η χρήση του μοντέλου ασαφούς λογικής p-norm για την αποκοπή των εναλλακτικών και των προσδιορισμό τιμών για τα επιμέρους κριτήρια, η χρήση περιβάλλοντος πολλαπλών πρακτόρων που επέτρεπε την αυτόνομη λειτουργία μονάδων λογισμικού και τον σχεδιασμό του συστήματος με λογικές συνεργασίας, και, τέλος, η χρήση εξυπηρετητή εφαρμογών ιστού για την ανάπτυξη εφαρμογής ιστού κατάλληλης για απομακρυσμένη χρήση του συστήματος.

Από την τριβή με το αντικείμενο προκύπτει ότι η μεθοδολογία UTA* μπορεί να χρησιμοποιηθεί σε πολλές εφαρμογές, και μπορεί να λειτουργήσει και ως ευφυής πράκτορας κάτω από ένα περιβάλλον πολλαπλών πρακτόρων. Επίσης, η αρχιτεκτονική του περιβάλλοντος πολλαπλών πρακτόρων προσφέρει πολλά πλεονεκτήματα στο σχεδιαστή ενός συστήματος ώστε να μπορεί να κατανείμει το φόρτο σε μονάδες λογισμικού που μπορούν να λειτουργούν αυτόνομα. Συγκεκριμένα, το περιβάλλον JADE ήταν πολύ καλή επιλογή διότι υποστηρίζει ολόκληρο το πρότυπο της FIPA με μια επιδεκτική σε τεράστιες αυξήσεις πρακτόρων αρχιτεκτονική. Επιπλέον, το εργαλείο struts δίνει την δυνατότητα για ανάπτυξη εφαρμογών ιστού με πολύ ευέλικτο και συνεκτικό τρόπο.

Ως μελλοντικές επεκτάσεις μπορούν να αναφερθούν η μετατροπή του συστήματος προσωποποιημένες υπηρεσίες για άλλους τύπους αντικειμένων πέραν των επιστημονικών άρθρων και η συστηματική αξιολόγησή του από χρήστες ως προς την προστιθέμενη αξία που προσφέρει για βελτίωση τμημάτων του μοντέλου και της υλοποίησης. Επίσης, ενδιαφέρων θα ήταν να κατασκευαστεί μηχανισμός αυτόματης επεξεργασίας των άρθρων για την παραγωγή μετα-δεδομένων. Τέλος, θα μπορούσε να δοκιμαστεί πάνω από πολλές πλατφόρμες πρακτόρων και να τροποποιηθεί ώστε να αποδίδει καλλίτερα σε κατανεμημένο σε πολλούς υπολογιστές περιβάλλον.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Joon Ho Lee, "PROPERTIES OF EXTENDED BOOLEAN MODELS IN INFORMATION RETRIEVAL", In Proceedings of the 17th ACM SIGIR International Conference on Research and Development in Information Retrieval, 1994, 182-190.

[2] Joon Ho Lee, W. Y. Kim, M. H. Kim, Y. J. Lee "ON THE EVALUATION OF BOOLEAN OPERATORS IN THE EXTENDED BOOLEAN FRAMEWORK", In Proceedings of the 16th ACM SIGIR International Conference on Research and Development in Information Retrieval, 1993, 291-297.

[3] J. Siskos, D. Yannacopoulos "*UTASTAR: An ordinal regretion method for building additive value functions*", Investigacao Operational Volume 5, No 1, p. 39-53, June 1985.

[4] Salton, G., Buckley, C. "INTRODUCTION TO MODERN INFORMATION RETRIEVAL", McGraw-Hill Book Company, New York, 1982.

[5] The JADE project http://jade.tilab.com/

[6] The STRUTS project http://struts.apache.org/

[7] E. Jacquet-Lagreze, J Siskos "*Assessing a set of additive utility functions for multicriteria decision-making, the UTA method*", European Journal of Operational Research 10, 1982, p 151-164.

[8] The Protégé Ontology Editor and Knowledge Acquisition System http://protege.stanford.edu/

[9] P. Haase, J. Broekstra, M.Ehrig, M. Menken, P.Mika, M. Plechawski, P. Pyszlak, B. Schnizler, R. Siebes, S. Staab, C. Tempich "Bibster - A Semantics-Based Bibliographic Peer-to-Peer System" Proceedings of the International Semantic Web Conference (ISWC2004), November 9-11, 2004, Hiroshima, Japan

[10] Theo van Veen, Bill Oldroyd "Search and Retrieval in The European Library", D-Lib Magazine Volume 10 number 2 ISSN 1082-9873, February 2004.

[11] Liren Chen, Katia Sycara "WebMate: A Personal Agent for World-Wide Web Browsing and Searching" Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)

[12] Leonard N. Foner, "*A Multi-Agent Referral System for Matchmaking*" PAAM '96 Proceedings, London, England, 1996

[13] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. *"The PageRank citation ranking: Bringing order to the Web"* Stanford Digital Library Technologies Project, 1999.

[14] Yates, R.B., Neto,B.R.: MODERN INFORMATION RETRIEVAL. ACM Press (1999).

[15] The Fountation for Intelligent Agents (FIPA) standard, www.fipa.org

[16] The Hypersonic SQL Database Engine http://hsqldb.org/

[17] The Apache Project http://www.apache.org/

[18] The Apache Tomcat Project http://tomcat.apache.org/

[19] Final Release of Java Servlet 2.5 specification http://jcp.org/aboutJava/communityprocess/mrel/jsr154/index.html

[20] Hesselink, H. W., 'Preference Rankings in the Face of Uncertainty', 2003, 1 - 15

[21] Kollias, S., Stafylopatis, A., 'Future Prospects for Neural Networks-Selected Applications-Intelligent Agents', NTUA, 2001

[22] Kalenka, S., Jennings, N. R., 'Socially Responsible Decision Making by Autonomous Agents', ICCS-97, 1 – 14

[23] Klusch, M., 'Agent-Mediated Trading: Intelligent Agents and E-Business'

ΠΑΡΑΡΤΗΜΑ Α' – Η ΜΟΝΑΔΑ UTA*

Η μονάδα UTA* παρουσιάστηκε στο κεφάλαιο της υλοποίησης στην αντίστοιχη παράγραφο ενώ εδώ θα δούμε οδηγίες εγκατάστασης αλλά και παραδείγματα χρήσης.

Η μονάδα αυτή έχει υλοποιηθεί σε Java2 ενώ μπορεί να χρησιμοποιηθεί και από άλλα περιβάλλοντα σε Windows με χρήση OLE objects. Συγκεκριμένα υπάρχει έχει κατασκευαστεί ειδικά μια βιβλιοθήκη δυναμικής σύνδεσης (dynamic link library – dll), η οποία μπορεί να φορτωθεί από οποιαδήποτε εφαρμογή και να χρησιμοποιηθεί κανονικά όπως δείχνει ο κώδικας σε Microsoft Visual Basic σε επόμενη παραγράφο.

ΟΔΗΓΙΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ

Ωστόσο στο σύστημα πρέπει να υπάρχει ένα JAVA Virtual Machine (JVM) από την έκδοση 1.5 και έπειτα. Επίσης, χρειάζεται να υπάρχει στον τρέχοντα φάκελο, ή στο environmental variable PATH τα δυναμικής σύνδεσης βιβλιοθήκες (dlls) που παρέχονται από το Lindo.

ΠΑΡΑΔΕΙΓΜΑΤΑ ΧΡΗΣΗΣ

Σε αυτήν την παράγραφο θα εξετάσουμε ένα πρόβλημα UTA* το οποίο θα κατασκευάσουμε και θα επιλύσουμε με δύο τρόπους· με Java και με Visual Basic. Στο τέλος υπάρχουν δύο πίνακες ενός MS Excel αρχείου στο οποίο ορίζετε και πάλι το ίδιο πρόβλημα.

Το παράδειγμα αφορά σε μέσα μεταφοράς και έχει ως εξής:

Μέσα μεταφοράς	Τιμή(Fr)	Χρονική Διάρκεια (min)	Άνεση	Προδιάταξη
RER:	3.0	10.0	1.0	1
METRO-1:	4.0	20.0	2.0	2
METRO-2:	2.0	20.0	0.0	2
BUS:	6.0	40.0	0.0	3
TAXI:	30.0	30.0	3.0	4
Intervals:	2	3	3	
Best:	2.0	10.0	3.0	
Worst:	30.0	40.0	0.0	

Πίνακας 14: Παράδειγμα ενός UTA* προβλήματος που αφορά σε μέσα μεταφοράς.

Αυτό το πρόβλημα αρχικά μπορεί να γραφεί σε Java ως εξής:

Σχήμα 26: Παράδειγμα κώδικα Java για χρήση της μονάδας UTA*.

Στη συνέχεια ακολουθεί ο ίδιος κώδικας σε Visual Basic με χρήση της μονάδας ως OLE Object.

```
Public Sub Uta_Example()
Dim best() As Double
Dim worst() As Double
Dim rank() As Long
Dim values() As Double
Dim intervals() As Long
Set problem = UtaSolve1.getUtaProblem
    problem.setD 0.052
 ReDim best(2) = (2, 10, 3)
   problem.setBest best()
ReDim worst(2) = (30, 40, 0)
   problem.setWorst worst()
ReDim rank(4) = (1, 2, 2, 3, 4)
   problem.setRank rank()
ReDim intervals(2) = (2, 3, 4)
problem.setIntervals intervals()
ReDim values(5, 3) = ((4, 10, 1) ,  (4, 20, 2),  (2, 20, 0),  (6, 40, 0),  (30, 30 , 3))
   For i = 1 To 5
      Dim row(2) As Double
      For j = 0 To 2
         row(j) = values(i, j + 1)
      Next j
      problem.addValues row()
   Next i
 UtaSolve1.setUtaProblem problem
results = UtaSolve1.getSolution
   Debug.Print results(1); results(2); results(3); results(4); results(5)
End Sub
```

Σχήμα 27: Παράδειγμα κώδικα MS Visual Basic για χρήση της μονάδας UTA*.

Το αρχείο excel έχει τρεις καρτέλες η κάθε μία από τις οποίες πρέπει να έχει συγκεκριμένο όνομα. Η πρώτη αφορά στη δομή του προβλήματος και έχει όνομα «criteria», η δεύτερη αφορά στις εναλλακτικές που έχουν καταταχθεί και ονομάζετε «input», ενώ η τρίτη αφορά σε εναλλακτικές που θέλουμε να κατατάξει ο αλγόριθμος και ονομάζεται «data».

Συγκεκριμένα θα δούμε την δομή που πρέπει να έχουν μέσα από το παραπάνω παράδειγμα στους τρεις πίνακες που ακολουθούν.

CriteriaName	intervals	best	worst	d
Price	2	2	30	0.05
minutes	3	10	40	
comfort	3	3	0	

Πίνακας 15: Παράδειγμα της καρτέλας του excel με όνομα «criteria». Εδώ ορίζονται τα κριτήρια του προβλήματος.

Παρατηρούμε ότι η πρώτη γραμμή έχει ονόματα τα οποία πρέπει να χρησιμοποιηθούν ως έχουν και τα κριτήρια να προστεθούν γραμμή-γραμμή.

AlName	Price	minutes	comfort	Rank
RER	3	10	1	1
METRO-1	4	20	2	2
METRO-2	2	20	0	2
BUS	6	40	0	3
TAXI	30	30	3	4

Πίνακας 16: Παράδειγμα της καρτέλας του excel με όνομα «input». Εδώ ορίζονται οι καταταγμένες από τον αποφασίζοντα εναλλακτικές.

Παρατηρούμε ότι η πρώτη γραμμή έχει συγκεκριμένα ονόματα· το πρώτο κελί έχει το όνομα AlName ενώ ακολουθούν τα ονόματα των κριτήριων όπως ορίστηκαν στην καρτέλα «criteria», και τελευταίο έρχεται το κελί με όνομα Rank. Οι εναλλακτικές προστίθενται από κάτω γραμμή-γραμμή.

AlName	Price	minutes	comfort
MyRER	3	10	1
MyMETRO-1	4	20	2
MyMETRO-2	2	20	0
MyBUS	6	40	0
MyTAXI	30	30	3

Πίνακας 17: Παράδειγμα της καρτέλας του excel με όνομα «output». Εδώ ορίζονται οι εναλλακτικές που θα καταταχθούν από το σύστημα.

Τα ονόματα και οι τιμές είναι αντίστοιχα με την προηγούμενη περίπτωση χωρίς τη στήλη Rank.

ΠΑΡΑΡΤΗΜΑ Β' – ΑΡΧΕΙΑ ΔΙΑΜΟΡΦΩΣΗΣ

Όλα τα αρχεία πρέπει να τοποθετηθούν στον φάκελο bin του tomcat. Το πρώτο αρχείο είναι το "persistence.properties" και περιέχει τις εξής γραμμές:

```
main=false
port=2000
container-name=PersistenceContainer
services=jade.core.persistence.PersistenceService;
jade.core.event.NotificationService;jade.core.mobility.AgentMobilityService
meta-db=hibernate_JADE_Persistence.properties
load-from=JADE-DB
```

σε αυτό καθορίζονται οι παράμετροι του persistence container ο οποίος διαχειρίζεται την βάση.

Το δεύτερο αρχείο είναι για τον main container του πολύ-πρακτορικού περιβάλλοντος με όνομα "jade.properties" και έχει ως εξής:

```
main=true
port=2000
services=jade.core.persistence.PersistenceService;
jade.core.event.NotificationService;jade.core.mobility.AgentMobilityService
meta-db=hibernate_JADE_Persistence.properties
agents=p:edu.ergasya.agents.profile.agents.PersistenceManagerAgent
```

Το επόμενο αρχείο είναι το "hibernate_JADE_Persistence.properties" και καθορίζονται σε αυτό παράμετροι που αφορούν τη σύνδεση με την βάση δεδομένων και το σύστημα διαχείρισής της.

```
## define query language constants / function names
hibernate.query.substitutions true 1, false 0, yes 'Y', no 'N'

## package imports
hibernate.query.imports net.sf.hibernate.test, net.sf.hibernate.eg
hibernate.query.factory_class org.hibernate.hql.classic.ClassicQueryTranslatorFactory

## HypersonicSQL
hibernate.dialect org.hibernate.dialect.HSQLDialect
hibernate.connection.driver_class org.hsqldb.jdbcDriver
hibernate.connection.username sa
hibernate.connection.password
hibernate.connection.url jdbc:hsqldb:./db/hsqldb/hibernate

## Hibernate Connection Pool
hibernate.connection.pool_size 1
hibernate.statement_cache.size 25
```

```
## Properties for external configuration of Proxool
hibernate.proxool.pool_alias pool1

## print all generated SQL to the console
hibernate.show_sql false

## set the maximum JDBC 2 batch size (a nonzero value enables batching)
hibernate.jdbc.batch_size 0

## use streams when writing binary types to / from JDBC
hibernate.jdbc.use_streams_for_binary true
```

Εδώ χρησιμοποίει το σύστημα διαχείρισης βάσης δεδομένων HypersonicSQL που βρίσκετε στον φάκελο: ./db/hsqldb/hibernate.