ΔΙΑΧΕΙΡΙΣΗ ΣΥΜΦΡΑΖΟΜΕΝΩΝ ΣΕ ΠΕΡΙΒΑΛΛΟΝΤΑ ΠΑΝΤΑΧΟΥ ΠΑΡΟΝΤΟΣ ΥΠΟΛΟΓΙΣΜΟΥ

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Πληροφορικής
Εξεταστική Επιτροπή

από τον

Κωνσταντίνο Γεωργούλα

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΟ ΛΟΓΙΣΜΙΚΟ

Ιούνιος 2008

ΑΦΙΕΡΩΣΗ

Στην οικογένεια μου.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, κ. Απόστολο Ζάρρα, για την σημαντική βοήθεια που μου παρείχε προκειμένου να φέρω εις πέρας την παρούσα διατριβή. Ιδιαίτερα να τον ευχαριστήσω για την αμέριστη συμπαράσταση του όλο το χρονικό διάστημα κατά τη συγγραφή της εργασίας. Θερμές ευχαριστίες τόσο στην οικογένεια μου όσο και στα κοντινά μου πρόσωπα για την υποστήριξη τους στην προσπάθεια μου να ολοκληρώσω επιτυχώς το δύσκολο αυτό εγχείρημα.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

ΠΕΡΙΛΗΨΗ

Κωνσταντίνος Γεωργούλας του Θωμά και της Ιωάννας. MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Ιούνιος, 2008. Διαχείριση Συμφραζομένων σε Περιβάλλοντα Πανταχού Παρόντος Υπολογισμού. Επιβλέποντας: Απόστολος Ζάρρας.

Σκοπός της παρούσας διατριβής είναι η ανάπτυξη του CoWSAMI, ενός ενδιάμεσου λογισμικού που δίνει τη δυνατότητα ανάπτυξης εφαρμογών οι οποίες είναι ενήμερες συμφραζομένων και οι οποίες λειτουργούν σε ένα περιβάλλον πανταχού παρόντος υπολογισμού. Σε ένα τέτοιο περιβάλλον κάθε οντότητα μπορεί να έχει το ρόλο ενός παρόχου συμφραζομένων ή ενός καταναλωτή συμφραζομένων ή ακόμη και των δύο μαζί. Οι πάροχοι συμφραζομένων παρέχουν τις πληροφορίες τους μέσω μιας ή περισσοτέρων υπηρεσιών Διαδικτύου. Κάθε χρήστης μπορεί να καθορίσει τα συμφραζόμενα για τα οποία ενδιαφέρεται, τα οποία ορίζονται σαν ένα σύνολο από ιδιότητες, οι οποίες λέγονται ιδιότητες συμφραζομένων. Οι ιδιότητες συμφραζομένων οργανώνονται από το χρήστη σε σχέσεις συμφραζομένων. Κάθε σχέση συμφραζομένων χαρακτηρίζεται από το όνομα της και το σύνολο των ιδιοτήτων της. Τα συμφραζόμενα αποθηκεύονται στη βάση δεδομένων του CoWSAMI, που είναι εγκατεστημένη στη συσκευή του παρόχου/καταναλωτή συμφραζομένων, σε πλειάδες. Η συλλογή τιμών για τις ιδιότητες κάθε σχέσης συμφραζομένων γίνεται με βάση κανόνες που έχει ορίσει ο χρήστης και οι οποίοι είναι αποθηκευμένοι και αυτοί στη βάση δεδομένων. Οι κανόνες αυτοί αντιστοιχίζουν ιδιότητες των σχέσεων συμφραζομένων σε λειτουργίες των υπηρεσιών Διαδικτύου που προσφέρει ένας πάροχος συμφραζομένων. Η κλήση των λειτουργιών έχει ως αποτέλεσμα την επιστροφή τιμών για τις αντίστοιχες ιδιότητες. Στόχος είναι να μπορούν οι χρήστες να θέτουν SQL ερωτήσεις στο CoWSAMI το οποίο με τη σειρά του 1) εντοπίζει παρόχους που μπορούν να συνεισφέρουν με συμφραζόμενα στον εμπλουτισμό των σχέσεων συμφραζομένων που ορίζει ο χρήστης 2) συλλέγει τα συμφραζόμενα από αυτούς, ανεξάρτητα του είδους διεπαφής μέσω της οποίας προσφέρονται 3) κατασκευάζει απάντηση στα SQL ερωτήματα με βάση τα συμφραζόμενα.

EXTENDED ABSTRACT IN ENGLISH

Georgoulas, Konstantinos. MSc, Computer Science Department, University of Ioannina, Greece. June, 2008. Context Management in Pervasive Environment Computing. Thesis Supervisor: Apostolos Zarras.

The rapid growth of wireless technology, coupled with advances in mobile computing devices, such as PDA's and mobile telephones, have fundamentally changed the way of designing and developing distributed systems. Current users of portable devices are increasing rapidly and the use of applications in mobile networks is part of our everyday life. Applications should not only be easy to use, but it should head towards the direction of pervasive environment computing. In such environments applications should be available everywhere at any time. For this reason the last few years an effort is undergone making these environments capable of offering a more friendly interactive experience between user and applications. In this effort context awareness contributes considerably, by enabling applications collecting information from their users' environment and then adapt their functionality according to its state, taking into account the increase of their effectiveness and usability.

The aim of the present thesis is the development of CoWSAMI, a middleware infrastructure that enables the development of context aware applications in a pervasive environment. Main characteristic of this middleware is that it allows the development of context aware applications in environments where context providers join and leave continuously and moreover, have limited calculating and communicating resources (e.g. battery, memory, stocking spaces). Thus this particular middleware allows the development of context aware applications in environments where context providers are quite heterogeneous. For example in an environment such as a metropolitan wireless network, users roaming in it, which act as context providers, use different type of devices (e.g. laptops, PDA's, mobile telephones) with different operating systems, different programming languages but also with different

communication protocols. Furthermore, one important element of heterogeneity in such environments is the existence of different interfaces of users' applications which are used for context gathering. Contemplating that semantically same context is provided from different providers via different/several interfaces, it becomes perceptible how much heterogeneous is such an environment and how difficult is the development of context aware applications. CoWSAMI overcomes all the aforementioned difficulties and allows the development of context aware applications in such environments.

Particularly, in CoWSAMI each entity may play the role of a context provider, the role of a context consumer or both. Context providers provide contextual information through one or more Web services. Context is specified as a set of context attributes. Context attributes are organised by the user in context relations. Each context relation is characterized by its name and its attributes. Contextual information is stored in the database of CoWSAMI, which is installed at the device of context provider/consumer, in form of tuples. The gathering of values for the attributes of each context relation is based on rules that are user defined. Rules are stored in the database and context attributes correspond to Web services operations using those rules. Calling these operations has as a result the return of values for the corresponding attributes. Our objective is that the users place SQL queries in CoWSAMI, which 1) locates context providers that contribute with contextual information in the enrichment of context relations that the user defines, 2) collects context from providers, despite of the interface through which they offer it, 3) answers the SQL queries according to the contextual information it gathers.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

1.1. Πανταχού Παρών Υπολογισμός και Ενημερότητα Συμφραζομένων

Η ραγδαία εξέλιξη της ασύρματης τεχνολογίας σε συνδυασμό με τα πλεονεκτήματα που παρέχει η χρήση φορητών συσκευών, όπως τα PDA's και τα κινητά τηλεφώνα, έχουν αλλάξει ριζικά τον τρόπο κατασκευής και σχεδίασης κατανεμημένων συστημάτων. Σε αντίθεση με τον υπολογισμό στην περίπτωση ενός σταθερού δικτύου, στα ασύρματα δίκτυα όλη η διαδικασία υποφέρει από περιορισμούς που προκύπτουν εξαιτίας των ελλιπών πόρων και εξαιτίας των συχνών αλλαγών στην διαθεσιμότητα των πόρων αυτών. Η τεχνολογία του ενδιάμεσου λογισμικού προσπάθησε να παρακάμψει αυτές τις δυσκολίες για να υποστηρίξει τον κατανεμημένο υπολογισμό σε ασύρματα δίκτυα στα οποία υπάρχουν πολλές κινητές συσκευές. Αυτό που κατάφερε το ενδιάμεσο λογισμικό ήταν να κρύψει τις λεπτομέρειες των λειτουργιών του δικτύου και να δώσει την ευκαιρία στους προγραμματιστές των εφαρμογών να εστιάσουν στην υλοποίηση της εφαρμογής αυτής καθ' αυτής. Μπορεί με τον τρόπο αυτό να ωφελήθηκαν οι εφαρμογές όσον αφορά στην εύκολη ανάπτυξη τους αλλά αυτό πλέον δεν είναι αρκετό. Οι σημερινοί χρήστες φορητών συσκευών αυξάνονται καθημερινά με ταχύτατους ρυθμούς και πλέον η χρήση εφαρμογών σε κινητά δίκτυα έχει μπει σχεδόν στην καθημερινότητα όλων. Επομένως οι εφαρμογές δεν αρκεί να είναι εύκολες στην υλοποίηση τους αλλά πρέπει να οδεύουν προς την κατεύθυνση της δημιουργίας περιβαλλόντων πανταχού παρόντος υπολογισμού. Σε ένα τέτοιο περιβάλλον οι εφαρμογές πρέπει να είναι διαθέσιμες παντού και πάντα στον περιβάλλοντα χώρο των χρηστών. Οι χρήστες και οι υποχρεώσεις τους αποτελούν πλέον το κεντρικό στοιχείο ενδιαφέροντος ενώ τεχνικά ζητήματα και συσκευές περνούν σε δεύτερη μοίρα. Για το λόγο αυτό άλλωστε τα τελευταία

χρόνια γίνεται προσπάθεια προκειμένου τα περιβάλλοντα αυτά να προσφέρουν μια πιο φυσική και φιλική αλληλεπίδραση μεταξύ χρήστη και εφαρμογών.

Στην προσπάθεια αυτή συμβάλει σημαντικά η *ενημερότητα συμφραζομένων*, δηλαδή η δυνατότητα των εφαρμογών να συλλέγουν πληροφορίες από το περιβάλλον του χρήστη και στη συνέχεια να προσαρμόζουν τη λειτουργία τους ανάλογα με την κατάσταση που επικρατεί σε αυτό, με γνώμονα την αύξηση της αποτελεσματικότητας και της χρηστικότητάς τους. Για να γίνουν όμως πιο σαφή όλα τα παραπάνω είναι απαραίτητο να δοθεί ο ορισμός της έννοιας συμφραζόμενα και κατ' επέκταση των εφαρμογών που υποστηρίζουν ενημερότητα συμφραζομένων. Ανατρέχοντας στο παρελθόν εντοπίζονται πολλοί διαφορετικοί ορισμοί για το τι είναι συμφραζόμενα. Αρχικά σαν συμφραζόμενα ορίστηκαν [16] οι πληροφορίες για τη θέση και την ταυτότητα ανθρώπων και αντικειμένων που βρίσκονται κοντά σε κάποιον καθώς επίσης και οι αλλαγές που συμβαίνουν σε αυτές. Αργότερα, ως συμφραζόμενα θεωρήθηκε [15] οτιδήποτε είχε να κάνει με τη θέση, το περιβάλλον, την ταυτότητα και τον χρόνο ενός χρήστη. Άλλη προσέγγιση θεωρούσε σαν συμφραζόμενα [2] τα συστατικά στοιχεία του περιβάλλοντος του χρήστη που ο υπολογιστής γνώριζε. Εν τέλει η πιο ακριβής και σαφής διατύπωση για τον όρο συμφραζόμενα [7] λέει ότι συμφραζόμενα είναι οποιεσδήποτε πληροφορίες μπορεί να χρησιμοποιηθούν για να χαρακτηρίσουν την κατάσταση οντοτήτων, όπως είναι ένα άτομο, μια τοποθεσία ή ένα αντικείμενο, οι οποίες είναι σχετικές με την αλληλεπίδραση του χρήστη με την εφαρμογή, συμπεριλαμβανομένου του χρήστη και της εφαρμογής.

Ο στόχος της παρούσας διατριβής είναι η ανάπτυξη του CoWSAMI, ενός ενδιάμεσου λογισμικού που δίνει τη δυνατότητα ανάπτυξης εφαρμογών οι οποίες είναι ενήμερες συμφραζομένων και οι οποίες λειτουργούν σε ένα περιβάλλον πανταχού παρόντος υπολογισμού. Κύριο χαρακτηριστικό αυτού του ενδιάμεσου λογισμικού είναι η δυνατότητα ανάπτυξης εφαρμογών με ενημερότητα συμφραζομένων σε περιβάλλοντα όπου οι πάροχοι συμφραζομένων εισέρχονται και εξέρχονται από το περιβάλλον συνεχώς και επιπλέον έχουν περιορισμένους υπολογιστικούς και επικοινωνιακούς πόρους (π.χ. μπαταρία, μνήμη, αποθηκευτικούς χώρους). Πέραν αυτού όμως το συγκεκριμένο ενδιάμεσο λογισμικό επιτρέπει την υλοποίηση εφαρμογών που είναι ενήμερες συμφραζομένων σε περιβάλλοντα όπου οι πάροχοι συμφραζομένων είναι αρκετά ανομοιογενείς. Αν για παράδειγμα θεωρηθεί ένα περιβάλλον που υποστηρίζεται από ένα μητροπολιτικό ασύρματο δίκτυο θα παρατηρηθεί

ότι οι χρήστες που κινούνται μέσα σε αυτό, οι οποίοι αποτελούν παρόχους συμφραζομένων, χρησιμοποιούν συσκευές διαφορετικού τύπου (π.χ. φορητούς υπολογιστές, PDA's, κινητά τηλέφωνα) με διαφορετικά λειτουργικά συστήματα η καθεμία, με διαφορετικές γλώσσες προγραμματισμού για την υλοποίηση των λειτουργιών τους αλλά και με διαφορετικά πρωτόκολλα επικοινωνίας. Όλα αυτά τα προβλήματα, λίγο έως πολύ αντιμετωπίζονται από πλατφόρμες ενδιάμεσου λογισμικού που προηγήθηκαν του CoWSAMI. Επιπλέον όλων αυτών όμως ένα ακόμη πιο σημαντικό στοιχείο ανομοιομορφίας τέτοιων περιβαλλόντων είναι οι διαφορετικές διεπαφές (interfaces) που παρέχονται από τις εφαρμογές των χρηστών και οι οποίες χρησιμοποιούνται για τη συλλογή των συμφραζομένων. Αν αναλογιστεί κανείς ότι μπορεί να παρέχονται σημασιολογικά ίδια συμφραζόμενα από διαφορετικούς παρόχους με διαφορετικού τύπου διεπαφές γίνεται αντιληπτό πόσο ανομοιογενές είναι ένα τέτοιο περιβάλλον και πόσο δύσκολο είναι να αναπτυχθούν εφαρμογές ενήμερες συμφραζομένων. Το CoWSAMI, που προτείνεται σε αυτή τη διατριβή, παρακάμπτει όλες τις προαναφερθείσες δυσκολίες και επιτρέπει την ανάπτυξη εφαρμογών με ενημερότητα συμφραζομένων σε τέτοια περιβάλλοντα. Στα επόμενα κεφάλαια θα γίνει μια περιγραφή των βασικών συστατικών ενός τέτοιου περιβάλλοντος, θα περιγραφεί με αναλυτικό τρόπο η αρχιτεκτονική του ενδιάμεσου λογισμικού που προτείνεται ως λύση στο συγκεκριμένο πρόβλημα, θα παρουσιαστούν οι τρόποι υλοποίησης των δομικών στοιχείων και τέλος θα γίνει αξιολόγηση της λειτουργίας του εστιάζοντας στην αποδοτικότητα και τη χρηστικότητα του.

1.2. Δομή της Διατριβής

Η παρούσα διατριβή αποτελείται από 6 κεφάλαια. Στο Κεφάλαιο 2 παρουσιάζονται οι σημαντικότερες εργασίες που έχουν γίνει για την υλοποίηση πλατφορμών ενδιάμεσου λογισμικού με στόχο την ανάπτυξη εφαρμογών με ενημερότητα συμφραζομένων. Οι πληροφορίες που αναλύονται για κάθε διαφορετική προσέγγιση αντιμετώπισης του προβλήματος έχουν να κάνουν με την αρχιτεκτονική της εκάστοτε προτεινόμενης λύσης, με τις μεθόδους συλλογής των συμφραζομένων, με τις τεχνικές διαχείρισης των συμφραζομένων καθώς και με τα μοντέλα που χρειάζονται για τον καθορισμό και αποθήκευση των συμφραζομένων.

Στο Κεφάλαιο 3 υπάρχει αρχικά η περιγραφή του WSAMI, το οποίο είναι το υπόβαθρο για την ανάπτυξη του ενδιάμεσου λογισμικού που προτείνεται σε αυτή τη διατριβή. Το WSAMI

είναι μια δομή ενδιάμεσου λογισμικού κύριο χαρακτηριστικό της οποίας είναι το ότι υποστηρίζει την ανάπτυξη υπηρεσιών Διαδικτύου σε φορητές συσκευές οι οποίες έχουν περιορισμένους υπολογιστικούς και επικοινωνιακούς πόρους. Στη συνέχεια του κεφαλαίου παρουσιάζεται αναλυτικά το CoWSAMI, το ενδιάμεσο λογισμικό που προτείνεται στη συγκεκριμένη διατριβή ως μια λύση για την ανάπτυξη και λειτουργία εφαρμογών που διαθέτουν ενημερότητα συμφραζομένων. Όλες οι υπηρεσίες που παρέχει το ενδιάμεσο λογισμικό περιγράφονται αναλυτικά ως προς τον τρόπο υλοποίησης τους αλλά και ως προς τις λειτουργίες που προσφέρει η καθεμιά στον χρήστη. Επίσης, περιγράφεται η αρχιτεκτονική του και επεξηγούνται οι βασικές αρχές στις οποίες στηρίζεται καθώς και οι τεχνικές καθορισμού των συμφραζομένων αλλά και εκείνες της συλλογής και αποθήκευσης τους.

Το Κεφάλαιο 4 περιέχει τα αποτελέσματα που προέκυψαν μετά από τις πειραματικές μετρήσεις που έγιναν κατά τη φάση λειτουργίας του προτεινόμενου ενδιάμεσου λογισμικού. Πιο αναλυτικά, το κεφάλαιο αυτό περιλαμβάνει διαγραμματικές απεικονίσεις των αποτελεσμάτων και σχόλια σχετικά με την απόδοση του CoWSAMI όπως αυτή προκύπτει μετά τη διεξαγωγή των πειραμάτων. Το κριτήριο που επιλέχθηκε για μελέτη της απόδοσης του CoWSAMI ήταν ο χρόνος απόκρισης των βασικών υπηρεσιών του. Πέραν όμως αυτών των μετρήσεων παρουσιάζεται και μια πρώτη μελέτη σχετικά με τη χρηστικότητα του ενδιάμεσου λογισμικού η οποία κατά κύριο λόγο στηρίζεται στη χρήση του γραφικού περιβάλλοντος διεπαφής που δημιουργήθηκε για την πιο εύκολη και ουσιαστική χρήση του. Πιο συγκεκριμένα, παρατίθενται στατιστικά στοιχεία που προέκυψαν μετά από απαντήσεις σε ερωτηματολόγια που δόθηκαν σε διάφορους χρήστες που χρησιμοποίησαν το ενδιάμεσο λογισμικό.

Το Κεφάλαιο 5 περιέχει χρήσιμες οδηγίες για την εγκατάσταση και χρήση του CoWSAMI. Περιλαμβάνει ένα αναλυτικό εγχειρίδιο εγκατάστασης του ενδιάμεσου λογισμικού το οποίο περιγράφει όλες τις ενέργειες από τη λήψη μέχρι την εγκατάσταση όλων των προαπαιτουμένων εφαρμογών λογισμικού μαζί με τις ρυθμίσεις που απαιτούνται για αυτές αλλά και για το λειτουργικό σύστημα. Όσο για τον οδηγό χρήσης περιλαμβάνει πλήρη περιγραφή του τρόπου χρήσης του γραφικού περιβάλλοντος των υπηρεσιών του ενδιάμεσου λογισμικού με συγκεκριμένα παραδείγματα και γραφική απεικόνιση της κατάστασης που βρίσκεται σε κάθε περίπτωση, προκειμένου να είναι απλούστατη η πρώτη επαφή ενός χρήστη με το CoWSAMI.

Το κεφάλαιο 6 περιλαμβάνει τα συμπεράσματα που προέκυψαν μετά την εκπόνηση της συγκεκριμένης εργασίας αλλά και πιθανές επεκτάσεις του CoWSAMI που μπορούν να υλοποιηθούν στο μέλλον.

ΚΕΦΑΛΑΙΟ 2. ΣΧΕΤΙΚΗ ΒΙΒΛΙΟΓΡΑΦΙΑ

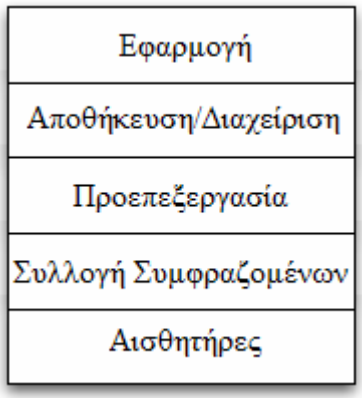
2.1. Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιαστούν διάφορες προσεγγίσεις που έχουν ήδη προταθεί και υλοποιηθεί για ανάπτυξη εφαρμογών με ενημερότητα συμφραζομένων. Πρόκειται για διαφορετικές πλατφόρμες που έχουν στόχο να δώσουν δυνατότητα στο χρήστη να μπορεί με εύκολο και απλό τρόπο να συλλέγει πληροφορίες που αφορούν το περιβάλλον του και στη συνέχεια να μπορεί να τροποποιεί τη συμπεριφορά του και τη συμπεριφορά των εφαρμογών του. Στις παρακάτω παραγράφους θα γίνει αναφορά στους τρόπους με τους οποίους επιτυγχάνεται αυτός ο στόχος και θα γίνουν φανερές οι διαφορές των προτάσεων. Πιο συγκεκριμένα, θα παρουσιαστούν οι αρχιτεκτονικές των προτεινόμενων πλατφορμών, οι τρόποι με τους οποίους γίνεται ο εντοπισμός των παρόχων των συμφραζομένων, τα μοντέλα που χρησιμοποιούνται για τη διαχείριση και αποθήκευση των συμφραζομένων αλλά και ο τρόπος επεξεργασίας τους.

Πριν όμως από την παρουσίαση των λύσεων καλό είναι να γίνει μια αναφορά σε μια κοινή αρχιτεκτονική [3] που χρησιμοποιείται από τα πιο πολλά συστήματα και πλατφόρμες που υλοποιήθηκαν τα τελευταία χρόνια. Μπορεί τα επίπεδα της αρχιτεκτονικής αυτής να διαφέρουν από σύστημα σε σύστημα σε θέματα που έχουν να κάνουν με το εύρος των

λειτουργιών του κάθε επιπέδου, τη θέση εγκατάστασης τους (π.χ. σε φορητούς ή σε κεντρικούς υπολογιστές) ή την ονομασία τους αλλά σχεδόν πάντα υπάρχουν πέντε επίπεδα. Από αυτά το χαμηλότερο έχει να κάνει με τους αισθητήρες, το δεύτερο με τη συλλογή των συμφραζομένων, το τρίτο με την εξαγωγή υψηλότερου επιπέδου συμφραζομένων, το τέταρτο με την αποθήκευση και διαχείριση των συμφραζομένων και το πέμπτο με τις εφαρμογές αυτές καθαυτές. Περιγράφοντας αναλυτικότερα την αρχιτεκτονική αξίζει να αναφερθεί ότι το πρώτο επίπεδο αποτελείται συνήθως από μια *συλλογή αισθητήρων* είτε φυσικών είτε εικονικών είτε λογικών. Οι εικονικοί αισθητήρες είναι εφαρμογές ή υπηρεσίες που παρέχουν πρωτογενή δεδομένα ενώ οι λογικοί αισθητήρες είναι εφαρμογές που παρέχουν υψηλότερου επιπέδου πληροφορία συνδυάζοντας δεδομένα από φυσικούς και εικονικούς αισθητήρες μαζί με επιπλέον πληροφορία από βάσεις δεδομένων. Το δεύτερο επίπεδο είναι υπεύθυνο για τη *συλλογή των συμφραζομένων*. Χρησιμοποιεί τους κατάλληλους οδηγούς (device drivers) για την περίπτωση συλλογής συμφραζομένων από φυσικούς αισθητήρες ή API's για την περίπτωση συλλογής συμφραζομένων από εικονικούς ή λογικούς αισθητήρες. Το τρίτο επίπεδο το οποίο λέγεται *επίπεδο Προεπεξεργασίας*, δεν υλοποιείται σε κάθε δομή αλλά σκοπός του είναι η εξαγωγή υψηλότερου επιπέδου συμφραζομένων συνδυάζοντας τα δεδομένα που συλλέγονται από το δεύτερο επίπεδο. Το *επίπεδο αποθήκευσης και διαχείρισης* οργανώνει τα συγκεντρωμένα συμφραζόμενα και τα προσφέρει στο χρήστη μέσω μιας κοινής διεπαφής. Ο χρήστης μπορεί να λάβει αυτά τα συμφραζόμενα είτε με *σύγχρονο* είτε με *ασύγχρονο* τρόπο. Στην πρώτη περίπτωση ο χρήστης ζητά κάθε φορά μέσω απομακρυσμένων κλήσεων την ενημέρωση για κάποια αλλαγή που συνέβη στο περιβάλλον του ενώ στη δεύτερη περίπτωση δηλώνει μια φορά το ενδιαφέρον του για κάποια συμφραζόμενα και ενημερώνεται αυτόματα σε κάθε αλλαγή που συμβαίνει σε αυτά. Στο υψηλότερο επίπεδο που λέγεται επίπεδο *εφαρμογής* υπάρχουν οι εφαρμογές και εκεί υλοποιούνται οι αντιδράσεις, δηλαδή οι αλλαγές συμπεριφοράς, του χρήστη και των εφαρμογών του στα διάφορα γεγονότα που λαμβάνουν χώρα στο περιβάλλον.

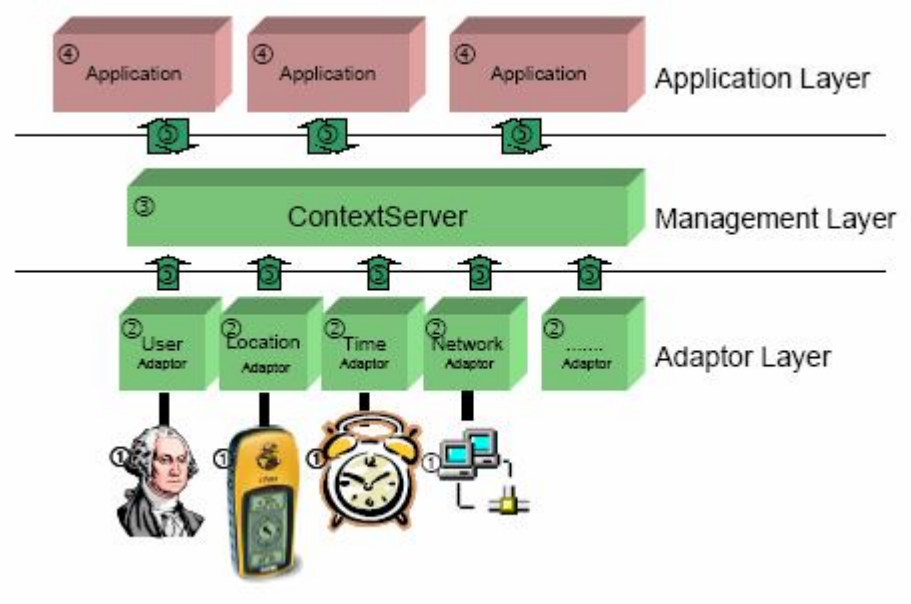
Έχοντας πλέον γίνει μια γενική παρουσίαση της αρχιτεκτονικής που ακολουθείται από τις πιο πολλές δομές που αναπτύχθηκαν για να υποστηρίξουν εφαρμογές με ενημερότητα συμφραζομένων θα περιγραφούν στη συνέχεια οι διάφορες προτάσεις που δημοσιεύτηκαν τα τελευταία χρόνια.



Σχήμα 2.1 Κοινή Αρχιτεκτονική Συστημάτων για Ανάπτυξη Εφαρμογών με Ενημερότητα Συμφραζομένων

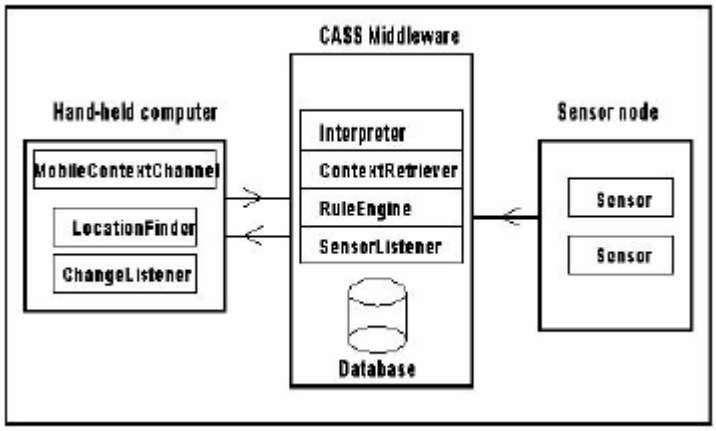
2.2. Αρχιτεκτονική

Μια από τις λύσεις που προτάθηκε είναι αυτή του Hydrogen [11]. Η αρχιτεκτονική του συστήματος έχει τρία επίπεδα: Το πρώτο επίπεδο που λέγεται επίπεδο Adaptor είναι υπεύθυνο για τη συλλογή των συμφραζομένων από τους αισθητήρες και την παράδοση τους στο δεύτερο επίπεδο. Τα συμφραζόμενα που συλλέγονται στο επίπεδο αυτό κατηγοριοποιούνται σε πέντε ομάδες οι οποίες είναι: ο χρόνος, ο τόπος, η συσκευή, ο χρήστης και το δίκτυο. Το δεύτερο επίπεδο, το οποίο λέγεται επίπεδο διαχείρισης περιλαμβάνει τον ContextServer ο οποίος μέσω των μεθόδων του, δίνει τη δυνατότητα στις εφαρμογές να βρίσκουν ή να δηλώνουν το ενδιαφέρον τους για κάποια συμφραζόμενα. Με τη βοήθεια του Context Server κάθε συσκευή μπορεί να μοιράζεται πληροφορίες με άλλες που βρίσκονται στην εμβέλεια της και τελικά να χρησιμοποιεί συμφραζόμενα που έχουν συλλεχθεί από απομακρυσμένους αισθητήρες που υπό άλλες συνθήκες δεν θα μπορούσε να επικοινωνήσει. Το τρίτο επίπεδο αποτελείται από τις εφαρμογές που εγκαθίστανται και λειτουργούν στο συγκεκριμένο σύστημα. Εξαιτίας του παραπάνω διαχωρισμού σε επίπεδα υπάρχει η δυνατότητα δύο ή περισσότερες εφαρμογές να έχουν πρόσβαση ταυτόχρονα στα ίδια συμφραζόμενα. Εκτός αυτού οι εφαρμογές μπορούν να αποκτούν τα συμφραζόμενα με δύο τρόπους: είτε με σύγχρονο τρόπο κάνοντας απευθείας ερώτηση στον ContextServer για ένα συγκεκριμένο συμφραζόμενο είτε με ασύγχρονο τρόπο δηλώνοντας την επιθυμία τους να ενημερώνονται για οποιαδήποτε αλλαγή συμβαίνει σε συγκεκριμένο συμφραζόμενο. Όσον αφορά την επικοινωνία μεταξύ των τριών αυτών επιπέδων γίνεται με τη χρήση ενός πρωτοκόλλου ανταλλαγής XML μηνυμάτων.



Σχήμα 2.2 Αρχιτεκτονική του Hydrogen[11]

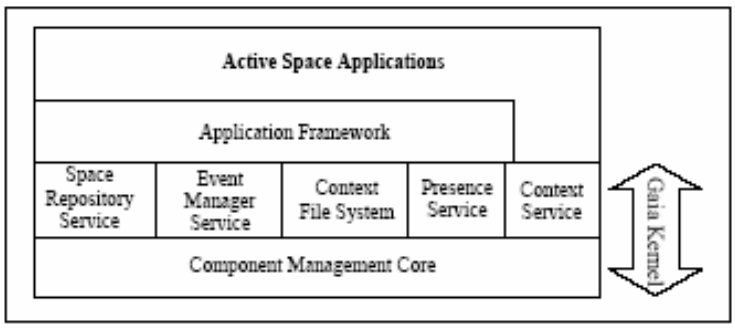
Μια άλλη πρόταση είναι το CASS [9], το οποίο είναι μια κεντρικοποιημένη δομή ενδιάμεσου λογισμικού της οποίας χαρακτηριστικό είναι η χρήση υψηλότερου επιπέδου συμφραζομένων. Η αρχιτεκτονική του CASS έχει ως εξής: αποτελείται από κόμβους αισθητήρων οι οποίοι είναι ηλεκτρονικοί υπολογιστές στους οποίους συνδέονται ένας ή περισσότεροι φυσικοί αισθητήρες. Οι εφαρμογές του είναι εγκατεστημένες σε φορητούς υπολογιστές οι οποίοι δεν επικοινωνούν άμεσα με τους κόμβους αισθητήρων αλλά απευθύνονται στην κεντρική δομή του ενδιάμεσου λογισμικού για να ενημερωθούν για τα συμφραζόμενα. Η κεντρική αυτή δομή του CASS είναι εγκατεστημένη σε ένα κεντρικό υπολογιστή και διαθέτει μια βάση δεδομένων στην οποία αποθηκεύονται συμφραζόμενα, δεδομένα χρηστών και εφαρμογών αλλά και κανόνες και συμπεριφορές σχετικές με συγκεκριμένες εφαρμογές. Αποτελείται από κλάσεις με διαφορετικές αρμοδιότητες η καθεμιά. Κεντρικό στοιχείο της πλατφόρμας είναι η RuleEngine και η κλάση Sensor Listener η οποία λαμβάνει ενημερώσεις από τους αισθητήρες για τυχών αλλαγές στα συμφραζόμενα και τις αποθηκεύει στη βάση δεδομένων. Ο ContextRetriever ανακτά από τη βάση τα αποθηκευμένα συμφραζόμενα. Οι δύο αυτές κλάσεις μπορούν να χρησιμοποιούν τις υπηρεσίες του Interpreter. Τέλος ο ChangeListener, ο οποίος βρίσκεται στους φορητούς υπολογιστές, έχει επικοινωνιακές δυνατότητες και επιτρέπει σε αυτούς να αντιλαμβάνονται αλλαγές που γίνονται σε συμφραζόμενα.



Σχήμα 2.3 Αρχιτεκτονική του CASS[9]

Η Gaia [14] μία άλλη πλατφόρμα ενδιάμεσου λογισμικού, δημιουργήθηκε με στόχο την ανάπτυξη και λειτουργία εφαρμογών σε active spaces. Σε αντίθεση με το CASS, η Gaia είναι μια κατανεμημένη δομή ενδιάμεσου λογισμικού η οποία συνδυάζει οντότητες λογισμικού και ανομοιογενείς δικτυακές συσκευές που υπάρχουν σε ένα φυσικό χώρο. Τα τρία βασικά κομμάτια της αρχιτεκτονικής της Gaia είναι ο Gaia Kernel, το Gaia application Framework και οι εφαρμογές. Ο Πυρήνας της Gaia (Gaia Kernel) περιέχει ένα σύστημα εγκατάστασης και διαχείρισης για κατανεμημένα αντικείμενα και ένα σύνολο από αλληλοσχετιζόμενες υπηρεσίες που χρησιμοποιούν οι εφαρμογές. Το σύστημα διαχείρισης και εγκατάστασης φορτώνει, μεταφέρει, δημιουργεί, καταστρέφει όλα τα συστατικά στοιχεία και τις εφαρμογές της Gaia. Τα συστατικά αυτά είναι κατανεμημένα αντικείμενα και για την απομακρυσμένη αλληλεπίδραση τους χρησιμοποιείται η CORBA. Οι υπηρεσίες του πυρήνα της Gaia είναι πέντε: α) Ο event manager ο οποίος είναι υπεύθυνος για τη διάδοση των γεγονότων που λαμβάνουν χώρα στο περιβάλλον της πλατφόρμας. Για να το πετύχει αυτό στηρίζεται σε ένα μοντέλο επικοινωνίας που περιλαμβάνει παρόχους γεγονότων, καταναλωτές γεγονότων και κανάλια μετάδοσης γεγονότων. Σε κάθε τέτοιο κανάλι υπάρχουν πάροχοι που δίνουν πληροφορία για κάποιες αλλαγές που έγιναν και καταναλωτές που λαμβάνουν αυτή την πληροφορία. β) Η context service επιτρέπει στις εφαρμογές να ρωτούν για συμφραζόμενα και να εκδηλώνουν το ενδιαφέρον τους για αλλαγές που συμβαίνουν σε κάποια από αυτά προκειμένου να προσαρμόζουν τη συμπεριφορά τους. Τα συμφραζόμενα συλλέγονται από τους παρόχους συμφραζομένων που μπορεί να είναι φυσικοί αισθητήρες ή αντικείμενα που παράγουν συμφραζόμενα υψηλότερου επιπέδου. γ) Η presence service διατηρεί ενημερωμένη την πληροφορία σχετικά με το ποιες οντότητες, είτε ψηφιακές είτε φυσικές, υπάρχουν ανά

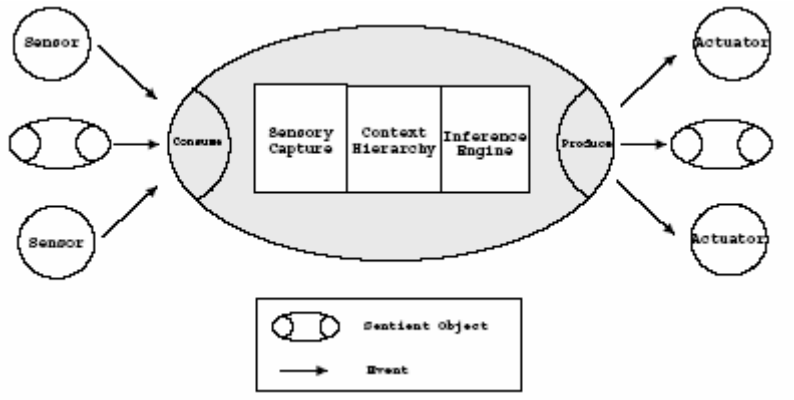
πάσα στιγμή στο χώρο δ) Το space repository το οποίο αποθηκεύει πληροφορίες για τις οντότητες υλικού και λογισμικού που υπάρχουν στο περιβάλλον. Παρέχει δε και λειτουργίες για την εύρεση αυτών των οντοτήτων, από τις εφαρμογές, βάση κάποιων χαρακτηριστικών τους. ε) Το context file system (CFS) χρησιμοποιείται για να κάνει τα προσωπικά δεδομένα των χρηστών αυτομάτως γνωστά στις εφαρμογές με την παρουσία τους. Το CFS δημιουργεί μια ιεραρχία εικονικών φακέλων όπου τα συμφραζόμενα αναπαρίστανται με φακέλους ενώ στοιχεία του μονοπατιού ενός φακέλου αναπαριστούν τύπους συμφραζομένων και τιμές. Έτσι για παράδειγμα στο φάκελο /location:/RM2401/situation:/meeting υπάρχουν αρχεία που σχετίζονται με τα συμφραζόμενα ότι στο δωμάτιο RM2401 διεξάγεται μια συνάντηση (δηλαδή location==RM2401 && situation==meeting).



Σχήμα 2.4 Αρχιτεκτονική της Gaia[14]

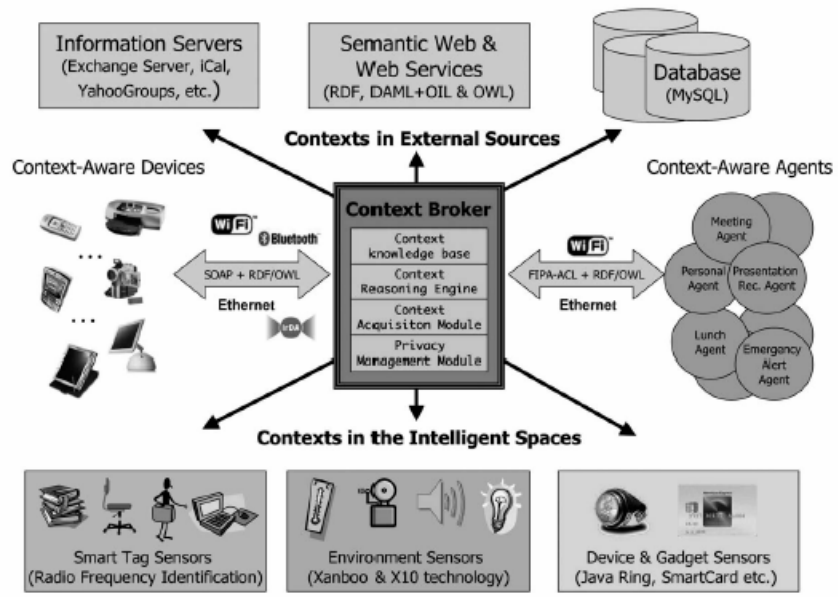
Πέρα από τις προαναφερόμενες λύσεις μπορούμε να εστιάσουμε στο CORTEX [4] το οποίο στηρίζεται στη χρήση sentient objects. Στη συγκεκριμένη λοιπόν πλατφόρμα οι εφαρμογές δημιουργούν sentient objects τα οποία συλλέγουν τα συμφραζόμενα και στη συνέχεια στηριζόμενα σε κανόνες που ορίζονται από τον χρήστη τροποποιούν τη συμπεριφορά της εφαρμογής. Σύμφωνα με τους συγγραφείς, ένα sentient αντικείμενο είναι μια οντότητα η οποία αποτελείται από τρία μέρη:Sensory Capture, Context Hierarchy, Inference Engine. Μέσω των διεπαφών του, ένα sentient αντικείμενο επικοινωνεί με τους αισθητήρες, οι οποίοι παράγουν γεγονότα, και τους actuators, οι οποίοι είναι οντότητες που καταναλώνουν γεγονότα. Κάθε sentient object παράγει γεγονότα που προκύπτουν μετά από συνδυασμό και ερμηνεία όλων των πληροφοριών που συλλέγονται από τους αισθητήρες. Η παραγωγή τέτοιων γεγονότων γίνεται σε τρεις φάσεις. Αρχικά επεξεργάζονται τα συμφραζόμενα που συλλέχθηκαν από τους αισθητήρες για να διαχειριστεί η αβεβαιότητα τους. Στη συνέχεια

δομείται μια ιεραρχία συμφραζομένων και έπειτα μέσω της inference engine παράγονται τα γεγονότα βάσει των κανόνων που έχουν οριστεί από τον χρήστη. Τα γεγονότα αυτά καταναλώνονται από τους actuators οι οποίοι στην ουσία μετατρέπουν αυτά τα γεγονότα σε εντολές επηρεάζοντας έτσι την συμπεριφορά μιας εφαρμογής που τους χρησιμοποιεί.



Σχήμα 2.5 Αρχιτεκτονική του CORTEX[4]

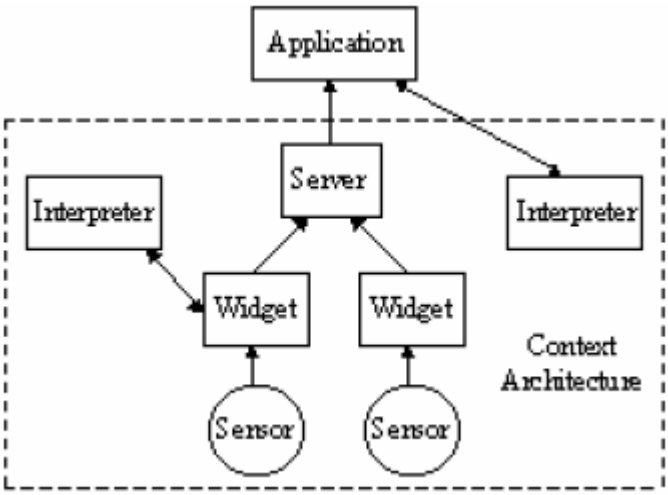
Το CoBrA [5] είναι επίσης μια πρόταση για την υποστήριξη εφαρμογών με ενημερότητα συμφραζομένων. Το βασικό συστατικό αυτής της πλατφόρμας είναι μια συλλογή οντολογιών η οποία λέγεται COBRA-ONT και χρησιμοποιείται για την μοντελοποίηση των συμφραζομένων στο περιβάλλον. Κεντρικό συστατικό της αρχιτεκτονικής της πλατφόρμας είναι ο context broker, ένας εξυπηρετητής που είναι εγκατεστημένος σε ένα ισχυρό κεντρικό υπολογιστή, σκοπός του οποίου είναι να διατηρεί τα συμφραζόμενα εκ μέρους των συσκευών του περιβάλλοντος και να προστατεύει την ιδιωτικότητα των χρηστών ακολουθώντας τους κανόνες και την πολιτική που έχει καθορίσει ο καθένας για λογαριασμό του. Ο context broker αποτελείται από τέσσερα κομμάτια: την Context Knowledge Base η οποία αποθηκεύει μόνιμα όλα τα συμφραζόμενα, την Context Reasoning Engine η οποία αποφασίζει για την αντίδραση της πλατφόρμας με βάση τη γνώση που υπάρχει από τα συμφραζόμενα, το Context Acquisition Module που είναι μια βιβλιοθήκη από διαδικασίες που χρησιμοποιούνται από τους χρήστες για αναζήτηση και απόκτηση συμφραζομένων και το Policy Management Module το οποίο είναι ένα σύνολο κανόνων μέσω των οποίων ο core broker αποφασίζει τι δικαίωμα έχει κάθε οντότητα στο να μοιραστεί συμφραζόμενα αλλά και ποιες είναι εκείνες οι οντότητες που πρέπει να ενημερωθούν για κάποιες αλλαγές στο περιβάλλον του συστήματος.



Σχήμα 2.6 Αρχιτεκτονική του CoBrA[5]

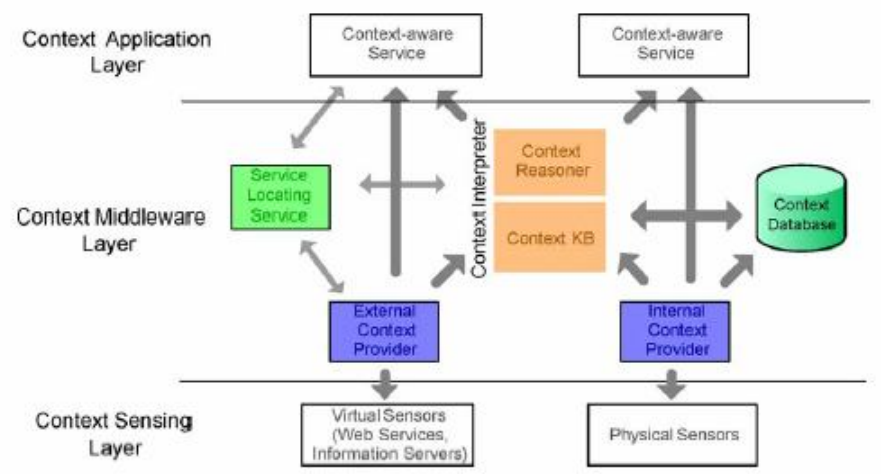
Επιπροσθέτως αναφορά πρέπει να γίνει στο Context Toolkit [8] το οποίο είναι ένα σύστημα που σχεδιάστηκε χρησιμοποιώντας αντικειμενοστραφή προσέγγιση. Το συγκεκριμένο σύστημα αποτελείται από τρία είδη αντικειμένων: τα widgets, τους servers και τους interpreters. Και τα τρία είδη αντικείμενων εκτελούνται και αρχικοποιούνται αυτόνομα. Το context widget είναι κάτι παρόμοιο με ένα user interface widget, το οποίο καθορίζεται από τις ιδιότητες και τα callbacks του. Οι ιδιότητες είναι συμφραζόμενα τα οποία συλλέγονται από τους αισθητήρες του περιβάλλοντος και στη συνέχεια μπορούν να αποσταλούν σε άλλα συστατικά του συστήματος. Τα callbacks είναι τύποι γεγονότων, για την εμφάνιση των οποίων το widget ενημερώνει τα συστατικά μέρη του συστήματος τα οποία του έχουν ζητήσει ενημέρωση σε τυχών αλλαγές. Κάθε context server συλλέγει τα συμφραζόμενα που σχετίζονται με μια οντότητα π.χ. ένα άτομο. Άρα υπάρχουν τόσοι context servers όσες και οι οντότητες του συστήματος. Οι εφαρμογές ζητούν από τον context server της κάθε οντότητας τα συμφραζόμενα για τα οποία ενδιαφέρονται. Πρέπει να σημειώσουμε ότι τόσο τα widgets όσο και οι context servers αποθηκεύουν μόνιμα τα συμφραζόμενα που συλλέγουν. Οι context interpreters από την μεριά τους αυτό που κάνουν είναι να συνδυάσουν πληροφορία που περιγράφει την κατάσταση του περιβάλλοντος και να παράγουν μια νέα γενικευμένη πληροφορία. Οι context interpreters μπορούν να χρησιμοποιηθούν από widgets, context servers, εφαρμογές ακόμη και από άλλους interpreters. Για την επικοινωνία των τριών αυτών ειδών αντικειμένων υπάρχει ένα άλλο είδος αντικείμενων, το BaseObject, τις λειτουργίες των

οποίων κληρονομούν τα άλλα τρία αντικείμενα. Ένα αντικείμενο BaseObject παρέχει τις βασικές υπηρεσίες για την επικοινωνία των κατανεμημένων συστατικών του συστήματος και την απόκρυψη των λεπτομερειών που σχετίζονται με την ετερογένεια αυτών των συστατικών.



Σχήμα 2.7 Αρχιτεκτονική του ContextToolkit[8]

Μια άλλη πρόταση είναι το SOCAM [10], οι δημιουργοί του οποίου πρότειναν ένα κατανεμημένο ενδιάμεσο λογισμικό για την ανάπτυξη υπηρεσιών με ενημερότητα συμφραζομένων. Το SOCAM αποτελείται από πέντε βασικά συστατικά τα οποία είναι: οι context providers, ο context interpreter, η context database, οι context-aware services και η service locating service. Χαρακτηριστικό αυτής της πρότασης είναι η χρήση οντολογιών για την περιγραφή των συμφραζομένων. Κάθε context provider λαμβάνει συμφραζόμενα είτε από φυσικούς αισθητήρες είτε από εικονικούς αισθητήρες και τα αναπαριστά σαν γεγονότα με τη μορφή OWL περιγραφών. Ο context interpreter παρέχει υπηρεσίες για την επεξεργασία των συμφραζομένων που συλλέχθηκαν από τους context providers. Η context database αποθηκεύει τις οντολογίες των συμφραζομένων. Οι context-aware services αποτελούν το επίπεδο εφαρμογής και είναι αυτές που χρησιμοποιούν τα διαφόρων επιπέδων συμφραζόμενα και τροποποιούν τη συμπεριφορά τους ανάλογα με την κατάσταση που επικρατεί στο περιβάλλον τους. Όσο για τη service location service κύριο μέλημα της είναι να παρέχει το μηχανισμό που επιτρέπει στις εφαρμογές και τους χρήστες τον εντοπισμό των context providers και του context interpreter.



Σχήμα 2.8 Αρχιτεκτονική του SOCAM[10]

Τελειώνοντας θα γίνει αναφορά στο PICO [12] το οποίο είναι ένα άλλο ενδιάμεσο λογισμικό για ανάπτυξη εφαρμογών με ενημερότητα συμφραζομένων η αρχιτεκτονική του οποίου στηρίζεται σε οντότητες λογισμικού που λέγονται delegents και συσκευές που λέγονται camileuns. Οι delegents εκπροσωπούν τις camileuns σε κοινότητες που σκοπό έχουν την πιο αποδοτική χρήση των πόρων τους. Έτσι λοιπόν συσκευές που βρίσκονται σε κοντινές αποστάσεις μέσα στον ίδιο χώρο μπορούν να ανταλλάσουν δεδομένα που έχουν συλλέξει με τους αισθητήρες τους και επομένως οι delegents τους να συγκροτούν μια κοινότητα. Η αρχιτεκτονική του PICO αποτελείται από τέσσερα επίπεδα. Το πρώτο και χαμηλότερο στην ιεραρχία επίπεδο είναι αυτό των camileuns το οποίο αποτελείται από το υλικό για τον υπολογισμό και την επικοινωνία. Το δεύτερο επίπεδο είναι αυτό που προσφέρει το API για την επικοινωνία των delegents με τα camileuns. Στο τρίτο επίπεδο βρίσκουμε το delegent επίπεδο το οποίο αποτελείται από delegents που δημιουργήθηκαν για να φέρουν εις πέρας εργασίες και υπηρεσίες διαφόρων κοινοτήτων. Τέλος στο τέταρτο επίπεδο, το οποίο λέγεται επίπεδο κοινότητας, γίνεται η δημιουργία κοινοτήτων και δηλώνεται η λειτουργικότητα τους μαζί με το ενδιαφέρον τους και το στόχο τους. Οποιαδήποτε σειρά από γεγονότα η οποία αλλάζει τα συμφραζόμενα μπορεί να οδηγήσει σε δημιουργία κοινοτήτων delegents. Να σημειωθεί ότι για κάθε camileun μπορεί να υπάρχουν ένας ή περισσότεροι delegents που την αντιπροσωπεύουν. Έτσι λοιπόν οι κοινότητες των delegents μπορούν και παρέχουν τις υπηρεσίες του ενδιάμεσου λογισμικού σε ετερογενείς συσκευές.

2.3. Εντοπισμός Παρόχων Συμφραζομένων

Στην παράγραφο αυτή θα παρατεθούν οι τρόποι με τους οποίους οι διάφορες πλατφόρμες που αναλύθηκαν πιο πάνω εντοπίζουν τους παρόχους συμφραζομένων. Μια τέτοια διαδικασία εντοπισμού είναι χρήσιμη σε περιβάλλοντα όπου συνεχώς μεταβάλλονται και μπορεί ανά πάσα χρονική στιγμή κάποιοι αισθητήρες να τεθούν εκτός λειτουργίας ή κάποιοι άλλοι να προστεθούν ή ακόμη και κάποιοι πάροχοι συμφραζομένων (π.χ. εικονικοί ή λογικοί αισθητήρες) να γίνονται ή όχι διαθέσιμοι. Όσον αφορά στο Hydrogen, εξαιτίας του ότι χρησιμοποιεί συμφραζόμενα που παρέχονται από αισθητήρες οι οποίοι είναι τοπικά εγκατεστημένοι δεν απαιτείται μηχανισμός εντοπισμού άλλων παρόχων. Άλλωστε η ανταλλαγή συμφραζομένων μεταξύ των context servers που είναι εγκατεστημένοι σε διαφορετικές συσκευές απαλλάσσει τη δομή του Hydrogen από τέτοιου είδους μηχανισμούς μιας και οποιαδήποτε πληροφορία απαιτείται από άλλους παρόχους παρέχεται μέσω των context servers τους. Το SOCAM από τη μεριά του παρέχει την υπηρεσία service locating service η οποία έχει τη δυνατότητα να εντοπίζει τις αλλαγές στη διαθεσιμότητα των παρόχων συμφραζομένων (δηλ ποιοι φεύγουν και ποιοι έρχονται) και επιπλέον επιτρέπει στου παρόχους να διαφημίζουν τα συμφραζόμενα που παρέχουν. Στη Gaia ο συγκεκριμένος μηχανισμός υλοποιείται από την Context service η οποία διαθέτει μια λίστα καταγραφής όλων των παρόχων συμφραζομένων που υπάρχουν στο active space. Στο CASS οι αισθητήρες είναι ενσωματωμένοι σε υπολογιστές, τους λεγόμενους κόμβους αισθητήρων, οι οποίοι μέσω του Sensor Listener εντοπίζονται άμεσα. Στο CORTEX οι πάροχοι συμφραζομένων, οι οποίοι είναι οντότητες λογισμικού παρέχουν μια XML περιγραφή για το τι είδους γεγονότα λογισμικού παράγουν. Αυτές οι XML περιγραφές υπάρχουν σε ειδικές βιβλιοθήκες από τις οποίες τα sentient objects μπορούν εντοπίσουν ποιοι αισθητήρες υπάρχουν και να επιλέξουν από ποιον θα συλλέγουν συμφραζόμενα. Όσο για το PICO, κάθε πάροχος συμφραζομένων εκπροσωπείται από ένα delegent που είναι μια οντότητα λογισμικού η οποία έχει επικοινωνιακές δυνατότητες και μπορεί να εντοπίζει άλλους delegents για να συλλέξει συμφραζόμενα.

2.4. Διαδικασία Συλλογής Συμφραζομένων

Σχετικέ με τη διαδικασία συλλογής των συμφραζομένων μπορούμε να παρατηρήσουμε ότι στην περίπτωση του Context Toolkit χρησιμοποιείται η οντότητα του context widget η οποία συλλέγει τα συμφραζόμενα από διάφορους αισθητήρες και στην συνέχεια τα προσφέρει στις

εφαρμογές οι οποίες έχουν εκδηλώσει ενδιαφέρον για κάποιο συγκεκριμένο συμφραζόμενο ή ακόμη και να απαντά σε ερωτήσεις που έχουν να κάνουν με πληροφορία που έχει συλλέξει. Με αυτό τον τρόπο υποκρύπτονται από τις εφαρμογές όλες οι δυσκολίες ανάκτησης των πληροφοριών, που προκύπτουν εξαιτίας της ανομοιογένειας των συσκευών που παρέχουν τα συμφραζόμενα. Στην περίπτωση του CASS υπάρχουν κόμβοι αισθητήρων οι οποίοι είναι υπολογιστές στους οποίους έχουν συνδεθεί διάφοροι αισθητήρες. Μέσω του SensorListener γίνονται αντιληπτές οι αλλαγές συμφραζομένων που συλλέγουν οι αισθητήρες που υπάρχουν στους κόμβους και έτσι συγκεντρώνεται η πληροφορία στην κεντρική βάση δεδομένων του CASS. Στο SOCAM υπάρχουν οι context providers οι οποίοι συλλέγουν πληροφορίες από διάφορους αισθητήρες, είτε φυσικούς είτε εικονικούς, και στη συνέχεια αναπαριστούν όλη αυτή τη συγκεντρωμένη πληροφορία σαν ένα context event με τη μορφή μιας OWL περιγραφής. Στο CoBrA υπάρχει το Context Acquisition Module το οποίο αποτελεί κομμάτι του Context Broker και είναι μια βιβλιοθήκη από λειτουργίες που παρέχουν τη δυνατότητα απόκτησης συμφραζομένων. Στη Gaia με τη βοήθεια της Event Manager Service δημιουργούνται κανάλια όπου κάθε πάροχος μπορεί να δηλώσει αλλαγές που συνέβησαν στο περιβάλλον του συστήματος ενώ ταυτόχρονα μπορούν εφαρμογές να πληροφορηθούν τις αλλαγές αυτές. Σε κάθε κανάλι μπορούν να συνεισφέρουν περισσότεροι του ενός πάροχοι γεγονότων και μπορούν να λαμβάνουν τις πληροφορίες πολλοί ταυτόχρονα καταναλωτές γεγονότων. Στο CORTEX η συλλογή των συμφραζομένων γίνεται μέσω των sentient objects με τη βοήθεια των διεπαφών που διαθέτουν για την επικοινωνία τους με τους αισθητήρες. Στο Hydrogen η συλλογή των συμφραζομένων γίνεται από το επίπεδο Adaptor με τη χρήση κάποιων προσαρμογέων για κατηγορίες συμφραζομένων όπως: ο χρόνος, ο τόπος, η συσκευή, ο χρήστης και το δίκτυο. Στο PICO είναι πολύ απλός ο τρόπος συλλογής των συμφραζομένων και επιτυγχάνεται μέσω των delegents και της επικοινωνίας που έχουν μεταξύ τους αφού καθένας από αυτούς σχετίζεται αποκλειστικά με μια μόνο συσκευή του περιβάλλοντος.

2.5. Μοντέλο Καθορισμού Συμφραζομένων

Σχετικά με το μοντέλο καθορισμού συμφραζομένων που χρησιμοποιούν οι διάφορες προτεινόμενες λύσεις που υλοποιήθηκαν διακρίνουμε πολλές διαφορετικές προσεγγίσεις. Ένα μοντέλο που χρησιμοποιείται από δυο συστήματα, το SOCAM και το CoBrA, είναι αυτό της χρήσης οντολογιών. Σε αυτές τις περιπτώσεις οι οντότητες/συσκευές του περιβάλλοντος στο οποίο εγκαθίστανται το κάθε σύστημα περιγράφονται αναλυτικά από οντολογίες. Στο CoBrA

υπάρχουν οντολογίες που περιγράφουν την τοποθεσία, τους agents είτε είναι άτομα είτε είναι software agents, τα συμφραζόμενα της τοποθεσίας που βρίσκεται ένας agent και τα γεγονότα στα οποία συμμετέχει ένας agent. Στο SOCAM τα συμφραζόμενα αναπαρίστανται σε κατηγορηματικό λογισμό $1^{ης}$ τάξης της μορφής Predicate(subject, value). Το subject ανήκει σε ένα σύνολο από οντότητες π.χ. άνθρωπος, θέση ενός αντικειμένου κ.τ.λ., το value είναι η τιμή που μπορεί να πάρει το subject ενώ το predicate μπορεί να πάρει τιμές από ένα σύνολο predicates π.χ. "βρίσκεται στη", "η κατάσταση του είναι" κ.τ.λ. Στη συγκεκριμένη πλατφόρμα η οντολογία στηρίζεται σε ένα ιεραρχικό μοντέλο δύο επιπέδων. Το πιο πάνω επίπεδο περιλαμβάνει τη γενική οντολογία ενώ το άλλο επίπεδο περιλαμβάνει τις εξειδικευμένες οντολογίες καθεμιά από τις οποίες περιγράφει ένα συγκεκριμένο πεδίο εφαρμογής. Η γενική οντολογία περιγράφει γενικά concepts όπως άνθρωπος, τοποθεσία, υπολογιστική οντότητα και δραστηριότητα ενώ οι εξειδικευμένες οντολογίες καθορίζουν περαιτέρω λεπτομέρειες των βασικών concepts. Προχωρώντας μπορούμε να αναφερθούμε στο Hydrogen το οποίο χρησιμοποιεί πέντε διαφορετικούς τύπους συμφραζομένων καθένας από τους οποίους αναπαρίσταται με ένα διαφορετικό είδος context αντικειμένων. Καθένα από αυτά τα είδη context αντικειμένων υλοποιείται εξειδικεύοντας το ίδιο βασικό είδος context αντικειμένου. Τα contexts αντικείμενα παρέχουν τη δυνατότητα για αναπαράσταση των συμφραζομένων σε XML μορφή για την αποστολή τους μέσω των context servers σε συσκευές που μπορούν να προσπελαστούν μέσω του δικτύου. Στη Gaia τα συμφραζόμενα καθορίζονται σαν 4-ary κατηγορήματα η δομή των οποίων έχει ως εξής: Context (<ContextType>,<Subject>, <Relater>,<Object>). Το ContextType αναφέρεται στον τύπο του συμφραζόμενου που περιγράφεται. Το subject μπορεί να είναι ένας άνθρωπος ή ένα αντικείμενο ή μια περιοχή ενώ το object είναι μια τιμή που σχετίζεται με το subject. Ο Relater συσχετίζει το subject με το object με τη χρήση συγκριτικών τελεστών, ρημάτων ή προθέσεων. Ένα παράδειγμα είναι Context(temperature,room3231,is,25,^{o}C). Στο Context Toolkit τα συμφραζόμενα μοντελοποιούνται σαν ιδιότητες και callbacks των widgets. Όσο για το CORTEX τα συμφραζόμενα που παράγονται από τους αισθητήρες είναι STEAM γεγονότα. Το STEAM είναι ένα ενδιάμεσο λογισμικό το οποίο χρησιμοποιείται για την επικοινωνία των συστατικών στοιχείων ενός ασύρματου περιβάλλοντος και η λειτουργία του οποίου βασίζεται σε γεγονότα.

2.6. Μέθοδοι Επεξεργασίας Συμφραζομένων

Αφού ληφθούν τα συμφραζόμενα με τους τρόπους που προαναφέραμε πρέπει εν συνεχεία να επεξεργαστούν προκειμένου να χρησιμοποιηθούν από τις εφαρμογές. Για το λόγο υπάρχουν διαδικασίες σε κάθε πλατφόρμα οι οποίες παρέχουν δυο διαφορετικών ειδών επεξεργασία στα συμφραζόμενα που μόλις ελήφθησαν από τους παρόχους. Η σύνθεση (aggregation) είναι η πρώτη μορφή επεξεργασίας που μπορεί να γίνει στα συμφραζόμενα και έχει να κάνει με τη συλλογή πολλών διαφορετικών συμφραζομένων που αφορούν μια συγκεκριμένη οντότητα και με την εξαγωγή υψηλότερου επιπέδου συμφραζομένων. Το άλλο είδος επεξεργασίας που μπορεί να επιτευχθεί είναι η ερμηνεία (interpretation), η οποία τροποποιεί/μεταφράζει (transform) την αρχική πληροφορία των συμφραζομένων σύμφωνα με κάποια ειδική γνώση. Στο Context Toolkit η σύνθεση υλοποιείται από τους context servers οι οποίοι συγκεντρώνουν τα συμφραζόμενα που αφορούν μια οντότητα. Όσο για την ερμηνεία, αυτή επιτυγχάνεται μέσω των Interpreters οι οποίοι τροφοδοτούνται με συμφραζόμενα τα οποία επεξεργάζονται μέσω της μεθόδου τους intrerpretData() παράγοντας έτσι την πληροφορία με διαφορετική μορφή και σημασία. Για παράδειγμα μπορεί ένας interpreter να λάβει πληροφορία για την τοποθεσία, τον ήχο και την ταυτότητα ανθρώπων και να εξάγει πληροφορία ότι μια συνάντηση πραγματοποιείται. Στο SOCAM ο context interpreter αποτελείται από τον context reasoner και την βάση γνώσεων (KB). Δουλειά του context reasoner είναι να παράγει συμφραζόμενα υψηλότερου επιπέδου βασιζόμενος στα πρωτογενή συμφραζόμενα, να εντοπίζει ασυνεπή δεδομένα στη KB και να αντιμετωπίζει συγκρούσεις συμφραζομένων στην KB. Για να επιτευχθούν τα παραπάνω ο reasoner ακολουθεί κάποιους κανόνες. Αυτό μπορεί να γίνει με δύο τρόπους: α) μέσω του ontology reasoning όπου χρησιμοποιείται μια reasoning μηχανή βασισμένη σε κανόνες που δηλώνονται σε γλώσσα RDF ή OWL ή β) μέσω user-defined rule-based reasoning όπου οι κανόνες που χρησιμοποιούνται δηλώνονται από τον ίδιο το χρήστη. Η KB αποτελείται από μια οντολογία συμφραζομένων μαζί με τα στιγμιότυπα της και παρέχει ένα API για να μπορούν τα συστατικά του συστήματος να ρωτούν, να προσθέτουν, να αφαιρούν και να τροποποιούν την οντολογία αυτή ή τα στιγμιότυπα της. Στο CoBrA υπάρχει η context reasoning engine η οποία είναι μια inference engine που κάνει reasoning με βάση την αποθηκευμένη γνώση των συμφραζομένων (context knowledge) που υπάρχει στην οντολογία CORBA-ONT. Όσον αφορά το CASS, η εξαγωγή υψηλότερου επιπέδου συμφραζομένων βασίζεται σε μια inference engine και σε μια βάση γνώσεων. Έτσι λοιπόν για την εξαγωγή υψηλού επιπέδου συμφραζομένων αποθηκεύονται στη βάση δεδομένων του CASS κανόνες με τη μορφή

πινάκων, κάθε εγγραφή του οποίου αποτελείται από καταστάσεις συμφραζομένων και μια αντίστοιχη συμπεριφορά/αντίδραση. Οι κανόνες αυτοί αποτελούν την λεγόμενη Knowledge Base την οποία συμβουλεύεται η Inference Engine του CASS κάθε φορά που γίνεται μια αλλαγή σε συμφραζόμενα. Σε περίπτωση που η inference engine εντοπίζει κανόνες που ικανοποιούνται από την υπάρχουσα κατάσταση, εφαρμόζει τη συμπεριφορά/αντίδραση που προσδιορίζεται σε καθένα από τους κανόνες που ικανοποιούνται. Στο CORTEX τα συμφραζόμενα οργανώνονται σε μια ιεραρχία συμφραζομένων. Η Inference engine του CORTEX με βάση τους κανόνες που σχετίζονται με αυτά είναι υπεύθυνη για την αλλαγή της συμπεριφοράς της εφαρμογής. Στη Gaia η επεξεργασία των συμφραζομένων γίνεται από την context service. Η υπηρεσία αυτή μπορεί να παράγει υψηλότερου επιπέδου συμφραζόμενα με τη βοήθεια κανόνων οι οποίοι επιβάλλουν την εκτέλεση first order logic πράξεων στα predicates των συμφραζομένων. Έτσι λοιπόν με τις λογικές πράξεις σύζευξης, διάζευξης, άρνησης μπορούν να παραχθούν πιο πολύπλοκα συμφραζόμενα όπως φαίνεται στον παρακάτω κανόνα Context (Number of people, Room 2401, >, 4) AND Context(Application, PowerPoint, is, Running) => Context(Social Activity, Room 2401, Is, Presentation).

2.7. Διαφορές CoWSAMI από Προηγούμενες Προσεγγίσεις

Μετά την ολοκλήρωση της παρουσίασης των σημαντικότερων λύσεων που έχουν προτεθεί ως τώρα για την επίλυση του προβλήματος της ενημερότητας συμφραζομένων των εφαρμογών ενός χρήστη αυτό που απομένει είναι ένας γενικός σχολιασμός για το τι καινούργιο προσφέρει το CoWSAMI. Το CoWSAMI λοιπόν έρχεται σαν μια νέα πρόταση στη λίστα των πλατφορμών που είχαν προταθεί για τη λύση του προβλήματος της ανάπτυξης εφαρμογών με ενημερότητα συμφραζομένων. Σε αντίθεση με τις προηγούμενες προσεγγίσεις το CoWSAMI αντιμετωπίζει τόσο το πρόβλημα των περιορισμένων πόρων των φορητών συσκευών όσο και της συνεχής μεταβολής της διαθεσιμότητας τους σε ένα τέτοιο περιβάλλον. Μέσω της χρήσης του WSAMI το οποίο είναι μια πλατφόρμα ενδιάμεσου λογισμικού για ανάπτυξη υπηρεσιών Διαδικτύου σε κινητές συσκευές, παρακάμπτονται όλες οι δυσκολίες που επιφέρουν οι περιορισμένοι πόροι. Μπορεί η ανάπτυξη υπηρεσιών διαδικτύου να απαιτεί πολλούς υπολογιστικούς μιας και στηρίζεται στη χρήση της XML, η οποία από τη φύση της για την επεξεργασία της απαιτεί μεγάλη υπολογιστική ισχύ, εντούτοις με τη χρήση του WSAMI η εγκατάσταση και λειτουργία υπηρεσιών διαδικτύου δεν απαιτεί πολλούς υπολογιστικούς πόρους. Όσο για το δεύτερο, δηλαδή τη συνεχή μεταβολή της

διαθεσιμότητας των συσκευών που υπάρχουν σε ένα τέτοιο περιβάλλον, η λύση δίνεται μέσω του δυναμικού τρόπου εντοπισμού των παρόχων συμφραζομένων. Με βάση το CoWSAMI κάθε συσκευή μπορεί να εντοπίσει οποιαδήποτε άλλη συσκευή ανάλογα με τη διεπαφή της υπηρεσίας Διαδικτύου που αυτή παρέχει. Γενικά στο CoWSAMI δεν χρησιμοποιείται κάποιος κεντρικός εξυπηρετητής όπου αποθηκεύεται η πληροφορία για τον εντοπισμό των παρόχων αλλά μέσω της επικοινωνίας όλων των συσκευών του περιβάλλοντος υπάρχει η δυνατότητα καθεμιά από αυτές να διατηρεί έναν αποθηκευτικό χώρο στον οποίο θα υπάρχουν οι πάροχοι των συμφραζομένων οι οποίοι είναι διαθέσιμοι κάθε στιγμή. Αυτό είναι άλλωστε και το χαρακτηριστικό που κάνει το CoWSAMI αποδοτικό ακόμη και σε τελείως αδόμητα περιβάλλοντα πανταχού παρόντος υπολογισμού. Πέραν αυτού μια ακόμη πιο σημαντική διαφορά του CoWSAMI σε σχέση με άλλες προτάσεις είναι ότι δεν επιβάλλει περιορισμούς στους παρόχους για προσφορά των συμφραζομένων μέσω μιας καθολικά αποδεκτής διεπαφής αλλά τους δίνει τη δυνατότητα να διαθέτουν οποιουδήποτε είδους διεπαφή επιθυμούν για τα συμφραζόμενα αρκεί να ακολουθεί τα πρότυπα των υπηρεσιών Διαδικτύου.

Προκειμένου να παρουσιαστούν συνοπτικά οι στόχοι του CoWSAMI θα παρατεθεί ένα σενάριο χρήσης του προτεινόμενου ενδιάμεσου λογισμικού το οποίο θα ακολουθηθεί και στη συνέχεια της παρούσας εργασίας. Στο συγκεκριμένο παράδειγμα [6] θεωρούμε ένα περιβάλλον πανταχού παρόντος υπολογισμού που υπάρχει στην πόλη Rocquencourt, κοντά στην πόλη των Βερσαλλιών της Γαλλίας. Στους δημότες και τους τουρίστες αυτής της πόλης παρέχονται μη επανδρωμένα οχήματα, τα CyberCars, προκειμένου να μετακινούνται μέσα σε αυτή. Τα οχήματα αυτά έχουν ένα ενσωματωμένο υπολογιστή, στον οποίο έχει εγκατασταθεί το CoWSAMI μέσω του οποίου παρέχονται πληροφορίες σχετικές με τα χαρακτηριστικά του οχήματος όπως ταχύτητα, θέση, μάρκα. Τα συγκεκριμένα οχήματα μπορούν να επικοινωνούν με άλλες οντότητες του περιβάλλοντος μέσω ενός ασύρματου δικτύου. Υπάρχει δε η δυνατότητα να υπάρχουν διαφορετικού τύπου CyberCars τα οποία έχουν κατασκευαστεί από διαφορετικούς κατασκευαστές και επομένως τα συμφραζόμενα που παρέχονται από CyberCars διαμέσου υπηρεσιών Διαδικτύου είναι δυνατόν να προσφέρονται μέσω διαφορετικών διεπαφών ανάλογα με τον εκάστοτε κατασκευαστή. Στα παρακάτω σχήματα παρουσιάζονται οι διαφορετικές διεπαφές των υπηρεσιών που παρέχουν τα συμφραζόμενα για την περίπτωση δύο διαφορετικών CyberCars. Στην περίπτωση των CyberFrogs υπάρχει μια υπηρεσία Διαδικτύου η οποία έχει έξι διαφορετικές μεθόδους προκειμένου να παρέχονται το αναγνωριστικό, η μάρκα, η ταχύτητα, τα αποθέματα καυσίμων και οι συντεταγμένες κάθε

CyberFrog. Αντίθετα, στην δεύτερη περίπτωση τα CyberCabs διαθέτουν μια υπηρεσία με μια μόνο μέθοδο η οποία όταν κληθεί επιστρέφει ένα μήνυμα που περιέχει όλες τις πληροφορίες ενός CyberCab.

Abstract part of WSAMI specification for CyberCab

```
<Abstract name = "CyberCab">
<Interface
hrefSchema="http://localhost:8080/CyberCab.wsdl">
</Abstract>
```

CyberCab Interface specification in WSDL

```
<wsdl:definitions>
---------------------------------------------------------------------------
<wsdl:message name="getStateResponse">
  <wsdl:part name="ID" type="xsd:int"/>
  <wsdl:part name="Brand" type="xsd:string"/>
  <wsdl:part name="Velocity" type="xsd:float"/>
  <wsdl:part name="Fuel" type="xsd:float"/>
  <wsdl:part name="XPos" type="xsd:float"/>
  <wsdl:part name="YPos" type="xsd:float"/>
</wsdl:message>
---------------------------------------------------------------------------
<wsdl:portType name="CyberCab">
<wsdl:operation name="getState">
  <wsdl:output message="impl:getStateResponse"
     name="getStateResponse" />
</wsdl:operation>
</wsdl:portType>
---------------------------------------------------------------------------
</wsdl:definitions>
```

Σχήμα 2.9 WSAMI Περιγραφή του CyberCab[6]

Επιπλέον, διατίθενται από το δήμο του Rocquencourt υπηρεσίες Διαδικτύου οι οποίες παρέχουν πληροφορίες για τα ξενοδοχεία και τα εστιατόρια της πόλης. Όπως και στην περίπτωση των CyberCars έτσι και για τα ξενοδοχεία και τα εστιατόρια υπάρχουν διαφορετικές διεπαφές υπηρεσιών. Επιπροσθέτως, το Δημαρχείο της πόλης παρέχει μια υπηρεσία για την θέση διαφόρων σημαντικών τοποθεσιών όπως μνημεία, νοσοκομεία, οργανισμούς.

Abstract part of WSAMI specification for CyberFrog

```
<Abstract name = "CyberFrog">
<Interface
hrefSchema="http://localhost:8080/CyberFrog.wsdl">
</Abstract>
```

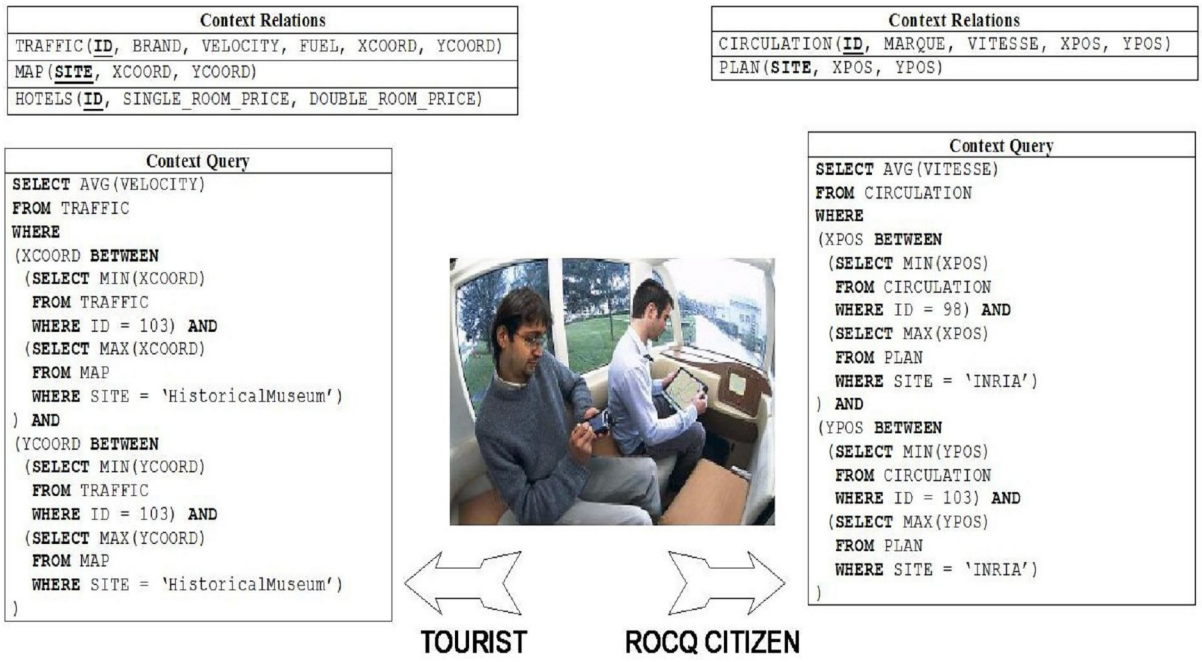
CyberFrog Interface specification in WSDL

```
<wsdl:definitions>
<wsdl:definitions>
<wsdl:message name="getVelocityResponse">
 <wsdl:part name="getVelocityReturn" type="xsd:float" />
</wsdl:message>
-----------------------------------------------------------------------------
<wsdl:portType name="CyberFrog">
<wsdl:operation name="getID">
  <wsdl:output message="impl:getIDResponse"
                          name="getIDResponse" />
</wsdl:operation>
<wsdl:operation name="getVelocity">
  <wsdl:output message="impl:getVelocityResponse"
                          name="getVelocityResponse" />
</wsdl:operation>
</wsdl:portType>
-----------------------------------------------------------------------------
</wsdl:definitions>
```

Σχήμα 2.10 WSAMI Περιγραφή του CyberFrog[6]

Στόχος μας είναι, ο επιβάτης ενός CyberCar να έχει τη δυνατότητα μέσω μιας φορητής συσκευής, στην οποία έχει εγκαταστήσει το CoWSAMI, να καθορίζει τα συμφραζόμενα για τα οποία ενδιαφέρεται. Κάθε επιβάτης φυσικά μπορεί να δηλώσει διαφορετικά συμφραζόμενα που τον ενδιαφέρουν σε σύγκριση με κάποιον άλλο. Για παράδειγμα ένας Άγγλος τουρίστας μπορεί να ενδιαφέρεται για την κίνηση στο κέντρο της πόλης και τα διαθέσιμα ξενοδοχεία σε αυτή ενώ ένας κάτοικος της πόλης μπορεί να ενδιαφέρεται μόνο για την κυκλοφοριακή κίνηση. Στην πρώτη περίπτωση ο Άγγλος τουρίστας θα δηλώσει τις σχέσεις συμφραζομένων που φαίνονται στο σχήμα 2.11. Η σχέση TRAFFIC αποτελείται από έξι ιδιότητες με ονόματα ID, BRAND, VELOCITY, FUEL, XCOORD, YCOORD οι οποίες αντιστοιχούν στο αναγνωριστικό, τη μάρκα, την ταχύτητα, τα αποθέματα καυσίμων και τις συντεταγμένες κάθε CyberCar που κυκλοφορεί στο περιβάλλον. Η δεύτερη σχέση με όνομα MAP αποτελείται από τρεις ιδιότητες και αναφέρεται στο όνομα και τις συντεταγμένες θέσης μιας τοποθεσίας. Όσο για την τρίτη σχέση με όνομα HOTELS η οποία διαθέτει επίσης τρεις

ιδιότητες προσδιορίζει το όνομα ενός ξενοδοχείου με τις τιμές του μονόκλινου και δίκλινου δωματίου αντίστοιχα. Ο Γάλλος επιβάτης με τη σειρά του μπορεί να δηλώσει τις σχέσεις συμφραζομένων CIRCULATION και PLAN. Η πρώτη έχει πέντε ιδιότητες με ονόματα:ID, MARQUE, VITESSE, XPOS, YPOS. Η δεύτερη έχει τρεις ιδιότητες με ονόματα: SITE, XPOS, YPOS και είναι παρόμοια με την σχέση MAP του Άγγλου τουρίστα. Με βάση τα παραπάνω, στόχος μας είναι να μπορούν οι χρήστες να θέτουν SQL ερωτήσεις στο CoWSAMI το οποίο με τη σειρά του 1) εντοπίζει παρόχους που μπορούν να συνεισφέρουν με συμφραζόμενα στον εμπλουτισμό των σχέσεων συμφραζομένων που ορίζει ο χρήστης, 2) συλλέγει τα συμφραζόμενα από αυτούς, ***ανεξάρτητα του είδους διεπαφής μέσω της οποίας προσφέρονται,*** 3) κατασκευάζει απάντηση στα SQL ερωτήματα με βάση τα συμφραζόμενα. Στο σχήμα 2.11 φαίνονται πιθανές ερωτήσεις τόσο του τουρίστα όσο και του κατοίκου της πόλης οι οποίοι επιβαίνουν στα CyberCars. Οι απαντήσεις που δίνονται προκύπτουν μετά από έλεγχο της θέσης των CyberCars που μπορούν να προσπελαστούν. Ο εντοπισμός και κατηγοριοποίηση των CyberCars σε CyberFrogs και CyberCabs γίνεται ανάλογα με τις διεπαφές των υπηρεσιών ενώ η συγκέντρωση των θέσεων των οχημάτων γίνεται ακολουθώντας τους κανόνες που υπάρχουν για κάθε κατηγορία CyberCar.



Σχήμα 2.11 Παραδείγματα SQL Ερωτημάτων στο CoWSAMI[6]

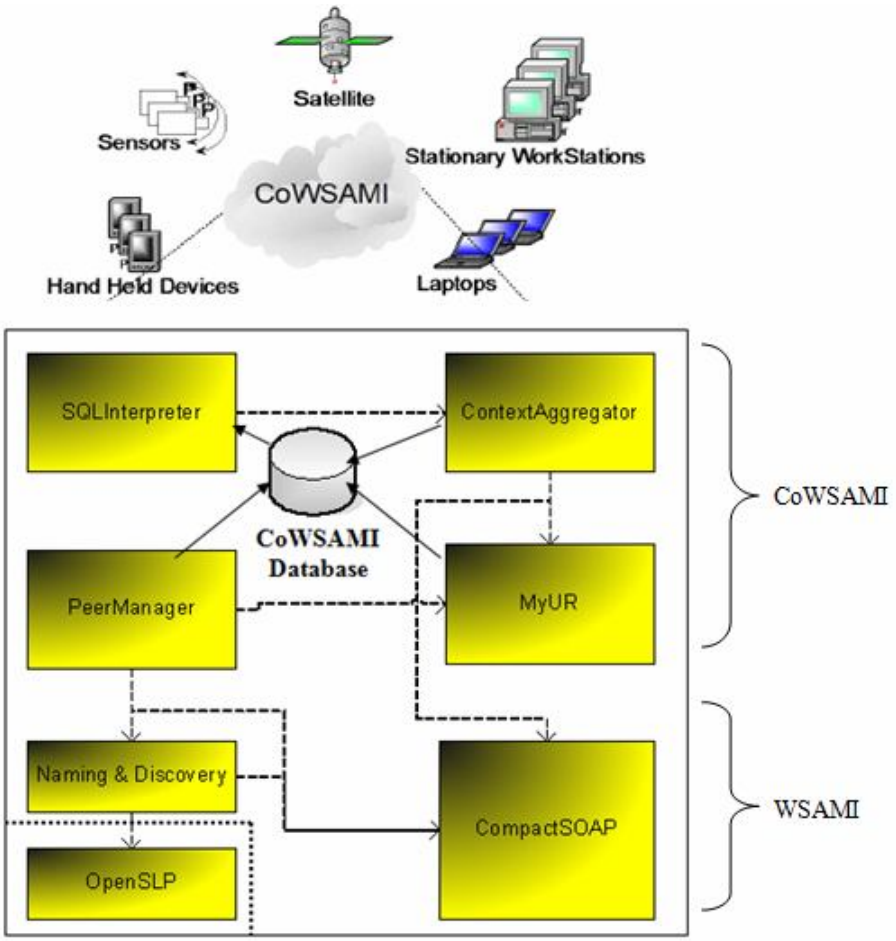
ΚΕΦΑΛΑΙΟ 3. COWSAMI

3.1. Αρχιτεκτονική CoWSAMI

Έχοντας περιγράψει μερικές από τις λύσεις που προτάθηκαν με σκοπό τη διευκόλυνση της ανάπτυξης και λειτουργίας εφαρμογών με ενημερότητα συμφραζομένων θα προχωρήσουμε στην περιγραφή του CoWSAMI. Στο κεφάλαιο αυτό θα περιγραφεί η αρχιτεκτονική του ενδιάμεσου λογισμικού και θα αναλυθεί ο τρόπος υλοποίησης όλων των υπηρεσιών του.

Όπως φαίνεται από το σχήμα 3.1 το CoWSAMI διαθέτει τέσσερις υπηρεσίες: τον PeerManager, τον ContextAggregator, τον SQLInterpreter και τη MyUR. Διαθέτει επίσης μία βάση δεδομένων στην οποία αποθηκεύονται τα συμφραζόμενα τα οποία συλλέγονται και άλλες πληροφορίες που αφορούν τους παρόχους συμφραζομένων. Ειδικότερα, οι αρμοδιότητες των υπηρεσιών είναι οι εξής: Ο PeerManager είναι υπεύθυνος για τον εντοπισμό των παρόχων συμφραζομένων. Ο Context Aggregator συλλέγει τα συμφραζόμενα και τα αποθηκεύει στη βάση δεδομένων. Ο SQLInterpreter χρησιμοποιείται για την επεξεργασία των συμφραζομένων και πιο συγκεκριμένα για την απάντηση SQL ερωτημάτων που γίνονται από το χρήστη πάνω στις σχέσεις της βάσης δεδομένων που περιέχει τα συμφραζόμενα. Τέλος η MyUR είναι μια υπηρεσία που χρησιμοποιείται για τη λήψη πληροφοριών/λεπτομερειών που αφορούν χαρακτηριστικά των υπηρεσιών των παρόχων συμφραζομένων όπως για παράδειγμα η διαθεσιμότητα τους και ο χρόνος απόκρισης τους.

Για την υλοποίηση όλων αυτών των υπηρεσιών είναι απαραίτητη η εγκατάσταση και λειτουργία του WSAMI. Το WSAMI είναι μια πλατφόρμα ενδιάμεσου λογισμικού η οποία επιτρέπει την εγκατάσταση και λειτουργία υπηρεσιών Διαδικτύου σε φορητές και σταθερές συσκευές. Λόγω του σχεδιασμού του επιτρέπει την εγκατάσταση και τη λειτουργία των υπηρεσιών αποφεύγοντας τη σπατάλη πολλών υπολογιστικών πόρων. Η συγκεκριμένη πλατφόρμα, μέσω του CSOAP πυρήνα της και της υπηρεσίας Naming & Discovery, η οποία παρέχει δυναμικό εντοπισμό υπηρεσιών Διαδικτύου, προσφέρει το κατάλληλο υπόβαθρο για την ανάπτυξη και λειτουργία των υπηρεσιών του CoWSAMI.



Σχήμα 3.1 Αρχιτεκτονική του CoWSAMI

Πριν περάσουμε στην αναλυτική παρουσίαση των δομικών στοιχείων του CoWSAMI καλό είναι να γίνει μια περιγραφή του περιβάλλοντος στο οποίο αυτό εγκαθίσταται και λειτουργεί. Θεωρούμε ότι το περιβάλλον αποτελείται από *οντότητες* καθεμιά από τις οποίες διαθέτει ένα

στιγμιότυπο του COWSAMI. Οι οντότητες αυτές μπορούν να συνδέονται μεταξύ τους μέσω ενός τοπικού δικτύου είτε ενσύρματου είτε ασύρματου. Το ρόλο αυτών των οντοτήτων μπορούν να τον παίξουν είτε σταθεροί υπολογιστές, είτε φορητοί υπολογιστές ακόμη και PDA's ή αισθητήρες. Σε ένα τέτοιο περιβάλλον κάθε οντότητα μπορεί να έχει το ρόλο ενός *παρόχου* συμφραζομένων ή ενός *καταναλωτή* συμφραζομένων ή ακόμη και των δύο μαζί. Οι πάροχοι συμφραζομένων παρέχουν τις πληροφορίες τους μέσω *μιας ή περισσοτέρων υπηρεσιών Διαδικτύου*. Αυτές οι υπηρεσίες καθορίζονται/περιγράφονται με τη χρήση της WSAMI γλώσσας, η οποία αποτελεί μια επέκταση της WSDL με σκοπό την πιο αναλυτική περιγραφή της συμπεριφοράς υπηρεσιών Διαδικτύου. Η περιγραφή κάθε υπηρεσίας Διαδικτύου αποτελείται από δύο κομμάτια: το *abstract* κομμάτι και το *concrete* κομμάτι. Το abstract κομμάτι της περιγραφής καθορίζει την διεπαφή της υπηρεσίας ενώ το concrete κομμάτι περιέχει πληροφορίες σχετικές με τον τρόπο κλήσης της υπηρεσίας, όπως για παράδειγμα η διεύθυνση που βρίσκεται εγκατεστημένη η υπηρεσία, το πρωτόκολλο επικοινωνίας που χρησιμοποιείται (βλέπε σχήμα 2.10). Η διεπαφή της υπηρεσίας προσδιορίζεται από μια WSDL περιγραφή της οποίας το URI αναφέρεται στο abstract κομμάτι της περιγραφής της υπηρεσίας. Οι διεπαφές δύο υπηρεσιών είναι ίδιες όταν είναι συντακτικά όμοια τα abstract κομμάτια των περιγραφών τους.

Σε ένα τέτοιο περιβάλλον κάθε χρήστης μπορεί να καθορίσει τα συμφραζόμενα για τα οποία ενδιαφέρεται. Στο CoWSAMI τα συμφραζόμενα ορίζονται σαν ένα σύνολο από ιδιότητες οι οποίες λέγονται *ιδιότητες συμφραζομένων*. Μάλιστα κάθε πάροχος συμφραζομένων μπορεί να προσφέρει περισσότερες από μια τιμές που αφορούν διαφορετικές ιδιότητες συμφραζομένων (π.χ. ένα CyberCar αναφέρει την ταχύτητα και τη μάρκα του) ενώ μπορεί να υπάρχουν και διαφορετικοί πάροχοι συμφραζομένων που προσφέρουν σημασιολογικά ίδια πληροφορία (π.χ. δυο CyberCars αναφέρουν την ταχύτητα τους μέσω διαφορετικών διεπαφών υπηρεσιών Διαδικτύου). Οι ιδιότητες συμφραζομένων οργανώνονται από το χρήστη σε *σχέσεις συμφραζομένων*. Η κάθε σχέση συμφραζομένων χαρακτηρίζεται από το όνομα της και το σύνολο των ιδιοτήτων της. Με βάση αυτή την οργάνωση λοιπόν τα συμφραζόμενα αποθηκεύονται στη βάση δεδομένων του CoWSAMI, που είναι εγκατεστημένη στη συσκευή του παρόχου/καταναλωτή συμφραζομένων, σε πλειάδες. Η συλλογή τιμών για τις ιδιότητες κάθε σχέσης συμφραζομένων γίνεται με βάση κανόνες που έχει ορίσει ο χρήστης και οι οποίοι είναι αποθηκευμένοι και αυτοί στη βάση δεδομένων. Οι κανόνες αυτοί αντιστοιχίζουν ιδιότητες των σχέσεων συμφραζομένων σε λειτουργίες των υπηρεσιών Διαδικτύου που

προσφέρει ένας πάροχος συμφραζομένων. Η κλήση των λειτουργιών έχει ως αποτέλεσμα την επιστροφή τιμών για τις αντίστοιχες ιδιότητες. Σημειωτέων δε ότι για κάθε σχέση συμφραζομένων μπορούν να υπάρχουν πολλοί διαφορετικοί κανόνες, καθένας από τους οποίους προσδιορίζει τη συμπλήρωση των ιδιοτήτων μιας σχέσης συμφραζομένων με τιμές που παρέχονται από παρόχους με διαφορετικές διεπαφές. Τέτοιοι πάροχοι είναι στο παράδειγμα μας (παρ. 2.7) τα CyberCabs και τα CyberFrogs. Τα CyberCabs παρέχουν τιμές μέσω μιας μόνο μεθόδου (getState()) ενώ τα CyberFrogs έχουν έξι διαφορετικές μεθόδους που επιστρέφουν τιμές. Αν λοιπόν ένας καταναλωτής κατασκευάσει μια σχέση συμφραζομένων η οποία έχει μια μόνο ιδιότητα και η οποία συμπληρώνεται με τις ταχύτητες των CyberCars του περιβάλλοντος τότε μπορούν να δηλωθούν δύο διαφορετικοί κανόνες για τη συλλογή τιμών για την ιδιότητα αυτής της σχέσης συμφραζομένων. Άλλος κανόνας θα δηλωθεί για τη συλλογή τιμών από τα CyberCabs αφού θα χρησιμοποιείται η μέθοδος getState() και άλλος κανόνας για τη συλλογή τιμών από τα CyberFrogs αφού στην περίπτωση αυτή θα χρησιμοποιείται η μέθοδος getVelocity().

Για να επιτευχθούν τα παραπάνω υπάρχουν στη συσκευή κάθε χρήστη οι υπηρεσίες PeerManager, ContextAggregator, SQLInterpreter και MyUR του CoWSAMI. Ο PeerManager είναι υπεύθυνος για το εντοπισμό όλων των διαθέσιμων παρόχων συμφραζομένων του περιβάλλοντος για τους οποίους ενδιαφέρεται ο κάθε χρήστης. Ο εντοπισμός αυτός βασίζεται στην Naming & Discovery υπηρεσία που διαθέτει το WSAMI, η οποία κατηγοριοποιεί τους παρόχους συμφραζομένων ανάλογα με την διεπαφή της υπηρεσίας που προσφέρουν όπως αυτή περιγράφεται στο abstract κομμάτι της WSAMI περιγραφή τους. Ο PeerManager αποτελείται από έξι διαφορετικές λειτουργίες. Οι λειτουργίες αυτές δίνουν τη δυνατότητα σε ένα πάροχο/καταναλωτή να φέρει εις πέρας τα εξής:

- Να εισάγει στο περιβάλλον μια νέα υπηρεσία που προσφέρει, δηλαδή να κάνει γνωστή την παρουσία της στους άλλους χρήστες
- Να εξάγει από το περιβάλλον μια υπηρεσία που μέχρι τότε παρείχε, δηλαδή να ενημερώσει τους άλλους χρήστες ότι παύει πλέον να παρέχει τη συγκεκριμένη υπηρεσία
- Να ανακαλύψει όλες τις διαφορετικές διεπαφές υπηρεσιών διαδικτύου που παρέχονται από τις οντότητες του συστήματος και οι οποίες είναι εκείνη τη στιγμή προσβάσιμες από αυτόν. Κάθε διεπαφή ορίζει μια διαφορετική κατηγορία υπηρεσιών.

- Να συγκεντρώνει και να αποθηκεύει τις υπηρεσίες των οντοτήτων που παρέχουν μια συγκεκριμένη διεπαφή, δηλαδή τις υπηρεσίες που ανήκουν σε μια συγκεκριμένη κατηγορία. Για τη συγκεκριμένη λειτουργία υπάρχουν τρεις διαφορετικές πολιτικές που μπορούν να ακολουθηθούν από το χρήστη και αφορούν το κάθε πότε γίνεται η συλλογή αυτών των διευθύνσεων. Σύμφωνα με την πρώτη πολιτική η ενημέρωση γίνεται μετά από αίτημα του χρήστη, η δεύτερη πολιτική προτείνει την περιοδική ενημέρωση ανά τακτά χρονικά διαστήματα ενώ η τρίτη πολιτική υλοποιεί την συνεχή ενημέρωση και διατηρεί την πληροφορία κάθε χρονική στιγμή ενημερωμένη.

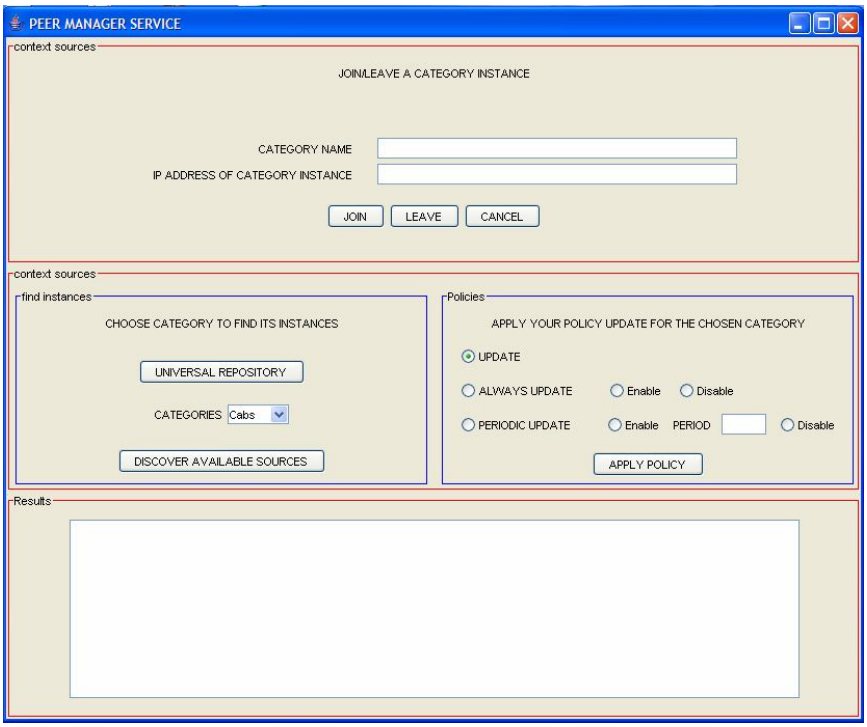
Ο ContextAggregator με τη σειρά του δίνει τη δυνατότητα στους χρήστες να ορίσουν τις σχέσεις συμφραζομένων που τους ενδιαφέρουν αλλά και τους κανόνες σύμφωνα με τους οποίους θα συλλέγονται τιμές για τις ιδιότητες των σχέσεων αυτών. Είναι επίσης υπεύθυνος για τη συλλογή των πληροφοριών που απαιτούνται για τη συμπλήρωση των ιδιοτήτων των σχέσεων συμφραζομένων. Ο ContextAggregator για να πετύχει όλα αυτά παρέχει τρεις λειτουργίες. Η πρώτη δίνει τη δυνατότητα στον καταναλωτή συμφραζομένων να καθορίσει τη σχέση των συμφραζομένων που επιθυμεί με τις ιδιότητες συμφραζομένων που τον ενδιαφέρουν. Η δεύτερη λειτουργία χρησιμοποιείται από τον καταναλωτή συμφραζομένων για τον καθορισμό των κανόνων σύμφωνα με τους οποίους συμπληρώνονται οι σχέσεις συμφραζομένων που έχει ήδη ορίσει. Φυσικά ο καταναλωτής συμφραζομένων έχει την ευχέρεια να δηλώσει πολλούς κανόνες για κάθε σχέση συμφραζομένων και αυτό μπορεί να το κάνει για όσες σχέσεις συμφραζομένων αυτός επιθυμεί. Τέλος η τρίτη λειτουργία του ContextAggregator κατά την εκτέλεση της συμπληρώνει τις ιδιότητες της εκάστοτε σχέσης συμφραζομένων με τιμές που συγκεντρώνει από τις διάφορες υπηρεσίες Διαδικτύου που καλεί σύμφωνα με τους κανόνες που ο καταναλωτής συμφραζομένων έχει δηλώσει εκ των προτέρων.

Ο SQLInterpreter επιτρέπει στον χρήστη να επεξεργαστεί, μέσω υποβολής SQL ερωτημάτων, τα συμφραζόμενα που έχουν συλλεχθεί και τα οποία βρίσκονται αποθηκευμένα στη βάση δεδομένων με τη μορφή πλειάδων τιμών που αντιπροσωπεύουν τιμές για τις ιδιότητες των σχέσεων συμφραζομένων. Πέρα από απλά ερωτήματα σε γλώσσα SQL, ο χρήστης μπορεί να θέσει ερωτήματα με ορισμένους περιορισμούς οι οποίοι έχουν να κάνουν με το πλήθος των παρόχων που θα ερωτηθούν για να γίνει η συλλογή των συμφραζομένων ή ακόμη και με κάποιες προϋποθέσεις που πρέπει να πληρούν οι κόμβοι για να συνεισφέρουν στην

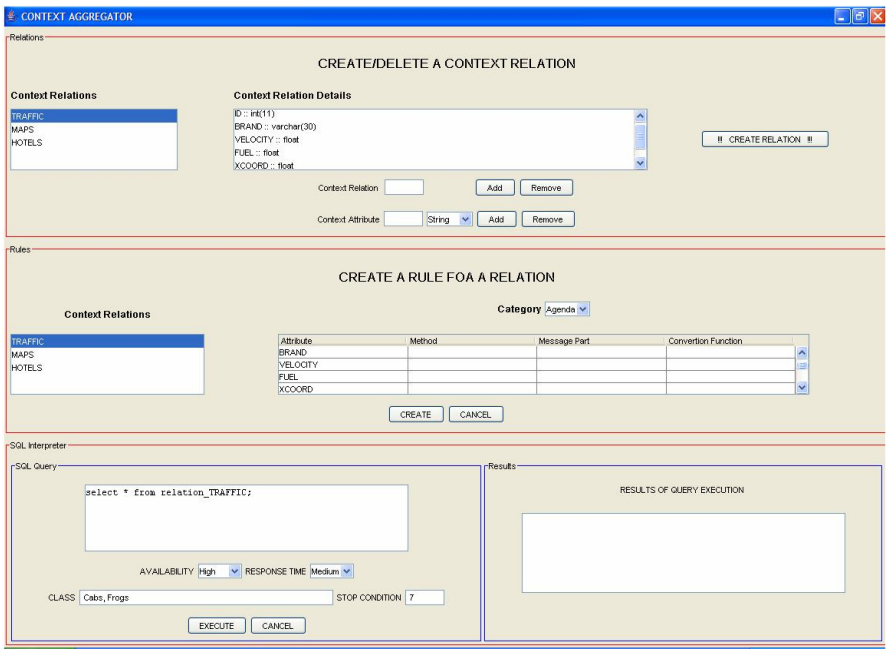
συμπλήρωση μια σχέσης και οι οποίες έχουν σχέση με τη διαθεσιμότητα και το χρόνο απόκρισης μιας υπηρεσίας.

Τέλος η MyUR δίνει στο χρήστη τη δυνατότητα να λάβει πληροφορίες για τις διαφορετικές κατηγορίες υπηρεσιών που διατίθενται στο περιβάλλον του. Η λήψη των πληροφοριών γίνεται από ένα κεντρικό αποθηκευτικό χώρο που βρίσκεται εγκατεστημένος στο περιβάλλον του CoWSAMI. Η διαχείριση του αποθηκευτικού αυτού χώρου είναι αρμοδιότητα του διαχειριστή του CoWSAMI που είναι υπεύθυνος να διατηρεί ενημερωμένες τις πληροφορίες που αφορούν τις κατηγορίες των υπηρεσιών που διατίθενται στο περιβάλλον. Στην κεντρική αυτή αποθήκη υπάρχουν καταχωρημένες οι κατηγορίες των υπηρεσιών που είναι διαθέσιμες στο περιβάλλον. Οι πληροφορίες του MyUR αντλούνται από τις συσκευές των χρηστών του περιβάλλοντος προκειμένου να βοηθηθούν οι χρήστες κατά τη δήλωση των κανόνων τους. Η συγκεκριμένη αποθήκη δεν περιέχει πληροφορίες για τις διευθύνσεις που παρέχονται διάφορες υπηρεσίες αλλά το μόνο που παρέχει είναι η ενημέρωση για την ύπαρξη κάποιων κατηγοριών υπηρεσιών.

Πριν όμως προχωρήσουμε στην ανάλυση της αρχιτεκτονικής του CoWSAMI πρέπει να γίνει αναφορά σε ένα άλλο σημαντικό χαρακτηριστικό του, το οποίο υλοποιήθηκε για την όσο το δυνατόν πιο εύκολη χρήση του. Έχει δημιουργηθεί μια γραφική διεπαφή χρήστη που δίνει στον χρήστη ένα περιβάλλον πολύ πιο φιλικό και προσιτό ούτως ώστε να πετύχει τους στόχους του. Πιο συγκεκριμένα έχουν δημιουργηθεί δύο γραφικά περιβάλλοντα εργασίας, από τα οποία το ένα παρέχει τις λειτουργίες του PeerManager ενώ στο δεύτερο έχουν ενσωματωθεί οι λειτουργίες του ContextAggregator και του SQLInterpreter. Η δημιουργία του έγινε με στόχο την διευκόλυνση του καταναλωτή συμφραζομένων στον ορισμό των σχέσεων συμφραζομένων αλλά και των κανόνων που καθορίζουν τον τρόπο συμπλήρωσης τους αλλά και τον πιο γρήγορο και εύκολο τρόπο εκτέλεσης όλων του λειτουργιών του PeerManager.



Σχήμα 3.2 Γραφικό Περιβάλλον της Υπηρεσίας Peer Manager



Σχήμα 3.3 Γραφικό Περιβάλλον της Υπηρεσίας του Context Aggregator

3.2. Δομή του WSAMI

Το WSAMI όπως έχει ήδη αναφερθεί είναι μια πλατφόρμα κατανεμημένου ενδιάμεσου λογισμικού η οποία αποτελείται από δύο βασικά συστατικά: τον πυρήνα (CSOAP Broker) και την υπηρεσία Naming & Discovery.

Ο πυρήνας του ενδιάμεσου λογισμικού στηρίζεται σε μια υλοποίηση του SOAP πρωτοκόλλου το οποίο επιτρέπει την εγκατάσταση και κλήση υπηρεσιών Διαδικτύου σε φορητές συσκευές, οι οποίες διαθέτουν περιορισμένους υπολογιστικούς και επικοινωνιακούς πόρους. Μαζί με το CSOAP πυρήνα παρέχονται το InstallWSAMI εργαλείο, το οποίο επιτρέπει την εγκατάσταση και ρύθμιση υπηρεσιών και το WSDL2WSAMI εργαλείο, το οποίο παράγει την WSAMI περιγραφή μιας υπηρεσίας δοθείσας της WSDL περιγραφή της.

Η υπηρεσία Naming & Discovery (ND) σκοπό έχει τον εντοπισμό των παρόχων συμφραζομένων που υπάρχουν στο περιβάλλον ενός καταναλωτή συμφραζομένων. Σε κάθε καταναλωτή/πάροχο που χρησιμοποιεί το CoWSAMI υπάρχει εγκατεστημένη αυτή η υπηρεσία. Η υπηρεσία ND διατηρεί στη συσκευή κάθε καταναλωτή/παρόχου συμφραζομένων δύο αποθηκευτικούς χώρους στους οποίους αποθηκεύονται πληροφορίες για τις υπηρεσίες Διαδικτύου που είναι διαθέσιμες στο περιβάλλον. Ο ένας αποθηκευτικός χώρος ο οποίος είναι η τοπική αποθήκη διατηρεί πληροφορίες για τις υπηρεσίες που έχει εγκαταστήσει ένας πάροχος συμφραζομένων στην συσκευή του και τις οποίες παρέχει στους υπόλοιπους. Η απομακρυσμένη αποθήκη η οποία αποτελεί τον δεύτερο αποθηκευτικό χώρο που διατηρεί η ND, λειτουργεί με τη λογική μιας τοπικής κρυφής μνήμης και περιέχει τις πληροφορίες για όλες τις απομακρυσμένες υπηρεσίες Διαδικτύου που έχουν εντοπιστεί στο παρελθόν. Σε κάθε αποθήκη η πληροφορία που αποθηκεύεται για κάθε υπηρεσία είναι το URI της WSAMI περιγραφή της. Μέσω των δύο αποθηκών είναι δυνατή η διεκπεραίωση αιτημάτων για εντοπισμό υπηρεσιών δοθέντος ενός URI που αντιστοιχεί στην περιγραφή της διεπαφής που πρέπει να προσφέρουν. Σε κάθε περίπτωση ο εντοπισμός μιας υπηρεσίας βασίζεται στο αν η δοθείσα διεπαφή είναι όμοια με τη διεπαφή υπηρεσιών που εντοπίζονται. Επομένως ο αλγόριθμος εντοπισμού των υπηρεσιών έχει ως εξής:

- Δοθέντος του URI της προς αναζήτηση διεπαφής στην ND, γίνεται έλεγχος τόσο στην τοπική όσο και στην απομακρυσμένη αποθήκη της συσκευής όπου είναι εγκατεστημένη η ND για το αν υπάρχουν αποθηκευμένα URI’s περιγραφών υπηρεσιών που προσφέρουν τη διεπαφή που καθορίζεται από το δοθέν URI.

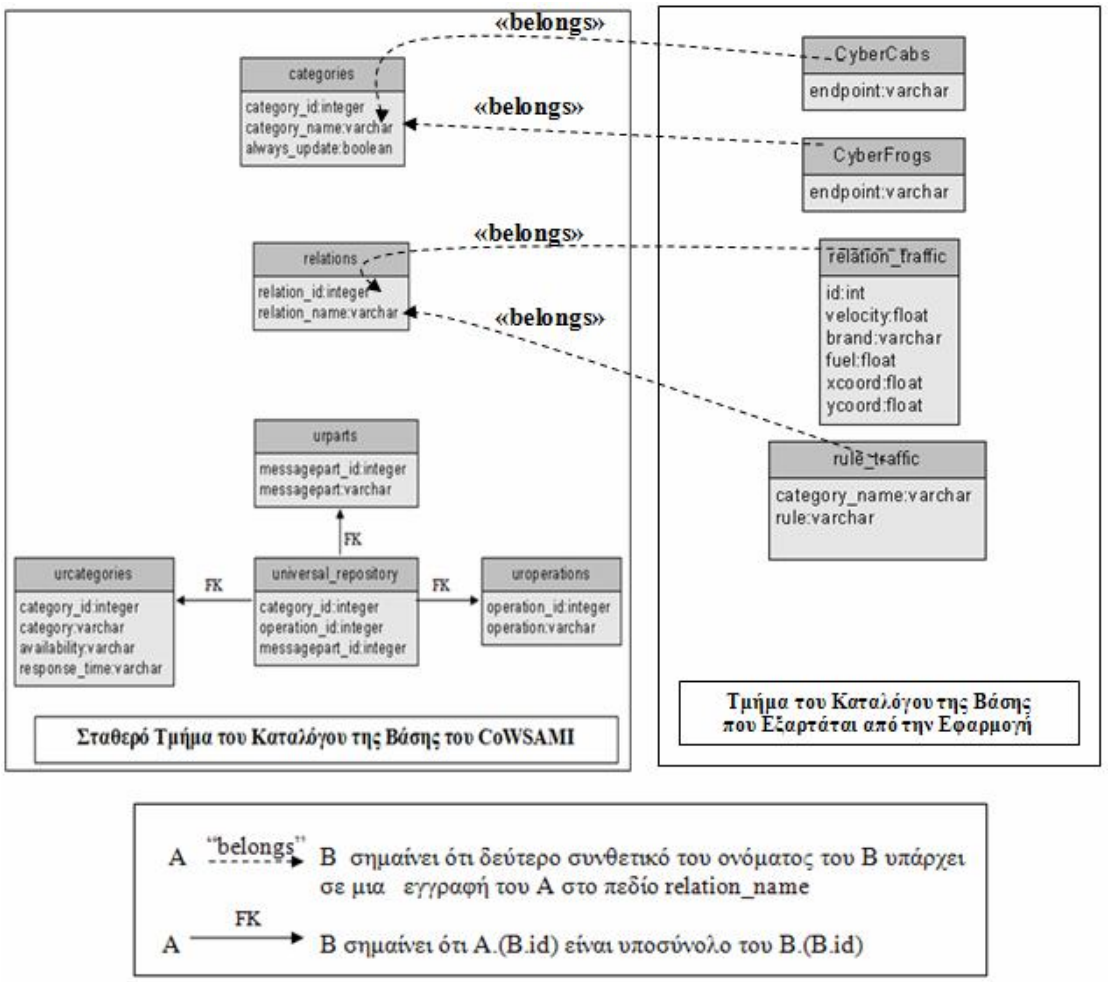
- Αν στις δύο αποθήκες δεν βρεθεί κανένα αποτέλεσμα το ερώτημα διαβιβάζεται σε άλλες ND υπηρεσίες που είναι προσβάσιμες εκείνη τη στιγμή. Η κάθε ND εκτελεί την ίδια λειτουργία που έγινε από την ND υπηρεσία στην οποία ξεκίνησε η αναζήτηση. Τα αποτελέσματα της αναζήτησης επιστρέφονται στον χρήστη που έκανε την ερώτηση αρχικά.

Ο εντοπισμός των απομακρυσμένων ND υπηρεσιών, που απαιτείται στο δεύτερο βήμα του αλγορίθμου λειτουργίας της ND υπηρεσίας, στηρίζεται στο πρωτόκολλο SLP [17]. Με τη χρήση του πρωτοκόλλου αυτού οι συσκευές στις οποίες εκτελείται η ND υπηρεσία μπορούν να εντοπιστούν. Προκειμένου να λειτουργήσει το SLP πρωτόκολλο υπάρχει εγκατεστημένος σε κάθε συσκευή ένας SLP Server. Κάθε SLP Server κάνει περιοδικά multicast σε μια συγκεκριμένη διεύθυνση (224.0.1.22) μια αίτηση αναζήτησης συσκευών που παρέχουν την ND υπηρεσία. Οι SLP Servers λαμβάνουν αυτή την αίτηση και αν διαθέτουν την υπηρεσία ND στέλνουν ένα απευθείας μήνυμα στον server που έκανε την ερώτηση. Με τον τρόπο αυτό οποιαδήποτε συσκευή λειτουργεί στο περιβάλλον που περιγράφουμε και διαθέτει εγκατεστημένο ένα SLP Server μπορεί να εντοπίσει τη διεύθυνση στην οποία διατίθεται η κάθε ND υπηρεσία.

3.3. Βάση Δεδομένων του CoWSAMI

Σημαντικό στοιχείο της υλοποίησης του CoWSAMI αποτελεί η βάση δεδομένων που υπάρχει εγκατεστημένη στη συσκευή κάθε καταναλωτή/παρόχου που χρησιμοποιεί το CoWSAMI. Σε αυτή τη βάση δεδομένων υπάρχουν σχέσεις οι οποίες χρησιμοποιούνται από όλες τις υπηρεσίες του CoWSAMI. Επίσης κάποιες λειτουργίες των υπηρεσιών οδηγούν στην δυναμική δημιουργία νέων σχέσεων. Στο σχήμα 3.4 φαίνεται το σχήμα της βάσης δεδομένων που υπάρχει στο δίσκο μιας συσκευής όταν εγκαθίσταται για πρώτη φορά σε αυτή το CoWSAMI. Στη σχέση categories αποθηκεύονται οι κατηγορίες υπηρεσιών που έχουν εντοπιστεί έστω και μια φορά ενώ στη σχέση relations αποθηκεύονται οι σχέσεις συμφραζομένων που δηλώνει ο καταναλωτής συμφραζομένων. Οι σχέσεις universal_repository, urcategories, uroperations, urparts χρησιμοποιούνται για την αποθήκευση των πληροφοριών που λαμβάνονται από την κεντρική αποθήκη του δικτύου μέσω της υπηρεσίας MyUR. Οι υπόλοιπες τέσσερις σχέσεις CyberFrogs, CyberCabs, relation_traffic και rule_traffic είναι παραδείγματα σχέσεων που δημιουργούνται κατά τη λειτουργία του CoWSAMI στο σενάριο που περιγράφηκε στην παράγραφο 2.7.

Οι σχέσεις CyberFrogs και CyberCabs δημιουργούνται όταν εισαχθεί για πρώτη φορά στο δίκτυο ένα CyberFrog ή ένα CyberCab αντίστοιχα. Σε καθεμιά από αυτές τις σχέσεις αποθηκεύονται οι διευθύνσεις στις οποίες είναι διαθέσιμα τα CyberFrogs ή τα CyberCars. Όσο για τις σχέσεις relation_traffic και rule_traffic δημιουργούνται όταν ο Άγγλος τουρίστας του παραδείγματος ορίσει την σχέση συμφραζομένων traffic. Η πρώτη σχέση έχει σαν πεδία τις ιδιότητες της σχέσης συμφραζομένων, τα οποία γεμίζουν με τιμές από τον ContextAggregator όταν αυτό ζητηθεί από τον χρήστη. Η δεύτερη σχέση χρησιμοποιείται για την καταχώρηση των κανόνων που αφορούν την συμπλήρωση της αντίστοιχης σχέσης συμφραζομένων. Τα ονόματα των δύο αυτών σχέσεων της βάσης δεδομένων είναι σύνθετα και αποτελούνται από τα προθέματα relation_ και rule_ αντίστοιχα ακολουθούμενα από το όνομα της σχέσης συμφραζομένων στην οποία αναφέρονται.

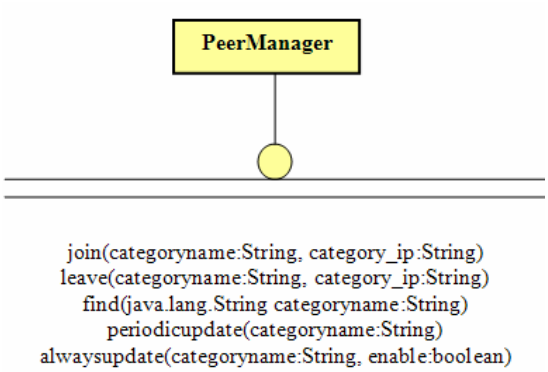


Σχήμα 3.4 Βάση Δεδομένων του CoWSAMI

Στις επόμενες παραγράφους θα γίνει αναλυτική περιγραφή των υπηρεσιών του CoWSAMI οπότε και θα αναλυθούν όλα τα χαρακτηριστικά των σχέσεων της βάσης δεδομένων που μόλις περιγράψαμε.

3.4. Υπηρεσία PeerManager

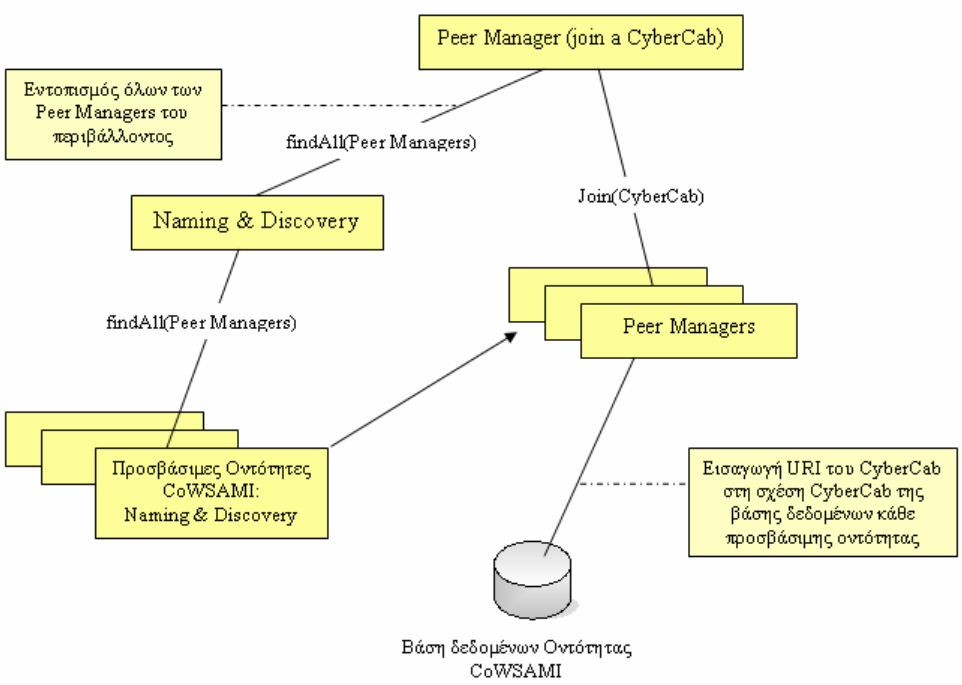
Στην παράγραφο αυτή θα γίνει μια εκτενής αναφορά στην υλοποίηση του Peer Manager και στον τρόπο με τον οποίο οι λειτουργίες του φέρουν εις πέρας την αποστολή τους. Πριν από αυτό όμως αξίζει να γίνει μια αναφορά στις βασικές αρχές λειτουργίας της υπηρεσίας αυτής. Ο μηχανισμός εντοπισμού παρόχων συμφραζομένων που υλοποιεί η υπηρεσία του PeerManager διαχειρίζεται μία σχέση στη βάση δεδομένων, με όνομα categories, στην οποία αποθηκεύονται όλες οι κατηγορίες υπηρεσιών που υπάρχουν στο περιβάλλον. Η σχέση αυτή περιλαμβάνει τρία πεδία από τα οποία το πρώτο είναι ένα μοναδικό αναγνωριστικό για κάθε κατηγορία, το δεύτερο είναι το όνομα της κατηγορίας που συμπίπτει με το όνομα της αντίστοιχης διεπαφής και το τρίτο προσδιορίζει την χρήση ή όχι της πολιτικής της συνεχούς ενημέρωσης. Κάθε φορά που εντοπίζεται μια υπηρεσία που δεν ανήκει σε μια ήδη καταγεγραμμένη στη σχέση categories κατηγορία, η υπηρεσία του PeerManager εισάγει μια καινούργια πλειάδα στη συγκεκριμένη σχέση. Πέρα όμως από τη συγκεκριμένη σχέση ο PeerManager δημιουργεί και μια νέα σχέση για κάθε διαφορετική κατηγορία που έχει εισαχθεί στη σχέση categories. Κάθε τέτοια σχέση έχει σαν όνομα το όνομα της κατηγορίας. Στη σχέση αυτή αποθηκεύονται όλα τα URIs που προσφέρουν την αντίστοιχη διεπαφή. Έτσι λοιπόν σε κάθε τέτοια σχέση υπάρχουν τόσα URIs όσες και οι διαθέσιμες υπηρεσίες της συγκεκριμένης κατηγορίας.



Σχήμα 3.5 Λειτουργίες του PeerManager

3.4.1. Υπηρεσίες Εισαγωγής και Εξαγωγής Υπηρεσιών από το περιβάλλον του CoWSAMI

Έχοντας λοιπόν αυτή τη βασική δομή αποθήκευσης, ο PeerManager μπορεί να εκτελέσει τις λειτουργίες του με πολύ μεγάλη ευκολία. Ξεκινώντας με τη λειτουργία εισαγωγής μιας υπηρεσίας στο περιβάλλον (join), το μοναδικό πράγμα που έχει να κάνει ο PeerManager, που βρίσκεται εγκατεστημένος στη συσκευή που παρέχει την υπηρεσία, είναι να εισάγει το URI της συγκεκριμένης υπηρεσίας στη σχέση της κατηγορίας στην οποία ανήκει και στη συνέχεια καλώντας όλους τους PeerManager που βρίσκονται στην εμβέλεια της, τους οποίους βρίσκει μέσω της ND, να τους πληροφορήσει για την ύπαρξη της. Στη συνέχεια οι PeerManagers με τη σειρά τους θα ενημερώσουν ή όχι την αντίστοιχη σχέση της βάσης δεδομένων τους με το συγκεκριμένο URI ανάλογα με το αν έχουν επιλέξει ή όχι την πολιτική της συνεχούς ενημέρωσης για τη κατηγορία που ανήκει η συγκεκριμένη υπηρεσία.

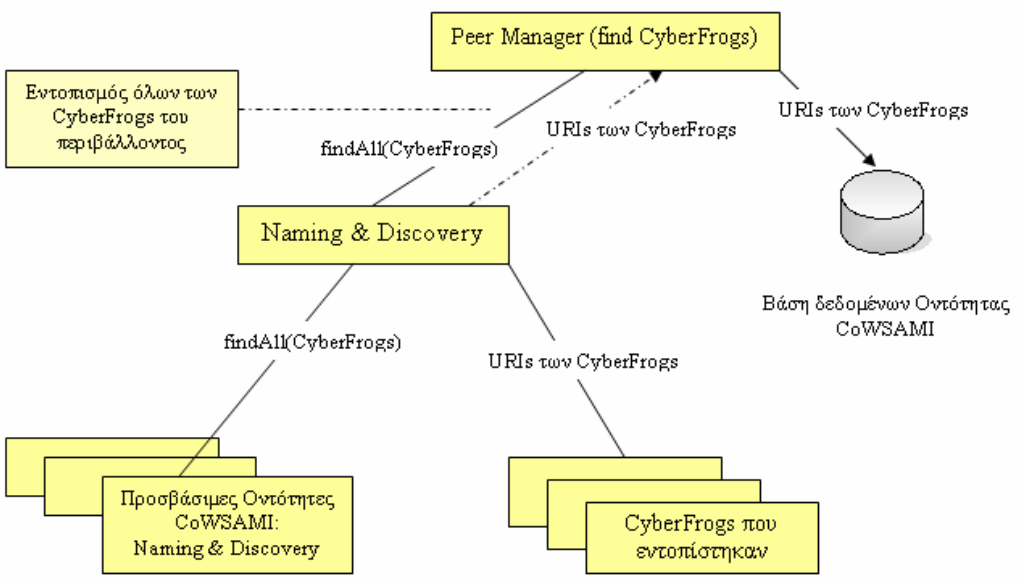


Σχήμα 3.6 Διαδικασία Εισαγωγής μιας Υπηρεσίας στο Περιβάλλον του CoWSAMI

Ακριβώς το αντίθετο χρειάζεται να κάνει στην περίπτωση αποχώρησης μιας υπηρεσίας από το περιβάλλον. Δηλαδή θα πρέπει να διαγράψει το URI αυτής της υπηρεσίας από την βάση δεδομένων και να πληροφορήσει τους άλλους PeerManager οι οποίοι θα διαγράψουν, αν έχουν επιλέξει την πολιτική της συνεχούς ενημέρωσης για την κατηγορία της υπηρεσία, την αντίστοιχη πλειάδα της σχέσης που υπάρχει στη βάση δεδομένων τους.

3.4.2. Πολιτικές της Λειτουργίας Ενημέρωσης Διαθέσιμων Υπηρεσιών

Όσον αφορά στη λειτουργία ενημέρωσης σχετικά με το ποιες υπηρεσίες μιας κατηγορίας είναι ανά πάσα στιγμή διαθέσιμες θα πρέπει να σημειωθεί ότι κάθε καταναλωτής συμφραζομένων έχει τη δυνατότητα να ορίσει οποιαδήποτε πολιτική ενημέρωσης επιθυμεί για κάθε κατηγορία υπηρεσιών και μπορεί βεβαίως να αλλάζει αυτή την πολιτική όποτε το επιθυμεί. Ξεκινώντας με την πιο απλή πολιτική η οποία είναι η ενημέρωση μετά από αίτημα του καταναλωτή συμφραζομένων, η οποία υλοποιείται με την λειτουργία find, το μόνο που χρειάζεται να γίνει είναι να κληθεί η υπηρεσία ND η οποία θα αναζητήσει όλες τις υπηρεσίες της κατηγορίας που ζήτησε ο χρήστης. Τα αποτελέσματα που προκύπτουν, δηλαδή τα URIs των υπηρεσιών, αποθηκεύονται στην αντίστοιχη σχέση στη βάση δεδομένων σβήνοντας ταυτόχρονα οποιαδήποτε άλλα προγενέστερα στοιχεία.



Σχήμα 3.7 Διαδικασία Ενημέρωσης Διαθέσιμων Υπηρεσιών μετά από Αίτημα του Χρήστη

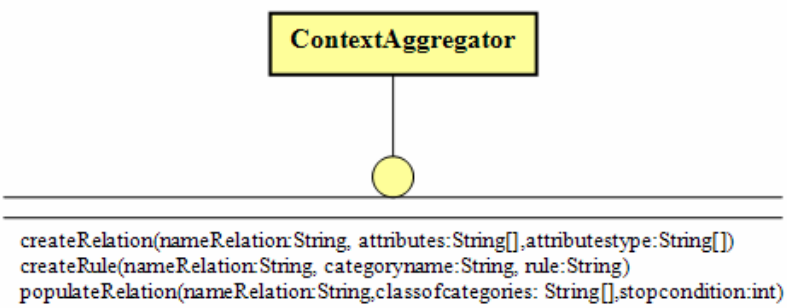
Στην περίπτωση που επιλεγεί η περιοδική ενημέρωση (periodic_update), ο καταναλωτής συμφραζομένων δηλώνει το όνομα της κατηγορίας για την οποία θέλει να εφαρμοστεί η συγκεκριμένη πολιτική καθώς επίσης και την περίοδο που θα μεσολαβεί μεταξύ δύο διαδοχικών ενημερώσεων και αυτό που γίνεται είναι ακριβώς το ίδιο με την ενημέρωση μετά από αίτημα μόνο που τώρα αυτό επαναλαμβάνεται συνεχώς ανά χρονικά διαστήματα ίσα με

την περίοδο που έδωσε ο χρήστης. Η διαδικασία αυτή σταματά μόνο όταν ο καταναλωτής συμφραζομένων δηλώσει ρητά ότι το επιθυμεί.

Τέλος η χρήση της συνεχής ενημέρωσης (always_update) επιτρέπει στον PeerManager να εισάγει στην βάση δεδομένων του, URIs υπηρεσιών για την παρουσία των οποίων ενημερώνεται από άλλους PeerManager οι οποίοι εισάγουν αυτές τις υπηρεσίες στο περιβάλλον. Σε περίπτωση μη υιοθέτησης αυτής της πολιτικής ο PeerManager της οντότητας δεν ενημερώνει τη βάση για υπηρεσίες για τις οποίες υπάρχει πληροφόρηση μέσω κάποιου άλλου PeerManager. Για την εφαρμογή της συγκεκριμένης πολιτικής το τρίτο πεδίο της σχέσης categories που περιγράφαμε στην αρχή της παραγράφου το οποίο λέγεται always_update(=συνεχής ενημέρωση) αποθηκεύει για κάθε διαφορετική κατηγορία την επιλογή του χρήστη για τη συγκεκριμένη πολιτική. Αρχικά η τιμή για το συγκεκριμένο πεδίο είναι true(=αληθής), δηλαδή είναι ενεργή η συνεχής ενημέρωση, αλλά κάθε φορά που ο καταναλωτής συμφραζομένων αλλάζει άποψη για τη χρήση ή όχι της συνεχούς πολιτικής ενημέρωσης για μια συγκεκριμένη κατηγορία το πεδίο always_update γίνεται true ή false ανάλογα με την επιλογή του.

3.5. Υπηρεσία ContextAggregator

Ο ContextAggregator με τις τρεις λειτουργίες του δίνει τη δυνατότητα στον καταναλωτή συμφραζομένων να δηλώνει με δυναμικό τρόπο τα συμφραζόμενα που τον ενδιαφέρουν, να τα αποθηκεύει τοπικά στη βάση δεδομένων του αλλά και να τα επεξεργάζεται κάνοντας SQL ερωτήματα.



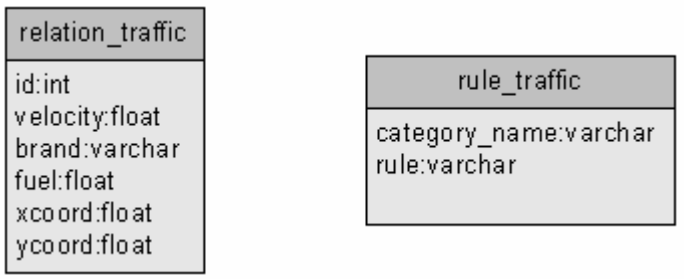
Σχήμα 3.8 Λειτουργίες του ContextAggregator

3.5.1. Λειτουργία Δημιουργία Σχέσης Συμφραζομένων

Η πρώτη λειτουργία λοιπόν του ContextAggregator, η createRelation(=δημιούργησε σχέση), επιτρέπει σε ένα καταναλωτή συμφραζομένων να δηλώσει το όνομα της σχέσης συμφραζομένων που επιθυμεί να δημιουργήσει καθώς επίσης και τις ιδιότητές της. Αυτό που έχει να κάνει ο καταναλωτής συμφραζομένων είναι να γράψει το όνομα της σχέσης, τα ονόματα των ιδιοτήτων της καθώς και τον τύπο δεδομένων για καθεμιά από αυτές τις ιδιότητες. Το CoWSAMI έχει οργανώσει έτσι τη βάση δεδομένων του κάθε καταναλωτή συμφραζομένων και για την συγκεκριμένη λειτουργία υπάρχει ήδη κατασκευασμένη μια σχέση με όνομα relations(=σχέσεις) η οποία αποτελείται από δύο πεδία από τα οποία το πρώτο αποθηκεύει το μοναδικό αναγνωριστικό κάθε σχέσης συμφραζομένων που δημιουργείται ενώ το δεύτερο διατηρεί το όνομα της κάθε σχέσης συμφραζομένων. Έτσι λοιπόν κατά τη δημιουργία μιας σχέσης συμφραζομένων το πρώτο πράγμα που πραγματοποιείται από τον ContextAggregator είναι η εισαγωγή μιας νέας πλειάδας στη σχέση αυτή. Όσον αφορά τις ιδιότητες της σχέσης συμφραζομένων διατηρούνται σε άλλη σχέση, η οποία δημιουργείται τη στιγμή που ο χρήστης αιτείται τη δημιουργία μιας σχέσης συμφραζομένων. Το όνομα της σχέσης αυτής προκύπτει από το πρόθεμα relation_ ακολουθούμενο από το όνομα της δηλωθείσας σχέσης συμφραζομένων. Τα πεδία είναι οι ιδιότητες της σχέσης συμφραζομένων και ο τύπος δεδομένων κάθε πεδίου καθορίζεται σύμφωνα με τα όσα είχε σημειώσει ο καταναλωτής συμφραζομένων κατά το αίτημα του. Στη συγκεκριμένη σχέση θα αποθηκεύονται οποιεσδήποτε τιμές συλλέγονται στο μέλλον από παρόχους που σχετίζονται με αυτή τη σχέση συμφραζομένων.

Πέρα όμως από τα παραπάνω πρέπει να σημειωθεί ότι ταυτόχρονα με όλες αυτές τις ενέργειες πραγματοποιείται και η δημιουργία μιας επιπλέον σχέσης η χρησιμότητα της οποίας είναι τεράστια, μιας και στη σχέση αυτή γίνεται η αποθήκευση των κανόνων που προσδιορίζουν το τρόπο εύρεσης και συγκέντρωσης τιμών που σχετίζονται με τη σχέση συμφραζομένων. Η σχέση έχει επίσης σύνθετο όνομα, αποτελούμενο από το πρόθεμα rule_ και το όνομα της σχέσης συμφραζομένων. Δύο πεδία απαρτίζουν τη σχέση αυτή εκ των οποίων το πρώτο έχει τίτλο categoryname και αποθηκεύει το όνομα μιας κατηγορίας ενώ το δεύτερο έχει τίτλο rule(=κανόνας) και αποθηκεύει τον κανόνα για τη σχέση. Η συμπλήρωση και διαχείριση της συγκεκριμένης σχέσης γίνεται από τη λειτουργία createRule του ContextAggregator, η οποία θα εξηγηθεί λεπτομερώς στην επόμενη παράγραφο. Έτσι λοιπόν στην περίπτωση του Άγγλου τουρίστα του παραδείγματος, κατά τη δημιουργία της σχέσης

TRAFFIC θα δημιουργηθούν οι παρακάτω σχέσεις στη βάση δεδομένων που υπάρχει στη συσκευή του (σχήμα 3.9).



Σχήμα 3.9 Σχέσεις της Βάσης Δεδομένων που Δημιουργούνται κατά τη Δήλωση μιας Σχέσης Συμφραζομένων

3.5.2. Λειτουργία Δημιουργίας Κανόνα μιας Σχέσης Συμφραζομένων

Έχοντας λοιπόν ένας χρήστης ορίσει τις σχέσεις συμφραζομένων και κατ' επέκταση και τα συμφραζόμενα του περιβάλλοντος για τα οποία ενδιαφέρεται, θα πρέπει να κάνει τη δήλωση των κανόνων κάθε σχέσης συμφραζομένων. Όπως έχει εξηγηθεί οι κανόνες κάθε σχέσης συμφραζομένων καθορίζουν τον τρόπο που συλλέγονται οι τιμές για τις ιδιότητες τους. Πριν όμως από οτιδήποτε άλλο χρειάζεται να παρουσιαστεί αναλυτικά η δομή ενός κανόνα και να επεξηγηθεί κάθε δομικό κομμάτι του. Παρατηρώντας κανείς ένα κανόνα (σχήμα 3.10) μπορεί να διακρίνει ότι αποτελείται από δύο τμήματα: την κατηγορία υπηρεσίας και το σώμα του κανόνα. Το πρώτο καθορίζει μια διεπαφή υπηρεσιών και το δεύτερο το σώμα του κανόνα.

ΚΑΤΗΓΟΡΙΑ ΥΠΗΡΕΣΙΑΣ	ΣΩΜΑ ΚΑΝΟΝΑ			
	1η ιδιότητα	μέθοδος	Επιστρεφόμενο τμήμα	Χρήση Συνάρτησης Μετατροπής
	2η ιδιότητα	Μέθοδος	Επιστρεφόμενο τμήμα	Χρήση Συνάρτησης Μετατροπής
	....			
	nη ιδιότητα	μέθοδος	Επιστρεφόμενο τμήμα	Χρήση Συνάρτησης Μετατροπής

Σχήμα 3.10 Δομή Κανόνα μιας Σχέσης Συμφραζομένων

Με βάση το πρότυπο των υπηρεσιών Διαδικτύου μια διεπαφή περιλαμβάνει ένα σύνολο από μεθόδους/λειτουργίες. Κάθε μέθοδος δέχεται σαν όρισμα ένα μήνυμα εισόδου και επιστρέφει σαν αποτέλεσμα ένα μήνυμα εξόδου σε αυτόν που την κάλεσε. Γενικά τα περιεχόμενα ενός μηνύματος χωρίζονται σε επιμέρους τμήματα.

Με βάση τα παραπάνω το σώμα του κανόνα αποτελείται από μια ακολουθία πλειάδων που αποτελούνται από τέσσερα επιμέρους τμήματα. Η πλειάδα έχει να κάνει με τον τρόπο συμπλήρωσης μιας ιδιότητας της σχέσης συμφραζομένων, οπότε εύκολα αντιλαμβάνεται κανείς ότι το σώμα του κανόνα αποτελείται από τόσες πλειάδες όσες είναι και οι ιδιότητες της σχέσης στην οποία ανήκει ο συγκεκριμένος κανόνας.

Πιο αναλυτικά:

- Το πρώτο τμήμα της κάθε τετράδας παρατηρούμε ότι περιέχει το όνομα μιας ιδιότητας της σχέσης συμφραζομένων για την οποία δημιουργείται ο κανόνας. Μάλιστα καθορίζει ότι τα υπόλοιπα τρία στοιχεία της τετράδας αφορούν την συμπλήρωση της ιδιότητας με αυτό το όνομα.
- Το δεύτερο στοιχείο περιέχει το όνομα της μεθόδου που πρέπει να κληθεί προκειμένου να συμπληρωθεί η συγκεκριμένη ιδιότητα. Η μέθοδος αυτή πρέπει να είναι μια από τις μεθόδους που παρέχουν οι υπηρεσίες της κατηγορίας που έχει δηλωθεί αρχικά μαζί με το σώμα του κανόνα.
- Το τρίτο στοιχείο καθορίζει το τμήμα του μηνύματος επιστροφής της μεθόδου που θα χρησιμοποιηθεί για τη συμπλήρωση της ιδιότητας της σχέσης συμφραζομένων.
- Το τέταρτο στοιχείο της τετράδας του κανόνα είναι το όνομα της υπηρεσίας μετατροπής.

Η υπηρεσία μετατροπής είναι μια υπηρεσία Διαδικτύου η οποία είναι τοπικά εγκατεστημένη από το χρήστη και σκοπός της είναι να δέχεται συμφραζόμενα και να τα τροποποιεί ως προς την αναπαράσταση τους (π.χ. τιμές που αναπαριστούν χιλιόμετρα να μετατραπούν στην αντίστοιχη τιμή σε μίλια). Στη συσκευή του καταναλωτή συμφραζομένων μπορεί να υπάρχουν πολλές τέτοιου είδους διαφορετικές υπηρεσίες η καθεμιά από τις οποίες εκτελεί μια διαφορετική μετατροπή. Για λόγους υλοποίησης θεωρείται ότι κάθε υπηρεσία τέτοιου είδους έχει μια μόνο μέθοδο η οποία ονομάζεται convert. Μέσω αυτής γίνεται η επεξεργασία και η επιστροφή των μετατρεπόμενων τιμών. Στα σχήματα 3.9 και 3.10 φαίνονται δύο κανόνες που

υπάρχουν στη βάση δεδομένων του Άγγλου τουρίστα του παραδείγματος οι οποίοι αφορούν τη συμπλήρωση της σχέσης συμφραζομένων TRAFFIC.

	ΣΩΜΑ ΚΑΝΟΝΑ			
CyberFrog	atr 1: Velocity	oper 1: getVelocity	message 1: getVelocityResponse	Conversion :
	atr 2: ID	oper 1: getID	message 1: getIDResponse	Conversion : Converter
	atr 3: Brand	oper 1: getBrand	message 1: getBrandResponse	Conversion :
	atr 4: Fuel	oper 1: getFuel	message 1: getFuelResponse	Conversion : Converter
	atr 5: XPos	oper 1: getXPos	message 1: getXPosResponse	Conversion :
	atr 6: YPos	oper 1: getYPos	message 1: getYPosResponse	Conversion : Converter //

Σχήμα 3.11 Κανόνας για τη Σχέση Συμφραζομένων TRAFFIC (CyberFrogs)

	ΣΩΜΑ ΚΑΝΟΝΑ			
CyberCab	atr 1: Velocity	oper 1: getState	message 1: getVelocityResponse	Conversion : Converter
	atr 2: ID	oper 1: getState	message 1: getIDResponse	Conversion :
	atr 3: Brand	oper 1: getState	message 1: getBrandResponse	Conversion : Converter
	atr 4: Fuel	oper 1: getState	message 1: getFuelResponse	Conversion : Converter
	atr 5: XPos	oper 1: getState	message 1: getXPosResponse	Conversion :
	atr 6: YPos	oper 1: getState	message 1: getYPosResponse	Conversion : //

Σχήμα 3.12 Κανόνας για τη Σχέση Συμφραζομένων TRAFFIC(CyberCabs)

Στον κανόνα του σχήματος 3.11 φαίνεται ότι οι ιδιότητες της σχέσης συμφραζομένων TRAFFIC θα συμπληρωθούν με τιμές που παρέχουν μέθοδοι υπηρεσιών της κατηγορίας CyberFrogs. Αυτό δηλώνεται στο πεδίο categoryname της σχέσης rule_traffic. Για καθεμιά από τις έξι ιδιότητες της σχέσης συμφραζομένων υπάρχει μια μέθοδος των υπηρεσιών της κατηγορίας CyberFrogs που πρέπει να κληθεί αλλά και το τμήμα της κάθε μεθόδου που περιέχει την τιμή που θα συμπληρώσει την κάθε ιδιότητα. Για τη συμπλήρωση της ιδιότητα brand (=μάρκα) η οποία είναι η τρίτη κατά σειρά ιδιότητα (atr 3:) θα πρέπει να κληθεί η μέθοδος getBrand και να ληφθεί το τμήμα getBrandResponse της απάντησης της.

Στον κανόνα του σχήματος 3.12 παρατηρούμε ότι η ίδια σχέση συμφραζομένων (TRAFFIC) θα συμπληρωθεί με τιμές οι οποίες θα συλλεχθούν όταν κληθούν υπηρεσίες της κατηγορίας CyberCabs. Σε αντίθεση με τον προηγούμενο κανόνα στην συγκεκριμένη περίπτωση υπάρχει μια μόνο μέθοδος η οποία πρέπει να κληθεί για τη συλλογή τιμών για τις έξι ιδιότητες της σχέσης συμφραζομένων. Η μέθοδος αυτή είναι η getState(). Όμως για καθεμιά από τις έξι ιδιότητες θα χρησιμοποιηθεί διαφορετικό τμήμα του μηνύματος επιστροφής. Όπως φαίνεται στο σχήμα τόσο η ιδιότητα brand (atr 3:) όσο και η ιδιότητα fuel (atr 4:) θα πάρουν τιμή μέσω

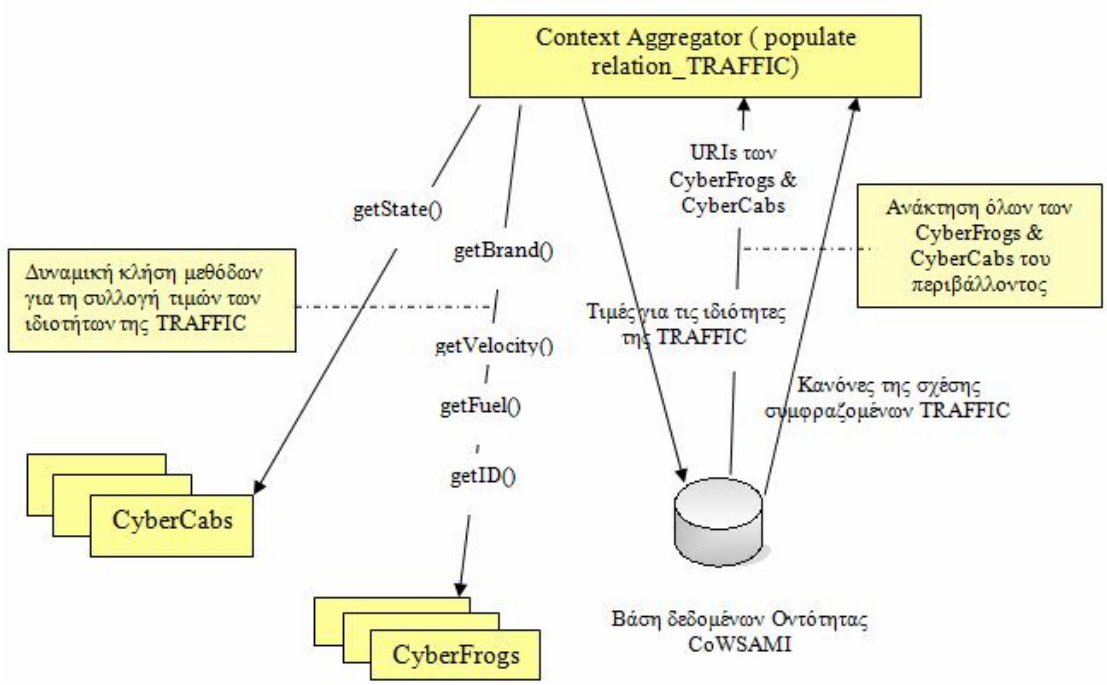
της μεθόδου getState(), μόνο που η πρώτη θα λάβει την τιμή που επιστρέφει το τμήμα getBrandResponse ενώ η δεύτερη θα λάβει την τιμή του τμήματος getFuelResponse.

3.5.3. Λειτουργία Συλλογής Τιμών για τις Ιδιότητες μιας Σχέσης Συμφραζομένων

Η τρίτη λειτουργία του ContextAggregator είναι η populateRelation η οποία συμπληρώνει μια δοθείσα σχέση συμφραζομένων με τις τιμές που συγκεντρώνονται όταν κληθούν και επιστρέψουν τις τιμές τους εκείνες οι μέθοδοι των υπηρεσιών που ορίζονται από τους κανόνες της. Μάλιστα το πρώτο πράγμα που γίνεται κατά τη διάρκεια εκτέλεσης αυτής της λειτουργίας είναι η ανάγνωση κάθε εγγραφής που υπάρχει στη σχέση κανόνων της δοθείσας σχέσης συμφραζομένων. Για κάθε κανόνα γίνονται τα παρακάτω:

- εντοπίζονται από την αντίστοιχη σχέση της βάσης δεδομένων όλες οι υπηρεσίες της κατηγορίας η οποία είναι δηλωμένη στο πρώτο πεδίο της εγγραφής που περιέχει τον κανόνα.
- στη συνέχεια για κάθε υπηρεσία που έχει εντοπιστεί γίνεται κλήση των μεθόδων της, οι οποίες δηλώνονται στο κυρίως σώμα του κανόνα. Με τον τρόπο αυτό συλλέγονται οι τιμές που αντιστοιχούν στις ιδιότητες της σχέσης
- μετά από αυτό γίνεται η εισαγωγή των τιμών αυτών στη σχέση της βάσης δεδομένων που αντιστοιχεί στη συγκεκριμένη σχέση συμφραζομένων και αρχίζει να γίνεται η συμπλήρωση. Αξίζει να αναφερθεί ότι αν στον κανόνα έχει δηλωθεί η χρήση μεθόδου μετατροπής για κάποιες τιμές, οι τιμές αυτές πρώτα δέχονται την επεξεργασία της υπηρεσία μετατροπής και μετά εισάγονται στον πίνακα της σχέσης.

Από τα παραπάνω λοιπόν προκύπτει ότι για κάθε κανόνα που είναι δηλωμένος θα προκύψουν τόσες εγγραφές στη σχέση της βάσης δεδομένων που αντιστοιχεί στη σχέση συμφραζομένων όσες είναι και οι υπηρεσίες της κατηγορίας που καθορίζεται στον κανόνα. Ακολουθώντας αυτή τη διαδικασία η σχέση θα συμπληρωθεί με τόσες εγγραφές όσες όλες οι υπηρεσίες των κατηγοριών που ο χρήστης έχει δηλώσει στους κανόνες. Πρέπει να αναφερθεί επίσης ότι η populateRelation δεν εκτελείται αυτόνομα/άμεσα διαμέσου εντολής του χρήστη αλλά ενεργοποιείται ανάλογα με τα ερωτήματα που γίνονται μέσω του SQLInterpreter.



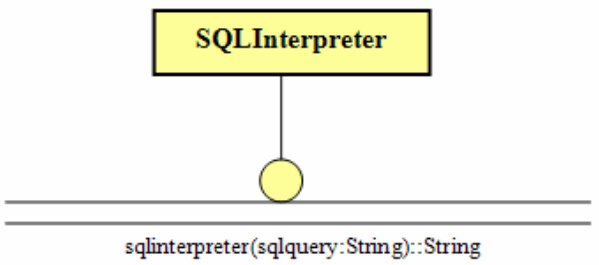
Σχήμα 3.13 Διαδικασία Συλλογής Τιμών για τις Ιδιότητες μιας Σχέσης Συμφραζομένων

3.6. Υπηρεσία SQLInterpreter

Ο SQLInterpreter με τη λειτουργία sqlinterpret() συμβάλλει στην επεξεργασία και διαχείριση των συμφραζομένων που έχουν συλλεχθεί. Πιο αναλυτικά αυτό που συμβαίνει είναι η εκτέλεση απλών SQL ερωτημάτων στις σχέσεις της βάσης δεδομένων που περιέχουν τις πλειάδες τιμών που αφορούν τις σχέσεις συμφραζομένων που έχουν δημιουργηθεί. Αυτό που γίνεται από τη λειτουργία του SQLInterpreter κατά την εκτέλεση ενός τέτοιου ερωτήματος είναι το εξής: Στην αρχή γίνεται μια επεξεργασία του ερωτήματος και εντοπίζονται οι σχέσεις συμφραζομένων οι οποίες συνεισφέρουν στην απάντηση του ερωτήματος. Έπειτα γίνεται συμπλήρωση των σχέσεων συμφραζομένων που εντοπίστηκαν ότι συνεισφέρουν στο ερώτημα κάνοντας κλήση της λειτουργίας populateRelation για καθεμιά από τις σχέσεις συμφραζομένων. Μετά τα παραπάνω γίνεται η άμεση εκτέλεση του ερωτήματος του καταναλωτή συμφραζομένων πάνω στα ενημερωμένα πλέον στοιχεία και παράγονται τα αποτελέσματα τα οποία αποτυπώνονται και αυτά με τη χρήση του γραφικού περιβάλλοντος σε ειδικό πλαίσιο αποτελεσμάτων.

Πέρα όμως από τον κλασσικό τρόπο υποβολής ερωτημάτων SQL, η συγκεκριμένη υλοποίηση του SQLIntrepreter επιτρέπει την υποβολή ερωτημάτων με την εισαγωγή κάποιων επιπλέον περιορισμών. Οι περιορισμοί που μπορεί να προσθέσει ο καταναλωτής συμφραζομένων

έχουν να κάνουν με τη διαθεσιμότητα και το χρόνο απόκρισης των υπηρεσιών που συνεισφέρουν στη συμπλήρωση μιας σχέσης συμφραζομένων. Επίσης μπορούν να τεθούν περιορισμοί που σχετίζονται με το πλήθος των παρόχων που ο καταναλωτής συμφραζομένων επιθυμεί να ερωτηθούν προκειμένου να συλλεχθούν και να επεξεργαστούν τα συμφραζόμενα που ζητά.



Σχήμα 3.14 Λειτουργία του SQLInterpreter

Αναλυτικότερα οι επιλογές για τους περιορισμούς που μπορεί να θέσει ο χρήστης είναι τέσσερις:

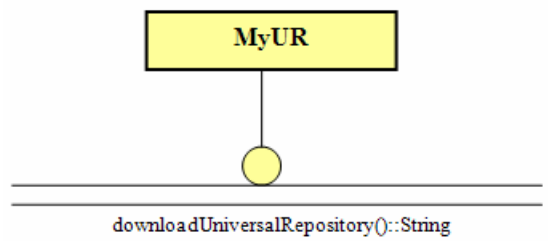
- availability (=διαθεσιμότητα) και response time (=χρόνος απόκρισης) οι οποίοι αναφέρονται στα χαρακτηριστικά που πρέπει να έχουν οι κατηγορίες των υπηρεσιών που θα συνεισφέρουν στην συμπλήρωση μιας σχέσης. Στο CoWSAMI θεωρείται ότι οι τιμές που μπορούν να πάρουν η διαθεσιμότητα και ο χρόνος απόκρισης μιας υπηρεσίας είναι τρεις και πιο συγκεκριμένα είναι: High(υψηλή), Medium(μέτρια), Low(χαμηλή). Έτσι λοιπόν οι τιμές που μπορεί να δώσει ο καταναλωτής συμφραζομένων για την διαθεσιμότητα και το χρόνο απόκρισης κατά την υποβολή ερωτήματος είναι οποιοσδήποτε συνδυασμός μεταξύ των τριών αυτών επιτρεπτών επιλογών. Οι τιμές που έχει κάθε κατηγορία για καθένα από αυτά τα δύο χαρακτηριστικά είναι σταθερές και δηλώνονται στην κεντρική αποθήκη του CoWSAMI. Από την κεντρική αυτή αποθήκη μέσω της υπηρεσίας MyUR μπορεί ο κάθε χρήστης να λάβει αυτές τις πληροφορίες και να τις αποθηκεύσει στη βάση δεδομένων του.
- class (= κλάση κατηγοριών) είναι ένα πεδίο όπου ο καταναλωτής συμφραζομένων έχει τη δυνατότητα να δηλώσει ένα σύνολο από κατηγορίες υπηρεσιών τις οποίες θα χρησιμοποιήσει ο SQLInterpreter για να απαντήσει ένα ερώτημα. Σε κάθε περίπτωση

όμως πρέπει οι κατηγορίες που τελικά θα συνεισφέρουν στην απάντηση του ερωτήματος να ικανοποιούν τόσο τους περιορισμούς για διαθεσιμότητα και χρόνο απόκρισης αλλά και να ανήκουν στην συγκεκριμένη κλάση που δηλώθηκε.

- Stop condition (=συνθήκη τερματισμού) μέσω της οποίας ο καταναλωτής συμφραζομένων μπορεί να δώσει έναν αριθμό ο οποίος θα είναι το άνω φράγμα στο πλήθος των συσκευών που θα τους ζητηθεί να προσφέρουν τις επιστρεφόμενες τιμές των υπηρεσιών τους.

3.7. Υπηρεσία MyUR

Για τη συλλογή των πληροφοριών που βρίσκονται αποθηκευμένες στην κεντρική αποθήκη που αναφέρθηκε πιο πάνω υπάρχει η υπηρεσία MyUR. Η υπηρεσία αυτή χρησιμοποιείται για τη μεταφορά δεδομένων από την κεντρική αποθήκη στον τοπικό δίσκο του κάθε καταναλωτή συμφραζομένων. Τα δεδομένα αυτά είναι απαραίτητα για την εκτέλεση των SQL ερωτημάτων από τον SQL Interpreter.



Σχήμα 3.15 Λειτουργία της MyUR

Προκειμένου να επιτευχθεί το παραπάνω χρειάζεται ο χρήστης στη βάση δεδομένων του να έχει σχέσεις κατάλληλα συσχετιζόμενες μεταξύ τους. Την εργασία αυτή αναλαμβάνει το CoWSAMI το οποίο κατά την εγκατάσταση του δημιουργεί στη βάση δεδομένων κάθε χρήστη τέσσερις σχέσεις με ονόματα universalrepository, urcategories, uroperations και urparts. Στις τρεις τελευταίες σχέσεις αποθηκεύονται τα ονόματα όλων των κατηγοριών υπηρεσιών, μεθόδων και τμημάτων μηνυμάτων επιστροφής αντίστοιχα μαζί με τα μοναδικά αναγνωριστικά τους. Στη σχέση urcategories υπάρχουν επιπλέον δύο πεδία στα οποία αποθηκεύονται οι τιμές για την διαθεσιμότητα και το χρόνο απόκρισης που χαρακτηρίζει κάθε διαφορετική κατηγορία υπηρεσιών. Η σχέση universalrepository περιέχει τρία πεδία στα

οποία αποθηκεύονται τα αναγνωριστικά, που υπάρχουν στις άλλες τρεις σχέσεις, με τέτοιο τρόπο προκειμένου να γίνει η σωστή συσχέτιση μεταξύ κατηγορίας, μεθόδου και τμήματος μηνύματος επιστροφής. Έτσι λοιπόν με τη χρήση αυτής της υπηρεσίας ο χρήστης μπορεί να ξέρει ανά πάσα στιγμή τι κατηγορίες υπηρεσιών υπάρχουν, τι μεθόδους έχει η καθεμιά αλλά ακόμη και τι δομή έχει το μήνυμα που επιστρέφει η κάθε μέθοδος. Στοιχεία ιδιαίτερα χρήσιμα κατά τη δήλωση κανόνων αλλά και κατά την αναζήτηση υπηρεσιών.

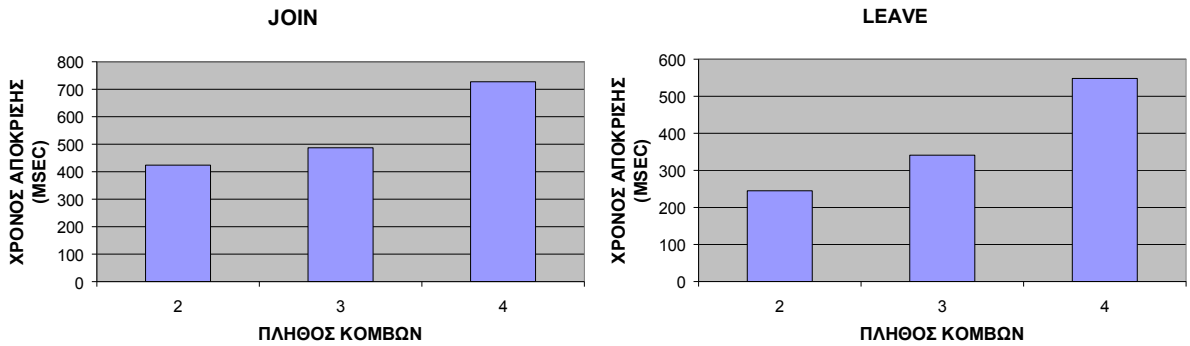
ΚΕΦΑΛΑΙΟ 4. ΠΕΙΡΑΜΑΤΙΚΕΣ ΜΕΤΡΗΣΕΙΣ

4.1. Απόδοση CoWSAMI

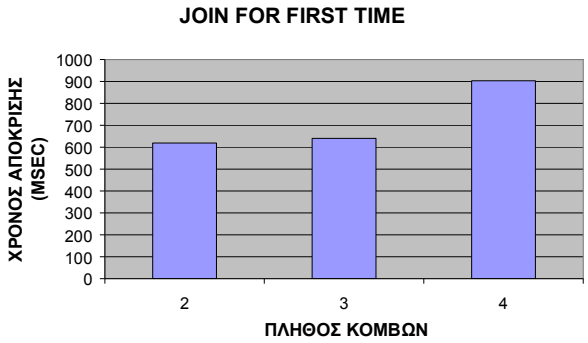
Στο συγκεκριμένο κεφάλαιο θα παρουσιαστούν ορισμένες μετρήσεις που έγιναν προκειμένου να εκτιμηθεί η απόδοση του CoWSAMI. Μεγαλύτερη προσοχή θα δοθεί στις λειτουργίες που παρέχονται από τον PeerManager και τον ContextAggregator. Τα χαρακτηριστικά που μετρήθηκαν έχουν να κάνουν με το χρόνο εκτέλεσης των λειτουργιών που παρέχουν οι δύο παραπάνω υπηρεσίες. Για καθεμιά από τις λειτουργίες των υπηρεσιών αυτών έγινε μέτρηση του χρόνου εκτέλεσης σε διαφορετικά σενάρια χρήσης. Οι πειραματικές μετρήσεις που θα παρατεθούν στη συνέχεια πραγματοποιήθηκαν σε ένα περιβάλλον το οποίο αποτελείται από τέσσερις υπολογιστές στους οποίους είχε εγκατασταθεί το CoWSAMI. Δύο από αυτούς συνδέονταν μέσω ασύρματου δικτύου και οι άλλοι δύο μέσω ενός τυπικού ενσύρματου δικτύου 100Mbps. Πιο συγκεκριμένα όσον αφορά τα τεχνικά χαρακτηριστικά των κόμβων αυτών πρέπει να πούμε ότι αποτελούνται από επεξεργαστές Pentium 4 και μνήμες χωρητικότητας 256, 512, 768 και 1024MB αντίστοιχα.

Σχετικά με την υπηρεσία του PeerManager, τα παρακάτω σχήματα δίνουν το χρόνο απόκρισης κατά την εκτέλεση των λειτουργιών της εισόδου και εξόδου (join και leave) υπηρεσιών Διαδικτύου που παρέχονται από παρόχους συμφραζομένων στο περιβάλλον. Οι πειραματικές μετρήσεις για τις παραπάνω λειτουργίες αναφέρονται σε τρία διαφορετικά σενάρια. Σε καθένα από τα διαφορετικά σενάρια αυξάνεται κατά ένα το πλήθος των κόμβων που είναι διαθέσιμοι στο περιβάλλον. Οποιοσδήποτε τέτοιος κόμβος μπορεί να παίξει τόσο το ρόλο του παρόχου όσο και το ρόλο του καταναλωτή συμφραζομένων. Αυτό που παρατηρείται από τη γραφική απεικόνιση των μετρήσεων (σχ.5.1 και σχ.5.2) είναι αύξηση του χρόνου απόκρισης των λειτουργιών ανάλογη με το πλήθος των κόμβων που βρίσκονται στο

περιβάλλον. Το συγκεκριμένο φαινόμενο εξηγείται πλήρως αν αναλογιστεί κανείς τον τρόπο υλοποίησης των συγκεκριμένων λειτουργιών όπου σε κάθε περίπτωση εισόδου ή εξόδου ενός παρόχου συμφραζομένων στο περιβάλλον, όλοι οι κόμβοι ενημερώνονται προκειμένου να διατηρήσουν στη βάση δεδομένων τους τα έγκυρα μόνο στοιχεία, στην περίπτωση που έχουν ενεργοποιήσει την πολιτική της συνεχούς ενημέρωσης για την κατηγορία των υπηρεσιών στην οποία ανήκει ο πάροχος που εισέρχεται/εξέρχεται στο περιβάλλον.



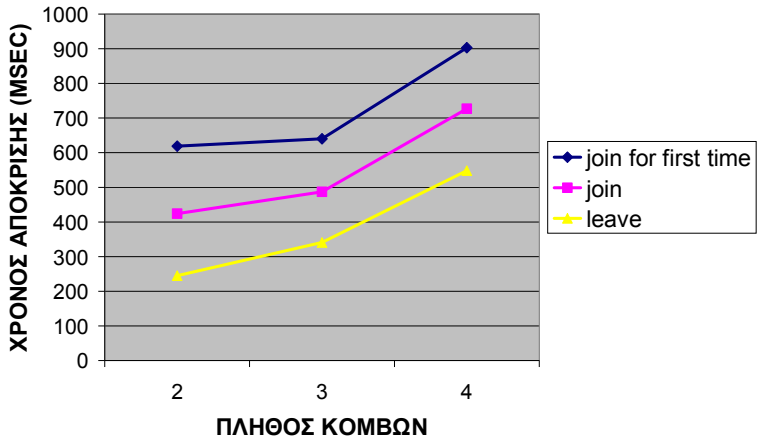
Σχήμα 4.1 Χρόνοι Απόκρισης κατά την Είσοδο και Έξοδο Υπηρεσιών στο Περιβάλλον



Σχήμα 4.2 Χρόνος Απόκρισης κατά την Είσοδο Υπηρεσίας για Πρώτη Φορά στο Περιβάλλον

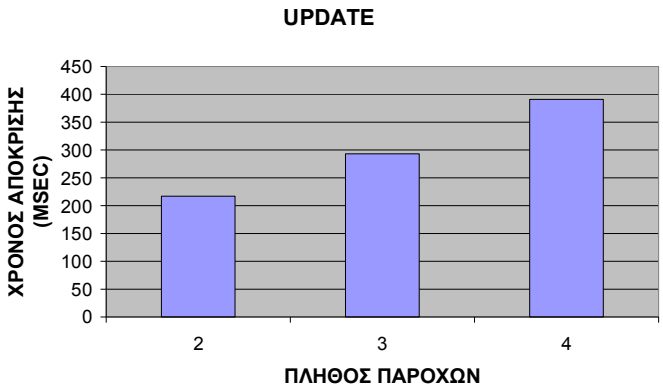
Αξιοσημείωτο είναι το γεγονός του αυξημένου χρόνου απόκρισης στην περίπτωση που ένας πάροχος που ανήκει σε μια νέα κατηγορία υπηρεσιών εισέρχεται για πρώτη φορά στο περιβάλλον. Αυτό είναι απολύτως δικαιολογημένο αφού στην περίπτωση που κάποια κατηγορία γνωστοποιείται στο περιβάλλον μέσω της εισόδου ενός παρόχου, όλοι οι κόμβοι που ενδιαφέρονται για τη συγκεκριμένη κατηγορία (στο πείραμα αυτό ισχύει για όλους) θα πρέπει να δημιουργήσουν μια νέα σχέση στη βάση δεδομένων τους στην οποία θα

αποθηκεύουν τις διευθύνσεις των παρόχων αυτής της κατηγορίας ενώ σε οποιαδήποτε άλλη περίπτωση εισόδου ενός παρόχου η συγκεκριμένη σχέση υπάρχει ήδη οπότε δεν χρειάζεται επιπλέον χρόνος για να δημιουργηθεί.



Σχήμα 4.3 Χρόνοι Απόκρισης του PeerManager κατά την Είσοδο/Εξοδο Υπηρεσιών στο Περιβάλλον

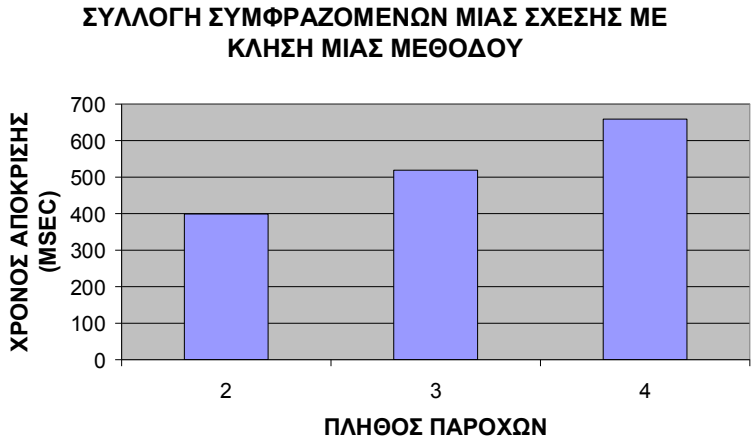
Όσον αφορά την λειτουργία που παρέχει τη δυνατότητα ενημέρωσης του κάθε καταναλωτή συμφραζομένων για τις διαφορετικές υπηρεσίες μιας κατηγορίας που είναι διαθέσιμες από τους διάφορους παρόχους πρέπει να διακρίνουμε δύο περιπτώσεις. Στην περίπτωση της συνεχούς ενημέρωσης η εφαρμογή της πολιτικής απαιτεί ελάχιστο χρόνο μιας και το μόνο που απαιτείται είναι η εισαγωγή μιας εγγραφής στην τοπική βάση δεδομένων. Από εκεί και πέρα το χρόνο για την συνεχή ενημέρωση του καταναλωτή συμφραζομένων επωμίζονται οι λειτουργίες της εισόδου/εξόδου μιας υπηρεσίας στο δίκτυο. Για την περίπτωση των άλλων δύο πολιτικών, δηλαδή της περιοδικής ενημέρωσης και της ενημέρωσης μετά από αίτημα του καταναλωτή συμφραζομένων, ο χρόνος που απαιτείται είναι ο ίδιος και για τις δύο περιπτώσεις αφού στην ουσία οι πολιτικές υλοποιούνται με τον ίδιο τρόπο, κάνοντας χρήση της υπηρεσίας ND, με τη διαφορά ότι στην περιοδική ενημέρωση ακολουθείται η ίδια διαδικασία επαναληπτικά. Από τα πειράματα που έγιναν στους τέσσερις κόμβους που ήταν διαθέσιμοι παρατηρήθηκε γραμμική αύξηση του χρόνου απόκρισης της λειτουργίας σε συνάρτηση με τον αριθμό των κόμβων που ήταν διαθέσιμοι, κάτι αντίστοιχο που παρατηρήθηκε και στις προηγούμενες δύο λειτουργίες της ίδιας υπηρεσίας.



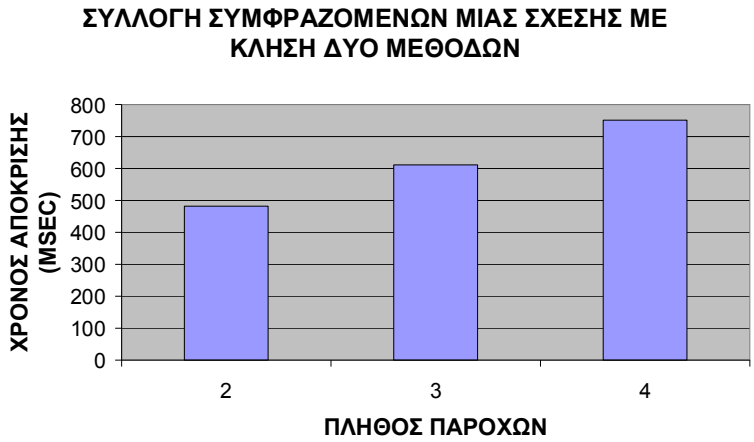
Σχήμα 4.4 Χρόνοι Απόκρισης Λειτουργίας Ενημέρωσης Διαθέσιμων Παρόχων Υπηρεσιών

Μετρήσεις όμως έγιναν και για τις λειτουργίες της υπηρεσίας του Context Aggregator από τις οποίες ενδιαφέρον παρουσιάζουν εκείνες που έχουν να κάνουν με τη συλλογή και αποθήκευση συμφραζομένων μιας σχέσης. Οι άλλες δύο λειτουργίες, η δημιουργία σχέσης και η δημιουργία κανόνα, απαιτούν χρόνο πολύ μικρό για να φέρουν εις πέρας την αποστολή τους αφού κατά την εκτέλεση τους γίνονται μόνο εγγραφές στην τοπική βάση δεδομένων του καταναλωτή συμφραζομένων. Έτσι δεν υπάρχουν μεγάλες καθυστερήσεις που υπεισέρχονται εξαιτίας της επικοινωνίας με άλλους κόμβους του δικτύου όπως συμβαίνει κατά τη συμπλήρωση μιας σχέσης. Επομένως γίνεται αντιληπτό ότι για την απόδοση του ContextAggregator σημαντικό ρόλο παίζει ο χρόνος απόκρισης στη συλλογή των συμφραζομένων που σχετίζονται με μια σχέση συμφραζομένων. Στο συγκεκριμένο πείραμα θεωρούμε ότι και οι τέσσερις κόμβοι του περιβάλλοντος παίζουν το ρόλο του παρόχου συμφραζομένων και μάλιστα οι υπηρεσίες που παρέχουν σχετίζονται με τη σχέση συμφραζομένων για την οποία εκτελείται η λειτουργία συλλογής και αποθήκευσης συμφραζομένων. Οι μετρήσεις αφορούν τη συμπλήρωση μιας σχέσης συμφραζομένων για την οποία κάθε φορά συνεισφέρουν οι δύο, τρεις ή οι τέσσερις πάροχοι. Πέρα όμως από αυτό μετρήσεις γίνονται και για τις περιπτώσεις συμπλήρωσης της σχέσης συμφραζομένων όπου η σχέση συμπληρώνεται μέσω μιας, δυο ή τριών μεθόδων μιας συγκεκριμένης υπηρεσίας που παρέχουν οι πάροχοι συμφραζομένων. Από τις μετρήσεις προκύπτει ότι ο χρόνος αυτός αυξάνεται όσο μεγαλύτερο είναι το πλήθος των παρόχων που υπάρχουν στο δίκτυο(σχ. 4.5 και 4.6). Πέρα όμως από αυτή τη διαπίστωση αξιοσημείωτο είναι το γεγονός της αύξησης του χρόνου απόκρισης της συγκεκριμένης λειτουργίας όταν για τη συμπλήρωση μιας σχέσης απαιτούνται όλο και περισσότερες κλήσεις μεθόδων μιας υπηρεσίας για τη συλλογή τιμών για

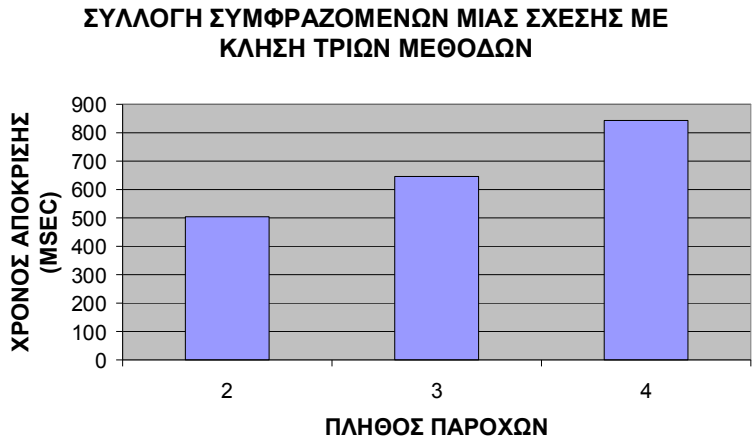
όλες τις ιδιότητες της σχέσης. Παρατηρείται λοιπόν το φαινόμενο ότι καθώς αυξάνεται ο αριθμός των μεθόδων μιας υπηρεσίας που χρειάζονται για να συμπληρωθούν οι ιδιότητες μιας σχέσης αυξάνεται και ο χρόνος που απαιτείται για να πραγματοποιηθεί αυτό (σχ. 4.5).



Σχήμα 4.5 Λειτουργίας Συλλογής Συμφραζομένων μιας Σχέση Συμφραζομένων με Κλήση μιας Μεθόδου

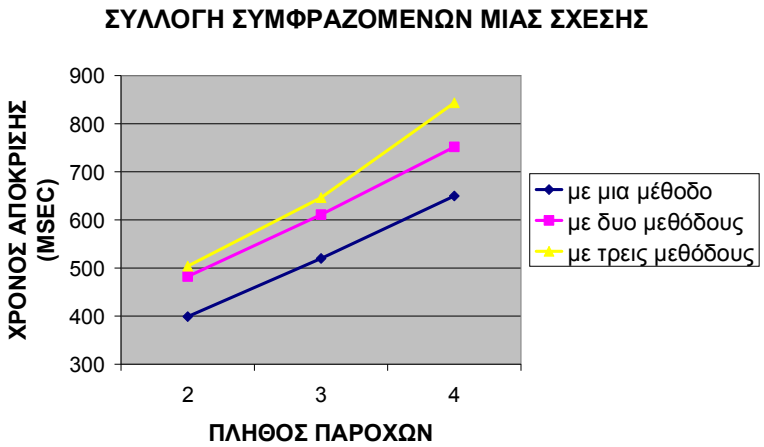


Σχήμα 4.6 Λειτουργίας Συλλογής Συμφραζομένων μιας Σχέση Συμφραζομένων με Κλήση δύο Μεθόδων



Σχήμα 4.7 Λειτουργίας Συλλογής Συμφραζομένων μιας Σχέση Συμφραζομένων με Κλήση τριών Μεθόδων

Έτσι λοιπόν είναι φυσικό σε ένα δίκτυο μια σχέση συμφραζομένων που χρειάζεται τη κλήση μιας μεθόδου για να συμπληρώσει τις ιδιότητες της να το κάνει πιο γρήγορα από κάποια άλλη που χρειάζεται δύο ή και περισσότερες κλήσεις μεθόδων αφού στη δεύτερη περίπτωση ο χρόνος που υπεισέρχεται για την κλήση και λήψη των επιστρεφόμενων τιμών υπολογίζεται δύο ή περισσότερες φορές όσες είναι οι μέθοδοι που απαιτούνται για τη συμπλήρωση της σχέσης. Στα παρακάτω σχήματα φαίνονται με διαγραμματικό τρόπο όλες αυτές οι παρατηρήσεις που αφορούν την απόδοση της υπηρεσίας του Context Aggregator.



Σχήμα 4.8 Λειτουργίας Συλλογής Συμφραζομένων μιας Σχέσης Συμφραζομένων

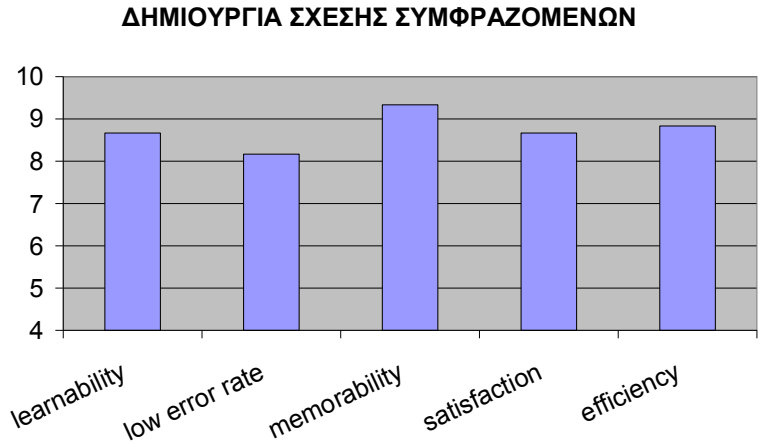
4.2. Χρηστικότητα Γραφικού Περιβάλλοντος Διεπαφής

Πέρα από τη μέτρηση της απόδοσης του ενδιάμεσου λογισμικού με κριτήριο το χρόνο απόκρισης της κάθε υπηρεσίας πραγματοποιήθηκε και μια άλλη μελέτη που αφορούσε το γραφικό περιβάλλον διεπαφής του CoWSAMI. Πιο συγκεκριμένα μελετήθηκε και μετρήθηκε η χρηστικότητα του γραφικού περιβάλλοντος διεπαφής της υπηρεσίας του Context Aggregator καθώς επίσης το πως αυτό επηρεάζει τη χρήση των λειτουργιών που αυτή παρέχει. Πριν παρατεθούν όμως τα στοιχεία που συγκεντρώθηκαν μετά από αυτές τις μετρήσεις καλό είναι να δοθεί ένας ορισμός για το τι εννοούμε χρηστικότητα συστήματος. Ως χρηστικότητα [1] ορίζεται η ευκολία χρήσης και η αποδοχή ενός συστήματος από μια ομάδα χρηστών που χρησιμοποιεί το συγκεκριμένο σύστημα για να φέρει εις πέρας κάποιες εργασίες σε ένα συγκεκριμένο περιβάλλον. Η ευκολία χρήσης αντιπροσωπεύει την απόδοση και την ικανοποίηση των χρηστών ενώ η αποδοχή εκφράζει το πόσο το σύστημα χρησιμοποιείται.

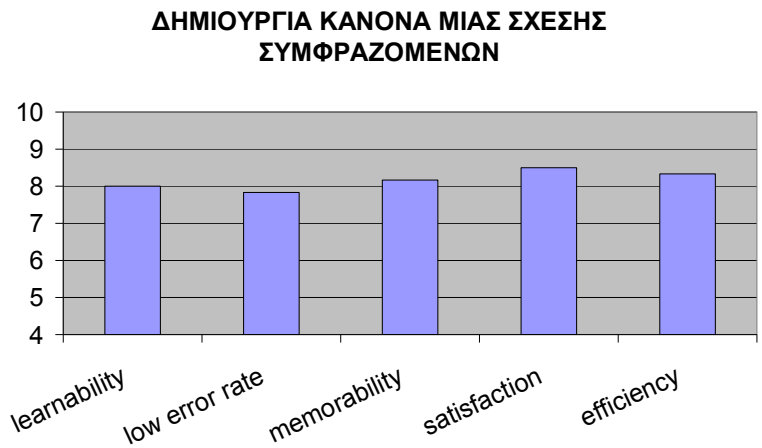
Η ιδιότητα της χρηστικότητας έχει ως επιμέρους χαρακτηριστικά τα παρακάτω: δυνατότητα εκμάθησης (learnability), αποδοτικότητα (efficiency), δυνατότητα απομνημόνευσης ενεργειών (memorability), χαμηλό ποσοστό σφαλμάτων (low error rate) και ικανοποίηση (satisfaction). Καθένα από τα παραπάνω χαρακτηριστικά συμβάλει με τον δικό του τρόπο στην ευκολία χρήσης και αποδοχής του συστήματος. Η learnability σχετίζεται με το κατά πόσο ο χρήστης δύναται να ξεκινήσει γρήγορα τη δουλειά του με το σύστημα, η efficiency σχετίζεται με το κατά πόσο ο χρήστης που έχει μάθει το σύστημα μπορεί να επιτύχει υψηλού επιπέδου παραγωγικότητα, η memorability σχετίζεται με το κατά πόσο ο χρήστης μπορεί να επιστρέψει στη χρήση του συστήματος μετά από ένα μεγάλο διάστημα αποχής, το low error rate σχετίζεται με το κατά πόσο ο χρήστης κάνει λάθη στη χρήση του συστήματος ενώ η satisfaction σχετίζεται με το κατά πόσο το σύστημα είναι ευχάριστο στη χρήση του.

Όλα αυτά τα χαρακτηριστικά μελετήθηκαν για την υπηρεσία του Context Aggregator και μάλιστα για καθεμιά από τις τρεις λειτουργίες του προκειμένου να μετρηθεί η χρηστικότητα του περιβάλλοντος διεπαφής που υλοποιήθηκε. Η τεχνική που χρησιμοποιήθηκε για τις μετρήσεις ήταν αυτή του ερωτηματολογίου. Στο ερωτηματολόγιο υπήρχαν έξι ερωτήσεις που μετρούσαν τα συγκεκριμένα χαρακτηριστικά για καθεμιά από τις τρεις λειτουργίες και επιπλέον υπήρχε η δυνατότητα καταγραφής σχολίων για διάφορα θέματα σχετικά με τη χρήση του CoWSAMI από τους χρήστες. Το ερωτηματολόγιο συμπληρώθηκε από πέντε διαφορετικούς χρήστες διαφορετικής εξοικείωσης με το σύστημα και σε γενικές γραμμές τα

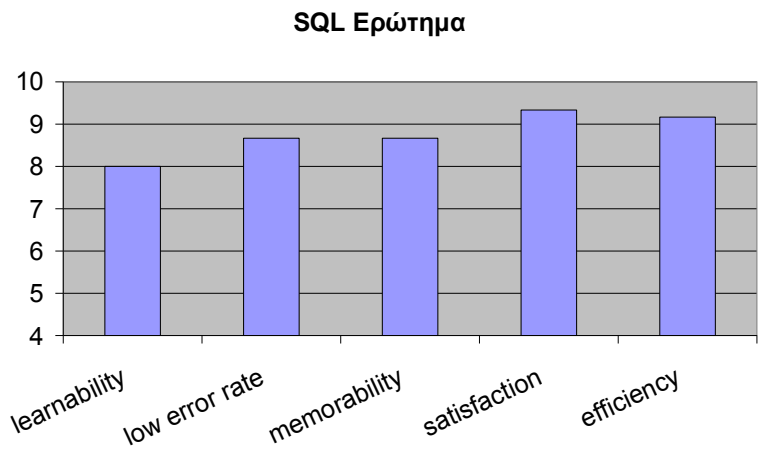
αποτελέσματα ήταν θετικά μιας και για τα πέντε χαρακτηριστικά ο μέσος όρος του βαθμού ικανοποίησης ξεπέρασε το 7 με άριστα το 10. Στα σχήματα αναπαρίστανται τα αποτελέσματα της έρευνας και για τις τρεις λειτουργίες για τα πέντε χαρακτηριστικά ξεχωριστά.



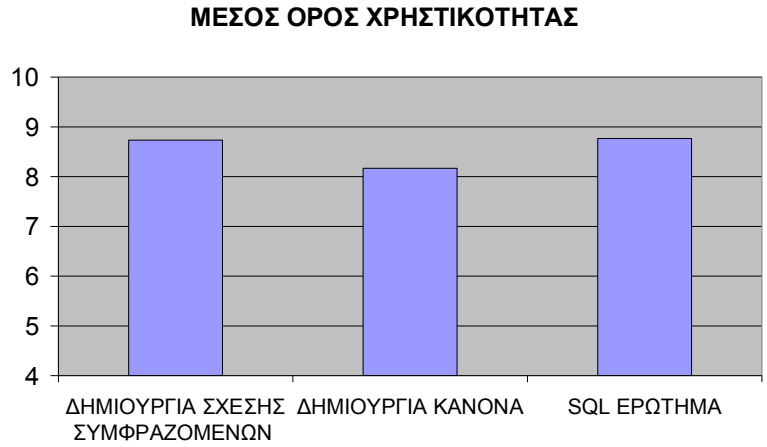
Σχήμα 4.9 Βαθμός Χρηστικότητας της Γραφικής Διεπαφής για τη Λειτουργία της Δημιουργίας Σχέσης Συμφραζομένων



Σχήμα 4.10 Βαθμός Χρηστικότητας της Γραφικής Διεπαφής για τη Λειτουργία της Δημιουργίας Κανόνα



Σχήμα 4.11 Βαθμός Χρηστικότητας της Γραφικής Διεπαφής για τη Λειτουργία της Εκτέλεσης SQL Ερωτημάτων



Σχήμα 4.12 Μέσος Όρος Βαθμού Χρηστικότητας της Γραφικής Διεπαφής για την Υπηρεσία του ContextAggregator

ΚΕΦΑΛΑΙΟ 5. ΕΓΧΕΙΡΙΔΙΟ ΕΓΚΑΤΑΣΤΑΣΗΣ ΚΑΙ ΧΡΗΣΗΣ

5.1. Εγχειρίδιο Εγκατάστασης

Στο κεφάλαιο παρατίθενται ορισμένες οδηγίες που έχουν να κάνουν με την εγκατάσταση του CoWSAMI σε ένα υπολογιστή αλλά και παραδείγματα που στόχο έχουν να διευκολύνουν ένα χρήστη που έρχεται για πρώτη φορά σε επαφή με το ενδιάμεσο λογισμικό. Υπάρχουν δύο παράγραφοι στο κεφάλαιο αυτό από τις οποίες η πρώτη αποτελεί έναν λεπτομερή οδηγό εγκατάστασης του ενδιάμεσου λογισμικού. Πιο συγκεκριμένα περιλαμβάνει όλες τις ενέργειες και ρυθμίσεις οι οποίες πρέπει να γίνουν από το χρήστη προκειμένου να λειτουργήσει σωστά το ενδιάμεσο λογισμικό και να δοθεί η δυνατότητα στο χρήστη να εκμεταλλευτεί όλες τις λειτουργίες που περιγράφηκαν στα προηγούμενα κεφάλαια. Όσον αφορά τη δεύτερη παράγραφο η οποία περιέχει ένα εγχειρίδιο χρήσης, περιλαμβάνει παραδείγματα τα οποία περιγράφουν με τη βοήθεια εικόνων την εμφάνιση και συμπεριφορά του CoWSAMI κατά τη διάρκεια χρήσης του. Για κάθε υπηρεσία που προσφέρει το CoWSAMI υπάρχουν οδηγίες και εικόνες οι οποίες κατευθύνουν με σαφή τρόπο τον χρήστη που επιθυμεί να τις χρησιμοποιήσει για πρώτη φορά.

Όπως προαναφέρθηκε στη συγκεκριμένη παράγραφο παρατίθενται ένας πλήρης οδηγός εγκατάστασης του CoWSAMI ο οποίος περιλαμβάνει όλα εκείνα τα βήματα που πρέπει να ακολουθηθούν έτσι ώστε το ενδιάμεσο λογισμικό να εγκατασταθεί και να είναι πλήρως λειτουργικό σε ένα προσωπικό υπολογιστή. Τα βήματα που περιγράφονται έχουν να κάνουν τόσο με τη ρύθμιση κάποιων παραμέτρων του λειτουργικού συστήματος όσο με την

προσθήκη κάποιων στοιχείων που αφορούν τον Web Server αλλά και με την εγκατάσταση και ρύθμιση της βάσης δεδομένων που θα πρέπει να υπάρχει στον τοπικό δίσκο του υπολογιστή κάθε χρήστη. Επομένως η διαδικασία που είναι υποχρεωτική για κάθε χρήστη που επιθυμεί να χρησιμοποιήσει το CoWSAMI είναι η ακόλουθη:

1. Εγκατάσταση του Cygwin ακολουθώντας τις οδηγίες που υπάρχουν στη διεύθυνση HTTP://WWW.CYGWIN.COM/.

2. Λήψη του Tomcat 4.1.34 από τη διεύθυνση HTTP://TOMCAT.APACHE.ORG/DOWNLOAD-41.CGI και συγκεκριμένα την έκδοση για τα Windows.

3. Αποσυμπίεση του αρχείου του Tomcat στον τοπικό δίσκο του υπολογιστή στη θέση C:\webservices και μετονομασία του φακέλου σε tomcat_folder.

4. Αποσυμπίεση του αρχείου του WSAMI στη θέση C:\webservices και μετονομασία του φακέλου σε wsami_folder.

5. Εγκατάσταση του MySQL 5.0 Community Server σύμφωνα με τις οδηγίες που παρέχονται στον ιστότοπο της MySQL (Προσοχή: να τεθεί ο κωδικός πρόσβασης του root σε '1234').

6. Δημιουργία μιας βάσης δεδομένων με όνομα wsamidb.

7. Μεταβίβαση όλων των προνομίων χρήσης της βάσης δεδομένων wsamidb στον χρήστη root εκτελώντας την παρακάτω εντολή «GRANT ALL PRIVILEGES ON *.* TO root@localhost IDENTIFIED BY '1234' WITH GRANT OPTION».

8. Αποσυμπίεση του αρχείου που αφορά στον mysql-connector-java-3.1.14 στον τοπικό δίσκο.

9. Αντιγραφή του αρχείου mysql-connector-java-3.1.14-bin.jar στο φάκελο C:\webservices\tomcat_folder\common\lib.

10. Εγκατάσταση της Java2SDK 1.4.2.

11. Εγκατάσταση του Apache-Ant-1.6.5.

12. Δημιουργία των παρακάτω μεταβλητών περιβάλλοντος των Windows:

    - ANT_HOME = C:\apache-ant-1.6.5-bin\ant

- CATALINA_HOME = C:\webservices\tomcat_folder
- JAVA_HOME = C:\j2sdk1.4.2
- CLASSPATH=.;C:\j2sdk1.4.2\jre\lib\ext\mysql-connector-java-3.1.14-bin.jar.

13. Αποσυμπίεση του OpenSlp Java Api στο φάκελο C:\j2sdk1.4.2.
14. Εκτέλεση των ενεργειών που περιγράφονται στην παράγραφο εγκατάστασης του WSAMI στο WSAMI User Guide.
15. Προσθήκη των παρακάτω δεδομένων στο αρχείο C:\webservices\ tomcat_folder\conf\server.xml και στις θέσεις που ακριβώς φαίνεται στο παράδειγμα:

```
<Context path="/wsami" docBase="wsami" debug="1" crossContext="true">
    . . . . . .
</Context>
<Context path="/wsami" docBase="wsami" debug="1" crossContext="true">
    . . . . . .
</Context>
```

```
<Resource name="jdbc/askisidb" auth="Container"
        type="javax.sql.DataSource"/>
 <ResourceParams name="jdbc/askisidb">
    <parameter>
    <name>factory</name>
    <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
    </parameter>
                                    <parameter>
                                      <name>maxActive</name>
                                      <value>10</value>
                                    </parameter>
                                    <parameter>
```

```
                                        <name>maxIdle</name>
                                        <value>5</value>
                                      </parameter>
                                      <parameter>
                                        <name>validationQuery</name>
                                        <value>SELECT 1</value>
                                      </parameter>
                                      <parameter>
                                        <name>testOnBorrow</name>
                                        <value>true</value>
                                      </parameter>
                                      <parameter>
                                        <name>testWhileIdle</name>
                                        <value>true</value>
                                      </parameter>
                                      <parameter>

<name>timeBetweenEvictionRunsMillis</name>
                                        <value>10000</value>
                                      </parameter>
                                      <parameter>
                                        <name>minEvictableIdleTimeMillis</name>
                                        <value>60000</value>
                                      </parameter>
        <parameter><name>username</name><value>kostas</value></parameter>
        <parameter><name>password</name><value>costas23</value></parameter>
        <parameter><name>driverClassName</name>
          <value>com.mysql.jdbc.Driver</value></parameter>
        <parameter><name>url</name>
          <value>jdbc:mysql://localhost:3306/wsamidb</value></parameter>
      </ResourceParams>
```

16. Ρύθμιση του αρχείου C:\webservices\tomcat_folder\webapps\wsami\WEB-INF\web.xml θέτοντας ως wsamibindir τον φάκελο C:\webservices\ wsami_folder\ dist\tomcat\bin.

17. Ρύθμιση του αρχείου C:\webservices\wsami_folder\services\ commonWindows.xml θέτοντας ως wsamidir τον φάκελο C:\webservices\wsami_folder\dist\tomcat.

18. Αντιγραφή του jar αρχείου που βρίσκεται στο φάκελο του java-connector στον φάκελο C:\j2sdk1.4.2\jre\lib\ext.

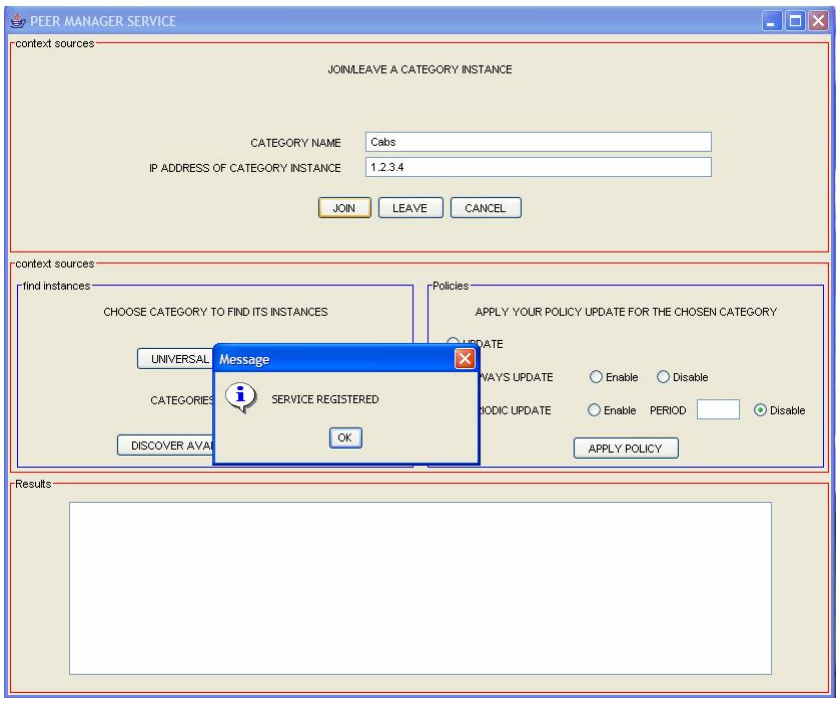
19. Ρύθμιση του αρχείου C:\webservices\wsami_folder\build.xml θέτοντας ως openslp.dir.Windows τον φάκελο C:\j2sdk1.4.2\slpapi.

5.2. Εγχειρίδιο Χρήσης

Στη συγκεκριμένη παράγραφο θα γίνει προσπάθεια να περιγραφούν με παραστατικό τρόπο οι ενέργειες που πρέπει να κάνει ένας χρήστης προκειμένου να αξιοποιήσει πλήρως τις δυνατότητες που του δίνει το CoWSAMI. Για όλες τις δυνατές λειτουργίες που παρέχονται υπάρχει λεπτομερής περιγραφή των ενεργειών του χρήστη αλλά και γραφική απεικόνιση του περιβάλλοντος που χρησιμοποιεί ο χρήστης μαζί με τις καταστάσεις που μπορεί να βρεθεί αυτό ανάλογα με τη φάση λειτουργίας του ενδιάμεσου λογισμικού.

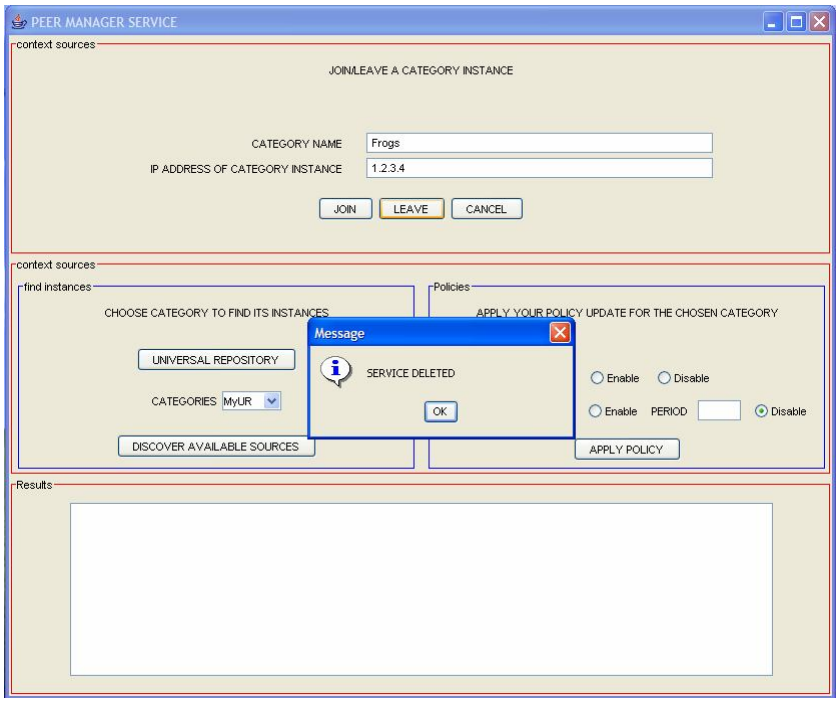
Ξεκινώντας με την υπηρεσία του PeerManager θα γίνει αναφορά στην λειτουργία που αφορά στην εισαγωγή μιας υπηρεσίας στο περιβάλλον. Για να επιτευχθεί αυτό ο χρήστης θα πρέπει να δηλώσει όπως φαίνεται και στο σχήμα 5.1 την κατηγορία της υπηρεσίας καθώς επίσης και την διεύθυνση του κόμβου/συσκευής στην οποία είναι διαθέσιμη. Στη συνέχεια με το πάτημα του κουμπιού JOIN η συγκεκριμένη υπηρεσία προστίθεται στη βάση δεδομένων όλων των συσκευών που υπάρχουν στο δίκτυο και έτσι εμφανίζεται και το μήνυμα που επιβεβαιώνει την πραγματοποίηση της προσθήκης.

Με παρόμοιο τρόπο επιτυγχάνεται και η αποχώρηση/διαγραφή μιας υπηρεσίας από το περιβάλλον. Ο χρήστης δηλώνει όπως και πριν την κατηγορία της υπηρεσίας και τη διεύθυνση του κόμβου στην οποία είναι διαθέσιμη και πατά το κουμπί LEAVE για να διαγραφεί η υπηρεσία. Μετά την επιτυχή διαγραφή της υπηρεσίας από τις βάσεις δεομένων των υπολοίπων χρηστών εμφανίζεται το ανάλογο μήνυμα όπως παρουσιάζεται στο σχήμα 5.2.

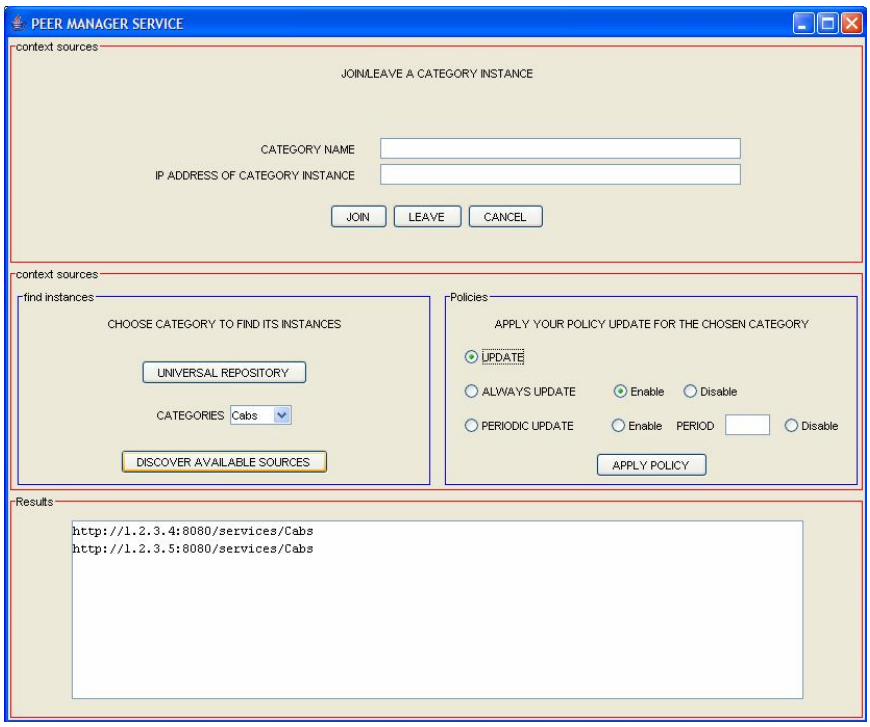


Σχήμα 5.1 Παράθυρο Επιτυχούς Εισόδου Υπηρεσίας στο Περιβάλλον

Όσον αφορά την εφαρμογή των πολιτικών ενημέρωσης που επιθυμεί ο χρήστης για μια συγκεκριμένη κατηγορία υπηρεσιών η διαδικασία είναι απλή και παρουσιάζεται γραφικά στα τρία επόμενα σχήματα. Στο σχήμα 5.3 περιγράφεται ο τρόπος με τον οποίο ο χρήστης μπορεί να ζητήσει την άμεση ενημέρωση για την ύπαρξη υπηρεσιών μιας συγκεκριμένης κατηγορίας. Όπως φαίνεται ο χρήστης το μόνο που πρέπει να κάνει είναι να επιλέξει από τη λίστα που βρίσκεται στο πλαίσιο με όνομα find instances την κατηγορία για την οποία επιθυμεί να ενημερωθεί και στη συνέχεια να σημειώσει την επιλογή UPDATE στο πλαίσιο με όνομα Policies. Πατώντας στη συνέχεια το πλήκτρο APPLY εμφανίζονται οι διευθύνσεις των υπηρεσιών της κατηγορίας, που είναι διαθέσιμες εκείνη τη στιγμή στο περιβάλλον, στο πλαίσιο των αποτελεσμάτων που βρίσκεται στο κάτω μέρος της οθόνης.

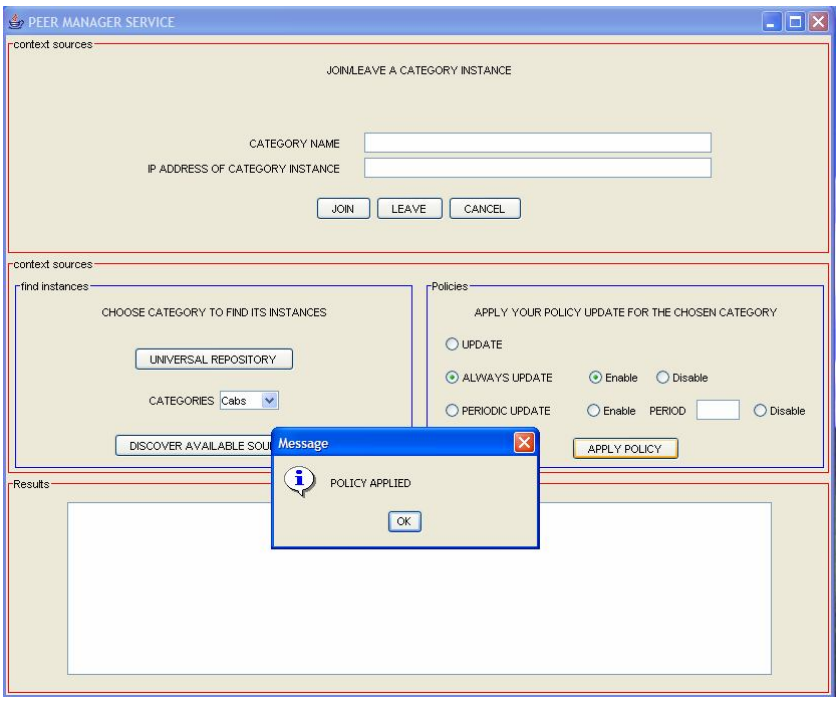


Σχήμα 5.2 Παράθυρο Επιτυχούς Εξόδου Υπηρεσίας από το Περιβάλλον



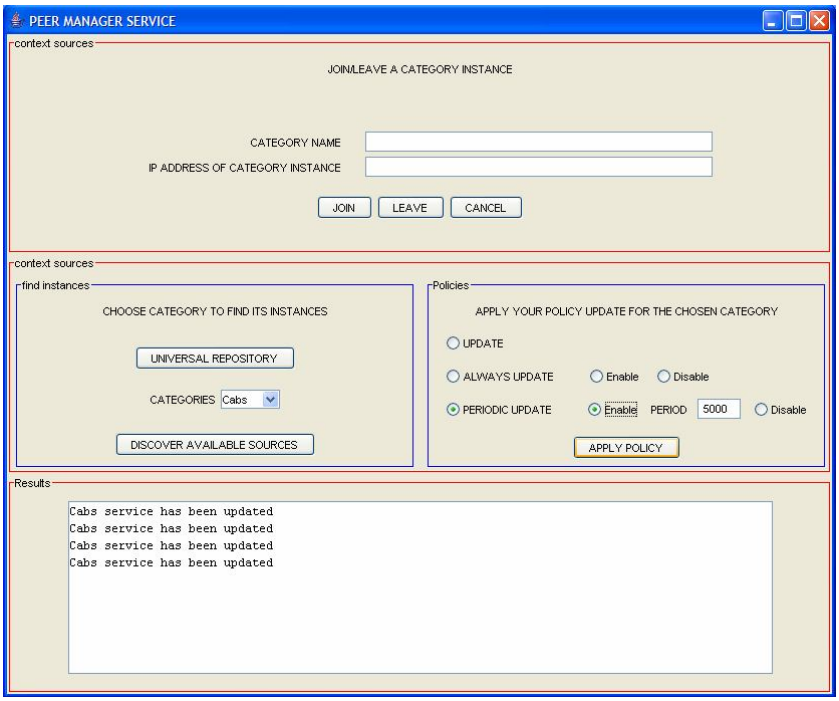
Σχήμα 5.3 Ενημέρωση για Διαθέσιμους Παρόχους Υπηρεσιών μιας Κατηγορίας

Ακολουθώντας τα ίδια βήματα μπορεί κάποιος να εφαρμόσει την πολιτική της συνεχούς ενημέρωσης για κάποια κατηγορία υπηρεσιών. Στην περίπτωση αυτή αφού ο χρήστης δηλώσει την κατηγορία στην οποία θέλει να εφαρμόσει την πολιτική, σημειώνει την επιλογή ALWAYS UPDATE και μια από τις επιλογές Enable ή Disable ανάλογα με το αν θέλει να εφαρμόσει τη πολιτική της συνεχούς ενημέρωσης ή να την καταργήσει. Σε κάθε περίπτωση εμφανίζεται ένα μήνυμα που πληροφορεί τον χρήστη για την υλοποίηση της επιθυμίας σαν αυτό του σχήματος 5.4.



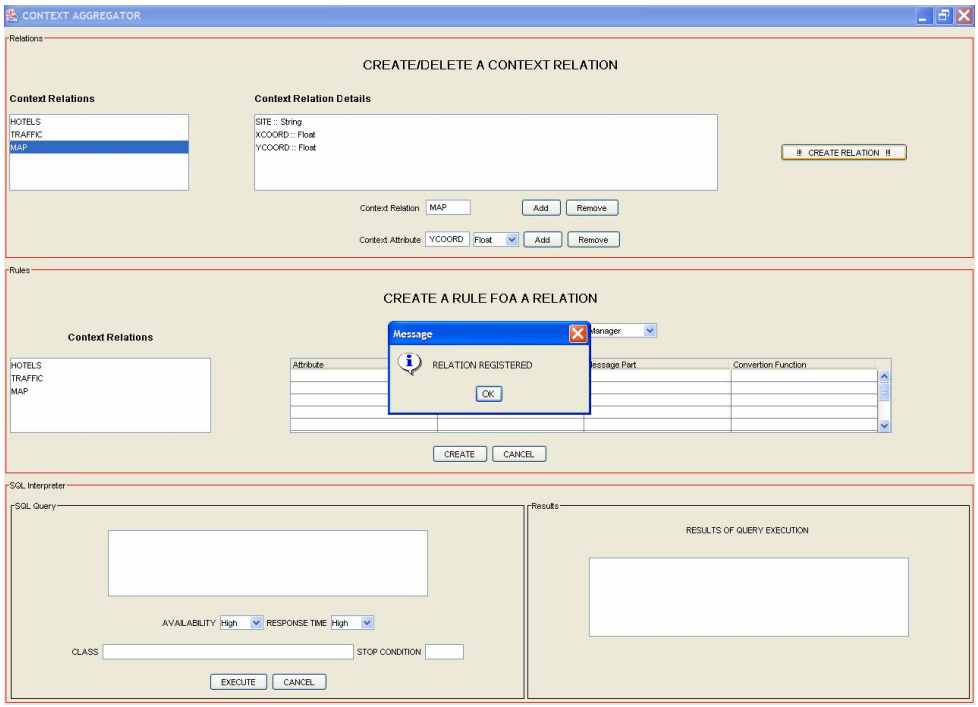
Σχήμα 5.4 Εφαρμογή της Πολιτικής Συνεχούς Ενημέρωσης

Για την εφαρμογή της τελευταίας από τις τρεις πολιτικές ενημέρωσης που μπορεί να εφαρμόσει ο χρήστης και η οποία είναι η περιοδική ενημέρωση θα πρέπει επιπλέον να δηλωθεί και η χρονική περίοδος που θα μεσολαβεί ανάμεσα σε δυο διαδοχικές ενημερώσεις για μια συγκεκριμένη κατηγορία υπηρεσιών. Από το σχήμα 5.5 γίνεται αντιληπτό ότι είναι απαραίτητο στην περίπτωση εφαρμογής της περιοδικής ενημέρωσης να δηλωθεί από το χρήστη η περίοδος, η οποία ορίζεται στην κλίμακα των msec. Κατά τη διάρκεια της περιοδικής ενημέρωσης μιας κατηγορίας εμφανίζεται ενημερωτικό κείμενο στο πλαίσιο των αποτελεσμάτων στο κάτω μέρος της οθόνης το οποίο πληροφορεί το χρήστη για την εκάστοτε επιτυχημένη ενημέρωση της κατηγορίας.



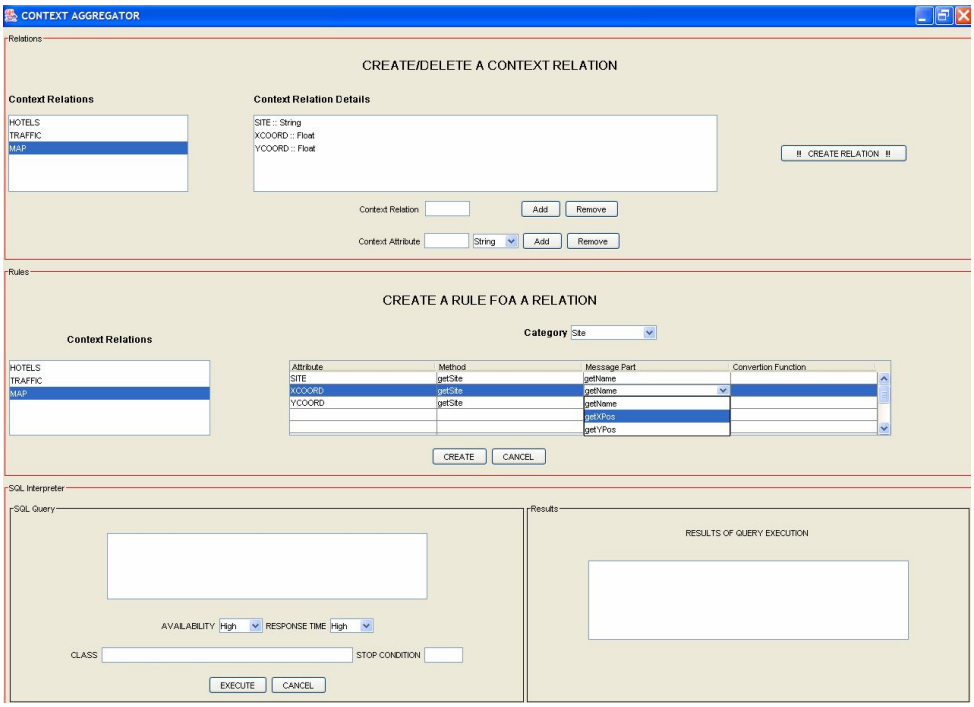
Σχήμα 5.5 Εφαρμογής της Πολιτικής της Περιοδικής Ενημέρωσης

Όσο για τη λειτουργία του ContextAggregator το πρώτο παράδειγμα που παρουσιάζεται έχει να κάνει με τη δημιουργία μιας σχέσης συμφραζομένων. Στο σχήμα 5.6 είναι εμφανές ότι ο χρήστης το πρώτο πράγμα που πρέπει να κάνει είναι να δηλώσει το όνομα της σχέσης συμφραζομένων που επιθυμεί να δημιουργήσει στο πλαίσιο κειμένου δίπλα από τη φράση Context Relation και στη συνέχεια να πιέσει το πλήκτρο Add που βρίσκεται δίπλα. Αμέσως εμφανίζεται στο αριστερό πλαίσιο το όνομα της σχέσης κάτω από τα ονόματα των ήδη υπαρχόντων σχέσεων συμφραζομένων. Έπειτα για κάθε ιδιότητα της σχέσης ο χρήστης πληκτρολογεί το όνομα και τον τύπο της στα πλαίσια δίπλα από τη φράση Context Attribute και πιέζει το κουμπί Add. Κάθε ιδιότητα που προστίθεται με αυτόν τον τρόπο εμφανίζεται στο πλαίσιο Context Relation Details όπως φαίνεται στο σχήμα. Προκειμένου όμως να δημιουργηθεί η σχέση και να καταχωρηθεί στη βάση δεδομένων θα πρέπει ο χρήστη να πατήσει το κουμπί CREATE RELATION και να περιμένει για το μήνυμα επιτυχής καταχώρησης της σχέσης.

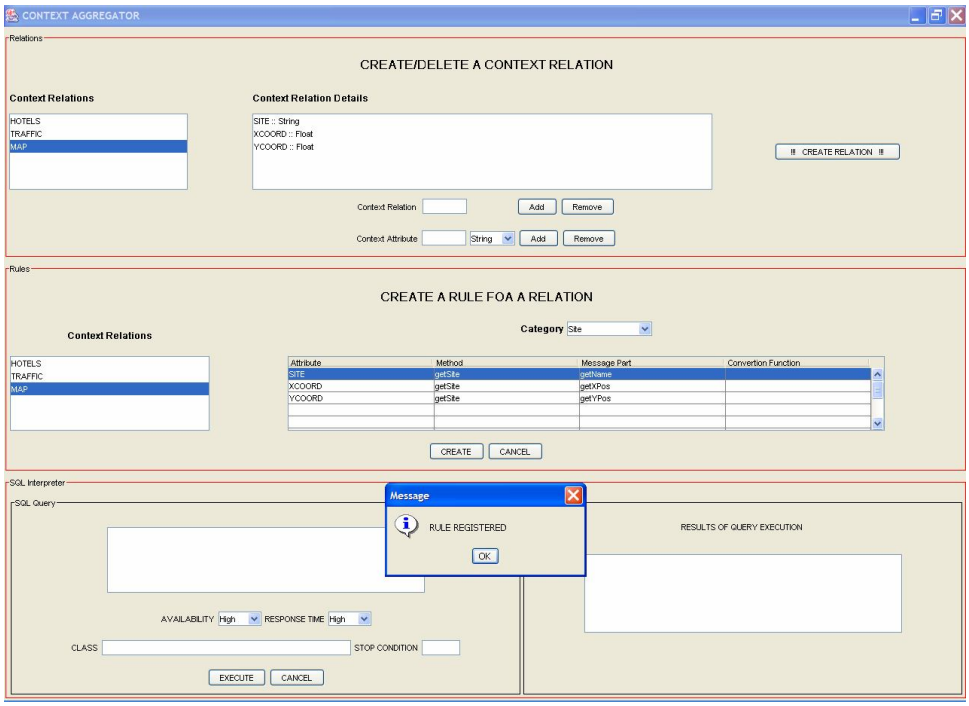


Σχήμα 5.6 Δημιουργία Σχέσης Συμφραζομένων

Η επόμενη λειτουργία που αναπαρίσταται πιο κάτω από τα σχήματα 5.7 και 5.8 είναι αυτή της δημιουργίας κανόνα για μια σχέση. Όπως φαίνεται από το πρώτο από τα δύο σχήματα ο χρήστης πρώτα από όλα πρέπει να επιλέξει την κατηγορία των υπηρεσιών οι μέθοδοι των οποίων θα συμπληρώνουν τις ιδιότητες της σχέσης συμφραζομένων που θα δηλωθεί εν συνεχεία στον κανόνα. Αφού λοιπόν επιλεγεί από την αντίστοιχη λίστα η κατηγορία υπηρεσίας, αυτό που ακολουθεί είναι η επιλογή της σχέσης συμφραζομένων στην οποία θα ανήκει ο προς δημιουργία κανόνας. Με την επιλογή της σχέσης συμφραζομένων από το πλαίσιο που βρίσκεται στο αριστερό μέρος της οθόνης με το όνομα Context Relations, εμφανίζονται στην πρώτη στήλη του πίνακα που βρίσκεται στο μέσο της οθόνης οι ιδιότητες της επιλεγμένης σχέσης. Αυτό που απομένει στο χρήστη είναι να δηλώσει για κάθε ιδιότητα της σχέσης που βρίσκεται σε διαφορετική γραμμή του πίνακα τη μέθοδο και το τμήμα του μηνύματος επιστρεφόμενης τιμής της μεθόδου το οποίο θα χρησιμοποιηθεί για την συμπλήρωση της (Σχήμα 5.7).



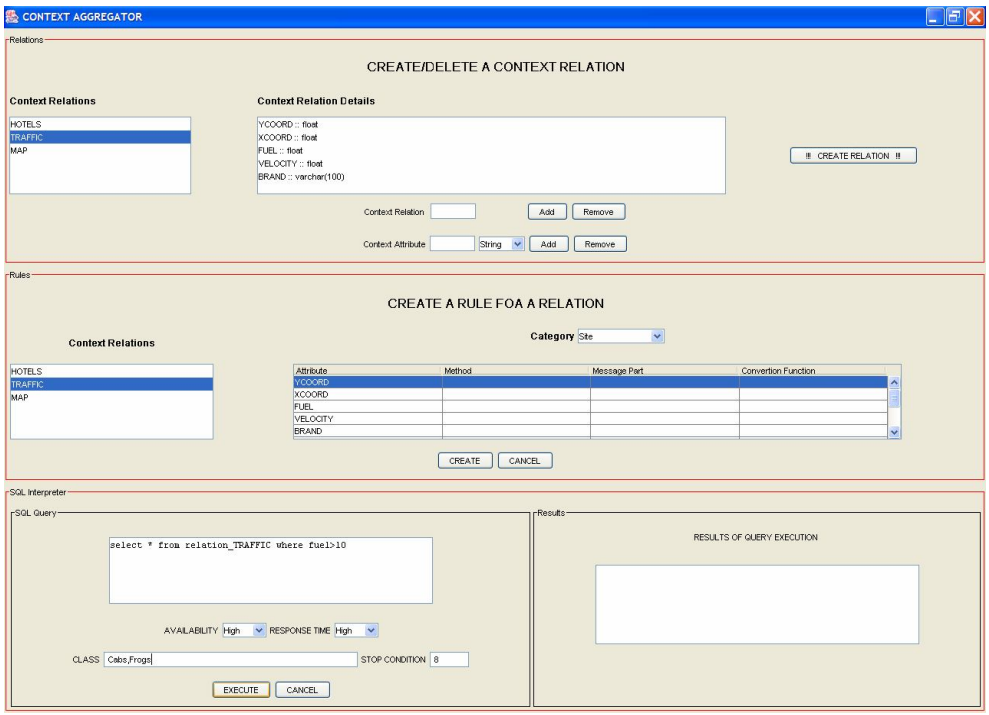
Σχήμα 5.7 Δημιουργία Κανόνα μιας Σχέσης Συμφραζομένων (α)



Σχήμα 5.8 Δημιουργία Κανόνα μιας Σχέσης Συμφραζομένων (β)

Αφού επιλεγούν από τις λίστες οι μέθοδοι και τα μηνύματα επιστρεφόμενης τιμής για κάθε ιδιότητα, ο χρήστης μπορεί να δηλώσει αν θα χρησιμοποιήσει συνάρτησης μετατροπής για

κάθε ιδιότητα γράφοντας το όνομα της συνάρτησης στην τελευταία στήλη του πίνακα που έχει το όνομα Convertion Function. Στην περίπτωση που η στήλη αυτή παραμείνει κενή σημαίνει πως δεν θα χρησιμοποιηθεί καμία συνάρτηση μετατροπής για καμία ιδιότητα της σχέσης. Έχοντας ακολουθήσει όλα αυτά τα βήματα το μόνο που απομένει στον χρήστη είναι να πιέσει το πλήκτρο CREATE και να περιμένει το αντίστοιχο ενημερωτικό μήνυμα επιτυχής δημιουργίας του κανόνα όπως αυτό του σχήματος 5.8.

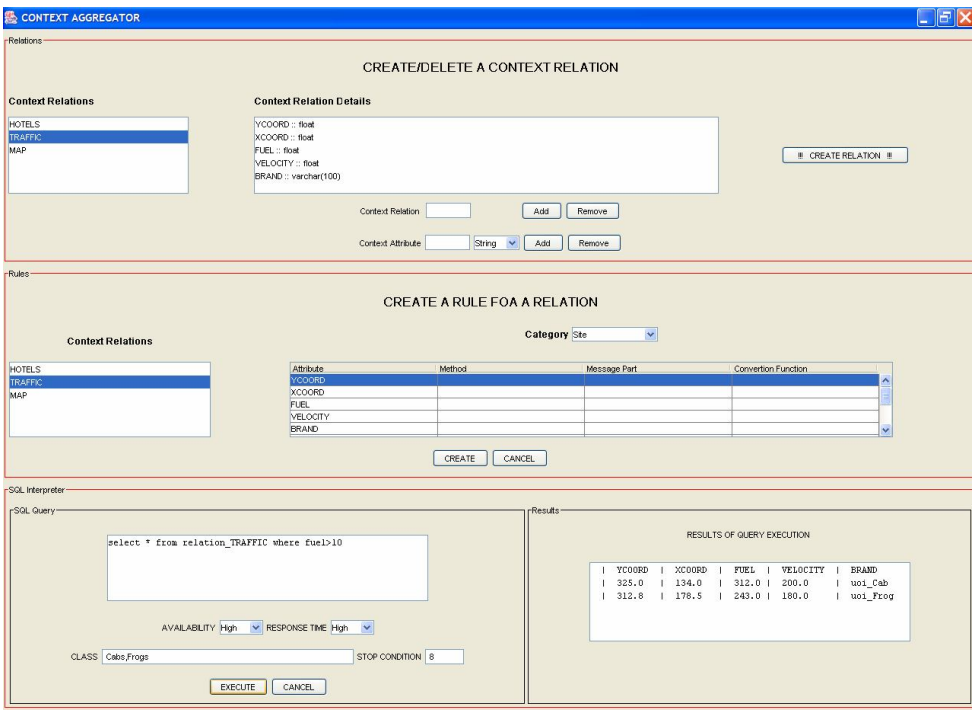


Σχήμα 5.9 Εκτέλεση SQL Ερωτήματος

Τελειώνοντας με τις δυνατότητες που παρέχει το γραφικό περιβάλλον του ContextAggregator θα παρουσιαστεί η διαδικασία υποβολής SQL ερωτήσεων προς τη βάση δεδομένων στην οποία καταχωρούνται τα συμφραζόμενα του περιβάλλοντος. Τα ερωτήματα υποβάλλονται ακολουθώντας το συντακτικό της SQL γλώσσας και αφορούν τις σχέσεις που αναπαριστούν σχέσεις συμφραζομένων που έχει ήδη δηλώσει ο χρήστης και οι οποίες έχουν συμπληρωθεί ακολουθώντας τους κανόνες που υπάρχουν για καθεμιά. Η μοναδική παραδοχή που έχει γίνει από το σύστημα και πρέπει να την τηρεί ο χρήστης σε αυτή τη φάση χρήσης του συστήματος είναι ότι τα ονόματα των σχέσεων που αναφέρονται σε σχέσεις συμφραζομένων αποτελούνται από το πρόθεμα relation_ ακολουθούμενο από το όνομα της σχέσης. Επομένως

ο χρήστης είναι απαραίτητο για να αναφερθεί σε ένα τέτοιο πίνακα να προσθέσει πριν το όνομα της σχέσης συμφραζομένων και το συγκεκριμένο πρόθεμα (Σχήμα 5.9).

Επιπλέον όλων αυτών πρέπει να συμπληρωθούν τα πεδία που αφορούν τη διαθεσιμότητα και το χρόνο απόκρισης των υπηρεσιών που επιθυμεί ο χρήστης να συνεισφέρουν στην συμπλήρωση των σχέσεων τις οποίες χρησιμοποιεί στο ερώτημα του καθώς επίσης να σημειωθούν ρητά ποιες κατηγορίες υπηρεσιών επιθυμεί να συνεισφέρουν ο χρήστης (πεδίο CLASS) ενώ τέλος πρέπει στο πεδίο STOP CONDITION να δοθεί ο αριθμός ο οποίος θα δηλώνει και το μέγιστο αριθμό κόμβων που επιθυμεί ο χρήστης να ερωτηθούν για τη συμπλήρωση των σχέσεων συμφραζομένων που συμμετέχουν στην ερώτηση του. Αφού όλα αυτά πραγματοποιηθούν και πατηθεί το κουμπί EXECUTE εμφανίζονται τα αποτελέσματα της ερώτησης στο δεξί πλαίσιο με όνομα RESULTS OF QUERY EXECUTION. Παράδειγμα τέτοιων αποτελεσμάτων απεικονίζεται στην εικόνα 5.10 όπου φαίνονται τα αποτελέσματα που αφορούν σε ερώτηση που συμμετέχει η σχέση συμφραζομένων με το όνομα TRAFFIC.



Σχήμα 5.10 Αποτελέσματα Εκτέλεσης SQL Ερωτήματος

ΚΕΦΑΛΑΙΟ 6. ΣΥΜΠΕΡΑΣΜΑΤΑ

6.1. Συμπεράσματα

Στη συγκεκριμένη εργασία παρουσιάσαμε το CoWSAMI, ένα ενδιάμεσο λογισμικό που δίνει τη δυνατότητα ανάπτυξης εφαρμογών οι οποίες είναι ενήμερες συμφραζομένων και οι οποίες λειτουργούν σε ένα περιβάλλον πανταχού παρόντος υπολογισμού. Κύριο χαρακτηριστικό αυτού του ενδιάμεσου λογισμικού είναι η δυνατότητα ανάπτυξης εφαρμογών με ενημερότητα συμφραζομένων σε περιβάλλοντα όπου οι πάροχοι συμφραζομένων εισέρχονται και εξέρχονται από το περιβάλλον συνεχώς και επιπλέον έχουν περιορισμένους υπολογιστικούς και επικοινωνιακούς πόρους (π.χ. μπαταρία, μνήμη, αποθηκευτικούς χώρους).

Σε αντίθεση με τις προηγούμενες προσεγγίσεις το CoWSAMI αντιμετωπίζει τόσο το πρόβλημα των περιορισμένων πόρων των φορητών συσκευών όσο και της συνεχής μεταβολής της διαθεσιμότητας τους σε ένα τέτοιο περιβάλλον. Παρέχει τη δυνατότητα δυναμικού τρόπου εντοπισμού των παρόχων συμφραζομένων δίνοντας έτσι λύση στο φαινόμενο της συνεχής μεταβολής της διαθεσιμότητας των συσκευών που υπάρχουν σε ένα τέτοιο περιβάλλον. Επίσης στο CoWSAMI δεν χρησιμοποιείται κάποιος κεντρικός εξυπηρετητής όπου αποθηκεύεται η πληροφορία για τον εντοπισμό των παρόχων. Επιπλέον το CoWSAMI δεν επιβάλλει περιορισμούς στους παρόχους για προσφορά των συμφραζομένων μέσω μιας καθολικά αποδεκτής διεπαφής αλλά τους δίνει τη δυνατότητα να διαθέτουν οποιουδήποτε είδους διεπαφή επιθυμούν για τα συμφραζόμενα αρκεί να ακολουθεί τα πρότυπα των υπηρεσιών Διαδικτύου.

Μετά την υλοποίηση του συγκεκριμένου λογισμικού πραγματοποιήσαμε πειραματικές μετρήσεις κατά τη φάση λειτουργίας του στις οποίες παρατηρήθηκε ότι ο χρόνος εκτέλεσης των λειτουργιών των υπηρεσιών του αυξάνεται γραμμικά σε σχέση με τον αριθμό των παρόχων και καταναλωτών συμφραζομένων που υπάρχουν στο περιβάλλον. Επιπλέον, διενεργήσαμε μια μελέτη σχετικά με τη χρηστικότητα του ενδιάμεσου λογισμικού η οποία κατά κύριο λόγο στηρίζεται στη χρήση του γραφικού περιβάλλοντος διεπαφής, της οποίας τα αποτελέσματα της ήταν σε γενικές γραμμές θετικά.

6.2. Μελλοντική Δουλειά

Μελλοντικός στόχος μας είναι η βελτίωση της απόδοσης του COWSAMI. Παρά το γεγονός ότι οι μετρήσεις που αφορούσαν το χρόνο απόκρισης των υπηρεσιών του ενδιάμεσου λογισμικού χαρακτηρίζονται αρκετά ικανοποιητικές, εντούτοις η μείωση του χρόνου αυτού συμβάλλει στη μεγαλύτερη αποδοχή του CoWSAMI από τους χρήστες. Μια επιλογή για την βελτίωση αυτής της απόδοσης είναι η πολυνηματική υλοποίηση των λειτουργιών του PeerManager και του ContextAggregator τρόπο έχοντας όμως πάντα υπόψη τους περιορισμένους πόρους των συσκευών στις οποίες εγκαθίστανται το CoWSAMI. Πέρα όμως από το θέμα της απόδοσης ένα ακόμη βήμα προς τη μελλοντική βελτίωση του CoWSAMI αποτελεί η δυνατότητα χρήσης όλων των δυνατοτήτων που παρέχει η γλώσσα SQL-P [13]. Μερικές από τις επεκτάσεις της SQL-P υλοποιήθηκαν όπως για παράδειγμα η δυνατότητα επιβολής περιορισμών από τους χρήστες που έχουν να κάνουν με τη διαθεσιμότητα και το χρόνο απόκρισης των υπηρεσιών, την class των κατηγοριών αλλά και την συνθήκη τερματισμού, αλλά η πλήρης υλοποίηση των επεκτάσεων της SQL-P είναι ένα επιπλέον στοιχείο αύξησης της ικανοποίησης των χρηστών του CoWSAMI.

ΑΝΑΦΟΡΕΣ

[1] Andreas Holzinger, “Usability engineering methods for software developers”, Communications of the ACM, V.48, P. 71-74, January 2005

[2] Brown, P. J. “The stick-e document: A framework for creating context-aware applications.”, In Proceedings of the Electronic Publishing, Palo Alto, pages 259–272, 1996

[3] Baldauf, M., Dustdar, S., and Rosenberg, F. “A Survey on Context Aware Systems”, Journal of Ad-Hoc and Ubiquitous Computing, 2005

[4] Biegel, G. and Cahill, V. “A Framework for Developing Mobile, Context-aware Applications”, In Proceedings of 2nd IEEE Conference on Pervasive Computing and Communications (Percom’04), 2004

[5] Chen, H., Finit, T., and Joshi, A. “An Ontology for Context-Aware Pervasive Computing Systems”, Knowledge Engineering Review 18, 2003

[6] Dionisis Athanasopoulos, Apostolos Zarras, Valérie Issarny, Evaggelia Pitoura, Panos Vassiliadis, "CoWSAMI: Interface-Aware Context Gathering in Ambient Intelligence Environments", In Pervasive and Mobile Computing Journal, 2007.

[7] Dey, A. K. and Abowd, G. D. “Towards a better understanding of context and context-awareness.”, In Proceedings of the Workshop on the What, Who, Where, When and How of Context-Awareness”, New York, ACM Press, 2000

[8] Dey, A. K. and Abowd, G. D. 1999. “A Context-based Infrastructure for Smart Environments”, In Proceedings of the International Workshop on Managing Interactions in Smart Environments (MANSE ’99), 114–128, 1999

[9] Fahy, P. and Clarke, S. “CASS - Middleware for Mobile Context-Aware Applications”, In Proceedings of the 2nd ACM SIGMOBILE International Conference on Mobile Systems, Applications and Services (MobiSys’04), 2004

[10] Gu, T., Pung, H.-K., and Zhang, D.-Q. 2005. “A Service-Oriented Middleware for Context-Aware Services”, Journal of Network and Computer Applications, 28, 1–18, 2005

[11] Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G., and Altmann, J. “Context-Awareness on Mobile Devices - the Hydrogen Approach”, In Proceedings of 36th IEEE”, Hawaii International Conference on System Sciences (HICSS’02), 292–302, 2002

[12] M. Kumar, B. A. Shirazi, S. K. Das, B. Y. Sung, D. Levine, and M. Singhal. "PICO :A Middleware Framework for Pervasive Computing", IEEE Pervasive Computing, 2(3):72–79, 2003.

[13] N. Folinas, P. Vassiliadis, E. Pitoura, E. Papapetrou, A. Zarras. CONTEXT-AWARE QUERY PROCESSING IN AD-HOC ENVIRONMENTS OF PEERS. Journal of Electronic Commerce in Organization, vol. 6, no. 1, pp. 38-62, IGI Global.

[14] Roman, M., Hess, C. K., Cerqueira, R., Ranganathan, A., Campbell, R. H., and Nahrstedt, K. "Gaia: A Middleware Infrastructure to Enable Active Spaces", IEEE Pervasive Computing 1, 4, 74–83,2002

[15] Ryan, N., Pascoe, J. and Morse, D. "Enhanced reality fieldwork: The context-aware archaeological assistent." Computer Applications in Archaeology, 1997

[16] Schilit, B. and Theimer, M. "Disseminating active map information to mobile hosts", IEEE Network, 8(5):22–32, 1994.

[17] WWW.OPENSLP.ORG

ΠΑΡΑΡΤΗΜΑ

Δημιουργήστε μια σχέση(relation) με όσα στοιχεία(attributes) επιθυμείτε και δηλώνοντας κάθε φορα τον τύπο τους.

1. Πόσο εύκολο ήταν να δημιουργήσετε την ζητούμενη σχέση;

   Πολύ δύσκολο 1 2 3 4 5 6 7 8 9 10 Εύκολο

2. Πόσα λάθη κάνατε εως ότου καταφέρετε να δημιουργήσετε τη σχέση;

   Πάρα πολλά 1 2 3 4 5 6 7 8 9 10 Κανένα

3. Θεωρείτε ότι θα μπορούσατε μετά από μικρο χρονικό διάστημα να δημιουργήσετε μια άλλη σχέση;

   Πολύ δύσκολα 1 2 3 4 5 6 7 8 9 10 Εύκολα

4. Πως χαρακτηρίζετε τη διεπαφή για την λειτουργία **"δημιουργία σχέσης"** ως προς την φιλικότητα προς το χρήστη;

   Καθόλου φιλική 1 2 3 4 5 6 7 8 9 10 Πολύ φιλική

5. Πως θα χαρακτηρίζατε τα βοηθητικά μηνύματα της εφαρμογής ως προς τη χρησιμότητα τους;

   Καθόλου χρήσιμα 1 2 3 4 5 6 7 8 9 10 Πολύ χρήσιμα

6. Θεωρείτε ότι η διεπαφή είναι σαφής προκειμένου να οδηγήσει το χρήστη στο στόχο του;

   Πολύ ασαφής 1 2 3 4 5 6 7 8 9 10 Σαφέστατη

7. Προσθέστε οποιοδήποτε σχόλιο σχετικά με τη χρήση της εφαρμογής στη συγκεκριμένη λειτουργία.

   .....
   .....
   .....
   .....

Σχήμα Π.1 Ερωτηματολόγιο για τη Μέτρηση της Χρηστικότητας της Γραφικής Διεπαφής (Λειτουργία Δημιουργίας Σχέσης Συμφραζομένων)

Δημιουργήστε ένα κανόνα(rule) για τη σχέση(relation) που δηλώσατε πιο πάνω.Για κάθε στοιχείο της σχέσης δηλώστε ποια μέθοδος πρέπει να κληθεί και ποιο κομμάτι της απάντησής της θα δώσει την τιμη στο συγκεκριμένο στοιχείο. Δηλώστε προειρετικά αν πρέπει να περάσει το κομμάτι της απάντησης της μεθόδου από μια υπηρεσία μετατροπής και ποια.

8. Πόσο εύκολο ήταν να δημιουργήσετε την ζητούμενη σχέση;

   Πολύ δύσκολο 1 2 3 4 5 6 7 8 9 10 Εύκολο

9. Πόσα λάθη κάνατε εως ότου καταφέρετε να δημιουργήσετε τη σχέση;

   Πάρα πολλά 1 2 3 4 5 6 7 8 9 10 Κανένα

10. Θεωρείτε ότι θα μπορούσατε μετά από μικρο χρονικό διάστημα να δημιουργήσετε μια άλλη σχέση;

    Πολύ δύσκολα 1 2 3 4 5 6 7 8 9 10 Εύκολα

11. Πως χαρακτηρίζετε τη διεπαφή για την λειτουργία **"δημιουργία σχέσης"** ως προς την φιλικότητα προς το χρήστη;

    Καθόλου φιλική 1 2 3 4 5 6 7 8 9 10 Πολύ φιλική

12. Πως θα χαρακτηρίζατε τα βοηθητικά μηνύματα της εφαρμογής ως προς τη χρησιμότητα τους;

    Καθόλου χρήσιμα 1 2 3 4 5 6 7 8 9 10 Πολύ χρήσιμα

13. Θεωρείτε ότι η διεπαφή είναι σαφής προκειμένου να οδηγήσει το χρήστη στο στόχο του;

    Πολύ ασαφής 1 2 3 4 5 6 7 8 9 10 Σαφέστατη

14. Προσθέστε οποιοδήποτε σχόλιο σχετικά με τη χρήση της εφαρμογής στη συγκεκριμένη λειτουργία.

    .....
    .....
    .....
    .....

Σχήμα Π.2 Ερωτηματολόγιο για τη Μέτρηση της Χρηστικότητας της Γραφικής Διεπαφής (Λειτουργία Δημιουργίας Κανόνα)

Γράψτε και εκτελέστε ένα SQL ερώτημα για τον πίνακα της σχέσης που δημιουργήσατε αρχικά.

15. Πόσο εύκολο ήταν να δημιουργήσετε την ζητούμενη σχέση;

Πολύ δύσκολο 1 2 3 4 5 6 7 8 9 10 Εύκολο

16. Πόσα λάθη κάνατε εως ότου καταφέρετε να δημιουργήσετε τη σχέση;

Πάρα πολλά 1 2 3 4 5 6 7 8 9 10 Κανένα

17. Θεωρείτε ότι θα μπορούσατε μετά από μικρο χρονικό διάστημα να δημιουργήσετε μια άλλη σχέση;

Πολύ δύσκολα 1 2 3 4 5 6 7 8 9 10 Εύκολα

18. Πως χαρακτηρίζετε τη διεπαφή για την λειτουργία **"δημιουργία σχέσης"** ως προς την φιλικότητα προς το χρήστη;

Καθόλου φιλική 1 2 3 4 5 6 7 8 9 10 Πολύ φιλική

19. Πως θα χαρακτηρίζατε τα βοηθητικά μηνύματα της εφαρμογής ως προς τη χρησιμότητα τους;

Καθόλου χρήσιμα 1 2 3 4 5 6 7 8 9 10 Πολύ χρήσιμα

20. Θεωρείτε ότι η διεπαφή είναι σαφής προκειμένου να οδηγήσει το χρήστη στο στόχο του;

Πολύ ασαφής 1 2 3 4 5 6 7 8 9 10 Σαφέστατη

21. Προσθέστε οποιοδήποτε σχόλιο σχετικά με τη χρήση της εφαρμογής στη συγκεκριμένη λειτουργία.

.....
.....
.....
.....

Σχήμα Π.3 Ερωτηματολόγιο για τη Μέτρηση της Χρηστικότητας της Γραφικής Διεπαφής (Λειτουργία Εκτέλεσης SQL Ερωτημάτων)

ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ

Ο Γεωργούλας Κωνσταντίνος γεννήθηκε στο Αγρινιο τον Ιανουάριο του 1981 οπου και πέρασε τα πρώτα χρόνια της ζωής του μέχρι το 1998 όταν εισήχθη ως προπτυχιακός φοιτητής στο Τμήμα Πληροφορικής του Οικονομικού Πανεπιστημίου Αθηνών. Ολοκλήρωσε τις προπτυχιακές σπουδές του τον Ιούνιο του 2002. Από το Σεπτέμβριο του 2004 εργάζεται ως καθηγητής Πληροφορικής της Δευτεροβάθμιας Εκπαίδευσης. Το Φεβρουάριο του 2005 έγινε δεκτός στο Πρόγραμμα Μεταπτυχιακών Σπουδων του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων από το οποίο αποφοίτησε τον Ιούνιο του 2008.