Σχεδιασμός και υλοποίηση γραφικών διεπαφών χρήστη για πρόσβαση σε MOF βάσεις γνώσεων

Αλατζίδης Χρήστος
Κρομμύδας Κωνσταντίνος

Πολυτεχνείο Κρήτης

Τμήμα Ηλεκτρονικών Μηχανικών & Μηχανικών Ηλεκτρονικών Υπολογιστών

Χανιά 2006

Αφιέρωση

Στις οικογένειές μας.

Ευχαριστίες

θέλουμε να ευχαριστήσουμε τον καθηγητή κ. Σταύρο Χριστοδουλάκη για την επίβλεψη και την καθοδήγησή του κατά τη διάρκεια της εκπόνησης της παρούσας διπλωματικής εργασίας. Επίσης, θέλουμε να τον ευχαριστήσουμε για τις σημαντικές εμπειρίες που μας προσέφερε κατά τη διάρκεια της εργασίας μας στο Εργαστήριο Κατανεμημένων Πληροφοριακών Συστημάτων και Εφαρμογών.

Θέλουμε να ευχαριστήσουμε θερμά τον Γιώργο Κοτόπουλο για τη συνεργασία του και για τη σημαντική βοήθεια που μας προσέφερε κατά τη διάρκεια του σχεδιασμού και της υλοποίησης της παρούσας εργασίας.

Επίσης, ευχαριστούμε ιδιαίτερα όλους τους συνεργάτες που δούλεψαν στο πρόγραμμα DBE για την αρμονική και δημιουργική συνεργασία που είχαμε.

Περίληψη

Η ραγδαία ανάπτυξη του Internet τα τελευταία χρόνια είχε σαν επακόλουθο την ανάπτυξη της βιομηχανίας της πληροφορικής. Εταιρείες, οργανισμοί κι επιχειρήσεις επιθυμούν να επωφεληθούν από αυτήν την ανάπτυξη για την παροχή προϊόντων και υπηρεσιών (B2C Business to Customer) ή για τη συνεργασία με άλλες επιχειρήσεις με σκοπό την παροχή σύνθετων υπηρεσιών (B2B Business to Business). Το κατανεμημένο περιβάλλον στο οποίο οι επιχειρήσεις αλληλεπιδρούν αποκαλείται ψηφιακό οικοσύστημα επιχειρήσεων. Μέσα σ'αυτό ζουν, αναπτύσσονται και πεθαίνουν όπως και στο φυσικό οικοσύστημα. Είναι όμως πρακτικά αδύνατο να εντοπιστεί η ακριβής υπηρεσία στον τεράστιο όγκο πληροφορίας που υπάρχει στο Internet. Για τον σκοπό αυτό έχουν δημιουργηθεί εργαλεία αναζήτησης τα οποία έχουν την δυνατότητα δημιουργίας ερωτήσεων και επισκόπησης αποτελεσμάτων με στόχο την εύρεση της απαραίτητης πληροφορίας στο κατανεμημένο περιβάλλον του Internet.

Ένα τέτοιο ψηφιακό οικοσύστημα επιχειρήσεων προτείνεται στα πλαίσια του ερευνητικού προγράμματος DBE (Digital Business Ecosystem) στο οποίο συμμετέχει το Εργαστήριο Διανεμημένων Πληροφοριακών Συστημάτων κι Εφαρμογών (MUSIC). Στο DBE υπάρχει ψηφιακή αναπαράσταση των επιχειρήσεων και των υπηρεσιών που αυτές προσφέρουν στον έξω κόσμο. Μέσα σε αυτό μπορούν να διεκπεραιώνονται επιχειρηματικές διαπραγματεύσεις, συνεργασίες, και συνδιαλλαγές διαμέσου των υπηρεσιών που παρέχονται. Η ανάγκη για αποτελεσματικές γραφικές διεπαφές χρήστη με σκοπό τη δημιουργία ερωτήσεων και την αναπαράσταση της πληροφορίας για τις επιχειρήσεις προέκυψε προκειμένου να είναι δυνατή η πρόσβαση στον μεγάλο όγκο της πληροφορίας.

Στο DBE χρησιμοποιείται η MOF (Meta Object Facility) αρχιτεκτονική του OMG (Object Management Group) για τον ορισμό των πληροφοριών που αναφέρονται στα μοντέλα των επιχειρήσεων και των υπηρεσιών που αυτές προσφέρουν. Η MOF αρχιτεκτονική δίνει τη δυνατότητα για τη δημιουργία ερωτήσεων με μεγάλη ακρίβεια που να εκμεταλλεύονται τη δομή της μέσω της χρήσης της γλώσσας ερωτήσεων QML (Query Metamodel Language) η οποία ορίζεται από το MOF και θέτει περιορισμούς σε MOF πληροφορία. Η QML

δημιουργήθηκε από το Εργαστήριο Διανεμημένων Πληροφοριακών Συστημάτων κι Εφαρμογών (MUSIC).

Στην παρούσα διπλωματική διατριβή, σχεδιάστηκαν και υλοποιήθηκαν δύο γραφικές διεπαφές οι οποίες επιτρέπουν τη δημιουργία ερωτήσεων και την αναπαράσταση MOF πληροφορίας με εύκολο και φιλικό προς το χρήστη τρόπο καλύπτοντας την παραπάνω ανάγκη για το πρόγραμμα DBE. Η μία υλοποίηση έχει αναπτυχθεί σαν plug-in για την πλατφόρμα Eclipse και η δεύτερη σαν web εφαρμογή πάνω στην πλατφόρμα OpenLaszlo. Οι δύο υλοποιήσεις χρησιμοποιούν κοινό μηχανιμό πρόσβασης και διαχείρισης δεδομένων επειδή στηρίζεται στο MVC (Model - Controller - View) πρότυπο σχεδίασης.

Περιεχόμενα

Κατάλογος σχημάτων

Κεφάλαιο 1
Εισαγωγή

1.1 Αναγκαιότητα

Μία βάση γνώσης είναι ένα σύστημα που χρησιμοποιείται για αποθήκευση και πρόσβαση σε δεδομένα που αναπαριστούν γνώση ενός εννοιολογικού πεδίου και τα οποία ακολουθούν κάποιο σχήμα όπως κανόνες, διαδικασίες, κλπ. Χρειάζεται να παρέχει διεπαφές προς τους χρήστες που επικοινωνούν με αυτή ώστε να μπορούν να έχουν πρόσβαση στη βάση δεδομένων που χρησιμοποιεί για να αποθηκεύει τα δεδομένα, η οποία μπορεί να βρίσκεται τοπικά ή να είναι κατανεμημένη σε ένα δίκτυο.

Σε ένα σύστημα το οποίο επικοινωνεί με βάση γνώσης που διαχειρίζεται μετα-δεδομένα, ένα ενδιαφέρον πρόβλημα είναι ο τρόπος με τον οποίο θα αναπαρίστανται τα μετα-δεδομένα ώστε να είναι κατανοητή η σημασία τους προς το χρήστη, αλλά και ο τρόπος που μπορεί ο χρήστης να χρησιμοποιήσει τα υπάρχοντα μετα-δεδομένα μέσω της αναπαράστασης αυτής για την κατασκευή ερωτήσεων προς τη βάση γνώσης ώστε να εκμεταλλευτεί τη δομή τους και να έχει ακριβέστερα αποτελέσματα.

Αν η βάση γνώσης αποθηκεύει την πληροφορία βάσει της MOF (MetaObject Facility) [6] αρχιτεκτονικής μετα-δεδομένων, μία αρχιτεκτονική τεσσάρων επιπέδων που δομεί την πληροφορία με τέτοιο τρόπο ώστε να εξασφαλίζει επεκτασιμότητα, τότε το πρόβλημα γίνεται ακόμα πιο πολύπλοκο καθώς το σύστημά μας πρέπει να

αναπαριστά τη δομή κάθε επιπέδου με τέτοιο τρόπο ώστε να μπορεί να υποστηρίξει την επεκτασιμότητα που παρέχει η MOF αρχιτεκτονική. Επίσης το γεγονός ότι κάθε επίπεδο πληροφορίας στο MOF αποτελεί στιγμιότυπο του προηγούμενου συνεπάγεται ότι για την κατασκευή ερωτήσεων προς ένα επίπεδο πρέπει να τεθούν περιορισμοί στο προηγούμενο, το οποίο απαιτεί γνώση της δομής του τελευταίου.

Ένα τέτοιο πρόβλημα κληθήκαμε να αντιμετωπίσουμε στα πλαίσια της παρούσας εργασίας· την πρόσβαση στην πληροφορία που περιέχει μία MOF βάση γνώσης από τον τελικό χρήστη με τέτοιο τρόπο ώστε να είναι κατανοητή η δόμηση της πληροφορίας και να μπορεί αυτός να εκμεταλλευθεί την δομή της στην κατασκευή ερωτήσεων, χωρίς να χρειάζεται να έχει γνώσεις για την γλώσσα ερωτήσεων που χρησιμοποιείται.

1.2 Το πρόγραμμα DBE (Digital Business Ecosystem)

Η εργασία αυτή πραγματοποιήθηκε μέσα στα πλαίσια του ερευνητικού προγράμματος DBE (Ψηφιακό Οικοσύστημα Επιχειρήσεων - Digital Business Ecosystem) [1] , στο οποίο συμμετέχει το Εργαστήριο Διανεμημένων Πληροφοριακών Συστημάτων και Εφαρμογών και σε αυτή την ενότητα θα αναφέρουμε λίγα πράγματα για αυτό.

Το πρόγραμμα DBE στοχεύει στη δημιουργία ενός περιβάλλοντος λογισμικού βασισμένου στο Internet, στο οποίο οι οργανισμοί που ζουν σε αυτό (επιχειρήσεις) συνεργάζονται, ανταγωνίζονται και συν-εξελίσσονται. Αυτό επιτυγχάνεται δίνοντας στις εφαρμογές υπηρεσιών διαδικτύου (web services) των επιχειρήσεων την ευκολία να οργανώνονται και να βελτιώνονται με ημιαυτόματο τρόπο, μιμούμενες εξελικτικά φαινόμενα τα οποία παρατηρούνται και στον φυσικό κόσμο. Ένα ψηφιακό επιχειρησιακό οικοσύστημα ορίζεται ως το εικονικό εκείνο περιβάλλον όπου υπάρχει ψηφιακή αναπαράσταση των επιχειρήσεων (π.χ. περιγραφή των λειτουργιών τους, των επιχειρησιακών μοντέλων τους, κλπ.) και των υπηρεσιών που αυτές προσφέρουν στον έξω κόσμο (π.χ. πώληση προϊόντων, ενοικίαση δωματίων, μεταφορά δεμάτων, κλπ.). Μέσα σε αυτό μπορούν να διεκπεραιώνονται επιχειρηματικές διαπραγματεύσεις, συνεργασίες, και συνδιαλλαγές διαμέσου των υπηρεσιών που παρέχονται. Οι επιχειρήσεις δρουν αυτόνομα και προσφέρουν υπηρεσίες (services)

που μπορεί να είναι σύνθεση πιο στοιχειωδών υπηρεσιών, κάποιες από τις οποίες μπορεί να προσφέρουν άλλες επιχειρήσεις του οικοσυστήματος. Φυσικά μια επιχείρηση θα διαλέξει το πιο ανταγωνιστικό service που γνωρίζει μέσα από το οικοσύστημα για να το ενσωματώσει στο δικό της service. Έτσι οι πιο ανταγωνιστικές επιχειρήσεις του οικοσυστήματος επιβιώνουν μακροπρόθεσμα.

Για την υλοποίηση των παραπάνω απαιτείται η δημιουργία ενός συνόλου από γραφικά εργαλεία, τα οποία θα αλληλεπιδρούν μεταξύ τους, δίνοντας τη δυνατότητα σε ένα χρήστη να περιγράψει ένα μοντέλο επιχείρησης και τις υπηρεσίες που θα παρέχει, αλλά και να αναζητήσει τόσο μοντέλα όσο και υπηρεσίες, και όλα μαζί υλοποιούν ένα περιβάλλον (το DBE Studio [2]) μέσα στο οποίο μπορεί ο οποιοσδήποτε χρήστης να ενεργήσει αναζητώντας, χρησιμοποιώντας ή δημιουργώντας μοντέλα επιχειρήσεων και υπηρεσίες.

Είναι σημαντικό σε αυτό το σημείο να αναφερθούν τα βασικά σημεία της σχεδίασης του DBE. Το πρόγραμμα DBE ακολουθεί την MDA (Model Driven Architecture) [4] προσέγγιση, μία αρχιτεκτονική στη μοντελοποίηση συστημάτων που χωρίζει την προδιαγραφή της λειτουργίας και συμπεριφοράς των συστημάτων από αυτήν της εφαρμογής τους σε μια συγκεκριμένη πλατφόρμα τεχνολογίας. Η δομή της πληροφορίας που διαχειρίζεται το DBE ορίζεται από τη MOF (MetaObject Facility) [6] αρχιτεκτονική, η οποία είναι μία πολυ-επίπεδη αρχιτεκτονική που έχει κατασκευαστεί με σκοπό τη διαχείριση μετά-δεδομένων με τρόπο που να εξασφαλίζει επεκτασιμότητα. Ως γλώσσα ερωτήσεων χρησιμοποιήθηκε η QML (Query Metamodel Language) [7], με την οποία μπορούμε να θέτουμε περιορισμούς στη MOF πληροφορία. Η τελευταία είναι αποθηκευμένη σε κατανεμημένες MOF βάσεις γνώσης, καθώς το DBE αποτελεί ένα peer-to-peer περιβάλλον.

Το λογισμικό του προγράμματος DBE είναι ανοικτό (open source) και κατ' επέκταση και η υλοποίηση της παρούσας εργασίας. Το λογισμικό είναι διαθέσιμο σε όποιον επιθυμεί στο site του SourceForge [3].

1.3 Στόχοι της εργασίας

Θέμα της εργασίας αυτής είναι η δημιουργία γραφικών διεπαφών για την πρόσβαση σε πληροφορία αποθηκευμένη σε MOF βάση γνώσεων. Συγκεκριμένα οι στόχοι της παρούσας εργασίας είναι:

- Μελέτη του τρόπου δόμησης και διαχείρισης της πληροφορίας που έχει αναπαρασταθεί με βάση την MOF αρχιτεκτονική.
- Σχεδιασμός και υλοποίηση του τρόπου αναπαράστασης της πληροφορίας αυτής, με όσο γίνεται απλούστερο τρόπο, αποκρύπτοντας όποια πολυπλοκότητα εμφανίζεται σε αυτήν.
- Σχεδιασμός και υλοποίηση εύχρηστων και αποτελεσματικών γραφικών διεπαφών προς το χρήστη για την πρόσβαση στην αποθηκευμένη πληροφορία και για τη δημιουργία ερωτήσεων προς αυτήν, χωρίς να απαιτούνται από αυτόν εξεζητημένες γνώσεις σχετικά με τη γλώσσα ερωτήσεων που χρησιμοποιείται.
- Σχεδιασμός και υλοποίηση του τρόπου αναπαράστασης, διαχείρισης και αποθήκευσης (στο τοπικό σύστημα αρχείων) των ερωτήσεων αυτών.

1.4 Δομή της εργασίας

Στο δεύτερο κεφάλαιο περιγράφονται οι τεχνολογίες που χρησιμοποιήθηκαν κατά το σχεδιασμό και υλοποίηση της παρούσας εργασίας. Επίσης παρατίθενται κάποιες εργασίες με σχετιζόμενο αντικείμενο μελέτης τις οποίες ξεχωρίσαμε και αναφέρονται τα βασικά πλεονεκτήματα και μειονεκτήματα τους σε σχέση με τις προδιαγραφές της εργασίας μας.

Στο τρίτο κεφάλαιο αναπτύσσεται η αρχιτεκτονική του συστήματός μας. Περιγράφεται ο τρόπος με τον οποίο σχεδιάστηκε και γιατί, οι μονάδες που την αποτελούν, οι οποίες αναλύονται σε εκτενή βαθμό ώστε να γίνει κατανοητή η χρησιμότητα της καθεμιάς και ο τρόπος με τον οποίο αλληλεπιδρούν.

Στο τέταρτο κεφάλαιο παρουσιάζεται η λειτουργικότητα που είναι επιθυμητό να υποστηρίζει η υλοποίηση της σχεδίασης με την τεχνική των περιπτώσεων χρήσης (use cases) και η υλοποίηση του συστήματος με αναλυτική περιγραφή των

δυνατοτήτων που παρέχει στον τελικό χρήστη, η οποία έγινε για δύο διαφορετικά περιβάλλοντα εργασίας.

Στο πέμπτο κεφάλαιο παρουσιάζονται τα αποτελέσματα δοκιμών χρήσης των γραφικών διεπαφών που υλοποιήθηκαν τόσο για τη σταθερότητα τους, όσο και για την ευχρηστία τους.

Στο έκτο και τελευταίο κεφάλαιο γίνεται μία ανακεφαλαίωση, αναφέρονται τα συμπεράσματα της εργασίας και επισημαίνονται οι μελλοντικές επεκτάσεις.

Κεφάλαιο 2

Επισκόπηση σχετικής έρευνας

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναφερθούν οι τεχνολογίες που χρησιμοποιήθηκαν κατά το σχεδιασμό και υλοποίηση της παρούσας εργασίας, για τις οποίες θα γίνει μία σύντομη περιγραφή των βασικών χαρακτηριστικών τους. Επίσης θα περιγραφούν και υπάρχουσες εργασίες με σχετιζόμενο αντικείμενο μελέτης, τις οποίες ξεχωρίσαμε και θα αναφερθούν τα πλεονεκτήματα και μειονεκτήματα τους σε σχέση με τις ανάγκες για τη δική μας εργασία.

2.2 Τεχνολογική βάση

Όπως έχει ήδη αναφερθεί η εργασία αυτή έχει εκπονηθεί στα πλαίσια του ερευνητικού προγράμματος DBE. Έτσι ήταν αναγκαίο να μελετηθούν και να χρησιμοποιηθούν τεχνολογίες οι οποίες έχουν υιοθετηθεί από το DBE κατά το σχεδιασμό του. Κατά το σχεδιασμό και την υλοποίηση της εργασίας αυτής όμως χρησιμοποιήθηκαν και άλλες τεχνολογίες, οι οποίες επίσης θα περιγραφούν στην παράγραφο αυτή.

2.2.1 MDA (Model Driven Architecture) αρχιτεκτονική συστημάτων

Η MDA (Model Driven Architecture) [4] είναι μια αρχιτεκτονική μοντελοποίησης συστημάτων του διεθνούς οργανισμού τυποποίησης OMG (Object Management Group) [10]. Βασικός στόχος της δημιουργίας της είναι να διαχωρίσει την προδιαγραφή λειτουργίας τους από αυτήν της εφαρμογής τους σε μια συγκεκριμένη πλατφόρμα τεχνολογίας. Δηλαδή αυτό που προτείνει η MDA είναι αρχικά η δημιουργία ενός ανεξάρτητου πλατφόρμας μοντέλου (PIM – Platform Independent Model) που να περιγράφει την λειτουργικότητα και τη συμπεριφορά της εφαρμογής που θέλουμε να κατασκευάσουμε, στη συνέχεια τη μετατροπή του μοντέλου αυτού σε ένα μοντέλο συγκεκριμένης πλατφόρμας (PSM – Platform Specific Model) και τέλος την μετατροπή του τελευταίου μοντέλου σε μία υλοποίηση εφαρμογής για την πλατφόρμα για την οποία σχεδιάστηκε.

Ακολουθώντας την αρχιτεκτονική αυτή ακόμη και αν αλλάξει η υλοποίηση της εφαρμογής, το PSM θα αλλάξει μόνο αν δεν καλύπτει τις ανάγκες της νέας υλοποίησης. Σε αυτήν την περίπτωση θα δημιουργηθεί ένα νέο PSM, το οποίο όμως θα προκύψει και πάλι από το ίδιο PIM, επειδή το τελευταίο έχει σχεδιαστεί με τέτοιο τρόπο ώστε να μην εξαρτάται από καμία πλατφόρμα υλοποίησης.

2.2.2 Αρχιτεκτονική MOF (MetaObject Facility)

Η αρχιτεκτονική MOF (MetaObject Facility) [6] του διεθνούς οργανισμού τυποποίησης OMG (Object Management Group) [10] έχει οριστεί με σκοπό τη διαχείριση μετα-δεδομένων με τρόπο που να εξασφαλίζει επεκτασιμότητα, παρέχοντας ένα πλαίσιο το οποίο θα υποστηρίζει κάθε είδους μετα-δεδομένα, και θα επιτρέπει την προσθήκη νέων ειδών όταν κρίνεται αναγκαίο.

Για την επίτευξη αυτού του σκοπού το MOF υιοθετεί μία πολυ-επίπεδη αρχιτεκτονική μετα-δεδομένων, η οποία βασίζεται στην κλασική αρχιτεκτονική μετα-δεδομένων τεσσάρων επιπέδων που είναι αρκετά δημοφιλής και χρησιμοποιείται σε πολλά γνωστά πρότυπα.

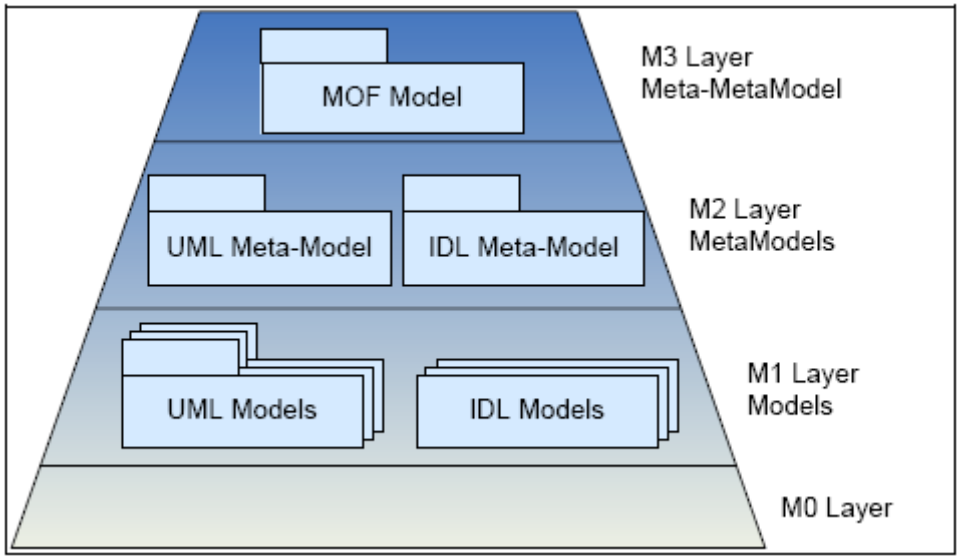
Η χρήση τεσσάρων επιπέδων θεωρείται ικανή να περιγράψει πλήρως τα μετα-δεδομένα. Τα επίπεδα αυτά είναι:

1. Το επίπεδο πληροφορίας (Information layer) ή M0. Αποτελείται από τα δεδομένα που επιθυμούμε να περιγράψουμε.
2. Το επίπεδο του Μοντέλου (Model layer) ή M1. Περιέχει τα μετα-δεδομένα τα οποία περιγράφουν τα δεδομένα στο επίπεδο πληροφορίας. Τα μετα-δεδομένα ομαδοποιούνται ανεπίσημα σε μοντέλα.
3. Το επίπεδο του μετα-μοντέλου (Metamodel layer) ή M2. Στο επίπεδο αυτό υπάρχουν οι περιγραφές οι οποίες προσδιορίζουν την δομή και την σημασιολογία των μετα-δεδομένων. Οι περιγραφές αυτές λέγονται μετα-μετα-δεδομένα (meta-metadata) και ομαδοποιούνται ανεπίσημα σε μετα-μοντέλα. Ένα μετα-μοντέλο είναι ουσιαστικά μία γλώσσα περιγραφής διαφορετικών ειδών μετα-δεδομένων.
4. Το επίπεδο του μετα-μετα-μοντέλου (Meta-metamodel layer) ή M3. Στο επίπεδο αυτό υπάρχει η περιγραφή (μία και μόνη) της δομής και της σημασιολογίας των μετα-μετα-δεδομένων (meta-metadata). Ένα μετα-μετα-μοντέλο είναι ουσιαστικά μία γλώσσα περιγραφής διαφορετικών ειδών μετα-μετα-δεδομένων.

Εάν το πλαίσιο μοντελοποίησης βασισμένο σε αυτή την αρχιτεκτονική σχεδιαστεί κατάλληλα, τότε:

- μπορεί να υποστηρίξει κάθε είδους μοντέλου και παραδείγματος μοντελοποίησης το οποίο είναι πρακτικά κατανοητό,
- μπορεί να επιτρέψει την συσχέτιση διαφορετικών ειδών μετα-δεδομένων,
- μπορεί να επιτρέψει την διαδοχική προσθήκη μετα-μοντέλων και νέων ειδών μετα-δεδομένων, και
- μπορεί να υποστηρίξει την ανταλλαγή αυθαίρετων μετα-δεδομένων (μοντέλα) και μετα-μετα-δεδομένων (μετα-μοντέλων) μεταξύ ομάδων οι οποίες χρησιμοποιούν το ίδιο μετα-μετα-μοντέλο.

Σχηματικά η MOF αρχιτεκτονική μετα-δεδομένων απεικονίζεται εικόνα 1.



Εικόνα 1: Η MOF Αρχιτεκτονική μετα-δεδομένων.

Ένα τυπικό παράδειγμα χρήσης της MOF αρχιτεκτονικής μετα-δεδομένων με μετα-μοντέλα για την αναπαράσταση των γλωσσών UML (Unified Model Language) και IDL(Interface Definition Language) .

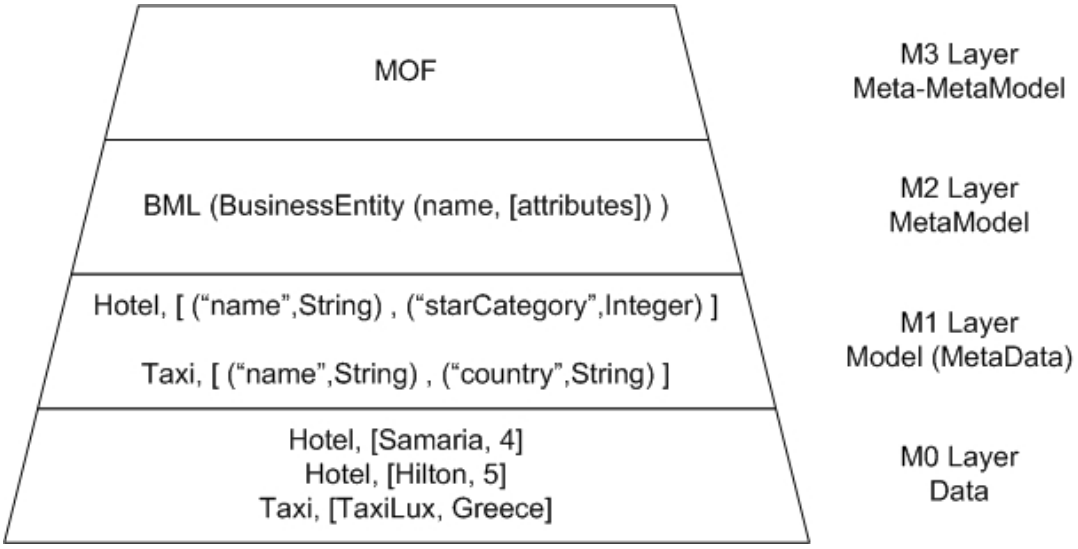
Το MOF ορίζει τέσσερις βασικές έννοιες (primitives) μοντελοποίησης:

- **Κλάσεις (Classes)**, οι οποίες χρησιμοποιούνται για την μοντελοποίηση των MOF μετά-αντικειμένων. Οι κλάσεις μπορούν να έχουν τριών ειδών ιδιότητες:
  - Χαρακτηριστικά (Attributes).
  - Λειτουργίες (Operations).
  - Αναφορές (References) μεταξύ κλάσεων.

  Οι κλάσεις μπορούν επιπλέον να περιέχουν Εξαιρέσεις (Exceptions) που μπορούν να προκληθούν από Λειτουργίες (Operations), Σταθερές (Constants), Τύπους Δεδομένων (DataTypes) όπως για παράδειγμα Boolean, Integer, String, Enumerations κλπ., Περιορισμούς (Constraints) όπως για παράδειγμα τον περιορισμό πως ένα Attribute "x:integer" είναι μονός αριθμός , και άλλα στοιχεία. Επιπλέον επιτρέπεται η κληρονομικότητα μεταξύ των κλάσεων, ενώ μία κλάση μπορεί να οριστεί σαν «αφηρημένη» (abstract). Τέλος μία κλάση μπορεί να οριστεί είτε σαν «φύλλο» (leaf), οπότε δεν μπορεί να έχει υπό-κλάσεις, είτε σαν «ρίζα» (root), οπότε δεν μπορεί να έχει υπέρ-κλάσεις.
- **Συσχετίσεις (Associations)**, οι οποίες χρησιμοποιούνται για την μοντελοποίηση των σχέσεων μεταξύ δύο μετά-αντικειμένων (Κλάσεων).
- **Τύποι Δεδομένων (DataTypes)**, οι οποίοι χρησιμοποιούνται για την μοντελοποίηση άλλων δεδομένων (π.χ. primitive τύποι, εξωτερικοί τύποι κτλ.).

- **Πακέτα (Packages)**, τα οποία χρησιμοποιούνται για την ομαδοποίηση των μοντέλων.

Για να γίνει περισσότερο κατανοητή η χρήση της αρχιτεκτονικής MOF θα παραθέσουμε ένα παράδειγμα εφαρμογής της στα πλαίσια του προγράμματος DBE. Για τις ανάγκες του DBE έχουν δημιουργηθεί διάφορα μετα-μοντέλα για την περιγραφή των επιχειρήσεων, των υπηρεσιών, κ.α. Ένα από αυτά, το BML (Business Model Language) χρησιμοποιείται για την περιγραφή επιχειρήσεων, με βάση τα πακέτα και τις κλάσεις που έχουν οριστεί σε αυτό (M2 επίπεδο). Τα μοντέλα επιχειρήσεων δημιουργούνται ως στιγμιότυπα του BML. Έτσι μπορούν να υπάρχουν διαφορετικά μοντέλα, όπως π.χ. ένα με στιγμιότυπο της κλάσης BusinessEntity του BML με όνομα "Hotel"και attributes "name" και "starCategory" και ένα με στιγμιότυπο της ίδιας κλάσης με όνομα "Taxi" και attributes "name" και "country" (M1 επίπεδο). Στη συνέχεια σε αυτά τα μοντέλα μπορούμε να αντιστοιχίσουμε συγκεκριμένα δεδομένα ως στιγμιότυπο των μοντέλων αυτών (M0 επίπεδο). Το παράδειγμα αυτό φαίνεται σχηματικά στην εικόνα 2.



Εικόνα 2: Παράδειγμα εφαρμογής του MOF στο DBE.

Με βάση την κλάση BusinessEntity του BML μετα-μοντέλου (M2 επίπεδο) ορίζονται ως διαφορετικά στιγμιότυπα δύο μοντέλα με BusinessEntity "Hotel" και "Taxi" (M1 επίπεδο), με βάση τα οποία ορίζονται τα δεδομένα (M0 επίπεδο).

2.2.3 MDR (MetaData Repository)

Το MDR (MetaData Repository) [9] του οργανισμού NetBeans [27] είναι μία βάση γνώσης μετα-δεδομένων. Είναι το κεντρικό μέρος στο οποίο αποθηκεύονται και

φυλάσσονται τα δεδομένα. Σε ένα repository μπορεί να υπάρχουν πολλαπλά αρχεία ή βάσεις δεδομένων για διανομή σε ένα δίκτυο (όπως και στην περίπτωση του DBE) ή μπορεί να είναι προσβάσιμο και απευθείας χωρίς να χρειάζεται μεσολάβηση δικτύου. Παρέχει διαχείριση της αποθηκευμένης πληροφορίας προς τους χρήστες, χωρίς αυτοί να έχουνε γνώση για αυτήν (το μετα-μοντέλο το οποίο ακολουθεί), μέσω XML. Χρησιμοποιείται το πρότυπο XMI (XML MetaData Interchange) [11], το οποίο είναι ουσιαστικά μετάφραση του MOF σε XML. Σε μία υλοποίηση η επικοινωνία αυτή γίνεται μέσω του προτύπου JMI (Java MetaData Intreface) [12], το οποίο είναι ουσιαστικά μία μετάφραση του MOF σε Java, δηλαδή παρέχει τη διεπαφή σε Java μέσω της οποίας μπορεί κάποιος να έχει πρόσβαση στο MDR.

Όσον αφορά στο DBE, το MDR κρατάει όλη την πληροφορία που αναφέρεται σε αυτό (μετα-μοντέλα, μοντέλα, οντολογίες, δεδομένα, κ.α.). Υπάρχουν μηχανισμοί για την επικοινωνία με το MDR ώστε να μπορούμε να έχουμε πρόσβαση με έναν αφαιρετικό τρόπο στα δεδομένα (π.χ. να κάνουμε ερωτήσεις ‘η να εξερευνήσουμε τις πληροφορίες που υπάρχουν για ένα μοντέλο). Η εργασία μας σχετίζεται με το MDR επειδή οι ερωτήσεις που γίνονται μέσω του interface που υλοποιήσαμε αναφέρονται στην πληροφορία που είναι αποθηκευμένη σε αυτό.

2.2.4 KB (Knowledge Base) / SR (Service Registry) υπηρεσίες

Οι υπηρεσίες KB (Βάση Γνώσης) και SR (Μητρώο Υπηρεσιών) [13] παρέχουν τη λειτουργικότητα για πρόσβαση στη βάση γνώσης του DBE μέσω διεπαφών. Υποστηρίζουν αποθήκευση, ανάκτηση και ερωτήσεις στα μοντέλα του DBE και στα δεδομένα μέσω μιας διεπαφής που έχει δημιουργηθεί για το σκοπό αυτό. Οι υπηρεσίες αυτές σχεδιάστηκαν και υλοποιήθηκαν από το Εργαστήριο Διανεμημένων Πληροφοριακών Συστημάτων και Εφαρμογών.

Η ύπαρξη δύο υπηρεσιών οφείλεται στο γεγονός ότι το DBE έχει σχεδιαστεί να λειτουργεί σε δύο διαφορετικά περιβάλλοντα. Στο Service Factory (παραγωγή υπηρεσιών) περιβάλλον δίδεται πρόσβαση σε όλα τα μοντέλα και τις οντολογίες που είναι αποθηκευμένα στη βάση γνώσης με την χρήση των οποίων κατασκευάζονται καινούριες υπηρεσίες (services), π.χ. από μία επιχείρηση. Σε αυτό το περιβάλλον η πρόσβαση γίνεται μέσω της KB. Στο Service Execution (εκτέλεση υπηρεσιών) περιβάλλον δίδεται πρόσβαση στις υπηρεσίες (services) οι οποίες είναι αποθηκευμένες στη βάση γνώσης. Η βάση γνώσης στο DBE είναι κατανεμημένη

στους κόμβους του peer-to-peer δικτύου στο οποίο αυτό λειτουργεί. Έτσι για παράδειγμα αν γίνει μία ερώτηση προς τη βάση γνώσης δεν θα επιστραφούν μόνο τα αποτελέσματα του κόμβου με τον οποίο επικοινωνεί ο χρήστης, αλλά και αποτελέσματα από τους υπόλοιπους κόμβους του δικτύου. Για κάθε υπηρεσία υπάρχει μία μοναδική περιγραφή, το "Service Manifest", που περιλαμβάνει τόσο το μοντέλο περιγραφής της υπηρεσίας (μετα-δεδομένα), όσο και τα δεδομένα που αναφέρονται σε αυτή (π.χ. τοποθεσία, όνομα, κόστος, ...). τους. Οι υπηρεσίες είναι προσβάσιμες από το μητρώο υπηρεσιών (Service Registry), εφόσον βέβαια παρέχεται η κατάλληλη διεπαφή. Σε αυτό το περιβάλλον η πρόσβαση γίνεται μέσω της SR.

2.2.5 Γλώσσα ερωτήσεων QML (Query Metamodel Language)

Η Query Metamodel Language (QML) [7] είναι μία αντικειμενοστραφής γλώσσα ερωτήσεων που χρησιμοποιείται για την πρόσβαση σε πληροφορία αποθηκευμένη σε MOF βάσεις γνώσεις. Δημιουργήθηκε, ως επέκταση της Object Constraint Language (OCL) 2.0 [14] του διεθνούς οργανισμού τυποποίησης OMG (Object Management Group) [10], με σκοπό τη δημιουργία ερωτήσεων και την έκφραση περιορισμών στα επίπεδα M3-M1 του MOF. Η ίδια είναι ένα M2 MOF μοντέλο κι έχει υλοποιηθεί σαν στιγμιότυπο του MOF μετα-μοντέλου. Ένα άλλο βασικό χαρακτηριστικό της είναι η δυνατότητα αναζήτησης MOF πληροφορίας με ασαφή κι όχι αυστηρό τρόπο και γι' αυτό τα αποτελέσματα ανακτώνται με βάση τη σχετικότητά τους ως προς τους περιορισμούς που τίθενται από την ερώτηση (ποσοστό επιτυχίας). Σχεδιάστηκε και υλοποιήθηκε από το Εργαστήριο Διανεμημένων Πληροφοριακών Συστημάτων και Εφαρμογών στα πλαίσια του προγράμματος DBE.

Τα βασικά χαρακτηριστικά που υποστηρίζει η QML σαν γλώσσα ερωτήσεων είναι τα εξής:

- Καθολική ύπαρξη (Universal quantification)
- Μερική ύπαρξη (Existential quantification)
- Σύζευξη (Conjunction)
- Προβολή (Projection)
- Μετονομασία (Renaming)
- Διήθηση (Filtering)

- Διάζευξη (Disjunction)
- Κατασκευή νέων στοιχείων (Construction of new elements)
- Σύνδεση (Join)
- Πολλαπλά έγγραφα (Multiple Documents)
- Καρτεσιανό γινόμενο (Cartesian product)
- Άρνηση (Negation)
- Άθροιση (Aggregates)
- Αριθμητικοί υπολογισμοί (Arithmetic computations)
- Ένωση (Union) (Όχι σε ολόκληρο document)
- Διαφορά (Difference) (Όχι σε ολόκληρο document)
- Ομαδοποίηση (Grouping) (Μέσω του let)
- Δήλωση μεταβλητών (Variable declaration)

Ένα πολύ απλό παράδειγμα ερώτησης σε QML είναι το παρακάτω:

```
Context BML:BusinessEntity complexQuery: BusinessOrganization
query:
attribute->select(name="StarCategory")
```

Με αυτή την ερώτηση προς το BML(Business Model Language) μεταμοντέλο ζητούνται οι επιχειρήσεις που έχουν σαν χαρακτηριστικό την κατηγορία τους σε αστέρια (attribute 'StarCategory').

Στο δεύτερο παράδειγμα που ακολουθεί και είναι περισσότερο πολύπλοκο θα παρουσιαστούν μερικές από τις δυνατότητες της QML σαν γλώσσα ερωτήσεων καθώς επίσης η σύνταξή της και μερικά από τα χαρακτηριστικά της που αναφέρθηκαν παραπάνω:

```
Context BML:BusinessEntity complexQuery: BusinessOrganization
query:
attribute->select(name="StarCategory")
and
attribute->select(name="Address")->
exists(type="ODM::HotelDomain::Address" and getTypeClassInstance()->
select(type.name="City")->exists(TheDTPRange.lexicalForm="Chania"))
```

Με αυτή την ερώτηση προς το BML(Business Model Language) μεταμοντέλο ζητούνται οι επιχειρήσεις που έχουν σαν χαρακτηριστικό την κατηγορία τους σε αστέρια (attribute 'StarCategory') και τη διεύθυνσή τους (attribute 'Address'), η

οποία να ορίζεται από την οντολογία HotelDomain και να έχει τιμή Χανιά (City "Chania").

Πρέπει να σημειωθεί ότι οι οντολογίες στο DBE ακολουθούν το μετα-μοντέλο ODM (Ontology Definition MetaModel) που σχεδιάστηκε και υλοποιήθηκε από το Εργαστήριο Διανεμημένων Πληροφοριακών Συστημάτων και Εφαρμογών στα πλαίσια του προγράμματος DBE. Ο λόγος που δημιουργήθηκε το ODM μετα-μοντέλο είναι για να υπάρχει δυνατότητα αναπαράστασης OWL (Ontology Web Language) [28] οντολογιών με βάση την MOF αρχιτεκτονική .

Η έκφραση attribute->select(name="StarCategory"), απαιτεί να υπάρχει ένα attribute με name "StarCategory". Η υπόλοιπη πρόταση απαιτεί ένα να υπάρχει ένα attribute με name "Address", type "ODM::HotelDomain::Address" και το στιγμιότυπο αυτού του Address να έχει name "City" και value "Chania". Επίσης πρέπει να αναφέρουμε ότι ο τύπος του Address απαιτείται να έχει οριστεί από διαφορετικό context; που ορίζεται από μία οντολογία που ακολουθεί το ODM μετα-μοντέλο. Τέλος μεταξύ των δύο προτάσεων υπάρχει μία σύζευξη (conjunction).

Παρατηρούμε ότι για να μπορέσουμε να εκφράσουμε μία πολύπλοκη ερώτηση σε QML, ώστε να εκμεταλλευθούμε τις δυνατότητες που αυτή προσφέρει, απαιτείται πολύ καλή γνώση της σύνταξής της

2.2.6 MVC (Model-View- Controller)

Το MVC (Model-View-Controller) [5], είναι ένα πρότυπο σχεδίασης (design pattern) που διαχωρίζει τη διαδικασία της πρόσβασης στα δεδομένα, τη λογική της εφαρμογής, την παρουσίαση των δεδομένων και την αλληλεπίδραση της εφαρμογής με το χρήστη.

Το MVC είναι πολύ απλό στην αρχιτεκτονική του αλλά πολύ χρήσιμο στη σχεδίαση εφαρμογών λογισμικού. Σύμφωνα με το πρότυπο αυτό, μια εφαρμογή χωρίζεται σε τρία βασικά μέρη: το μοντέλο (model), την όψη (view), και τους ελεγκτές (controllers).

Το μοντέλο αποτελεί τον πυρήνα της εφαρμογής, επειδή μέσα σε αυτό κρατείται τόσο η κατάσταση όσο και τα δεδομένα αυτής. Βασική αρχή η οποία πρέπει να τηρείται κατά τη σχεδίαση μιας εφαρμογής σύμφωνα με το MVC πρότυπο είναι ότι όλα τα δεδομένα τα οποία μπορούν να αλλάζουν τιμή θα πρέπει να

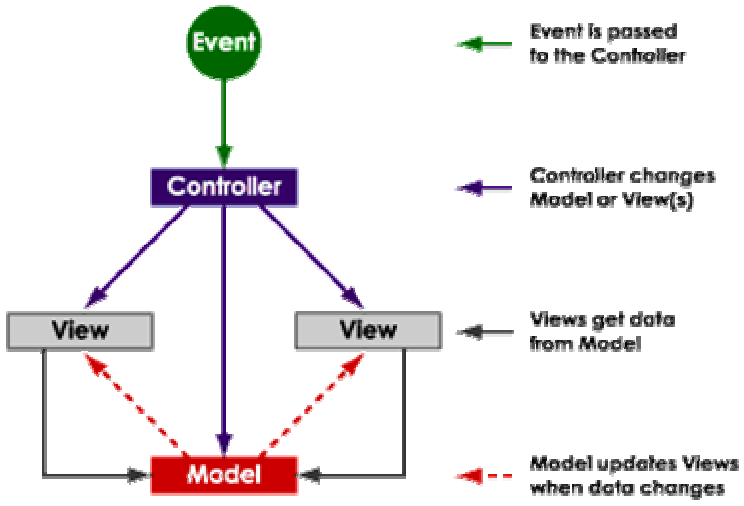
αποθηκεύονται στο μοντέλο. Παράλληλα το μοντέλο θα πρέπει να υλοποιεί κάποιο μηχανισμό ειδοποίησης (notification) ο οποίος θα ενημερώνει την εφαρμογή για πιθανές αλλαγές σε αυτό.

Η όψη αποτελεί το γραφικό μέρος της εφαρμογής στο οποίο αναπαριστώνται τα δεδομένα του μοντέλου. Όλα τα δεδομένα τα οποία παρουσιάζονται σε αυτήν θα πρέπει να έχουν αποθηκευτεί αρχικά στο μοντέλο.

Τέλος, οι ελεγκτές αποτελούν το τμήμα το οποίο συνδέει το μοντέλο με την όψη και έχει ως σκοπό ύπαρξης την επίτευξη επικοινωνίας μεταξύ του μοντέλου και της γραφικής αναπαράστασής του.

Όταν ο χρήστης ενεργήσει πάνω σε κάποιο στοιχείο της γραφικής διεπαφής της εφαρμογής τότε ο ελεγκτής αναλαμβάνει να ενημερώσει το μοντέλο αλλάζοντας τα δεδομένα που κρατάει. Έπειτα ο μηχανισμός ενημέρωσης, τον οποίο διαθέτει το μοντέλο, ανανεώνει τη γραφική απεικόνιση του μοντέλου (όψη) με τα νέα δεδομένα. Στο τελικό στάδιο ο χρήστης βλέπει την ανανεωμένη αναπαράσταση του μοντέλου, η οποία παρουσιάζεται σε αυτόν μέσω της γραφικής διεπαφής της εφαρμογής.

Στην εικόνα 3 φαίνεται ο τρόπος με τον οποίο επικοινωνούν τα διάφορα μέρη μιας εφαρμογής, που ακολουθεί το MVC πρότυπο σχεδίασης.



Εικόνα 3: Το MVC πρότυπο σχεδίασης.

Όταν ο χρήστης (actor) ενεργήσει πάνω σε κάποιο στοιχείο του γραφικού μέρους της εφαρμογής, ο ελεγκτής αναλαμβάνει να πραγματοποιήσει την αντίστοιχη αλλαγή στα δεδομένα του μοντέλου. Ο μηχανισμός ενημέρωσης του μοντέλου ανανεώνει τη γραφική απεικόνιση του μοντέλου με τα νέα δεδομένα

2.2.7 QbE (Query by Example)

Το QbE (Query by Example) [8] είναι μια μέθοδος για δημιουργία ερωτήσεων που επιτρέπει στο χρήστη να δημιουργήσει τις ερωτήσεις απλά ορίζοντας τιμές σε πίνακες, παρέχοντάς του ουσιαστικά ένα παράδειγμα των αποτελεσμάτων της ερώτησης. Ο χρήστης δεν αναγκάζεται να γράψει μια ερώτηση με μια έγκυρη σύνταξη, ή σε κάποια γλώσσα των ερωτήσεων, αλλά το μόνο που πρέπει να κάνει είναι να τοποθετήσει τους περιορισμούς που θέλει στο ανάλογο πεδίο του πίνακα. Κατά συνέπεια ο χρήστης έχει ένα επίπεδο αφαίρεσης μεταξύ της δημιουργίας μιας ερώτησης που θέλει με την ανάθεση των τιμών και της γλώσσας που θα υλοποιήσει την ερώτηση αυτή.

Οι γραμμές του πίνακα ομαδοποιούν τις τιμές που ορίζει ο χρήστης και υπονοούν τα κριτήρια που πρέπει να ικανοποιεί η ερώτηση. Τα κριτήρια που τοποθετούνται σε διαφορετικές γραμμές μπορούν να οριστούν διαφορετικά.

Με αυτήν την μέθοδο, μπορούμε να δημιουργήσουμε έναν φιλικό προς το χρήστη τρόπο δημιουργίας ερωτήσεων, που θα είναι χωρίς λάθη (καθώς το μόνο που κάνει ο χρήστης, είναι να ορίζει τιμές σε πεδία), δεν θα απαιτεί την εξειδικευμένη γνώση κάποιας γλώσσα και συγχρόνως θα δίνει τη δυνατότητα με έναν απλό τρόπο να δημιουργεί περίπλοκες ερωτήσεις.

Παραδείγματα:

- Παραδείγματος χάριν, για να βρούμε όλους τους πελάτες που έχουν ένα λογαριασμό στο branch SFU:

deposit	bname	account#	cname	balance
	SFU		P._x	

- για να βρούμε όλα τα ονόματα των branch που βρίσκονται στο Burnaby:

branch	bname	assets	bcity
	P.		¬ Burnaby

- για να βρούμε όλους τους πελάτες που έχουν λογαριασμό και στο SFU και στο MetroTown branch:

deposit	bname	account#	cname	balance
	SFU		P._x	
	MetroTown		_x	

- για να βρούμε όλους τους πελάτες που έχουν λογαριασμό σε ένα από τους δύο branch ή και στα δύο:

deposit	bname	account#	cname	balance
	SFU		P._x	
	MetroTown		P._y	

(Το P. πριν από τη μεταβλητή προκαλεί την εκτύπωση της ιδιότητας)

Το QbE είναι μία πολύ συνηθισμένη τεχνική που παρουσιάζεται στις γλώσσες ερωτήσεων.

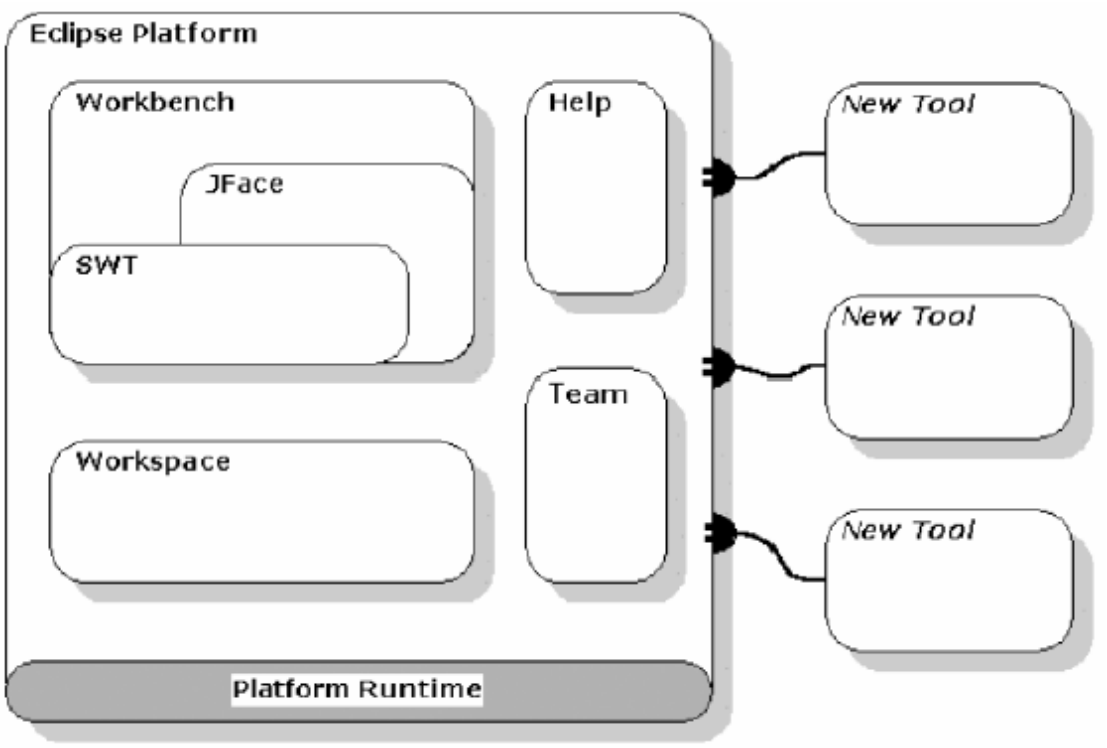
2.2.8 Πλατφόρμα Eclipse

Η πλατφόρμα Eclipse [15] είναι ένα εργαλείο ανάπτυξης λογισμικού. Βασικό χαρακτηριστικό της είναι το γεγονός ότι είναι χτισμένη πάνω σε ένα μηχανισμό ο οποίος επιτρέπει την εύρεση, την ανάπτυξη και την εκτέλεση μονάδων λογισμικού οι οποίες ονομάζονται plug-ins. Εκτός από ένα μικρό πυρήνα (Platform Runtime) όλη η λειτουργικότητα της Eclipse πλατφόρμας βρίσκεται σε plug-ins τα οποία έχουν εγκατασταθεί σε αυτήν. Κατά την εκκίνηση της πλατφόρμας Eclipse, ο πυρήνας της (Platform Runtime) ανακαλύπτει το σύνολο των διαθέσιμων plug-ins. Ένα plug-in ενεργοποιείται μόνο όταν υπάρχει ανάγκη να τρέξει ο κώδικας του. Τα βασικά χαρακτηριστικά της πλατφόρμας είναι τα ακόλουθα:

- Workbench: η επιφάνεια εργασίας της πλατφόρμας, δηλαδή το σύνολο των γραφικών στοιχείων της πλατφόρμας με τα οποία έρχεται σε επαφή ο χρήστης κατά τη διάρκεια της εργασίας του με την πλατφόρμα
- Workspace: χώρος εργασίας στο τοπικό σύστημα αρχείων, στον οποίο αποθηκεύονται πληροφορίες
- SWT (Standard Widget Toolkit): σύνολο από γραφικά δομικά στοιχεία υλοποιημένα με τέτοιο τρόπο ώστε να επιτρέπεται η ενσωμάτωσή τους με το τοπικό λειτουργικό σύστημα
- JFace: σύνολο γραφικών διεπαφών εμπλουτισμένα με κάποιες βασικές λειτουργίες, τις οποίες μπορεί εύκολα ο χρήστης να χρησιμοποιήσει, όπως δεντρικές αναπαραστάσεις, πίνακες, editors, παράθυρα ιδιοτήτων, κ.α.

- Help: μηχανισμός βοήθειας της πλατφόρμας που επιτρέπει παράλληλα τον εμπλουτισμό της με νέα εγχειρίδια
- Team: δυνατότητα διαχείρισης/ανάπτυξης μίας εργασίας από πολλούς χρήστες ταυτόχρονα μέσω της χρήσης κατάλληλων μηχανισμών αποθήκευσης διαμοιραζόμενων πόρων

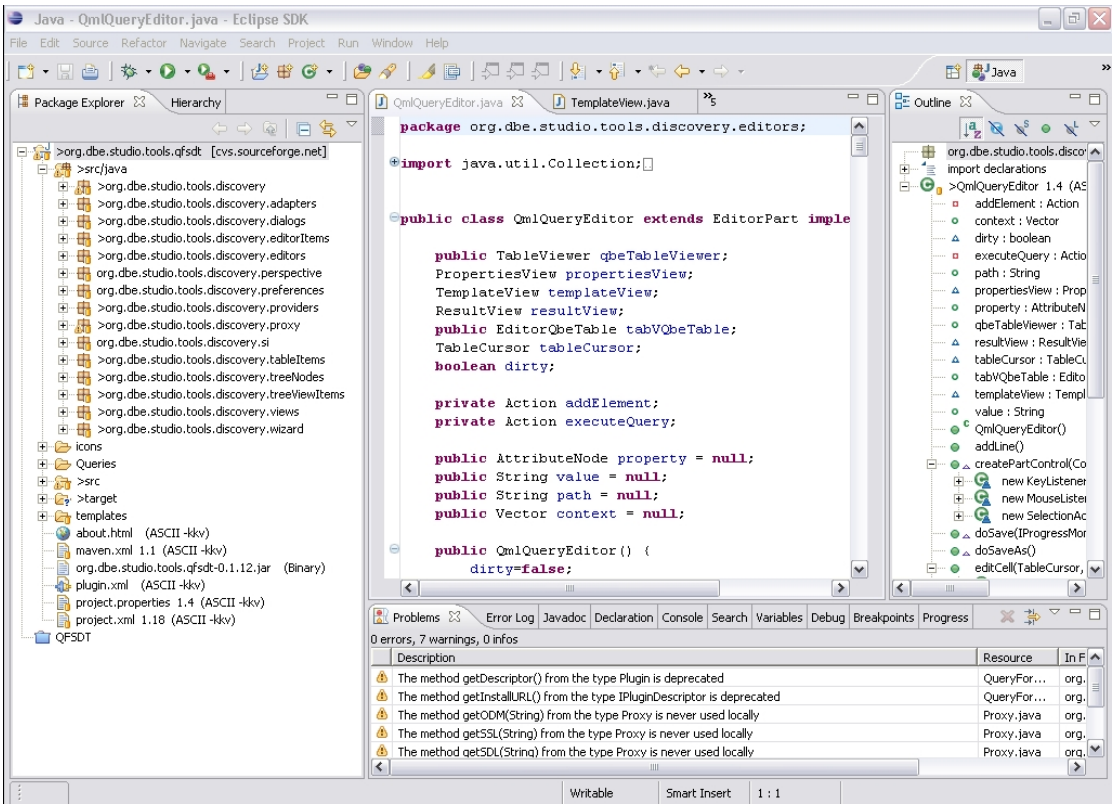
Η αρχιτεκτονική του Eclipse παρουσιάζεται στην εικόνα 4.



Εικόνα 4: Γενική αρχιτεκτονική του Eclipse.

Πάνω στο μικρό πυρήνα (Platform Runtime) του Eclipse προστίθενται plug-ins που εμπλουτίζουν τη λειτουργικότητά του.

Ένα παράδειγμα του interface του Eclipse παρουσιάζεται στην εικόνα 5. Σε αυτό φαίνονται δεντρικές αναπαραστάσεις, Editors και διάφορα παράθυρα ιδιοτήτων.



Εικόνα 5: Η πλατφόρμα ανάπτυξης λογισμικού Eclipse.

Το περιβάλλον ανάπτυξης εφαρμογών του Eclipse αποτελείται από ένα σύνολο Editor και παραθύρων ιδιοτήτων, που περιέχουν διάφορες γραφικές αναπαραστάσεις όπως δεντρικές, πίνακες κ.α.

2.2.9 OpenLaszlo

Το OpenLaszlo [16] είναι μία πλατφόρμα ανοικτού λογισμικού για τη δημιουργία εφαρμογών ιστού που παρέχουν τις δυνατότητες των γραφικών διεπαφών εφαρμογών γραφείου (desktop) χωρίς να απαιτούν εγκατάσταση.

Τα προγράμματα που έχουν αναπτυχθεί σε OpenLaszlo είναι γραμμένα στις γλώσσες XML [17] και JavaScript [18] και μεταγλωττίζονται με διαφανή τρόπο σε Macromedia Flash [19]. Το OpenLaszlo είναι ανεξάρτητο λειτουργικού συστήματος καθώς και ανεξάρτητο Web Browser. Μόνο απαιτούμενο για τον browser να υποστηρίζει Macromedia Flash.

Ένα παράδειγμα του interface του OpenLaszlo παρουσιάζεται στην εικόνα 6.

Εικόνα 6: Εφαρμογή που υλοποιήθηκε στην πλατφόρμα του OpenLaszlo.

Η εφαρμογή αυτή γράφτηκε σε XML και Javascript και μεταγλωττίστηκε σε γραφικά (Macromedia Flash).

2.3 Σχετιζόμενες εργασίες

Στις ακόλουθες υπο-ενότητες θα αναφερθούν μερικές σχετιζόμενες εργασίες με την δικιά μας. Επειδή η εργασία μας αναφέρεται στην κατασκευή γραφικών διεπαφών για πρόσβαση σε πληροφορία σε ένα αντικειμενοσταφές περιβάλλον οι εργασίες που αναζητούσαμε θα έπρεπε να είναι σχετικές με:

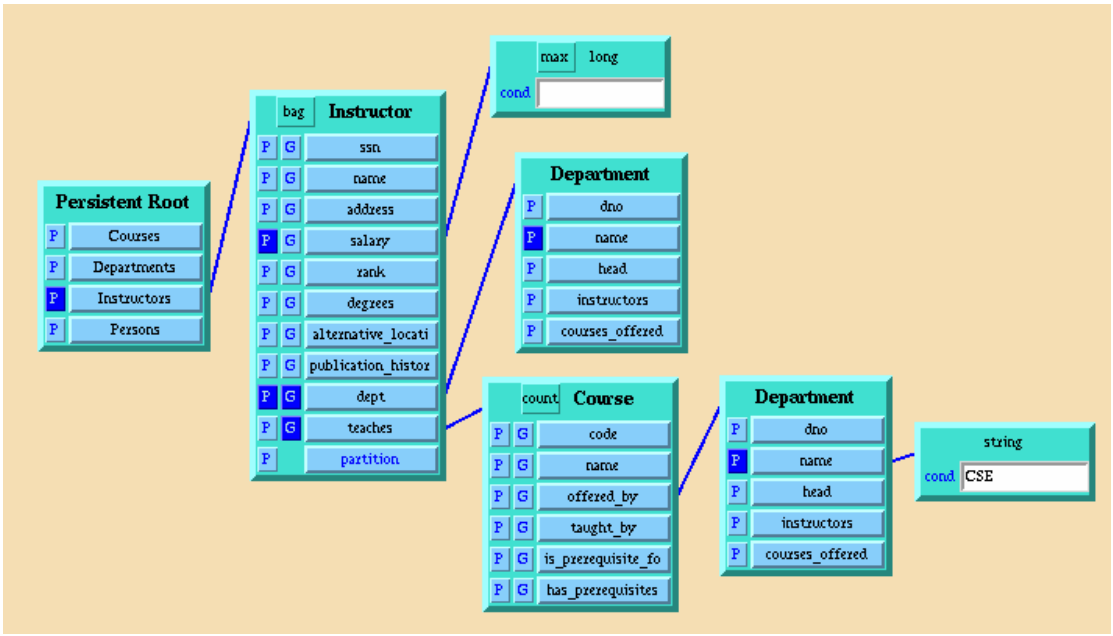
- γραφικές διεπαφές για κατασκευή ερωτήσεων
- γραφικές γλώσσες ερωτήσεων
- πρόσβαση πληροφορίας σε αντικειμενοστραγή περιβάλλοντα

Για τις εργασίες αυτές αναφέρουμε μία σύντομη περιγραφή και παραθέτουμε τα βασικά πλεονεκτήματα και μειονεκτήματα τους σε σχέση με τις προδιαγραφές της εργασίας μας.

2.3.1 Voodoo (Visual Object-Oriented Database language for ODMG OQL)

Το Voodoo (Visual Object-Oriented Database language for ODMG OQL) [20] είναι ένα εργαλείο κατασκευής ερωτήσεων για τη γλώσσα OQL (Object Query Language) [21]. Παρέχει μία γραφική διεπαφή για την δημιουργία OQL ερωτήσεων σε αντικειμενοστραφείς βάσεις δεδομένων. Καλύπτει σχεδόν όλες τις OQL ερωτήσεις και είναι αρκετά εύκολο για κάποιον να το μάθει και να το χρησιμοποιήσει.

Χρησιμοποιεί έναν ενιαίο τρόπο για την έκφραση των ερωτήσεων με τη χρήση ενός γενικού γραφικού στοιχείου. Η δομή των δεδομένων αναπαρίσταται εμφανίζοντας κάθε αντικείμενο σε ξεχωριστό πίνακα. Μεταξύ των πινάκων υπάρχουν σχέσεις διαφόρων τύπων. Τα κριτήρια αναπαρίστανται δίπλα από το στοιχείο της δομής στο οποίο αναφέρονται με τη δημιουργία ενός νέου γραφικού αντικειμένου. Ξεκινώντας από τη ρίζα ενός δέντρου που υπάρχει για κάθε ερώτηση (Persistent Root) ο χρήστης πλοηγείται στην πληροφορία, δημιουργώντας νέα γραφικά στοιχεία για κάθε αντικείμενο που επιλέγει. Μία OQL ερώτηση στο περιβάλλον VOODOO φαίνεται στην εικόνα 7.



Εικόνα 7: Αναπαράσταση OQL ερώτησης στο περιβάλλον VOODOO.

Εμφάνισε(P) όλους τους Instructors ομαδοποιημένους (G) σύμφωνα με το department name και το σύνολο των CSE courses που κάθε instructor διδάσκει(teaches).

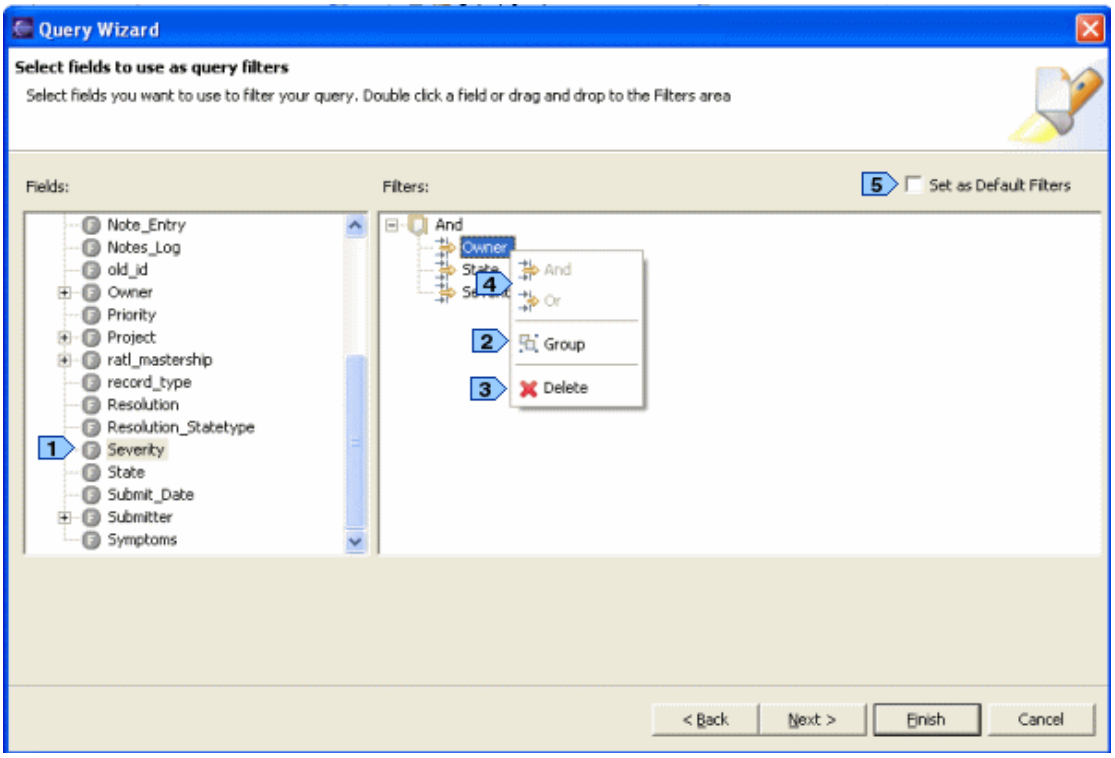
Αυτή η προσέγγιση είναι αποτελεσματική για τη δημιουργία κριτηρίων σε διαφορετικά αντικείμενα της δομής, παρέχοντας έναν εύκολο τρόπο να οριστούν οι συσχετίσεις τους, αλλά είναι περιοριστική για το χρήστη γιατί όταν η δομή ή τα αντικείμενα είναι μεγαλύτερα ή όταν ο χρήστης θέλει να ορίσει πολλά κριτήρια (πιο πολύπλοκη ερώτηση), ο γραφικός τρόπος αναπαράστασης της ερώτησης γίνεται πολύ μεγάλος επειδή για κάθε αντικείμενο εμφανίζονται όλες οι πληροφορίες που περιέχει, πολλές από τις οποίες δεν χρησιμοποιεί ο χρήστης στην ερώτηση καθιστώντας τη δημιουργία και διαχείριση της ερώτησης σχετικά δύσκολη και μεγάλη σε έκταση.

2.3.2 IBM Rational ClearQuest Client for Eclipse

Το IBM Rational ClearQuest [22] είναι ένα σύστημα (tracking system) που οργανώνει και αυτοματοποιεί τη διαχείριση συστήματος.

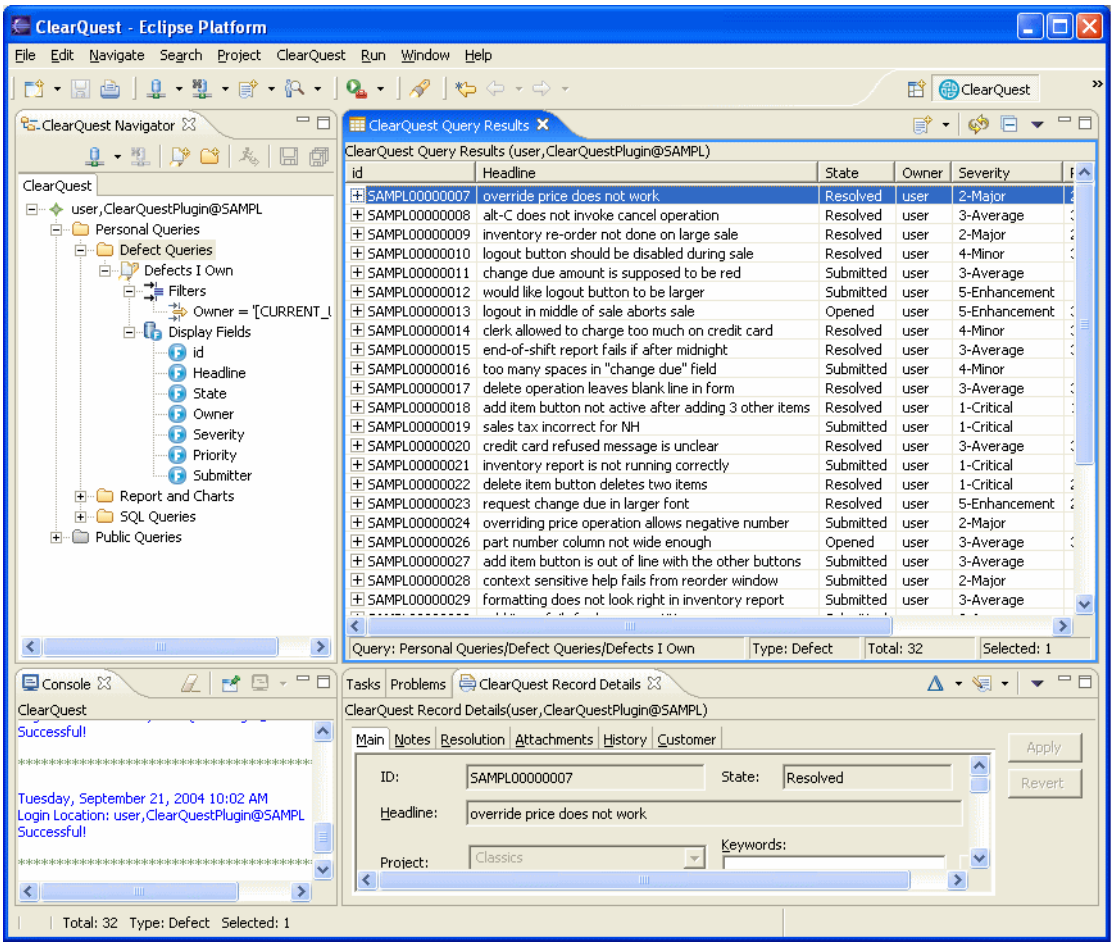
Για την δημιουργία ερωτήσεων χρησιμοποιεί έναν οδηγό (wizard), όπου επιλέγει από μία δεντρική αναπαράσταση πληροφοριών, τα πεδία στα οποία θέλει να θέσει περιορισμούς. Του δίνει ακόμα τη δυνατότητα να ομαδοποιήσει τους περιορισμούς με συζεύξεις και διαζεύξεις (and, or, group), όπως φαίνεται στην εικόνα 8.

Στο επόμενο βήμα του οδηγού, ο χρήστης ορίζει τις τιμές που επιθυμεί να έχουν τα πεδία που έχει επιλέξει στο προηγούμενο βήμα.



Εικόνα 8: Ο οδηγός του IBM Rational ClearQuest για τη δημιουργία μίας ερώτησης.
Ο χρήστης μπορεί να ομαδοποιήσει τα κριτήρια που έχει θέσει σύμφωνα με τις προτιμήσεις του.

Αφού έχει δημιουργήσει την ερώτηση ο χρήστης, τα αποτελέσματα εμφανίζονται σε έναν πίνακα, όπου κάθε στοιχείο περιέχει μία δεντρική αναπαράσταση με περισσότερες πληροφορίες ενώ ταυτόχρονα μπορεί να έχει και μία επισκόπηση της ερώτησης σε μία δεντρική πάλι δομή. Επίσης ο χρήστης μπορεί να ορίσει και τα περιεχόμενα τα αποτελεσμάτων εφόσον επιθυμεί. Τα αποτελέσματα φαίνονται στην εικόνα 9.



Εικόνα 9: Τα αποτελέσματα του IBM Rational ClearQuest.

Τα πεδία που εμφανίζονται στα αποτελέσματα έχουν οριστεί από το χρήστη και εμφανίζονται στη δεντρική δομή αριστερά. Ο χρήστης μπορεί να πλοηγηθεί στα αποτελέσματα βλέποντας περισσότερες λεπτομέρειες για αυτά.

Σε αυτή την υλοποίηση, η δεντρική αναπαράσταση των πληροφοριών τόσο στην επιλογή των ιδιοτήτων στη δημιουργία των περιορισμών, όσο και στην επισκόπηση της ερώτησης, παρέχει στο χρήστη όλες οι πληροφορίες με απλό, φιλικό και κατανοητό τρόπο, παρέχοντας παράλληλα και έναν εύκολο τρόπο πλοήγησης στην πληροφορία διατηρώντας την ιεραρχία.

Η χρήση του οδηγού για τη δημιουργία μιας ερώτησης είναι μια πολύ φιλική προς το χρήστη τεχνική επειδή καθοδηγεί το χρήστη για να εκτελέσει τις απαραίτητες ενέργειες με σωστή σειρά, μην επιτρέποντάς του να κάνει τα λάθη. Επιπλέον μια ερώτηση που έχει δημιουργηθεί μπορεί να επαναχρησιμοποιηθεί και έχει τη δυνατότητα να την τροποποιήσει.

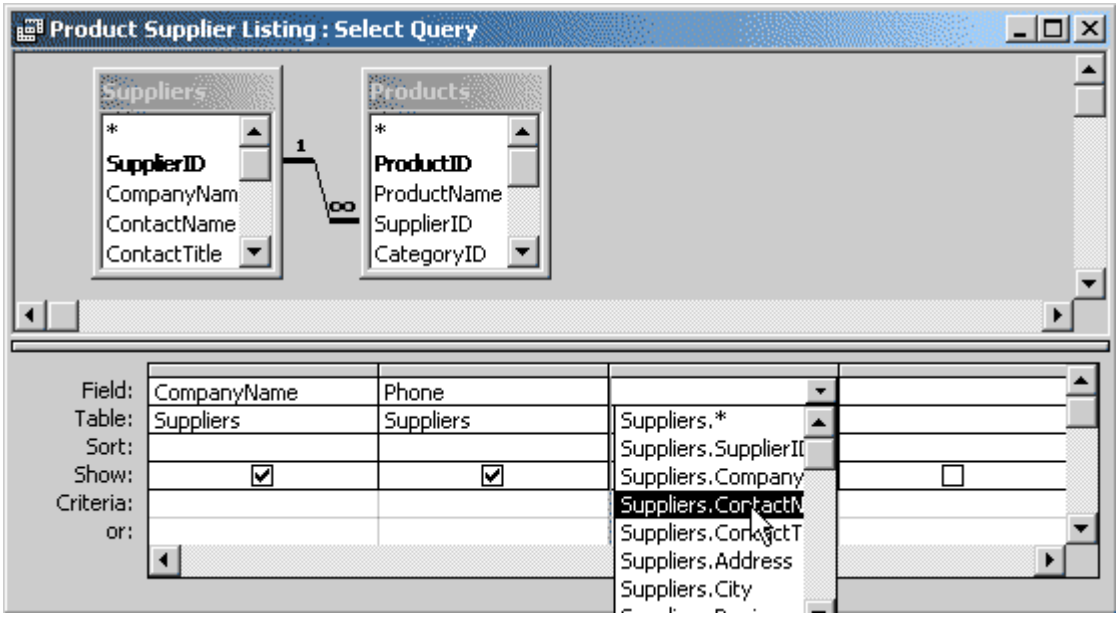
Τέλος, υπάρχει μια πολύ ενδιαφέρουσα προσέγγιση της εμφάνισης των αποτελεσμάτων. Ανάλογα με το εάν ο χρήστης θέλει, το δέντρο αποτελεσμάτων

μπορεί να επεκταθεί προκειμένου να εμφανιστούν περισσότερες πληροφορίες και να εκτελεστούν οι ενέργειες για το επιλεγμένο στοιχείο αποτελέσματος.

Από την άλλη πλευρά όμως το σύστημα αντίθετα με τις δικές μας προδιαγραφές λειτουργεί σε σχεσιακό περιβάλλον. Επίσης στην κατασκευή των ερωτήσεων ο τρόπος δημιουργίας περιορισμών δεν είναι φιλικός προς τον απλό χρήστη γιατί χρησιμοποιεί όρους όπως το AND (λογική σύζευξη) και το OR (λογική διάζευξη), που προέρχονται από το πεδίο της επιστήμης πληροφορικής, οι οποίες ίσως δεν είναι πάντα γνωστές στον απλό χρήστη και είναι πολύ πιθανό να τον αποπροσανατολίσουν και να τον οδηγήσουν σε δημιουργία λανθασμένων ερωτήσεων από αυτές που ήθελε να δημιουργήσει.

2.3.3 Microsoft Access

Το Microsoft Access [23] είναι ένα περιβάλλον που χρησιμοποιείται για τη δημιουργία και τη διαχείριση βάσεων δεδομένων. Το Microsoft Access έχει έναν ισχυρό μηχανισμό προκειμένου να δημιουργεί ερωτήσεις χρησιμοποιώντας μία γραφική διεπαφή με τον χρήστη. Ένας χρήστης δεν χρειάζεται να έχει τη γνώση μιας γλώσσας ερωτήσεων. Το μόνο πράγμα που πρέπει να κάνει είναι να καθορίσει τους πίνακες, τους περιορισμούς για αυτούς και τη συσχέτιση μεταξύ τους. Κατά συνέπεια δημιουργεί περίπλοκες ερωτήσεις με έναν εύκολο τρόπο χωρίς λάθη. Βασίζεται στο Query by Example (QbE) και παρέχει έναν εύκολο γραφικό τρόπο για να ορίσει σχέσεις μεταξύ των πινάκων. Στην εικόνα 10 παρουσιάζεται το περιβάλλον της Microsoft Access.



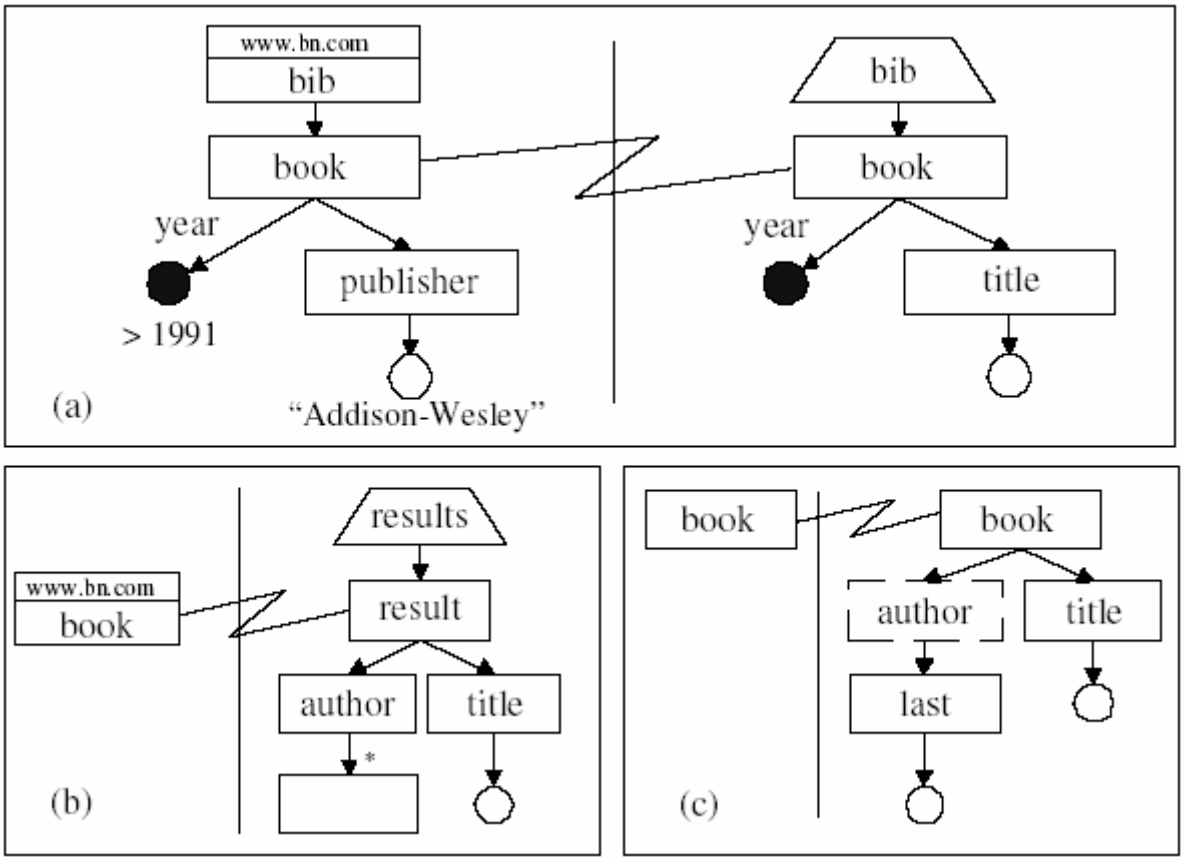
Εικόνα 10: Το περιβάλλον της Microsoft Access.
Ένας γραφικός τρόπος για την κατασκευή περιγραφικών και πολύπλοκων ερωτήσεων.

Το Microsoft Access προτείνει έναν εύκολο και περιγραφικό τρόπο για την κατασκευή πολύπλοκων ερωτήσεων, χωρίς να απαιτεί από το χρήστη εξειδικευμένες γνώσεις και είναι ευρέως διαδεδομένο. Όμως λειτουργεί για σχεσιακή βάση δεδομένων και αποτελεί εμπορικό προϊόν αντίθετα με τις προδιαγραφές μας.

2.3.4 XQbE (XQuery by Example)

Το XQbE (XQuery by Example) [24] είναι μία γραφική γλώσσα ερωτήσεων για την έκφραση ενός μεγάλου υποσυνόλου της XQuery με γραφικό τρόπο. Έχει δημιουργηθεί έτσι ώστε να μπορεί να χρησιμοποιηθεί και από απλούς χρήστες που δεν έχουν εξειδικευμένες γνώσεις και είναι επηρεασμένο από το QbE.

Χρησιμοποιεί δεντρικές αναπαραστάσεις ως βασικά γραφικά στοιχεία (όπως το QbE χρησιμοποιεί πίνακες) για να υποστηρίξει την ιεραρχική δομή της XML. Μία ή περισσότερες δεντρικές αναπαραστάσεις ορίζουν τα έγγραφα (documents) στην είσοδο της ερώτησης και μία δεντρική αναπαράσταση ορίζει το έγγραφο στην έξοδο της ερώτησης. Μπορούν να οριστούν περιορισμοί στις συσχετίσεις μεταξύ στοιχείων των αναπαραστάσεων καθώς και σε κάθε στοιχείο ξεχωριστά υποστηρίζοντας με τον τρόπο αυτό μεγάλο υποσύνολο της Xquery. Η δημιουργία ερωτήσεων με βάση το XQbE παρουσιάζεται στην εικόνα 11.



Εικόνα 11: Δημιουργία ερωτήσεων με χρήση του XQbE.

(a) Εμφάνιση των βιβλίων που εκδόθηκαν από τον "Addison-Wesley" μετά το 1991, συμπεριλαμβανομένου της χρονιάς και του τίτλου τους.

(b) Δημιουργία αντικειμένων result στα αποτελέσματα που περιέχουν τα αντικείμενα author και τον τίτλο.

(c) Εμφάνιση κάθε βιβλίο του τίτλου και των επωνύμων των συγγραφέων(διατηρώντας τη σειρά των βιβλίων όπως στο αρχικό document).

Το XQbE παρέχει μία διεπαφή φιλική προς το χρήστη και επιτρέπει τη δημιουργία μίας ερώτησης με εύκολο τρόπο. Όμως ακόμη και για τη δημιουργία μίας πολύ απλής ερώτησης χρειάζεται η χρήση πολλών γραφικών στοιχείων για την αναπαράσταση ολόκληρης της ιεραρχίας μέχρι το στοιχείο στο οποίο θέλουμε να θέσουμε περιορισμούς, το οποίο καθιστά την αναπαράσταση των ακόμα και απλών ερωτήσεων μεγάλη σε έκταση.

2.4 Ανακεφαλαίωση

Στο κεφάλαιο αυτό αναφέρθηκαν οι τεχνολογίες που χρησιμοποιήθηκαν κατά το σχεδιασμό και υλοποίηση της παρούσας εργασίας. Παρουσιάστηκαν η

αρχιτεκτονική μοντελοποίησης συστημάτων MDA, η MOF αρχιτεκτονική μεταδεδομένων, το MDR και ο τρόπος επικοινωνίας με αυτό στο DBE (KB/SR), η γλώσσα ερωτήσεων QML, το MVC πρότυπο σχεδίασης, η τεχνική QbE και οι πλατφόρμες υλοποίησης Eclipse και Laszlo.

Επίσης αναφέρθηκαν κάποιες υπάρχουσες εργασίες με σχετιζόμενο αντικείμενο μελέτης με το αντικείμενο της εργασίας αυτής τις οποίες θεωρήσαμε σημαντικό να παρουσιάσουμε και για τις οποίες αναφέραμε μία σύντομη περιγραφή και παραθέσαμε τα βασικά πλεονεκτήματα και μειονεκτήματα τους σε σχέση με τις ανάγκες της εργασίας μας. Όμως καμία από αυτές δεν κάλυπτε τις ανάγκες μας για τη δημιουργία μίας γραφικής διεπαφής φιλικής προς το χρήστη που να μπορεί να αναπαραστήσει με κατανοητό τρόπο την πολύπλοκη και μεγάλη σε έκταση MOF πληροφορία και να παρέχει μηχανισμούς ώστε να μπορεί ο χρήστης να τη χρησιμοποιήσει για τη δημιουργία ερωτήσεων.

Στο επόμενο κεφάλαιο θα παρουσιαστεί η αρχιτεκτονική του συστήματός μας, στην οποία θα περιγράφεται τον τρόπο με τον οποίο σχεδιάστηκε και γιατί, τις μονάδες που την αποτελούν και τον τρόπο με τον οποίο αλληλεπιδρούν μεταξύ τους.

Κεφάλαιο 3

Αρχιτεκτονική συστήματος

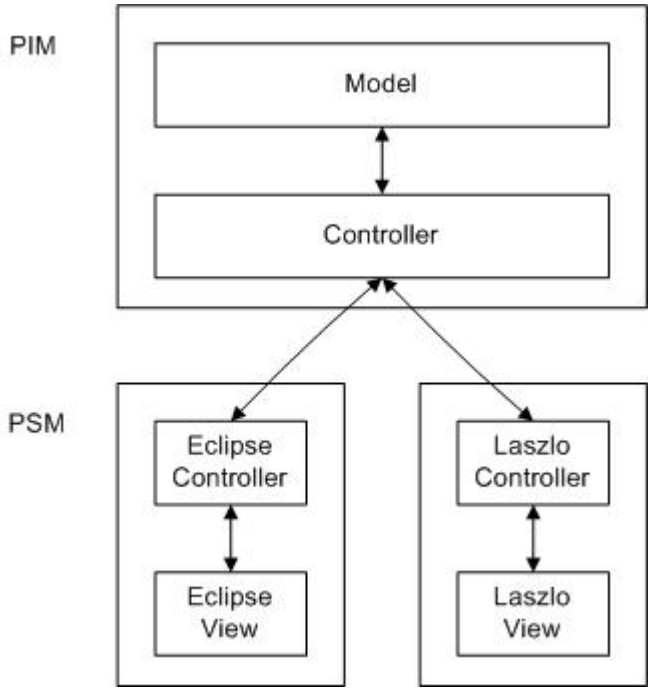
3.1 Εισαγωγή

Κατά το σχεδιασμό της εφαρμογής τρία ήταν τα βασικά μελήματα που έπρεπε να ικανοποιηθούν. Αρχικά, η γραφική διεπαφή θα πρέπει να παρέχει την λειτουργικότητα μεγάλου μέρους της QML (Query Metamodel Language) [7] ώστε οι ερωτήσεις που θα μπορεί να διαμορφώσει ένας χρήστης με γραφικό τρόπο να είναι αρκετά ισχυρές (σαν να τις διατύπωνε με QML) αλλά χωρίς να απαιτεί από τους χρήστες τις γνώσεις του συντακτικού της γλώσσας. Δεύτερον, η αξιοποίηση MOF (MetaObject Facility) [6] πληροφορίας όλων των επιπέδων για την διευκόλυνση των χρηστών κατά την διαμόρφωση της ερώτησής τους. Τέλος, το τρίτο μέλημα ήταν ο σχεδιασμός να γίνει με τέτοιο τρόπο ώστε να μπορούν να υλοποιηθούν διεπαφές από περισσότερα του ενός περιβάλλοντα. Για τις ανάγκες του DBE υλοποιήθηκαν δύο εφαρμογές, η μία σε περιβάλλον της πλατφόρμας Eclipse και η άλλη σε περιβάλλον διαδικτύου.

3.2 Γενική αρχιτεκτονική

Όπως έχει αναφερθεί, βασικός στόχος της παρούσας εργασίας είναι η δημιουργία γραφικών διεπαφών για πρόσβαση σε MOF (MetaObject Facility) [6] βάσεις γνώσης εκμεταλλευόμενοι τη δομή της πληροφορίας που περιέχουν για την κατασκευή ερωτήσεων.

Για την υποστήριξη περισσότερων της μίας υλοποίησης ακολουθήσαμε το MVC (Model-View-Controller) [5] πρότυπο σχεδίασης, χωρίζοντας το σύστημα σε τρία διακριτά μέρη, αυτό που κρατάμε την πληροφορία (Model), τη γραφική διεπαφή (View) και το τμήμα που αναλαμβάνει την επικοινωνία μεταξύ των δύο προηγουμένων (Controller), το οποίο μπορεί να επεκταθεί για να καλύψει επιπλέον ανάγκες της εκάστοτε υλοποίησης, όπως κάναμε εμείς για τις δικές μας υλοποιήσεις. Έτσι ουσιαστικά υπάρχει ένα τμήμα υλοποίησης ανεξάρτητου πλατφόρμας, PIM (Platform Independent Model) και ένα τμήμα που αναφέρεται στην εκάστοτε υλοποίηση, PSM (Platform Specific Model), το οποίο παρουσιάζεται στην εικόνα 12.

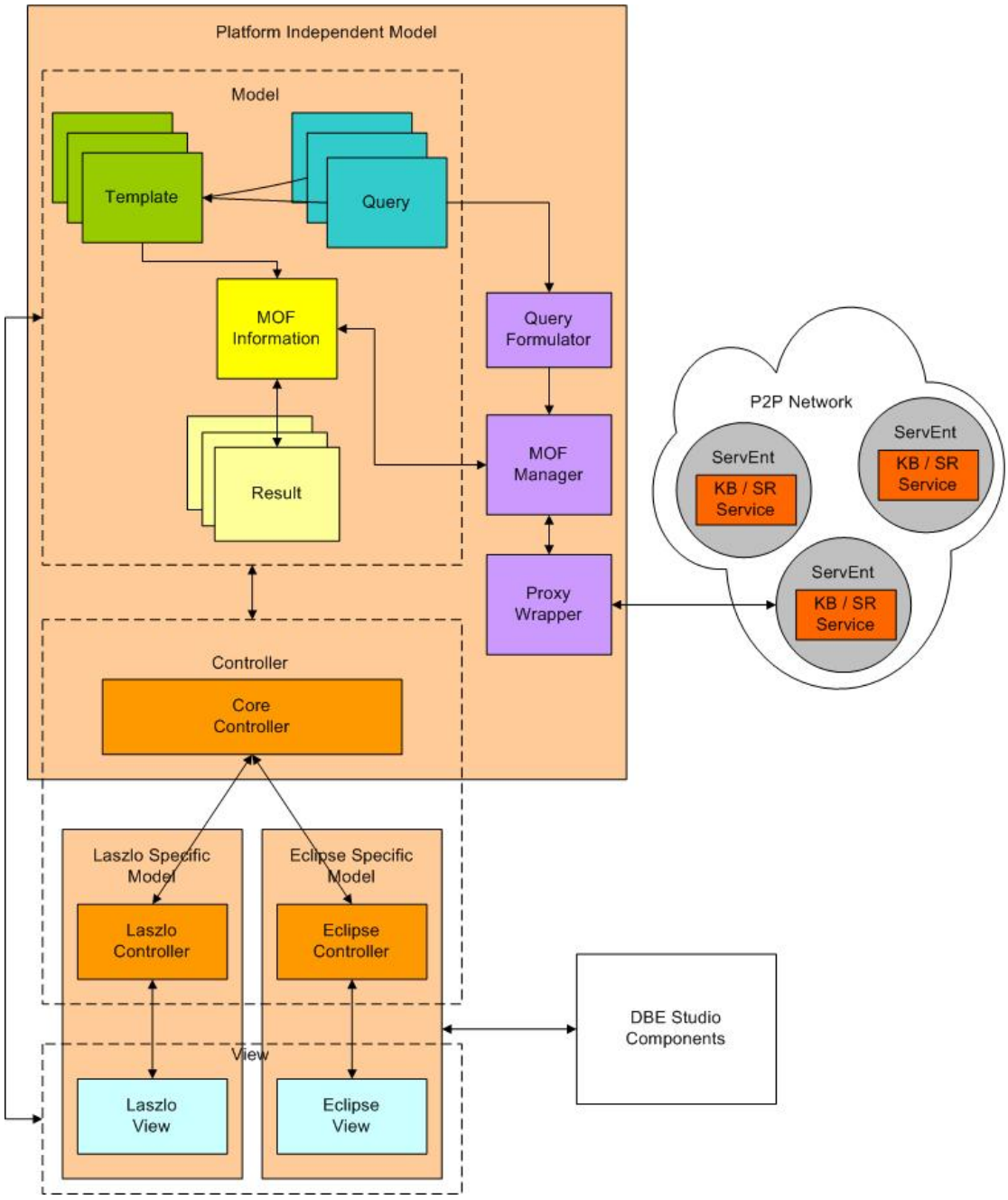


Εικόνα 12: Εφαρμογή MVC προτύπου σχεδίασης στην αρχιτεκτονική του συστήματός μας.

Κάθε υλοποίηση χωρίζεται σε τρία μέρη, Model που είναι ανεξάρτητο πλατφόρμας υλοποίησης, View που αναφέρεται σε συγκεκριμένη πλατφόρμα και Controller που αποτελείται από ένα κομμάτι που είναι κοινό για όλες τις υλοποιήσεις και ένα κομμάτι που αναφέρεται σε συγκεκριμένη πλατφόρμα.

Η αρχιτεκτονική του συστήματός μας αναλυτικότερα παρουσιάζεται στην εικόνα 13. Ο χρήστης κάνει τις επιλογές για τη δημιουργία ερώτησης ή εξερεύνησης της MOF πληροφορίας μέσω της διεπαφής (GUI). Το GUI ενημερώνει τον κατάλληλο controller για τις ενέργειες του χρήστη και εκείνος προωθεί τα ανάλογα αιτήματα στο μοντέλο πίσω από την εφαρμογή. Στο μοντέλο οι MOF πληροφορίες (μετα-μοντέλα, μοντέλα, κλπ) αποθηκεύονται στη μονάδα του MOF information. Τα templates περιέχουν αναφορές προς επιλεγμένα στοιχεία της MOF πληροφορίας και τα queries περιέχουν τιμές για τα στοιχεία των template. Επίσης υπάρχει μία δομή για την αποθήκευση των αποτελεσμάτων των ερωτήσεων (Result). Για την πρόσβαση στα δεδομένα που είναι αποθηκευμένα στη βάση γνώσης του DBE χρησιμοποιείται ο MOF Manager. Αυτός επίσης αναλαμβάνει να προωθήσει και τις ερωτήσεις που δημιουργεί ο Query Formulator με τη χρήση των templates και των queries που αναφέρονται σε αυτά. Ο MOF Manager προωθεί τα αιτήματα προς τη βάση γνώσης μέσω του Proxy Wrapper που αναλαμβάνει την επικοινωνία με αυτήν. Η βάση γνώσης στο DBE είναι κατανεμημένη σε ένα peer-to-peer (P2P) δίκτυο, όπου σε κάθε κόμβο (ServEnt) υπάρχει τοπική πληροφορία και κατάλληλοι μηχανισμοί για την επικοινωνία με αυτόν.

Ο σχεδιασμός του συστήματος έγινε με τέτοιο τρόπο ώστε να αποτελείται από αυτοδύναμες μονάδες και να μην υπάρχει εξάρτηση της μίας από την άλλη. Έτσι αν προκύψει μία αλλαγή σε μία μονάδα, π.χ. χρειαστεί να αλλάξει η γλώσσα ερωτήσεων, τότε θα χρειαστεί να αλλάξουμε μόνο τη μονάδα του Query Formulator. Επίσης η αναπαράσταση της δομής μίας ερώτησης ήταν επιθυμητό να αποτελείται από δύο διακριτές μονάδες. Στη μία μονάδα περιέχονται τα στοιχεία MOF (μετα-μοντέλο, μοντέλο, ή οντολογία) στα οποία θέλουμε να θέσουμε περιορισμούς (Template) και στην άλλη τους περιορισμούς αυτούς (Query). Ο διαχωρισμός αυτός επιτρέπει την ανεξαρτησία των στοιχείων που έχουμε επιλέξει από τις τιμές των περιορισμών που θέλουμε να πληρούν. Το πλεονέκτημα που δημιουργείται με αυτήν τη μοντελοποίηση είναι η δυνατότητα επαναχρησιμοποίησης ενός Template για τη δημιουργία πολλών ερωτήσεων (Queries).



Εικόνα 13: Γενική αρχιτεκτονική συστήματος.

Παρουσιάζονται οι μονάδες που αποτελούν το σύστημα και ο τρόπος επικοινωνίας τόσο μεταξύ τους όσο και με τη βάση γνώσης. Επίσης παρουσιάζεται ο τρόπος με τον οποίο η αρχιτεκτονική αυτή ακολουθεί το MVC πρότυπο σχεδίασης.

Σε γενικές γραμμές τα συστατικά μέρη του συστήματος μπορούν να ομαδοποιηθούν στις εξής δομικές μονάδες:

- Template: δομή που αποτελείται από αναφορές σε στοιχεία MOF (κάποιο μετα-μοντέλο, μοντέλο ή οντολογία).
- Query: δομή που περιέχει περιορισμούς που αναφέρονται στα στοιχεία ενός συγκεκριμένου template.

- MOF information: δομή που περιέχει την καθαυτό MOF πληροφορία, μετα-μοντέλα, μοντέλα, οντολογίες, δεδομένα.
- Result: δομή που αντιπροσωπεύει τα αποτελέσματα των αναζητήσεων.
- Query Formulator: δομή που είναι υπεύθυνη για τον σχηματισμό (σύνταξη) των ερωτήσεων στη γλώσσα ερωτήσεων (QML).
- MOF Manager: διαχειριστής της αποθηκευμένης MOF πληροφορίας (μετα-μοντέλα, μοντέλα, οντολογίες, δεδομένα).
- Proxy Wrapper: δομή που αναλαμβάνει την επικοινωνία με τη peer-to-peer δίκτυο από τις βάσεις γνώσης (KB/SR)
- Core Controller: δομή που ενημερώνει ανάλογα με τα συμβάντα το model και το View της εκάστοτε υλοποίησης.
- Laszlo Controller: επέκταση του Core Controller για την υλοποίηση που έγινε στην πλατφόρμα Open Laszlo.
- Laszlo View: η γραφική διεπαφή που υλοποιήθηκε στην πλατφόρμα Open Laszlo.
- Eclipse Controller: επέκταση του Core Controller για την υλοποίηση που έγινε στην πλατφόρμα Eclipse.
- Eclipse View: η γραφική διεπαφή που υλοποιήθηκε στην πλατφόρμα Eclipse. Το Eclipse View επικοινωνεί και με ένα σύνολο άλλων plugin που έχουν υλοποιηθεί για το DBE.

Στη συνέχεια ακολουθεί μία πιο αναλυτική περιγραφή των παραπάνω μονάδων.

3.2.1 Template

Το Template περιέχει τα στοιχεία MOF στα οποία θέλουμε να θέσουμε περιορισμούς. Αυτά μπορεί να είναι κάποιο από τα υποστηριζόμενα μετα-μοντέλα, ένα συγκεκριμένο μοντέλο ή μία συγκεκριμένη οντολογία. Βασικός σκοπός της σχεδίασης του template είναι η απόκρυψη της πολυπλοκότητας του MOF μετα-μοντέλου, μοντέλου ή της οντολογίας, λόγω του μεγάλου όγκου πληροφορίας που συνήθως περιέχει. Γι'αυτόν το λόγο το template περιέχει αναφορές σε κάποια από τα στοιχεία αυτά ώστε να μην είναι απαραίτητη η πλοήγηση στο μοντέλο κάθε φορά που πρέπει να δημιουργηθεί ένα Query με βάση αυτό το Template.

3.2.2 Query

Το Query περιέχει τους περιορισμούς που αναφέρονται στα στοιχεία του Template. Με αυτό τον τρόπο ανεξαρτητοποιούμε τις τιμές των περιορισμών από τα στοιχεία στα οποία αναφέρονται αυτοί, έχοντας έτσι τη δυνατότητα να δημιουργούμε νέα queries που να αναφέρονται σε ίδιο template χωρίς να χρειάζεται να δημιουργούμε για κάθε query νέο template.

3.2.3 MOF Information

Η μονάδα MOF Information περιέχει δομές στις οποίες αποθηκεύονται τα MOF μετα-μοντέλα, μοντέλα, οντολογίες και δεδομένα, που έχουν ανακτηθεί από τη βάση γνώσης σε κάθε εκτέλεση της εφαρμογής, δομημένα με τέτοιο τρόπο ώστε να μπορούμε να τα χρησιμοποιήσουμε στη συνέχεια.

3.2.4 Result

Για κάθε αποτέλεσμα δημιουργείται μία δομή (το Result) η οποία περιέχει αναφορές στην πληροφορία που είναι αποθηκευμένη στο MOF Information, αλλά περιέχει και επιπλέον πληροφορία ώστε να ομαδοποιεί την πληροφορία αυτή με διαφορετική δομή (με διαφορετική ιεραρχία).

3.2.5 Query Formulator

Ο Query Formulator είναι ο μηχανισμός σχηματισμού ερωτήσεων με τον οποίο δημιουργούνται δομημένες ερωτήσεις σε QML (τη γλώσσα που χρησιμοποιεί το σύστημα για να κάνει ερωτήσεις) χρησιμοποιώντας τα στοιχεία που περιέχονται στο Template και στο Query της ερώτησης την οποία προωθεί στη βάση γνώσης μέσω του MOF Manager.

3.2.6 MOF Manager

Η μονάδα του MOF Manager διαχειρίζεται τη MOF πληροφορία (μετα-μοντέλα, μοντέλα, οντολογίες και δεδομένα) που είναι αποθηκευμένη στο MOF Information. Επίσης είναι υπεύθυνη για την προώθηση αιτημάτων και ερωτήσεων προς τη βάση γνώσης μέσω του Proxy Wrapper, όπως και για την διανομή της πληροφορίας που ανακτά από τη βάση γνώσης προς τις κατάλληλες μονάδες.

3.2.7 Proxy Wrapper

Η επικοινωνία του συστήματος με τη βάση γνώσης γίνεται μέσω μίας διεπαφής που παρέχει. Στο σύστημά μας την επικοινωνία με τη διεπαφή αυτή αναλαμβάνει η μονάδα του Proxy Wrapper ώστε οι υπόλοιπες μονάδες του συστήματος να μην επηρεάζονται από τυχόν αλλαγές που μπορεί να προκύψουν στην επικοινωνία αυτή.

3.2.8 Core Controller

Η μονάδα του Core Controller αναλαμβάνει να ενημερώνει το View και το Model του συστήματός μας για αλλαγές που συμβαίνουν σ'αυτά σύμφωνα με το MVC πρότυπο σχεδίασης. Όταν π.χ. έρθουν νέα αποτελέσματα ή όταν προστεθεί ένα νέο στοιχείο σε ένα template, ενημερώνει το View της εφαρμογής με τα νέα δεδομένα. Επίσης δέχεται αιτήματα από το View, οπότε και αναλαμβάνει να ενημερώσει το Model, π.χ. ο χρήστης δίνει γραφικά τιμές σε ένα query.

3.2.9 Laszlo Controller

Ο Laszlo Controller είναι μία επέκταση του Core Controller για την υλοποίηση που έγινε στην πλατφόρμα Laszlo. Αναλαμβάνει να ενημερώσει το Laszlo View για τις αλλαγές που γίνονται στο Model και να ενημερώνει το Model για τα events που γίνονται στο Laszlo View.

3.2.10 Laszlo View

Το Laszlo View είναι η γραφική διεπαφή που υλοποιήθηκε στην πλατφόρμα Open Laszlo. Είναι ουσιαστικά Html σελίδες στις οποίες περιέχονται γραφικά, μέσω των οποίων γίνεται η επικοινωνία με το χρήστη, σε Macromedia Flash [19]

3.2.11 Eclipse Controller

Ο Eclipse Controller είναι επέκταση του Core Controller για την υλοποίηση που έγινε στην πλατφόρμα Eclipse. Αναλαμβάνει να ενημερώσει το Eclipse View για τις αλλαγές που γίνονται στο Model και να ενημερώνει το Model για τα events που γίνονται στο Eclipse View.

3.2.12 Eclipse View

Το Eclipse View είναι η γραφική διεπαφή που υλοποιήθηκε στην πλατφόρμα Eclipse. Αποτελείται από ένα Perspective στο Eclipse μέσα στο οποίο, υπάρχουν Views και ένας Editor για την αναπαράσταση των επιθυμητών πληροφοριών.

3.3 Ανακεφαλαίωση

Στο κεφάλαιο αυτό παρουσιάστηκε η αρχιτεκτονική σχεδιασμού του συστήματός μας, η οποία ακολούθησε το MVC πρότυπο σχεδίασης ώστε να μπορούν να υποστηριχθούν περισσότερες της μίας υλοποίησης (στα πλαίσια της εργασίας μας έγιναν δύο). Αναφέρθηκε ο διαχωρισμός του συστήματος σε ανεξάρτητες μονάδες, ώστε αν αλλάξει κάτι στις προδιαγραφές να μη χρειαστεί αλλαγή σε όλο το σύστημα, η χρήση δύο ξεχωριστών μονάδων, του template (αναφορά στα στοιχεία που θέλουμε να θέσουμε περιορισμούς) και του query (οι περιορισμοί στα στοιχεία του template) για τη δημιουργία ερωτήσεων και στη συνέχεια περιγράφηκαν αναλυτικά όλες οι μονάδες της αρχιτεκτονικής και ο τρόπος με τον οποίο αλληλεπιδρούν μεταξύ τους. Τέλος, το σύστημα υποστηρίζει το μεγαλύτερο μέρος της QML επιτρέποντας έτσι

στους χρήστες, με εύκολο και προσιτό τρόπο, να ερωτούν προς MOF βάσεις γνώσης που υποστηρίζουν αυτή τη γλώσσα.

Στο επόμενο κεφάλαιο θα παρουσιαστεί η υλοποίηση που έγινε για τις δύο πλατφόρμες του Eclipse και του Laszlo, αφού πρώτα αναφερθεί η λειτουργικότητα που υποστηρίζουν.

Κεφάλαιο 4

Υλοποίηση συστήματος

4.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναλυθεί η υλοποίηση του συστήματός μας που έγινε ακολουθώντας την MVC αρχιτεκτονική που περιγράψαμε στο προηγούμενο κεφάλαιο. Αρχικά θα αναφερθεί η λειτουργικότητα που επιθυμούμε να υποστηρίξουμε, την οποία θα παρουσιάσουμε με την τεχνική των περιπτώσεων χρήσης. Στη συνέχεια θα αναλυθούν λεπτομερώς οι δύο υλοποιήσεις που έγιναν στα πλαίσια της παρούσας εργασίας, σε Eclipse και στο Laszlo, όπου θα παρουσιάσουμε τον τρόπο με τον οποίο γίνεται η αναπαράσταση της επιθυμητής πληροφορίας και οι δυνατότητες που δίνονται στο χρήστη για να αλληλεπιδράσει με την εκάστοτε εφαρμογή. Θα παρουσιαστεί επιπλέον το υποσύνολο της QML που ήταν επιθυμητό να υποστηρίχθηκε από τις υλοποιήσεις. Τέλος θα αναφερθούν κάποιες παρατηρήσεις και σχόλια για τις πλατφόρμες στις οποίες έγιναν οι υλοποιήσεις.

4.2 Λειτουργικότητα συστήματος

Η λειτουργικότητα του συστήματος παρουσιάζεται με τη τεχνική των περιπτώσεων χρήσης.

Οι περιπτώσεις χρήσης (use cases) είναι μία τεχνική που παρέχει έναν «εύκολο» και συνοπτικό τρόπο παρουσίασης των λειτουργιών μίας εφαρμογής. Έχουν μορφή πίνακα και περιγράφουν τα χαρακτηριστικά της λειτουργίας και όλα τα πιθανά βήματα που ακολουθούνται σε αυτήν κατά την εξέλιξη της διαδικασίας (σε αυτά συμπεριλαμβάνονται το σενάριο επιτυχούς εξέλιξης, αλλά και οι περιπτώσεις «λάθους»).

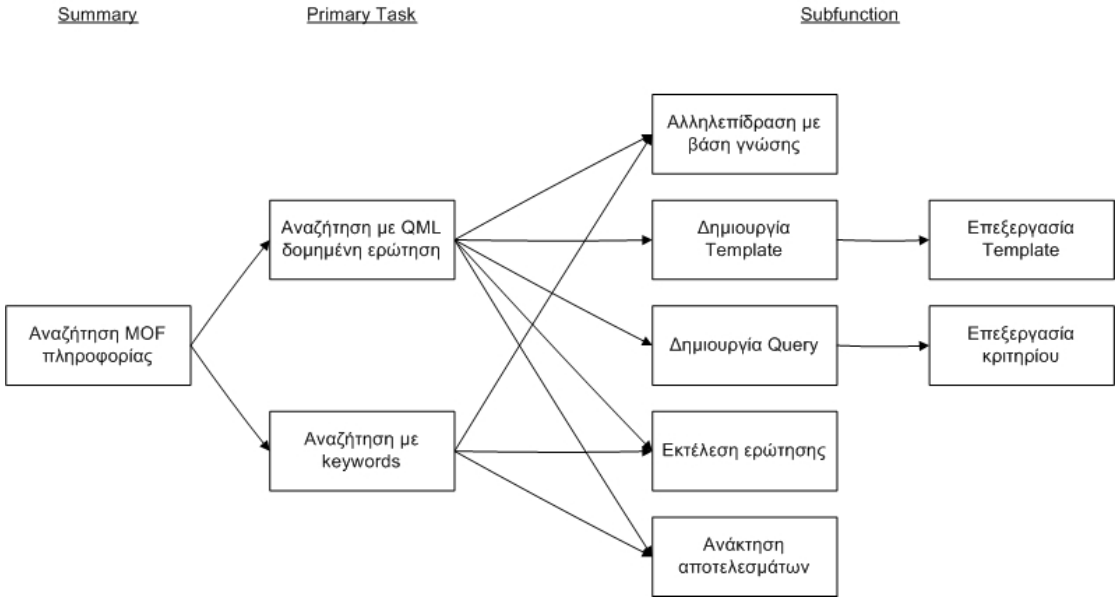
Κάθε περίπτωση χρήσης παρέχει ένα ή περισσότερα σενάρια που παρουσιάζουν τον τρόπο με τον οποίο το σύστημα πρέπει να αλληλεπιδράσει με τον τελικό χρήστη (ή κάποιο άλλο σύστημα) ώστε να επιτύχει έναν συγκεκριμένο στόχο.

Αυτές περιγράφονται κατά το σχεδιασμό της εφαρμογής, ώστε να αποτελέσουν τις κατευθυντήριες γραμμές κατά την υλοποίησή της, παρέχοντας παράλληλα και μία συνολική επισκόπηση για την εφαρμογή.

Επίσης πρέπει να αναφερθεί ότι για την περιγραφή των περιπτώσεων χρήσης έχουμε χρησιμοποιήσει το πρότυπο (template) του Alistair Cockburn [25].

4.2.1 Περιπτώσεις Χρήσης (Use Cases)

Στην εικόνα 14 που ακολουθεί παρατίθενται οι περιπτώσεις χρήσης σε ένα διάγραμμα το οποίο παρουσιάζει το επίπεδο (level) στο οποίο ανήκουν και τις μεταξύ τους συσχετίσεις.



Εικόνα 14: Διάγραμμα περιπτώσεων χρήσης

Στον πίνακα που ακολουθεί παρουσιάζονται συνοπτικά οι περιπτώσεις χρήσης με το επίπεδο (level) στο οποίο ανήκουν, τους βασικούς χαρακτήρες (primary actor) που συμμετέχουν σε αυτά, τον στόχο (goal) τους και μία σύντομη περιγραφή τους (brief).

#	**Level**	**Primary Actor**	**Goal**	**Brief**
1	Summary	Χρήστης	Αναζήτηση MOF πληροφορίας	Ανάκτηση αποτελεσμάτων που να πληρούν τους περιορισμούς που έχουν τεθεί στην ερώτηση
2	Primary Task	Χρήστης	Αναζήτηση με Keywords	Ανάκτηση αποτελεσμάτων που να πληρούν τους περιορισμούς που έχουν τεθεί στην ερώτηση
3	Primary Task	Χρήστης	Αναζήτηση με QML δομημένη ερώτηση	Ανάκτηση αποτελεσμάτων που να πληρούν τους περιορισμούς που έχουν τεθεί στην ερώτηση
4	Subfunction	Χρήστης	Αλληλεπίδραση με βάση γνώσης	Ο χρήστης αλληλεπιδρά με τη βάση γνώσης
5	Subfunction	Χρήστης	Δημιουργία template	Ο χρήστης δημιουργεί ένα template
6	Subfunction	Χρήστης	Επεξεργασία template	Ο χρήστης επεξεργάζεται ένα template
7	Subfunction	Χρήστης	Δημιουργία query	Ο χρήστης δημιουργεί ένα query
8	Subfunction	Χρήστης	Επεξεργασία κριτηρίου	Ο χρήστης επεξεργάζεται ένα κριτήριο
9	Subfunction	Χρήστης	Εκτέλεση ερώτησης	Ο χρήστης εκτελεί μία ερώτηση
10	Subfunction	Χρήστης	Ανάκτηση αποτελεσμάτων	Ανάκτηση των αποτελεσμάτων από το χρήστη

Αναλυτικά για κάθε περίπτωση χρήσης έχουμε:

Αναζήτηση MOF πληροφορίας

USE CASE 1	Αναζήτηση MOF πληροφορίας	
Goal in Context	Ανάκτηση αποτελεσμάτων που να πληρούν τους περιορισμούς που έχουν τεθεί στην ερώτηση	
Scope & Level	Σύστημα, Summary	
Preconditions	Καμία	
Success End Condition	Επιστρέφονται αποτελέσματα	
Failed End Condition	Δεν επιστρέφονται αποτελέσματα	
Primary, Secondary Actors	Χρήστης	
Trigger	Ο χρήστης επιλέγει να αναζητήσει MOF πληροφορία	
DESCRIPTION	**Step**	**Action**
	1	Αναζήτηση με QML δομημένη ερώτηση (#3)
SUB-VARIATIONS		**Branching Action**
	1	Αναζήτηση με Keywords (#2)

Αναζήτηση με Keywords

USE CASE 2	Αναζήτηση με Keywords
Goal in Context	Ανάκτηση αποτελεσμάτων που να πληρούν τους περιορισμούς που έχουν τεθεί στην ερώτηση
Scope & Level	Σύστημα, Primary Task
Preconditions	Καμία
Success End Condition	Επιστρέφονται αποτελέσματα
Failed End Condition	Δεν επιστρέφονται αποτελέσματα

Primary, Secondary Actors	Χρήστης	
Trigger	Ο χρήστης επιλέγει να αναζητήσει MOF πληροφορία με Keywords	
DESCRIPTION	**Step**	**Action**
	1	Αλληλεπίδραση με βάση γνώσης (#4)
	2	Εκτέλεση ερώτησης (#9)
	3	Ανάκτηση αποτελεσμάτων (#10)

Αναζήτηση με QML δομημένη ερώτηση

USE CASE 3	Αναζήτηση με QML δομημένη ερώτηση	
Goal in Context	Ανάκτηση αποτελεσμάτων που να πληρούν τους περιορισμούς που έχουν τεθεί στην ερώτηση	
Scope & Level	Σύστημα, Primary Task	
Preconditions	Καμία	
Success End Condition	Επιστρέφονται αποτελέσματα	
Failed End Condition	Δεν επιστρέφονται αποτελέσματα	
Primary, Secondary Actors	Χρήστης	
Trigger	Ο χρήστης επιλέγει να αναζητήσει MOF πληροφορία με QML δομημένη ερώτηση	
DESCRIPTION	**Step**	**Action**
	1	Αλληλεπίδραση με βάση γνώσης (#4)
	2	Δημιουργία template (#5)
	3	Δημιουργία query (#7)
	4	Εκτέλεση ερώτησης (#9)
	5	Ανάκτηση αποτελεσμάτων (#10)

Αλληλεπίδραση με βάση γνώσης

USE CASE 4	Αλληλεπίδραση με βάση γνώσης	
Goal in Context	Ο χρήστης αλληλεπιδρά με τη βάση γνώσης	
Scope & Level	Σύστημα, Subfunction	
Preconditions	Καμία	
Success End Condition	Ο χρήστης αλληλεπιδρά με τη βάση γνώσης	
Failed End Condition	Ο χρήστης δεν αλληλεπιδρά με τη βάση γνώσης	
Primary, Secondary Actors	Χρήστης	
Trigger	Ο χρήστης ζητάει αλληλεπίδραση με τη βάση γνώσης	
DESCRIPTION	**Step**	**Action**
	1	Γίνεται επικοινωνία με τη βάση γνώσης
	2	Προωθούνται αιτήματα προς αυτή
	3	Ανακτώνται πληροφορίες

Δημιουργία template

USE CASE 5	Δημιουργία template
Goal in Context	Ο χρήστης δημιουργεί ένα template
Scope & Level	Σύστημα, Subfunction
Preconditions	Να υπάρχει σύνδεση με τη βάση γνώσης
Success End Condition	Το template δημιουργείται
Failed End Condition	Το template δεν δημιουργείται
Primary, Secondary Actors	Χρήστης

Trigger	Ο χρήστης επιλέγει να δημιουργήσει ένα template	
DESCRIPTION	**Step**	**Action**
	1	Ανάκτηση MOF πληροφορίας στο οποίο θα αναφέρεται το template : • Μετα-μοντέλα • Μοντέλα (+ οντολογίες) • Οντολογίες
	2	Επιλογή στοιχείων από το μοντέλο πληροφορίας
	3	Ορισμός συζεύξεων μεταξύ των στοιχείων
	4	Ορισμός συνόλου αποτελεσμάτων από στοιχεία του μοντέλου

<u>Επεξεργασία template</u>

USE CASE 6	Επεξεργασία template	
Goal in Context	Ο χρήστης επεξεργάζεται ένα template	
Scope & Level	Σύστημα, Subfunction	
Preconditions	Να υπάρχει template	
Success End Condition	Στο template αποθηκεύονται οι νέες τιμές	
Failed End Condition	Στο template δεν αποθηκεύονται οι νέες τιμές	
Primary, Secondary Actors	Χρήστης	
Trigger	Ο χρήστης επιλέγει να επεξεργαστεί ένα template	
DESCRIPTION	**Step**	**Action**
	1	Προσθήκη/Αφαίρεση στοιχείων από το μοντέλο πληροφορίας
	3	Προσθήκη/Αφαίρεση συζεύξεων μεταξύ των στοιχείων
	4	Προσθήκη/Αφαίρεση αποτελεσμάτων από στοιχεία

		του μοντέλου

Δημιουργία Query

USE CASE 7	Δημιουργία query	
Goal in Context	Ο χρήστης δημιουργεί ένα query	
Scope & Level	Σύστημα, Subfunction	
Preconditions	Να υπάρχει template	
Success End Condition	Το query δημιουργείται	
Failed End Condition	Το query δεν δημιουργείται	
Primary, Secondary Actors	Χρήστης	
Trigger	Ο χρήστης επιλέγει να δημιουργήσει ένα query	
DESCRIPTION	**Step**	**Action**
	1	Επιλέγεται το template στο οποίο θα αναφέρεται το query
	2	Επεξεργασία κριτηρίου (#7)

Επεξεργασία κριτηρίου

USE CASE 8	Επεξεργασία κριτηρίου
Goal in Context	Ο χρήστης επεξεργάζεται ένα κριτηρίου
Scope & Level	Σύστημα, Subfunction
Preconditions	Να έχει δημιουργηθεί query
Success End Condition	Στο κριτήριο αποθηκεύονται οι νέες τιμές
Failed End Condition	Στο κριτήριο δεν αποθηκεύονται οι νέες τιμές
Primary,	Χρήστης

Secondary Actors		
Trigger	Ο χρήστης επιλέγει να επεξεργαστεί ένα κριτήριο	
DESCRIPTION	**Step**	**Action**
	1	Ο χρήστης δίνει τιμή για κάποια από τα παρακάτω χαρακτηριστικά του κριτηρίου: • Πράξη (Operation) • Τιμή (Value) • Βάρος (Weight)

Εκτέλεση ερώτησης

USE CASE 9	Εκτέλεση ερώτησης	
Goal in Context	Ο χρήστης εκτελεί μία ερώτηση	
Scope & Level	Σύστημα, Subfunction	
Preconditions	Να έχει δημιουργηθεί η ερώτηση	
Success End Condition	Η ερώτηση εκτελείται	
Failed End Condition	Η ερώτηση δεν εκτελείται	
Primary, Secondary Actors	Χρήστης	
Trigger	Ο χρήστης επιλέγει να εκτελέσει μία ερώτηση	
DESCRIPTION	**Step**	**Action**
	1	Σχηματισμός QML ερώτησης
	2	Η ερώτηση εκτελείται
SUB-VARIATIONS		**Branching Action**
	1	Σχηματισμός ερώτησης από keywords

Ανάκτηση αποτελεσμάτων

USE CASE 10	Ανάκτηση αποτελεσμάτων	
Goal in Context	Ανάκτηση των αποτελεσμάτων από το χρήστη	
Scope & Level	Σύστημα, Subfunction	
Preconditions	Να έχει εκτελεστεί μία ερώτηση	
Success End Condition	Γίνεται ανάκτηση των αποτελεσμάτων από το χρήστη	
Failed End Condition	Δεν γίνεται ανάκτηση των αποτελεσμάτων από το χρήστη	
Primary, Secondary Actors	Χρήστης	
Trigger	Να έχει εκτελεστεί μία ερώτηση	
DESCRIPTION	**Step**	**Action**
	1	Ανάκτηση των αποτελεσμάτων
	2	Επισκόπηση των αποτελεσμάτων
	3	Διαχείριση των αποτελεσμάτων
	4	Εκτέλεση ενεργειών στα αποτελέσματα

4.3 Υλοποίηση συστήματος

Στα πλαίσια της παρούσας εργασίας κατασκευάστηκαν δύο υλοποιήσεις για το σύστημα που έχει περιγραφεί. Οι υλοποιήσεις αυτές αφορούν σε διαφορετικά περιβάλλοντα. Η μία έχει γίνει στην πλατφόρμα του Eclipse [15], ενώ η άλλη έχει γίνει στην πλατφόρμα του OpenLaszlo [16].

Και στις δύο υλοποιήσεις, η διαδικασία κατασκευής μιας δομημένης QML ερώτησης απαιτεί την κατασκευή ενός Template κι ενός Query τα οποία είναι τα βασικά δομικά στοιχεία που συνθέτουν μια ερώτηση. Η δομή του καθενός από αυτά αναλύεται παρακάτω.

Ένα Template περιέχει από τα παρακάτω στοιχεία :

- Όνομα

- Περιγραφή (προαιρετική)
- Τύπος, αν το template αναφέρεται σε μετα-μοντέλο, μοντέλο ή οντολογία
- Τα στοιχεία του MOF μετα-μοντέλου, μοντέλου ή οντολογίας στα οποία αναφέρεται το template. Αυτά έχουν τα ακόλουθα χαρακτηριστικά
    - Όνομα (περιγράφει μοναδικά το κάθε στοιχείο)
    - Αναφορά (την ακριβή θέση του στην ιεραρχία του MOF μετα-μοντέλου, μοντέλου ή οντολογίας)
    - Τύπος (String, integer κλπ.)
- Συζεύξεις μεταξύ των στοιχείων
- Τα στοιχεία του MOF μετα-μοντέλου, μοντέλου ή οντολογίας τα οποία είναι επιθυμητό να επιστρέφονται σε μία ερώτηση που θα χρησιμοποιεί αυτό το Template.

Το Query αποτελείται από τα παρακάτω στοιχεία:

- Όνομα
- Αναφορά στο Template που χρησιμοποιεί
- Περιορισμοί στα στοιχεία του Template οι οποίοι έχουν τα ακόλουθα χαρακτηριστικά
    - Αναφορά (σε ποιο στοιχείο του Template αναφέρεται)
    - Τιμή (για τον περιορισμό)
    - Πράξη (ανάλογα με τον τύπο του στοιχείου στο οποίο αναφέρεται π.χ. οι πράξεις για τύπο String είναι = ,like και <>)
    - Βάρος σημασίας (Πόσο σημαντικός είναι ο περιορισμός αυτός για τη συνολική ερώτηση)

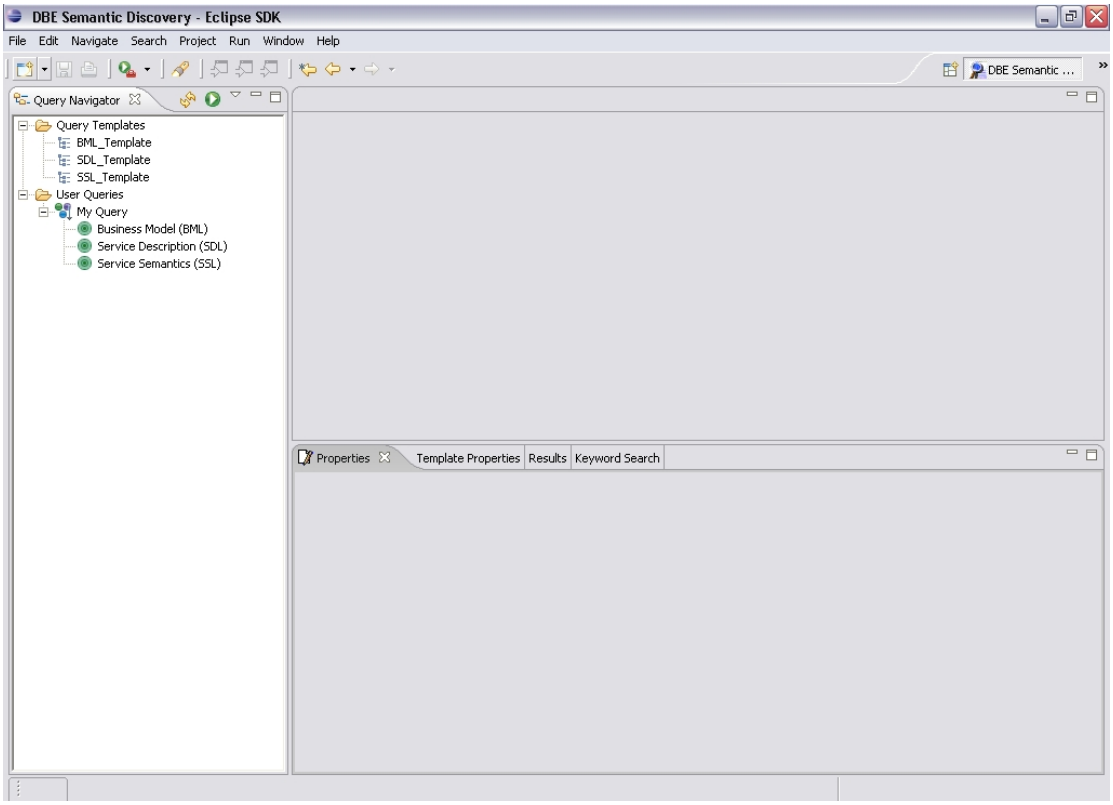
Τα χαρακτηριστικά του Template στην υλοποίηση για την πλατφόρμα του Eclipse φαίνονται στις εικόνες 17, 18, 19 και του Query στην 16.

Στην υλοποίηση για την πλατφόρμα του OpenLaszlo η διαδικασία δημιουργίας των Template και Query γίνεται αδιαφανώς από το χρήστη κατά τη διάρκεια δημιουργίας κριτηρίων (βλέπε παράρτημα Β, παράγραφο Β.1.3), ενώ στην υλοποίηση για την πλατφόρμα του Eclipse η διαδικασία δημιουργίας των Template (βλέπε παράρτημα Α, παράγραφο Α.2.1) και Query (βλέπε παράρτημα Α, παράγραφο Α.2.5) γίνεται με αναλυτικό τρόπο με επιλογές των στοιχείων που θα αποτελούν το καθένα από αυτά από το χρήστη.

Στη συνέχεια ακολουθεί αναλυτική περιγραφή των δύο υλοποιήσεων.

4.3.1 Υλοποίηση σαν Eclipse plug-in - Query Formulator and Semantic Discovery Tool (QFSDT)

Το QFSDT έχει υλοποιηθεί ως plug-in στην πλατφόρμα Eclipse. Στην επιφάνεια εργασίας του εργαλείου έχουμε ορίσει μία όψη (perspective) που φαίνεται στην εικόνα 15.



Εικόνα 15: Το perspective του QFSDT

Στο perspective αυτό έχουμε ορίσει πέντε περιοχές επισκόπησης (views) και μια περιοχή επεξεργασίας για τους επεξεργαστές των ερωτήσεων (editor area) :

- Η περιοχή επισκόπησης του πλοηγού των ερωτήσεων περιέχει πληροφορίες για τις ερωτήσεις (queries) για μοντέλα, δεδομένα και οντολογίες (μέσα σε μοντέλα) , για τα πρότυπα (templates) τα οποία είναι αποθηκευμένα στο τοπικό σύστημα αρχείων καθώς και τις ενέργειες που μπορούν να εφαρμοστούν σ'αυτά.
- Η περιοχή επισκόπησης των ιδιοτήτων περιέχει πληροφορίες για τα κριτήρια των queries.
- Η περιοχή επισκόπησης των ιδιοτήτων του template περιέχει πληροφορίες για το template.

- Η περιοχή επισκόπησης των αποτελεσμάτων περιέχει τα αποτελέσματα ενός δομημένου query και τις ενέργειες που μπορούν να εφαρμοστούν σ'αυτά.
- Η περιοχή επισκόπησης της γρήγορης αναζήτησης παρέχει έναν τρόπο για αναζήτηση με τη χρήση λέξεων-κλειδιών και τις ενέργειες που μπορούν να εφαρμοστούν στα αποτελέσματα.
- Η περιοχή του επεξεργαστή ερωτήσεων εμφανίζεται στην επάνω και δεξιά πλευρά της όψης (perspective) του εργαλείου. Σ'αυτήν ανοίγονται οι ερωτήσεις από έναν επεξεργαστή ερωτήσεων (επεξεργαστής ερωτήσεων για μοντέλα ή επεξεργαστής ερωτήσεων για δεδομένα, ανάλογα με τον τύπο της ερώτησης).

Το QFSDT έχει υλοποιηθεί ώστε να λειτουργεί στα δύο περιβάλλοντα λειτουργίας του DBE (Service Factory και Service Execution) με κοινό τρόπο και παρέχει στο χρήστη τη δυνατότητα να εναλλάσσεται μεταξύ των περιβαλλόντων αυτών αλλά και να ορίσει και συγκεκριμένη βάση γνώσης με την οποίο επιθυμεί να συνδεθεί.

Επίσης το QFSDT παρέχει τη δυνατότητα σε άλλα γραφικά εργαλεία του DBE Studio να συνδέονται με αυτό μέσω μίας διεπαφής που έχουμε δημιουργήσει ώστε να εκμεταλλεύονται την λειτουργικότητά του και να προσθέτουν λειτουργίες στα αποτελέσματα που αυτό επιστρέφει.

Είναι σημαντικό να αναφερθεί ότι έχει γραφτεί οδηγός χρήσης του QFSDT, ο οποίος ενσωματώνεται στη βοήθεια (Help) του Eclipse με την εγκατάσταση του DBE Studio.

Στις επόμενες υποενότητες θα περιγραφούν οι παραπάνω περιοχές καθώς και η αλληλεπίδραση μεταξύ τους.

4.3.1.1 Η περιοχή επισκόπησης του πλοηγού των ερωτήσεων (Query Navigator)

Ο πλοηγός ερωτήσεων (φαίνεται στην εικόνα 15) περιέχει τα templates και τα queries, τα οποία είναι αποθηκευμένα σ'έναν υποκατάλογο στο χώρο εργασίας του Eclipse (workspace) ο οποίος ονομάζεται "QFSDT\Queries". Τα templates και queries αποθηκεύονται ως .template και .query αρχεία αντίστοιχα και έχουν συγκεκριμένη XML μορφή (XML schema) που έχουμε ορίσει βάση του οποίου τα αποθηκεύουμε, τα διαβάζουμε και τα διαχειριζόμαστε μέσω ενός μηχανισμού που έχουμε υλοποιήσει για το σκοπό αυτό.

Στον πλοηγό των ερωτήσεων υπάρχει μία δεντρική αναπαράσταση που δείχνει την ιεραρχία των καταλόγων και αρχείων στον κατάλογο ”Queries”. Ο κατάλογος με όνομα “Query Templates” περιέχει templates και ο κατάλογος “User Queries” περιέχει queries ομαδοποιημένα σε γκρουπ (που όρισε ο χρήστης).

Μέσω αυτής της περιοχής επισκόπησης παρέχεται η δυνατότητα να εκτελεστούν ενέργειες που αφορούν στη διαχείριση των templates και queries και παρέχονται μέσω του Eclipse Controller. Ειδικότερα οι ενέργειες που αφορούν στη διαδικασία δημιουργίας των templates και queries και επεξεργασίας των πρώτων ξεκινούν οδηγούς (wizards) οι οποίοι καθοδηγούν το χρήστη σ’όλη τη διάρκεια της εκάστοτε διαδικασίας. Οι ενέργειες αυτές περιγράφονται αναλυτικά στο παράρτημα Α στις παραγράφους Α.2.1 – Α.2.6, Α.2.8, Α.2.9.

4.3.1.2 Η περιοχή του επεξεργαστή ερωτήσεων (Editor area)

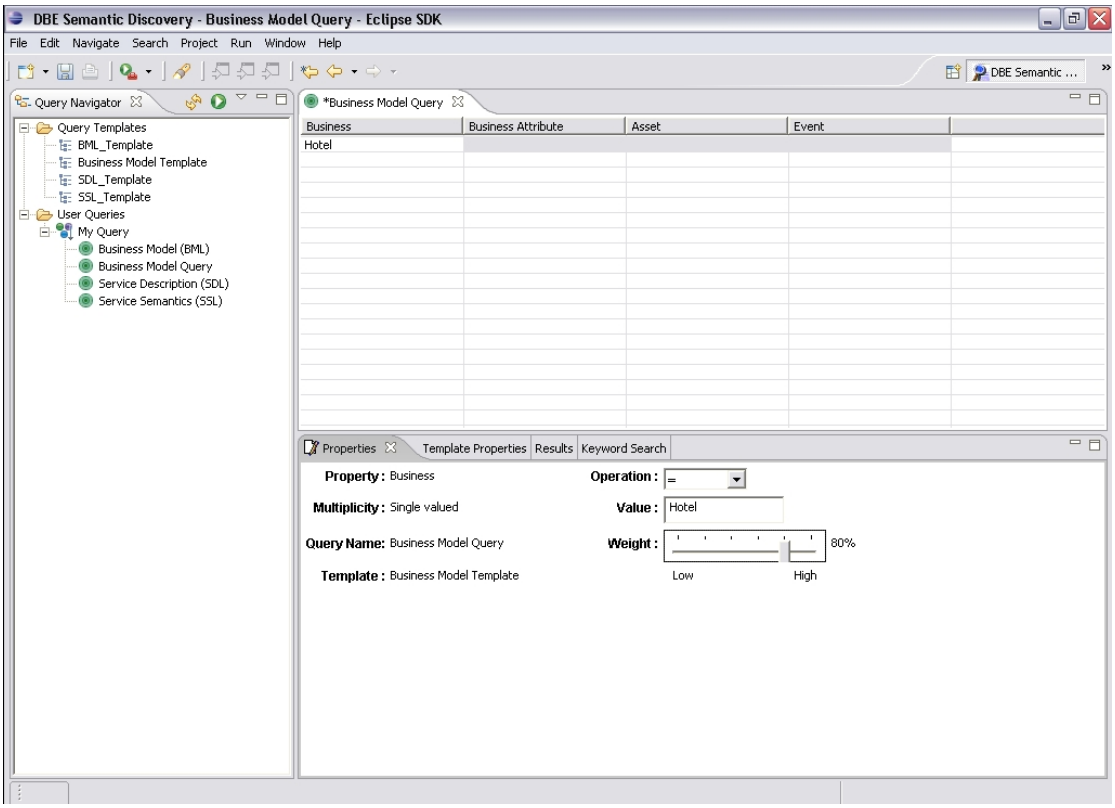
Η τεχνική που ακολουθήσαμε για να θέτει κριτήρια ένας χρήστης σε ένα query στην παρούσα υλοποίηση είναι η τεχνική QbE (Query by Example) [8], η οποία είναι πολύ διαδεδομένη και χρησιμοποιείται ευρέως επειδή είναι πολύ φιλική προς το χρήστη. Δεν απαιτεί εξειδικευμένες γνώσεις από αυτόν για οποιαδήποτε γλώσσα ερωτήσεων, αφού το μόνο που χρειάζεται να κάνει κάποιος είναι να δώσει τιμές στα κελιά ενός πίνακα, ορίζοντας ουσιαστικά με αυτό τον τρόπο κριτήρια για τα στοιχεία που αντιπροσωπεύουν οι στήλες του πίνακα.

Έτσι κάθε query στο QFSDT αναπαριστάται σαν ένας πίνακας, που περιέχεται στη δομή του editor που παρέχει το eclipse. Οι στήλες του πίνακα είναι τα στοιχεία του template στο οποίο είναι βασισμένο το query. Ο χρήστης δίνει τιμές σε κελιά του πίνακα τα οποία αναπαριστούν κριτήρια του query. Επίσης μπορεί να ορίζει και πράξη για το κάθε κριτήριο γράφοντας πριν από την τιμή του τον τελεστή που θέλει να εφαρμοστεί σε αυτό (π.χ. “like”). Το σύστημα αναγνωρίζει τον τελεστή (εφόσον μπορεί να εφαρμοστεί στο κριτήριο αυτό) και δίνει τις κατάλληλες τιμές στα στοιχεία του query. Για κάθε διαφορετικό query που ο χρήστης ή το σύστημα επιλέγει να ανοίξει, ανοίγει ένας νέος επεξεργαστής ερωτήσεων, editor.

Παράλληλα ο χρήστης έχει τη δυνατότητα να εκτελεί και ενέργειες πάνω στο query τις οποίες παρέχει ο Eclipse Controller. Μπορεί να εκτελέσει ένα query και να προσθέσει ένα στοιχείο στο template στο οποίο αναφέρεται (νέα στήλη στον πίνακα).

Οι ενέργειες που μπορεί να εκτελέσει κάποιος σε ένα query από τον editor περιγράφονται αναλυτικά στο παράρτημα Α στις παραγράφους Α.2.6, Α.2.9.

Η περιοχή του επεξεργαστή ερωτήσεων φαίνεται στην εικόνα 16 στο επάνω και δεξιά μέρος του perspective.



Εικόνα 16: Αναπαράσταση ενός query στο QFSDT.

Ένα query ανοίγεται στον επεξεργαστή ερωτήσεων στο επάνω και δεξιά μέρος του perspective και οι ιδιότητές του φαίνονται από κάτω στην περιοχή επισκόπησης των ιδιοτήτων του.

4.3.1.3 Η περιοχή επισκόπησης των ιδιοτήτων (Properties)

Κάθε κελί στον πίνακα του query (το οποίο αναπαριστά ένα κριτήριο) έχει κάποια χαρακτηριστικά. Η αναπαράσταση αυτών των χαρακτηριστικών γίνεται στην περιοχή επισκόπησης των ιδιοτήτων που φαίνεται στην εικόνα 16. Τα χαρακτηριστικά αυτά είναι :

- Σε ποιο template αναφέρεται το query
- Σε ποιο query ανήκει το χαρακτηριστικό
- Σε ποιο στοιχείο του template αναφέρεται ο περιορισμός που θέτει το κριτήριο
- Η τιμή του κριτηρίου

- Η πράξη η οποία εφαρμόζεται στα δεδομένα
- Το βάρος σημασίας που θα έχει το κριτήριο στο συνολικό query

Ανάλογα με το κελί το οποίο είναι επιλεγμένο, οι τιμές της περιοχής επισκόπησης των ιδιοτήτων αλλάζουν κατάλληλα. Ο χρήστης μπορεί να μεταβάλλει την τιμή, την πράξη και το βάρος σημασίας του κριτηρίου. Είναι σημαντικό να αναφερθεί ότι το βάρος σημασίας ενός κριτηρίου μπορεί να μεταβληθεί μόνο μέσω της περιοχής επισκόπησης των ιδιοτήτων.

Αλλάζοντας μία τιμή ο Eclipse Controller αναλαμβάνει να ενημερώσει το model για την αλλαγή αυτή (αλλάζει η τιμή ενός κριτηρίου στο query) και παράλληλα ενημερώνει και το ίδιο Eclipse View στην περιοχή του editor, διατηρώντας έτσι τη συνέπεια στα δεδομένα.

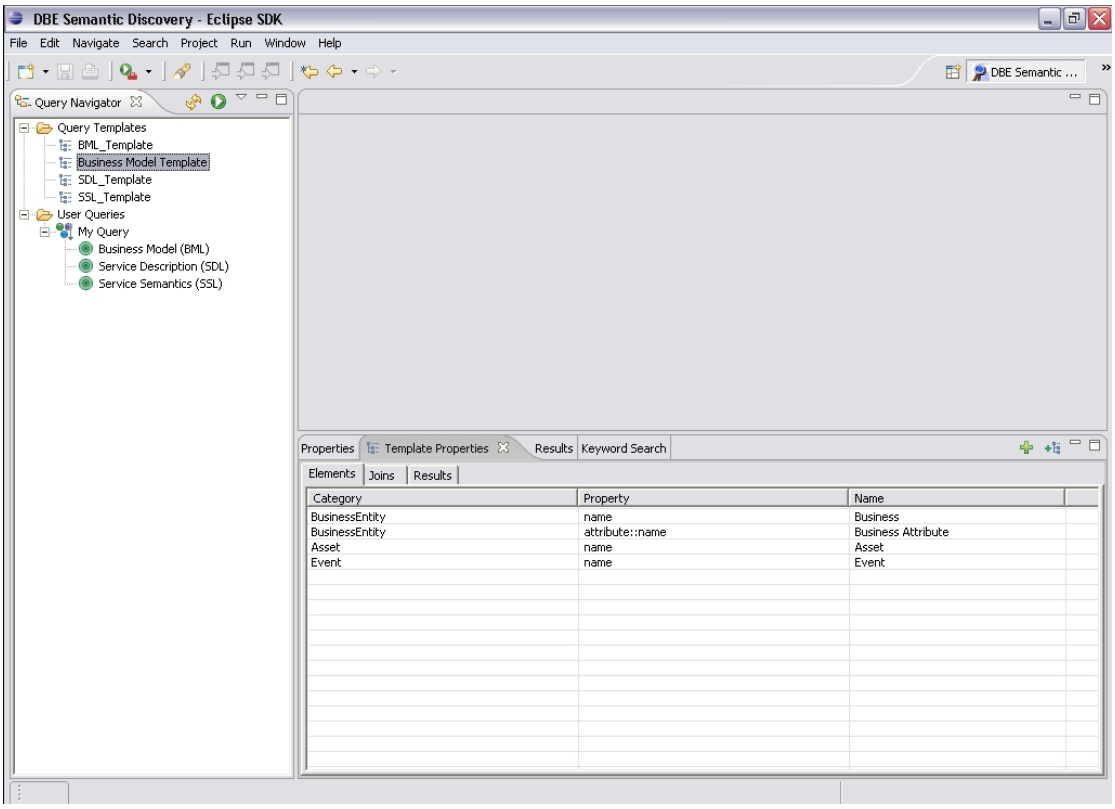
Η διαδικασία επεξεργασίας ενός query περιγράφεται αναλυτικά στο παράρτημα Α στην παράγραφο Α.2.6.

4.3.1.4 Η περιοχή επισκόπησης των ιδιοτήτων του template (Template Properties)

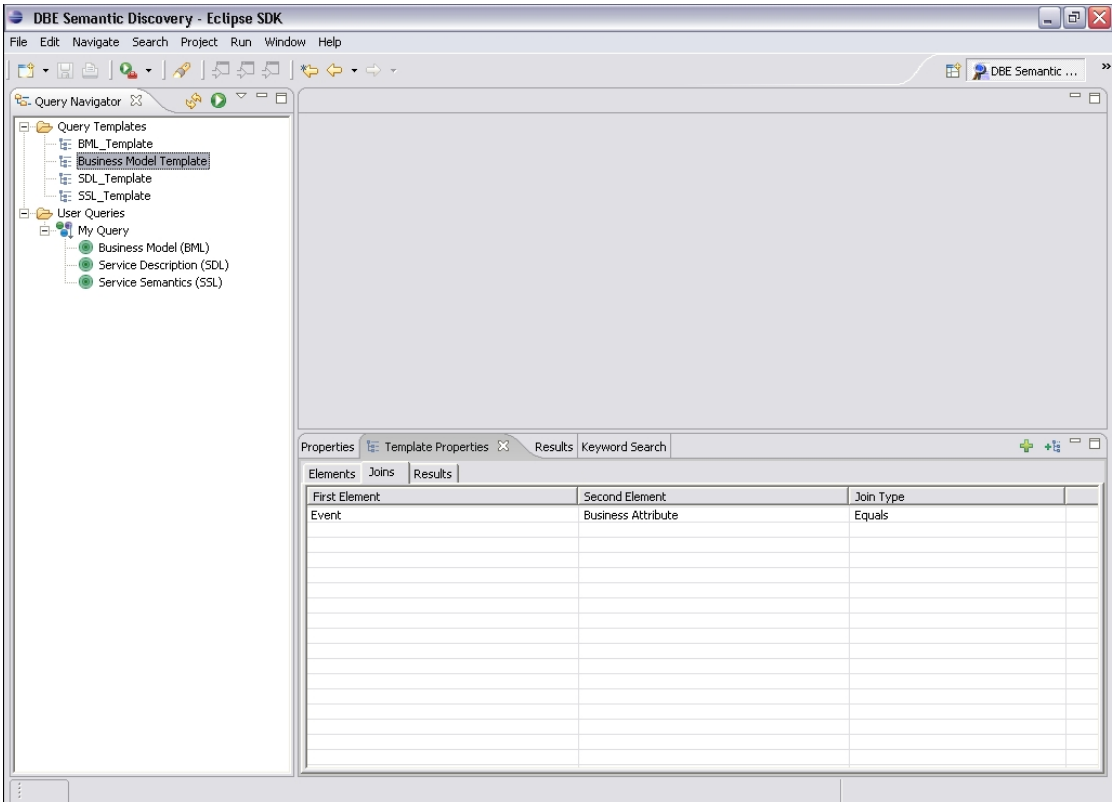
Η περιοχή επισκόπησης ιδιοτήτων template περιέχει πληροφορίες για το template το οποίο επιλέχτηκε ή το template στο οποίο βασίζεται ένα query. Οι πληροφορίες χωρίζονται σε φακέλους ανάλογα με το περιεχόμενό τους. Οι φάκελοι για τις πληροφορίες template είναι τρεις :

- Τα στοιχεία που συνθέτουν το template (εικόνα 17)
- Οι συζεύξεις μεταξύ των στοιχείων (εικόνα 18)
- Το σύνολο των αποτελεσμάτων που έχει ορίσει ο χρήστης (εικόνα 19)

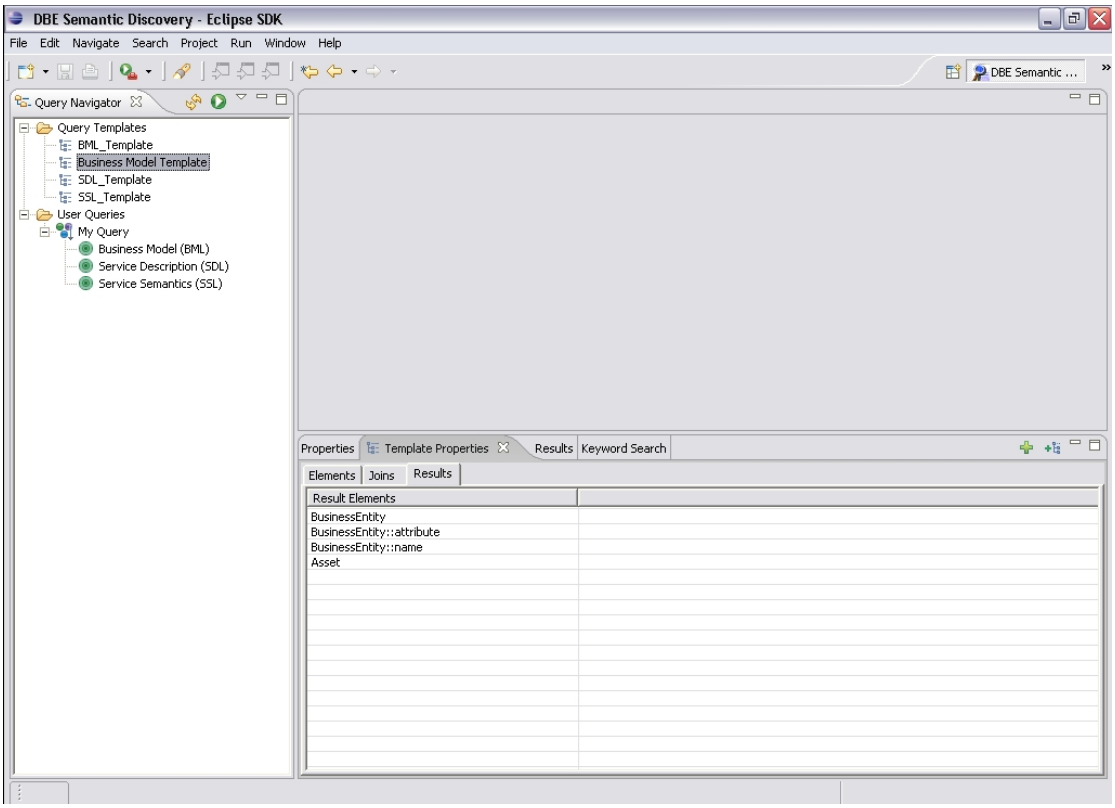
Πρέπει να αναφερθεί ότι σ'αυτήν την περιοχή επισκόπησης παρέχεται μέσω του Eclipse Controller η δυνατότητα επεξεργασίας του template (στα στοιχεία, τις συζεύξεις και τα αποτελέσματα) καθώς και ένας γρήγορος τρόπος προσθήκης ενός νέου στοιχείου στο template. Οι ενέργειες αυτές περιγράφονται αναλυτικά στο παράρτημα Α στην παράγραφο Α.2.4.



Εικόνα 17: Επισκόπηση των χαρακτηριστικών γνωρισμάτων που συνθέτουν το template.



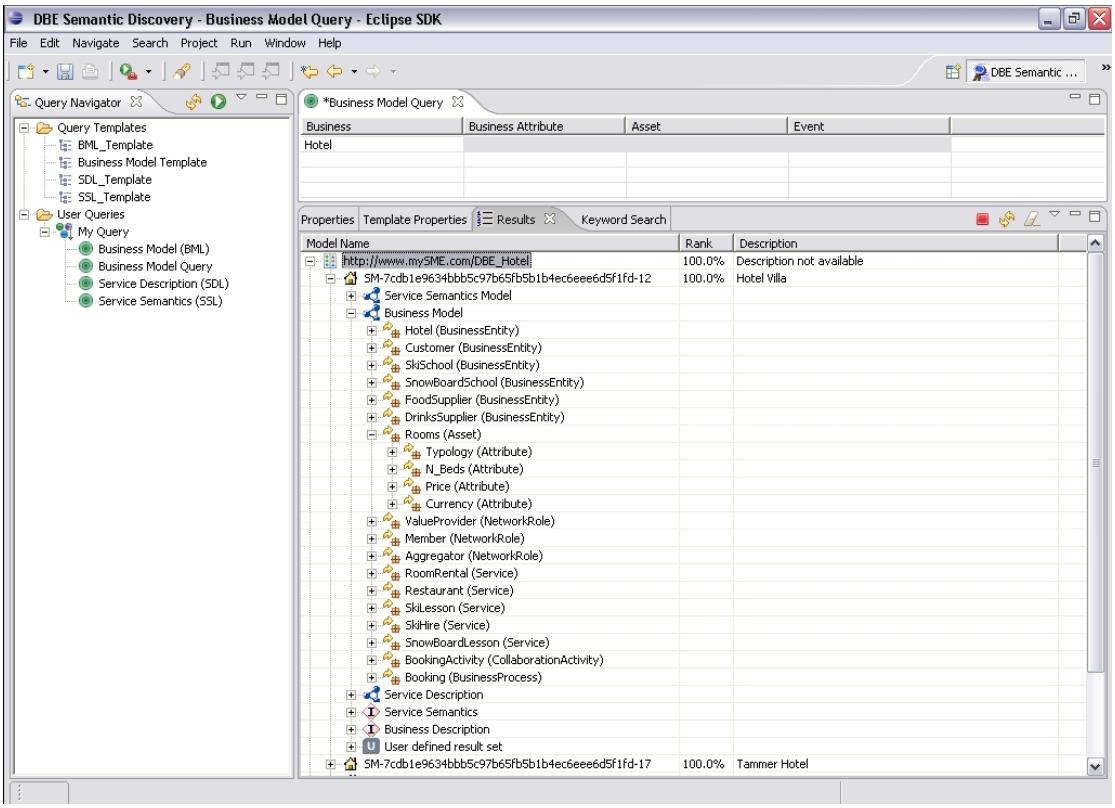
Εικόνα 18: Επισκόπηση των συζεύξεων στα χαρακτηριστικά γνωρίσματα του template.



Εικόνα 19: Επισκόπηση του συνόλου αποτελεσμάτων του template.

4.3.1.5 Η περιοχή επισκόπησης των αποτελεσμάτων (Results)

Αυτή η περιοχή επισκόπησης περιέχει τα αποτελέσματα μιας δομημένης ερώτησης που έχει εκτελεστεί. Κάθε αποτέλεσμα αναπαριστά, ανάλογα με τον τύπο του query, ένα μοντέλο ή μία υπηρεσία (service manifest), που σε κάθε περίπτωση πρέπει να ικανοποιεί τα κριτήρια που θέτει το query. Τα αποτελέσματα είναι ταξινομημένα κατά την σχετικότητα των αποτελεσμάτων με τα κριτήρια του query (ποσοστό επί τις εκατό). Ένα αποτέλεσμα μπορεί να έχει και κάποια περιγραφή. Η περιοχή επισκόπησης των αποτελεσμάτων φαίνεται στην εικόνα 20.



Εικόνα 20: Τα αποτελέσματα στο QFSDT.

Είναι σημαντικό να αναφερθεί πως η ανάκτηση των αποτελεσμάτων γίνεται με ασύγχρονο τρόπο λόγω του γεγονότος πως το σύστημα λειτουργεί σ'ένα P2P (peer-to-peer) περιβάλλον. Γι'αυτό το λόγο γίνεται ανανέωση της λίστας των αποτελεσμάτων κάθε λίγα δευτερόλεπτα από τον Eclipse Controller και να υπάρχουν νέα αποτελέσματα (αλλαγή στο Model) τα προσθέτει στη λίστα αποτελεσμάτων (αλλαγή στο View) πάλι ταξινομημένα ως προς τη σχετικότητα τους με τα κριτήρια της ερώτησης.

Ένα σύνολο από ενέργειες που παρέχονται από τον Eclipse Controller υπάρχουν στην παρούσα της περιοχής επισκόπησης :

- Πλοήγηση στα αποτελέσματα : Υπάρχει η δυνατότητα για την επισκόπηση των πληροφοριών που περιέχονται στα μοντέλα ή τα δεδομένα στα αποτελέσματα.
- Εκτέλεση υπηρεσίας : Αν τα αποτελέσματα είναι service manifests τότε υπάρχει η δυνατότητα της επίκλησης της υπηρεσίας. Μετά από αυτήν την ενέργεια, η υπηρεσία θα εκτελεστεί και ανάλογα με τον τύπο της διεπαφής που αυτή παρέχει θα εμφανιστεί ένα παράθυρο αλληλεπίδρασης με τον χρήστη.
- Εκτέλεση εγγεγραμμένων ενεργειών : Το QFSDT παρέχει μία διεπαφή μέσω της οποίας μπορεί μία άλλη εφαρμογή να χρησιμοποιήσει τις λειτουργικότητες

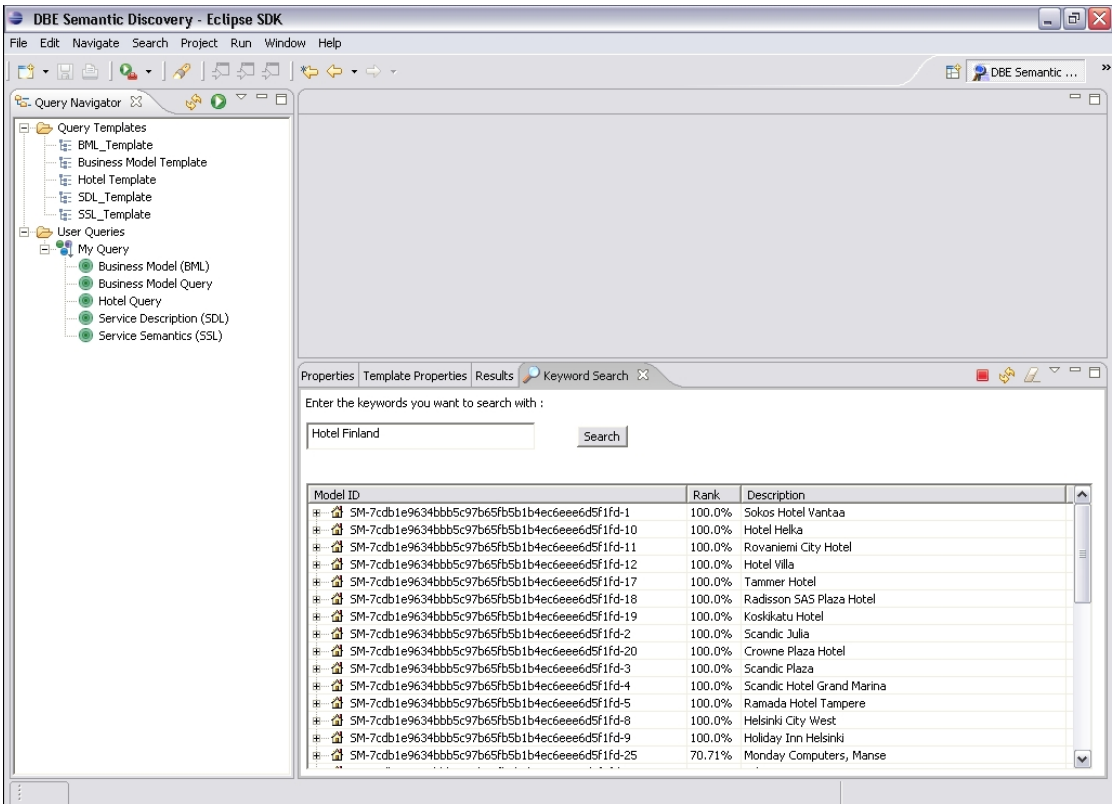
αναζήτησης που προσφέρει το QFSDT έτσι ώστε να μπορεί να χρησιμοποιήσει τα αποτελέσματα από μία αναζήτηση. Π.χ. άνοιγμα μοντέλου από τον BML Editor, Αποθήκευση μοντέλου στο δίσκο κλπ.

- Αναζήτηση για δεδομένα με χρήση συγκεκριμένου μοντέλου : Επιλέγοντας ο χρήστης συγκεκριμένο μοντέλο έχει τη δυνατότητα να δημιουργήσει άμεσα ένα template το οποίο να αναφέρεται στο μοντέλο αυτό με σκοπό την αναζήτηση δεδομένων.

Τα αποτελέσματα και οι λειτουργικότητές τους περιγράφονται αναλυτικά στο παράρτημα Α στις παραγράφους Α.2.10, Α.2.11.

4.3.1.6 Η περιοχή επισκόπησης της γρήγορης αναζήτησης (Keyword Search)

Η γρήγορη αναζήτηση είναι μια γρήγορη προσέγγιση αναζήτησης μοντέλων ή δεδομένων με τη χρήση λέξεων-κλειδιών. Τα αποτελέσματα μιας αναζήτησης με λέξεις-κλειδιά επιστρέφουν σ'αυτήν την περιοχή επισκόπησης και μπορούν να εκτελεστούν οι ίδιες ενέργειες όπως με αυτές των αποτελεσμάτων της περιοχής επισκόπησης των αποτελεσμάτων. Αυτό γίνεται επειδή ήταν επιθυμητό αυτή η περιοχή επισκόπησης να είναι ανεξάρτητη από όλες τις άλλες περιοχές επισκόπησης ώστε να μπορεί να χρησιμοποιηθεί από άλλα εργαλεία του Eclipse σαν ένας απλός και γρήγορος τρόπος αναζήτησης. Η περιοχή επισκόπησης της γρήγορης αναζήτησης φαίνεται στην εικόνα 21. Η γρήγορη αναζήτηση περιγράφεται αναλυτικά στο παράρτημα Α στην παράγραφο Α.2.12.



Εικόνα 21: Αναζήτηση με keywords στο QFSDT.

4.3.2 Υλοποίηση σε OpenLaszlo – DBE Service Discovery Portal

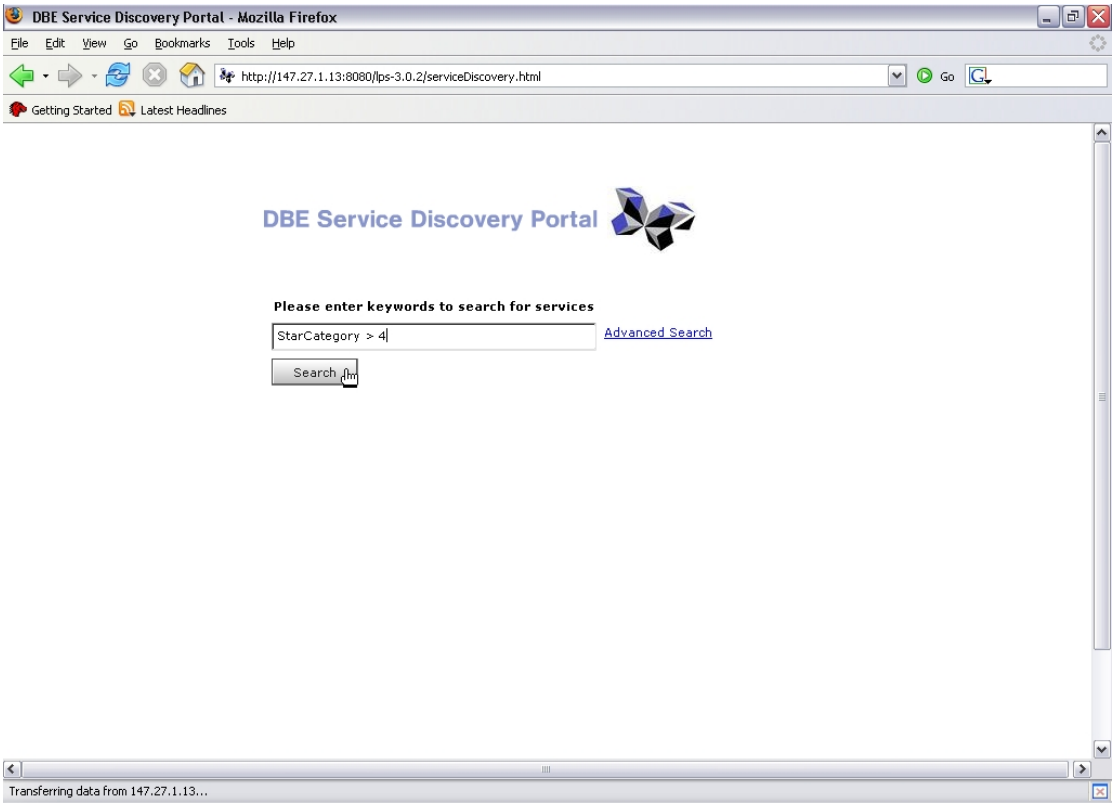
Το DBE Service Discovery Portal (Portal για συντομία) έχει υλοποιηθεί στην πλατφόρμα OpenLaszlo για να βγει μέρος της λειτουργικότητας του συστήματος σε web εφαρμογή. Ειδικότερα, μέσω του Portal ήταν επιθυμητό να γίνεται αναζήτηση μόνο υπηρεσιών. Αυτό περιορίζει το περιβάλλον λειτουργίας του Portal σε Service Execution περιβάλλον.

Το Portal αποτελείται από τρεις σελίδες :

- Απλή αναζήτηση με χρήση λέξεων κλειδιών
- Σύνθετη αναζήτηση με QML δομημένη ερώτηση (σε δεδομένα)
- Αποτελέσματα

4.3.2.1 Απλή αναζήτηση

Στη σελίδα αυτή που φαίνεται στην εικόνα 22 γίνεται απλή εισαγωγή λέξεων-κλειδιών με τις οποίες επιθυμεί ο χρήστης να αναζητήσει υπηρεσίες. Η λειτουργικότητα αυτή παρέχεται από τον Laszlo Controller και υπάρχει και στη σελίδα των αποτελεσμάτων. Αναλυτικά η διαδικασία αυτή περιγράφεται στο παράρτημα Β στην παράγραφο Β.1.2. Μετά την εκτέλεση της ερώτησης το σύστημα μεταβαίνει στη σελίδα των αποτελεσμάτων.



Εικόνα 22: Η σελίδα της απλής αναζήτησης.

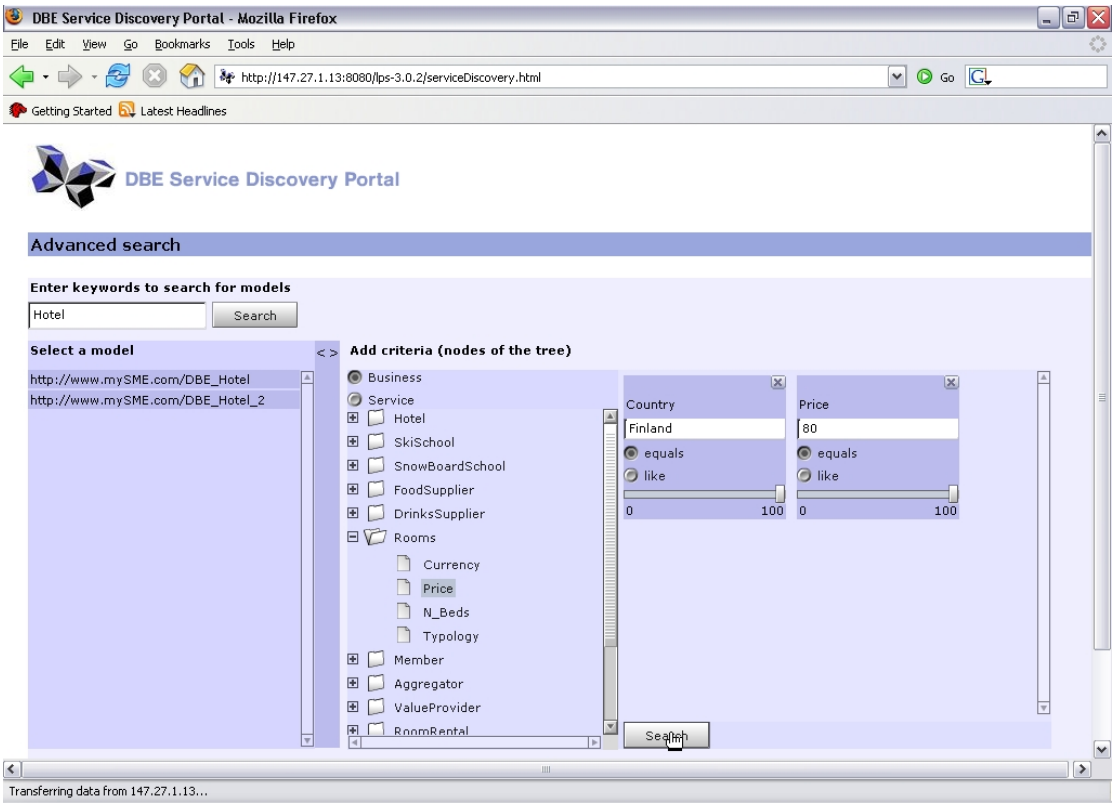
4.3.2.2 Σύνθετη αναζήτηση

Στη σελίδα της σύνθετης αναζήτησης που φαίνεται στην εικόνα 23 γίνεται κατασκευή δομημένων QML ερωτήσεων με παρόμοιο τρόπο με αυτόν της αναζήτησης δεδομένων στο QFSDT. Για να γίνει η αναζήτηση πρέπει να ακολουθηθούν τα εξής βήματα :

- Αναζήτηση μοντέλων με την εισαγωγή λέξεων-κλειδιών ώστε να ανακτηθούν μοντέλα πάνω στα οποία θα βασιστεί η ερώτηση.

- Επιλογή του κατάλληλου μοντέλου από τη λίστα των αποτελεσμάτων αναζήτησης. Μετά την επιλογή του μοντέλου εμφανίζεται μία δεντρική αναπαράστασή του.
- Επιλογή των στοιχείων του μοντέλου στα οποία θα δημιουργηθούν κριτήρια. Υπάρχει η δυνατότητα πλοήγησης σε όλη την πληροφορία του μοντέλου. Όταν ο χρήστης επιλέγει ένα στοιχείο δημιουργείται μία αναπαράσταση κριτηρίου.
- Εισαγωγή τιμών στα κριτήρια. Οι τιμές που μπορούν να εισαχθούν είναι :
  - Τιμή
  - Πράξη
  - Βάρος σημασίας κριτηρίου στην ερώτηση
- Εκτέλεση ερώτησης

Σε όλη αυτή τη διαδικασία ο Laszlo Controller αναλαμβάνει την επικοινωνία μεταξύ Model και Laszlo View ώστε να εμφανίζονται οι σωστές πληροφορίες ύστερα από κάθε ενέργεια του χρήστη.



Εικόνα 23: Η σελίδα της σύνθετης αναζήτησης.

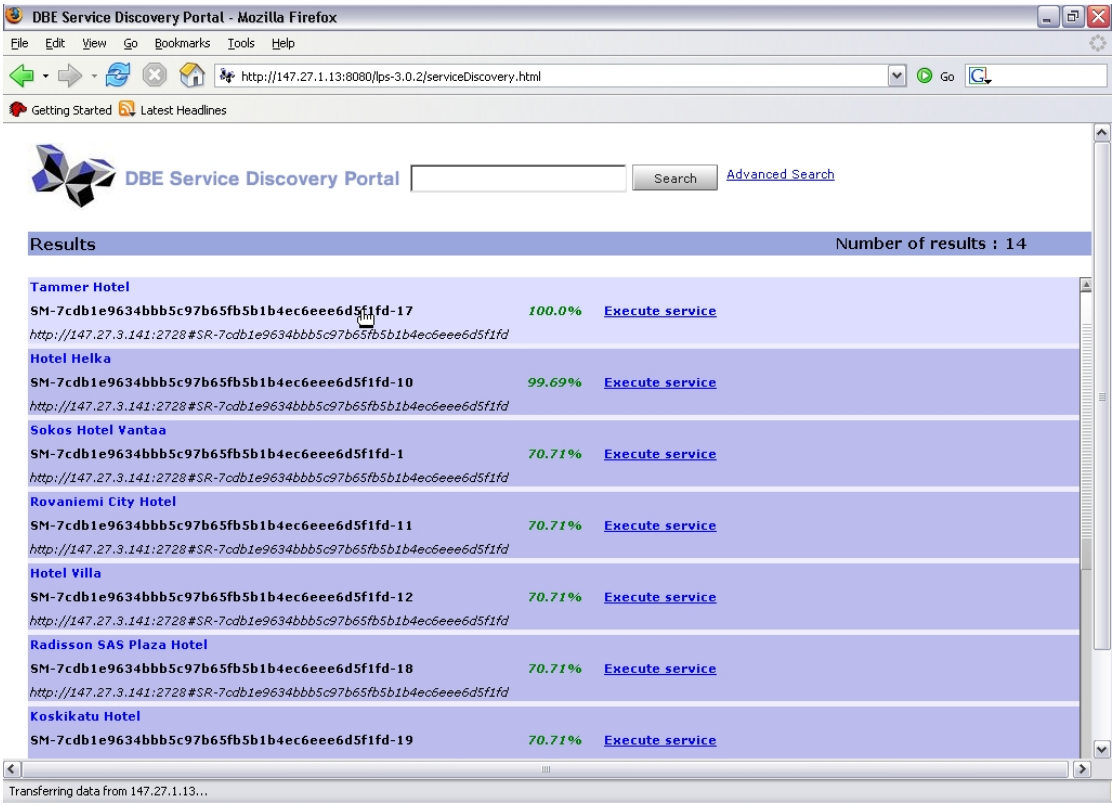
Μετά την εκτέλεση της ερώτησης το σύστημα μεταβαίνει στη σελίδα των αποτελεσμάτων.

Το template και το query όπως περιγράφτηκαν στο σχεδιασμό χρησιμοποιούνται κι εδώ για την κατασκευή της QML ερώτησης αλλά η δημιουργία τους γίνεται με αδιαφανή τρόπο προς τον χρήστη.

Αναλυτικά η διαδικασία αυτή περιγράφεται στο παράρτημα Β στην παράγραφο Β.1.3.

4.3.2.3 Αποτελέσματα

Μετά την εκτέλεση μιας ερώτησης ανακτώνται αποτελέσματα τα οποία εμφανίζονται σε μια λίστα ταξινομημένα κατά τη σχετικότητα των περιεχομένων των αποτελεσμάτων ως προς την ερώτηση. Η σελίδα των αποτελεσμάτων φαίνεται στην εικόνα 24.



Εικόνα 24: Η σελίδα των αποτελεσμάτων.

Για κάθε αποτέλεσμα ο χρήστης μπορεί να δει τις εξής πληροφορίες :

- Το διακριτικό νούμερο της υπηρεσίας (service manifest id)
- Μία περιγραφή της υπηρεσίας
- Η σχετικότητα του αποτελέσματος σε ποσοστό επί τις εκατό

- Το διακριτικό νούμερο του κόμβου από τον οποίο ανακτήθηκε
- Ένας σύνδεσμος για την εκτέλεση της υπηρεσίας που αναπαριστά. Με την επιλογή του συνδέσμου, θα ανοιχτεί ένας web browser για αλληλεπίδραση του χρήστη με την υπηρεσία εφόσον βέβαια αυτή είναι διαθέσιμη στο web.

Όπως και στην υλοποίηση στην πλατφόρμα του Eclipse έτσι και εδώ η ανάκτηση των αποτελεσμάτων γίνεται με ασύγχρονο τρόπο λόγω του γεγονότος πως το σύστημα λειτουργεί σ'ένα P2P (peer-to-peer) περιβάλλον. Γι'αυτό το λόγο γίνεται ανανέωση της λίστας των αποτελεσμάτων κάθε λίγα δευτερόλεπτα από τον Laszlo Controller και να υπάρχουν νέα αποτελέσματα (αλλαγή στο Model) τα προσθέτει στη λίστα αποτελεσμάτων (αλλαγή στο View) πάλι ταξινομημένα ως προς τη σχετικότητα τους με τα κριτήρια της ερώτησης.

Σε περίπτωση που ένας χρήστης επιθυμεί να δει περισσότερες πληροφορίες για μία υπηρεσία επιλέγει το αποτέλεσμα που την αντιπροσωπεύει. Θα εμφανιστεί ένα παράθυρο με πληροφορίες που αφορούν στην επιλεγμένη υπηρεσία.

Αναλυτικά η διαδικασία επισκόπησης στα αποτελέσματα περιγράφεται στο παράρτημα Β στην παράγραφο Β.1.4.

4.4 Υποστήριξη QML

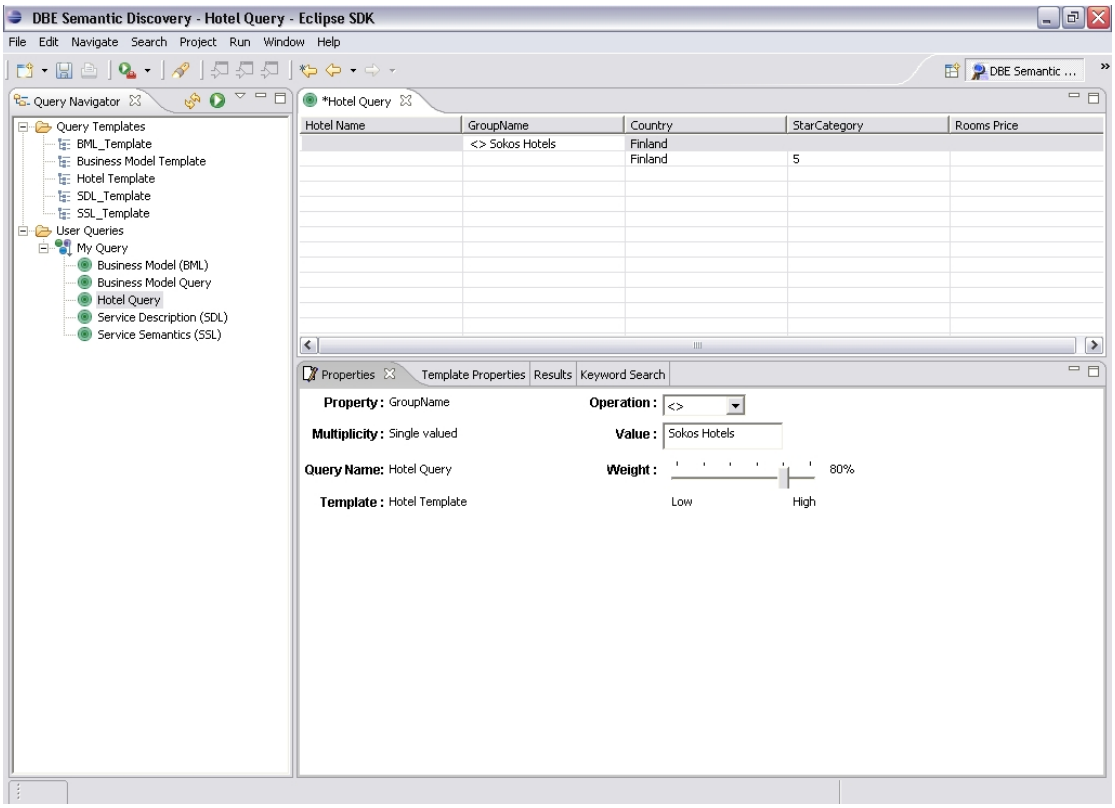
Για την κατασκευή ερωτήσεων που απευθύνονται στο MOF στο σύστημά μας χρησιμοποιήσαμε τη γλώσσα ερωτήσεων QML (Query Metamodel Language) [7]. Η QML παρέχει ένα μεγάλο σύνολο από λειτουργικότητες για την έκφραση περιορισμών στα επίπεδα M3-M1 του MOF. Στα πλαίσια του προγράμματος DBE ήταν επιθυμητό να δημιουργηθεί μία διεπαφή για αναζήτηση με απλά χαρακτηριστικά. Η υποστήριξη ολόκληρης της QML θα πρόσδιδε μεγάλη πολυπλοκότητα στην εφαρμογή, χάνοντας το βασικό στόχο για την υλοποίηση μίας απλής και εύχρηστης γραφικής διεπαφής. Έτσι επιλέχθηκε να χρησιμοποιηθεί ένα υποσύνολο της QML, το οποίο προσφέρει την επιθυμητή λειτουργικότητα για μία αποτελεσματική αναζήτηση, διατηρώντας ταυτόχρονα την απλότητα.

4.4.1 Υποστήριξη QML για την υλοποίηση στην πλατφόρμα Eclipse

Ο τρόπος με τον οποίο υποστηρίχθηκε η QML στην υλοποίηση που έγινε για την πλατφόρμα Eclipse φαίνεται παρακάτω:

- Στη δομή του πίνακα που αναπαριστά ένα query υποστηρίζουμε
  - Existential quantification στα κριτήρια κάθε γραμμής
  - Conjunction στις στήλες κάθε γραμμής του πίνακα
  - Disjunction ανάμεσα στις γραμμές του πίνακα
  - Negation μέσω του τελεστή '<>' στα κριτήρια
  - Filtering με τον ορισμό τιμών στα κριτήρια του query

  Στην εικόνα 25 παρουσιάζεται μία ερώτηση στην οποία φαίνονται τα παραπάνω χαρακτηριστικά.



Εικόνα 25: Υποστήριξη Existential Quantification, Conjunction, Disjunction, Negation και Filtering μέσα από ένα query στην Eclipse υλοποίηση.
- ***Κάθε γραμμή είναι ένας όρος exists.***
- ***Μεταξύ των κριτηρίων της γραμμής εφαρμόζεται σύζευξη (Conjunction).***
- ***Μεταξύ των γραμμών εφαρμόζεται διάζευξη (Disjunction).***
- ***Το Negation υποστηρίζεται με τον τελεστή '<>' στα κριτήρια.***
- ***Το Filtering γίνεται με την εισαγωγή τιμών στα κριτήρια.***

- Κατά τη δημιουργία ερώτησης έμμεσα υποστηρίζεται Variable declaration, το οποίο γίνεται αυτόματα από το σύστημα.
- Υποστηρίζονται Multiple Documents, επιλέγοντας πολλά διαφορετικά κομμάτια από ένα μετα-μοντέλο (multiple contexts) κατά τη διαδικασία δημιουργίας ενός template. Στο παράρτημα Α στην παράγραφο Α.2.1.2 παρουσιάζεται αναλυτικά ο τρόπος με τον οποίο επιλέγονται στοιχεία από διαφορετικά contexts. Για την δημιουργία ερωτήσεων σε multiple contexts το σύστημα υποστηρίζει Cartesian product σε διαφορετικά contexts.
- Τα Join υποστηρίζονται επιλέγοντας τα στοιχεία της σύζευξης και τον τύπο της κατά τη διαδικασία δημιουργίας ενός template. Στο παράρτημα Α στην παράγραφο Α.2.1.3 παρουσιάζεται αναλυτικά ο τρόπος με τον οποίο ορίζονται τα Joins.
- Μέσω του ορισμού αποτελεσμάτων κατά τη διαδικασία δημιουργίας ενός template υποστηρίζουμε Projection. Στο παράρτημα Α στην παράγραφο Α.2.1.4 παρουσιάζεται αναλυτικά ο τρόπος με τον οποίο ορίζονται τα αποτελέσματα και φαίνεται η υποστήριξη του Projection. Για κάθε στοιχείο των αποτελεσμάτων που ορίζει ο χρήστης το σύστημα κάνει αυτόματα Renaming σε αυτά.
- Δεν υποστηρίζονται Union, Difference, Grouping και Universal quantification, γιατί θα καθιστούσαν την εφαρμογή πολύπλοκη και δύσχρηστη, χωρίς να προσφέρουν μεγαλύτερη αποτελεσματικότητα στην αναζήτηση.
- Επίσης δεν υποστηρίζονται Construction of new elements, Aggregates και Arithmetic computations, γιατί δεν θα προσέφεραν μεγαλύτερη αποτελεσματικότητα στην αναζήτηση. Όμως θα μπορούσαν εύκολα να υποστηριχθούν κατά τη δημιουργία νέου query, όταν επιλέγουμε τα στοιχεία στα οποία θέλουμε να θέσουμε περιορισμούς κατά τη δημιουργία νέου template και όταν ορίζουμε αποτελέσματα στο template αντίστοιχα.

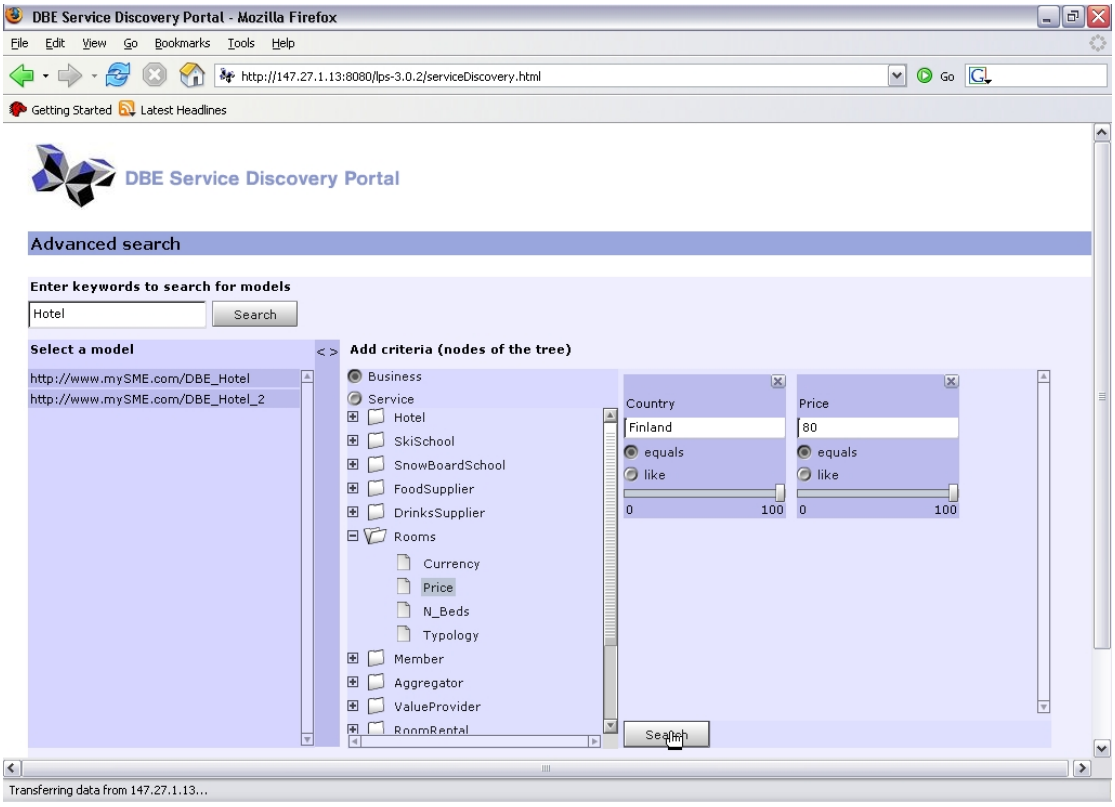
4.4.2 Υποστήριξη QML για την υλοποίηση στην πλατφόρμα OpenLaszlo

Για την υλοποίηση στο OpenLaszlo υποστηρίχθηκε μικρότερο κομμάτι της QML απ' ότι στην υλοποίηση για το Eclipse επειδή απευθύνεται αποκλειστικά σε απλούς χρήστες παρέχοντάς τους έναν πολύ εύκολο και γρήγορο τρόπο για

αναζήτηση υπηρεσιών. Ο τρόπος με τον οποίο υποστηρίχθηκε η QML στην υλοποίηση που έγινε για την πλατφόρμα OpenLaszlo φαίνεται παρακάτω:

- Στη δομή του πίνακα που αναπαριστά ένα query υποστηρίζουμε
  - Existential quantification για κάθε κριτήριο
  - Disjunction ανάμεσα κριτήρια
  - Filtering με τον ορισμό τιμών σε κάθε κριτήριο

  Στην εικόνα 26 παρουσιάζεται μία ερώτηση στην οποία φαίνονται τα παραπάνω χαρακτηριστικά.



Εικόνα 26: Υποστήριξη Existential Quantification, Disjunction και Filtering μέσα από ένα query στην OpenLaszlo υλοποίηση.
- Κάθε κριτήριο είναι ένας όρος exists.
- Μεταξύ των κριτηρίων εφαρμόζεται διάζευξη (Disjunction)
- Το Filtering γίνεται με την εισαγωγή τιμών στα κριτήρια.

- Κατά τη δημιουργία ερώτησης έμμεσα υποστηρίζεται Variable declaration, το οποίο γίνεται αυτόματα από το σύστημα.
- Υποστηρίζονται Multiple Documents, επιλέγοντας πολλά διαφορετικά κομμάτια από ένα μοντέλο (multiple contexts), το οποίο φαίνεται στο δέντρο που αναπαριστά το μοντέλο στην εικόνα 26.

- Τα υπόλοιπα χαρακτηριστικά της QML δεν τα υποστηρίξαμε σε αυτή την υλοποίηση γιατί θα καθιστούσαν την εφαρμογή πολύπλοκη και δύσχρηστη, χωρίς να προσφέρουν μεγαλύτερη αποτελεσματικότητα στην αναζήτηση.

4.5 Σχόλια – Παρατηρήσεις για τις πλατφόρμες Eclipse και Open Laszlo

Μετά την υλοποίηση των διεπαφών και στις δύο πλατφόρμες, ύστερα από αρκετό χρόνο ενασχόλησης με αυτές είμαστε πλέον σε θέση να κρίνουμε το κατά πόσο οι πλατφόρμες αυτές παρέχουν αρκετή λειτουργικότητα και πόσο εύκολη είναι η ανάπτυξη εφαρμογών σε αυτές.

Το Eclipse είναι μία ολοκληρωμένη πλατφόρμα ανάπτυξης εφαρμογών που παρέχει πάρα πολλές δυνατότητες. Αυτό που το κάνει μοναδικό είναι οι βιβλιοθήκες SWT με τις οποίες μπορείς να δημιουργήσεις δομικά γραφικά στοιχεία μέσα στο Eclipse με υπερβολικά εύκολο τρόπο. Βέβαια μερικές φορές ο τρόπος με τον οποίο διαχειρίζεται τα αντικείμενα αυτά είναι λίγο περιοριστικός και μπορεί να δημιουργήσει κάποια προβλήματα, όπως συναντήσαμε εμείς στο συγχρονισμό των threads, ενώ δεν μπορεί να τρέχει ανεξάρτητα του Eclipse αφού χρησιμοποιεί βιβλιοθήκες του. Επίσης ένα άλλο σημαντικό στοιχείο που κάνει ξεχωριστό το Eclipse είναι η δυνατότητα που δίνεται μέσα από το εργαλείο για διαχείριση/ανάπτυξη μίας εργασίας από πολλούς χρήστες ταυτόχρονα μέσω της χρήσης κατάλληλων μηχανισμών αποθήκευσης διαμοιραζόμενων πόρων με πολύ εύκολο τρόπο, κάτι που είναι πολύ χρήσιμο όταν πολλά άτομα δουλεύουν στην ίδια εργασία. Με αυτό τον τρόπο αποφεύγεται η επικάλυψη σε τμήματα κώδικα τα οποία τροποποιούν πολλά άτομα ταυτόχρονα.

Η πλατφόρμα OpenLaszlo παρέχει ένα εύκολο τρόπο δημιουργίας γραφικών σε Macromedia Flash [19] χωρίς να απαιτείται καθόλου γνώση της τεχνολογίας αυτής. Έχει πολλές δυνατότητες αλλά είναι αρκετά δύσκολη η διαχείριση των αντικειμένων που χρησιμοποιεί για να αποθηκεύει τις πληροφορίες που αναπαριστούν τα γραφικά στοιχεία που παράγει και μερικές φορές αρκετά περιοριστική. Χαρακτηριστικό είναι το πρόβλημα που αντιμετωπίσαμε στην αναπαράσταση μοντέλων σε δέντρο, καθώς δε μπορούσαμε να χρησιμοποιήσουμε τον

τρόπο που παρέχεται για αρχικοποίηση, εξαιτίας μεγάλου μεγέθους των μοντέλων μας, αναγκαζόμενοι να δημιουργούμε για κάθε στοιχείο του δέντρου ένα νέο γραφικό αντικείμενο. Παρόλα αυτά, λαμβάνοντας υπόψη επίσης και το γεγονός ότι είναι μία αρκετά καινούργια τεχνολογία η οποία θα εξελιχθεί τα επόμενα χρόνια, είναι μία πολύ καλή πλατφόρμα για την δημιουργία web εφαρμογών.

4.6 Ανακεφαλαίωση

Στο παρόν κεφάλαιο έγινε μία περιγραφή της υλοποίηση του συστήματός μας σε δύο διαφορετικές πλατφόρμες. Έτσι, ακολουθώντας την MVC αρχιτεκτονική μπορέσαμε να επαναχρησιμοποιήσουμε τμήματα του συστήματός μας για δύο φαινομενικά ασύμβατες πλατφόρμες, του Eclipse και του Open Laszlo. Η πρώτη υλοποίηση στην πλατφόρμα του Eclipse παρέχει ένα τρόπο αναζήτησης μοντέλων και υπηρεσιών με προχωρημένες δυνατότητες εκμεταλλευόμενο πολλά από τα χαρακτηριστικά της γλώσσας ερωτήσεων QML. Η δεύτερη υλοποίηση στην πλατφόρμα του OpenLaszlo παρέχει έναν πολύ εύκολο και γρήγορο τρόπο για αναζήτηση υπηρεσιών χρησιμοποιώντας μόνο τα χαρακτηριστικά της QML τα οποία είναι αναγκαία για την αναζήτηση αυτή.

Στο επόμενο κεφάλαιο θα γίνει περιγραφή της διαδικασίας που ακολουθήσαμε για τον ελέγχου του συστήματός μας ως προς τη σταθερότητα αλλά και την ευχρηστία του, ο οποίος έγινε τόσο από εμάς, όσο και από χρήστες (ως προς την ευχρηστία του).

Κεφάλαιο 5

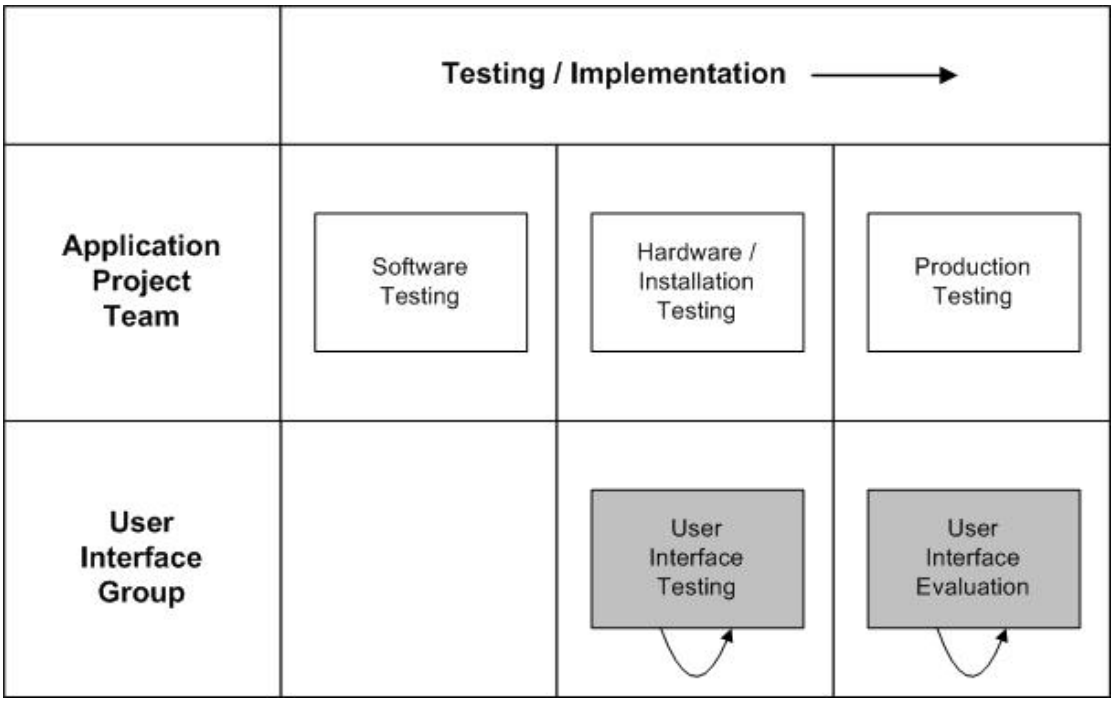
Έλεγχος σταθερότητας και ευχρηστίας συστήματος

5.1 Εισαγωγή

Στο κεφάλαιο αυτό θα περιγραφεί η διαδικασία ελέγχου του συστήματός μας ως προς τη σταθερότητα και την ευχρηστία του. Για τη διαδικασία αυτή βασιστήκαμε σε συγκεκριμένη μέθοδο που παρουσιάζεται στο βιβλίο της Deborah J. Mayhew, "Principles and Guidelines in Software User Interface Design" [26]. Για τη σταθερότητα θα περιγράψουμε αρχικά τις προδιαγραφές που θέλαμε να υποστηρίζει το σύστημά μας κατά τη λειτουργία του, τις περιπτώσεις που ελέγξαμε και τα συμπεράσματα στα οποία καταλήξαμε μετά τον έλεγχο των περιπτώσεων αυτών. Για την ευχρηστία του συστήματός μας θα περιγράψουμε τα σενάρια που ζητήθηκαν να εκτελέσουν οι χρήστες που δοκίμασαν το σύστημα, τα σχόλιά τους και την δίκη μας εκτίμηση μας για την αλληλεπίδραση χρηστών με το σύστημά μας.

5.2 Μέθοδος ελέγχου συστήματος

Η διαδικασία ελέγχου του συστήματός βασίστηκε σε μία μέθοδο που προτείνει η Deborah J. Mayhew στο βιβλίο της "Principles and Guidelines in Software User Interface Design" [26] και περιγράφεται σχηματικά στην εικόνα 27.



Εικόνα 27: Μέθοδος ελέγχου συστήματος από το βιβλίο της Meyhew "Principles and Guidelines in Software User Interface Design" [26].

Αρχικά ελέγχονται μικρά κομμάτια της υλοποίησης, στη συνέχεια ολόκληρη η υλοποίηση και τελικά γίνεται έλεγχος του συστήματος ως ολοκληρωμένη εφαρμογή.

Σύμφωνα με τη μέθοδο αυτή μπορούμε να χωρίσουμε τον έλεγχο αυτό σε τρία κομμάτια:

- Στο πρώτο κομμάτι ελέγχονται αυτόνομα κομμάτια κώδικα ως προς τη σωστή λειτουργίας τους.
- Στο δεύτερο κομμάτι ελέγχεται η σωστή εγκατάσταση και λειτουργία της εφαρμογής σε διάφορες πλατφόρμες. Κατά τη διάρκεια του ελέγχου γίνεται αξιολόγηση του συστήματος και από χρήστες που εκτελούν συγκεκριμένα σενάρια και από τα στοιχεία που θα συλλεγούν (σχόλια, απαντήσεις σε ερωτήσεις) μπορεί να προκύψει ανάγκη για αλλαγές στην εφαρμογή.

- Στο τρίτο κομμάτι μετά την υλοποίηση συνεχίζεται ο έλεγχος της σωστής λειτουργίας του συστήματος και συλλέγονται σχόλια από χρήστες που το δοκιμάζουν, ώστε αν υπάρχει ανάγκη να γίνουν διορθώσεις στην εφαρμογή (π.χ. νέες εκδόσεις) ή να δημιουργηθούν νέα προϊόντα αν οι ανάγκες των χρηστών το απαιτούν.

Στα πλαίσια της διπλωματικής επικεντρωθήκαμε στα δύο πρώτα κομμάτια της μεθόδου, ενώ στις επόμενες ενότητες θα περιγραφεί η διαδικασία ελέγχου που περιλαμβάνει το δεύτερο κομμάτι, όταν έχει ολοκληρωθεί ουσιαστικά η υλοποίηση (ως προς τη λειτουργικότητα που έπρεπε να καλύψει) και ελέγχεται η λειτουργία της ως ολοκληρωμένο σύστημα.

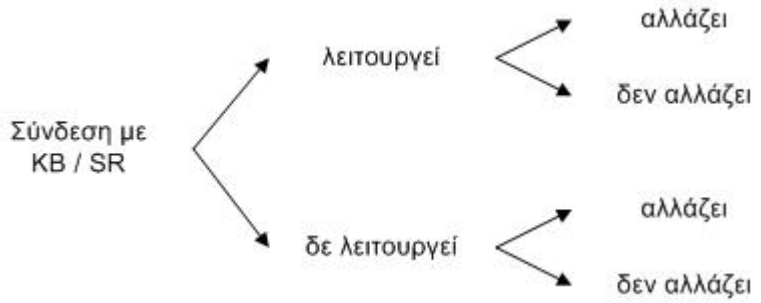
5.3 Έλεγχος σταθερότητας συστήματος

Πριν τον έλεγχο της σταθερότητας του συστήματος είναι βασικό να έχουν οριστεί οι προδιαγραφές που είναι επιθυμητό να ικανοποιεί αυτό και οι περιπτώσεις που προκύπτουν από αυτές και οι οποίες θα ελεγχθούν.

Έτσι για το σύστημά μας είναι απαραίτητο να ικανοποιούνται οι παρακάτω προδιαγραφές:

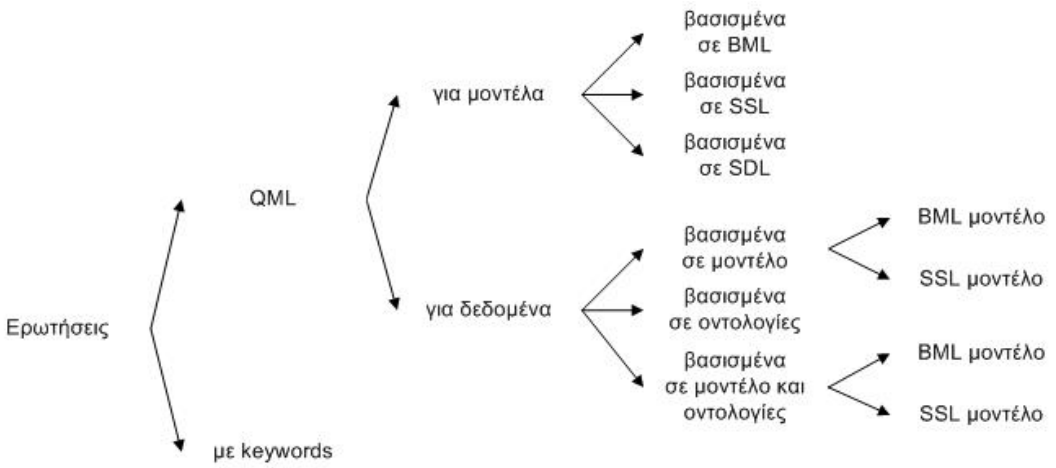
- Επικοινωνία με KB / SR κατανεμημένη σε peer-to-peer δίκτυο (στο Open Laszlo υπάρχει μόνο SR). Οι δυνατές καταστάσεις για αυτή την περίπτωση φαίνονται στο διάγραμμα της εικόνας 28.
- Ερωτήσεις βασισμένες σε QML για μοντέλα, δεδομένα και οντολογίες (στο Open Laszlo δε γίνονται QML ερωτήσεις για μοντέλα) και βασισμένες σε λέξεις-κλειδιά. Οι δυνατές καταστάσεις για αυτή την περίπτωση φαίνονται στο διάγραμμα της εικόνας 29.
- Δημιουργία, διαχείριση template και queries και επαναχρησιμοποίησή τους (μόνο στην πλατφόρμα Eclipse). Οι δυνατές καταστάσεις για αυτή την περίπτωση φαίνονται στο διάγραμμα της εικόνας 30.
- Εμφάνιση των αποτελεσμάτων και εκτέλεση λειτουργιών πάνω σε αυτά. Οι δυνατές καταστάσεις για αυτή την περίπτωση φαίνονται στο διάγραμμα της εικόνας 31.

Επίσης να αναφερθεί ότι έγινε έλεγχος για τη σωστή καθοδήγηση των χρηστών (εμφάνιση κατάλληλων οδηγιών) και για τα μηνύματα λάθους (ποια και πότε) που εμφανίζονται.



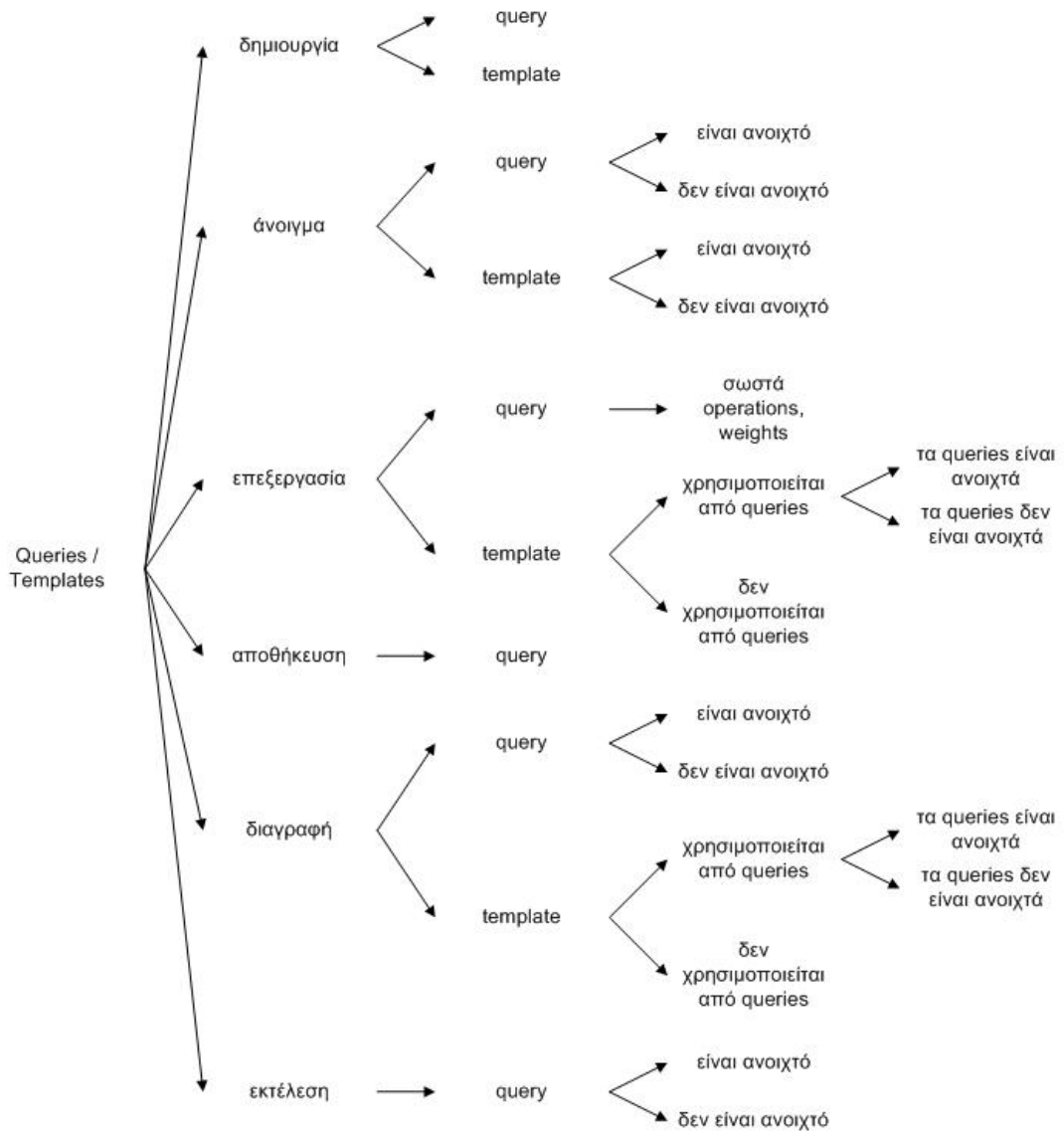
Εικόνα 28: Έλεγχος σταθερότητας συστήματος για επικοινωνία με KB / SR.

Οι δυνατές καταστάσεις στις οποίες μπορεί να βρεθεί η σύνδεση του συστήματος με την KB / SR. Το γεγονός ότι λειτουργεί η σύνδεση με KB/SR σημαίνει ότι υπάρχει σύνδεση με τη βάση γνώσης που έχει οριστεί από τον χρήστη ή το σύστημα ενώ το γεγονός ότι αλλάζει ή όχι η σύνδεση σημαίνει ότι ο χρήστης ορίζει άλλη βάση γνώσης με την οποία θέλει να συνδεθεί.



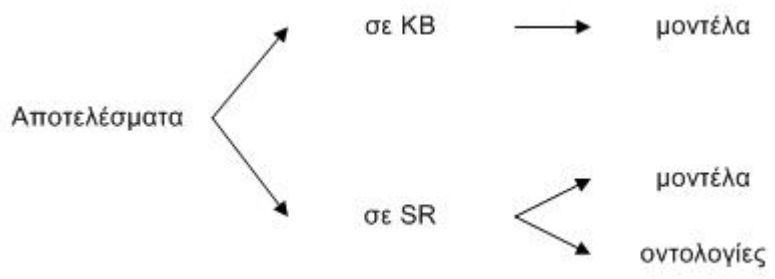
Εικόνα 29: Έλεγχος σταθερότητας συστήματος για δημιουργία ερωτήσεων.

Οι δυνατές καταστάσεις με τις οποίες μπορεί να γίνει μία ερώτηση. Μια ερώτηση σε μοντέλα πρέπει να αναφέρεται σε συγκεκριμένο μεταμοντέλο. Επειδή στις υλοποιήσεις που κάναμε είχαμε τρία μεταμοντέλα (BML – Business Model Language, SSL – Service Semantics Language, SDL – Service Description Languages), και γι'αυτό ο έλεγχος πρέπει να γίνει σε καθένα από αυτά ξεχωριστά, στα οποία βέβαια η διαχείριση γίνεται με κοινό τρόπο. Ένα μοντέλο μπορεί να περιέχει και έννοιες από οντολογίες (με αναφορά σ'αυτές), καθώς έχει δημιουργηθεί ένα μεταμοντέλο σε MOF (ODM – Ontology Definition Metamodel) για την αναπαράσταση οντολογιών σε αυτό.



Εικόνα 30: Έλεγχος σταθερότητας συστήματος για queries και templates.

Οι δυνατές καταστάσεις που υπάρχουν στο σύστημα για τα queries και τα templates και οι ενέργειες που μπορούν να εφαρμοστούν σ'αυτά.



Εικόνα 31: Έλεγχος σταθερότητας συστήματος για τα αποτελέσματα.

Οι δυνατές καταστάσεις για τα αποτελέσματα του συστήματος.

Ελέγχοντας τις προδιαγραφές μέσω όλων των δυνατών καταστάσεων στις οποίες μπορούν να βρεθούν που περιγράφηκαν παραπάνω και με όλους τους δυνατούς συνδυασμούς κατορθώσαμε να εξασφαλίσουμε τη σωστή λειτουργία του συστήματός μας σε όλες τις δυνατές καταστάσεις που μπορεί να βρεθεί. Ο έλεγχος έγινε τόσο για την εφαρμογή στην πλατφόρμα Eclipse όσο και για την πλατφόρμα του Open Laszlo (σε όσες περιπτώσεις έχουν νόημα για την κάθε υλοποίηση), ενώ πρέπει να αναφερθεί ότι κατά τη διάρκεια του ελέγχου διορθώθηκαν όσα λάθη εντοπίστηκαν. Αυτά ήταν κυρίως μηνύματα λάθους ή καθοδήγησης που δεν εμφανίζονταν ή δεν έπρεπε να εμφανιστούν και δυνατότητες που είχε ο χρήστης ενώ κανονικά δεν θα έπρεπε κάτω από ορισμένες συνθήκες.

5.4 Έλεγχος ευχρηστίας συστήματος

Με τον έλεγχο ευχρηστίας του συστήματος αυτό που θέλαμε να ελέγξουμε ήταν ουσιαστικά αν μπορεί ένας χρήστης να αντεπεξέλθει σε ένα σχετικά απαιτητικό σενάριο, χωρίς να γνωρίζει την εφαρμογή από πριν, με απλή περιγραφή των βημάτων που πρέπει να ακολουθήσει. Επειδή έγιναν δύο υλοποιήσεις ο έλεγχος έγινε για κάθε υλοποίηση χωριστά. Πρέπει να αναφερθεί ότι οι υλοποιήσεις που έγιναν στα πλαίσια αυτής της εργασίας δοκιμάστηκαν με επιτυχία στο review του DBE που έγινε τον Ιανουάριο του 2006. Επίσης εδώ και πέντε μήνες το σύστημά μας χρησιμοποιείται σε test λειτουργία με πραγματικές συνθήκες από developers, από SMEs και από άλλους χρήστες. Από τη διαδικασία αυτή είχαμε feedback για bugs, επιθυμητά features κ.α. τα οποία διορθώσαμε ή υλοποιήσαμε. Τέλος σε ένα evaluation για User Interfaces που έγινε για τα εργαλεία του DBE Studio είχαμε πολύ θετικά σχόλια.

5.4.1 Έλεγχος ευχρηστίας συστήματος Eclipse

Για την εφαρμογή του Eclipse ήταν αναγκαίο να τους αναλύσουμε κάποια βασικά χαρακτηριστικά για το DBE, για να κατανοήσουν τι αναζητούμε, και την MOF αρχιτεκτονική που χρησιμοποιεί, για να μπορούν να κατανοήσουν έννοιες όπως το μετα-μοντέλο και το μοντέλο, ενώ τους αναφέρθηκε και η αρχιτεκτονική templates

και queries που χρησιμοποιούμε στο σύστημά μας, για να κατανοήσουν τη διαδικασία που ακολουθούμε για τη δημιουργία μίας ερώτησης.

Ο έλεγχος έγινε από πέντε χρήστες που δεν είχαν σχέση με το ερευνητικό πρόγραμμα DBE και δεν είχαν εξειδικευμένες γνώσεις με τους ηλεκτρονικούς υπολογιστές.

Το σενάριο που εκτέλεσαν ζητούσε τα εξής:

- Δημιουργία template με τα εξής στοιχεία του BML μεταμοντέλου:
  - BusinessEntity.name
  - BusinessEntity.attribute.name
  - Asset.name
- Δημιουργία query με βάση το template αυτό
- Εισαγωγή των εξής κριτηρίων στο query
  - BusinessEntity.name = Hotel
  - BusinessEntity.attribute.name = StarCategory
  - Asset.name = Rooms
- Εκτέλεση του query
- Πλοήγηση στα μοντέλα που επιστρέφονται (στο BML κομμάτι τους)
- Προσθήκη του στοιχείου Service.name στο template
- Εισαγωγή επιπλέον κριτηρίου Service.name = Transportation
- Εκτέλεση του query
- Πλοήγηση στα μοντέλα που επιστρέφονται (στο BML κομμάτι τους)
- Δημιουργία template για συγκεκριμένο μοντέλο με στοιχεία
  - Hotel.country
  - Rooms.price
  - Transportation.starting-point
- Δημιουργία query με βάση το template αυτό
- Εισαγωγή των εξής κριτηρίων στο query
  - Hotel.country = Finland
  - Rooms.price < 80
  - Transportation.starting-point = Airport με βάρος 70%
- Εκτέλεση του query
- Πλοήγηση στις υπηρεσίες που επιστρέφονται (στα δεδομένα τους)
- Εκτέλεση μίας υπηρεσίας

Σε γενικές γραμμές όλοι οι χρήστες βρήκαν κατανοητή τη διαδικασία, σχολίασαν θετικά την καθοδήγηση και τα μηνύματα που εμφανίζονταν και δε βρέθηκαν σε κατάσταση που να μη μπορούν να βρουν τι να κάνουν για να συνεχίσουν.

Λαμβάνοντας υπόψη και το γεγονός ότι έρχονταν πρώτη φορά σε επαφή με την εφαρμογή ο έλεγχος δείχνει ότι ένας χρήστης που γνωρίζει το περιβάλλον εργασίας του DBE θα μπορέσει να την χρησιμοποιήσει με επιτυχία για αναζήτηση μοντέλων ή υπηρεσιών, ακόμη και αν δε γνωρίζει για την αρχιτεκτονική των templates και queries που ακολουθήσαμε, γιατί θα διαπιστώσει ότι για τη δημιουργία μίας ερώτησης (query) είναι απαραίτητο να επιλέξει ένα template στο οποίο θα αναφέρεται το query αυτό. Οπότε θα κατανοήσει ότι πρώτα πρέπει να δημιουργήσει ένα template που να ικανοποιεί τις ανάγκες του στην αναζήτηση μοντέλων ή δεδομένων και στη συνέχεια να δημιουργήσει ένα query στο template αυτό.

5.4.2 Έλεγχος ευχρηστίας συστήματος Laszlo

Για το Laszlo ήταν αναγκαίο να τους αναλύσουμε κάποια βασικά χαρακτηριστικά για το DBE, για να κατανοήσουν τι αναζητούμε, και την MOF αρχιτεκτονική που χρησιμοποιεί, για να μπορούν να κατανοήσουν έννοιες όπως το μετα-μοντέλο και το μοντέλο.

Ο έλεγχος έγινε από πέντε χρήστες που δεν είχαν σχέση με το ερευνητικό πρόγραμμα DBE και δεν είχαν εξειδικευμένες γνώσεις με τους ηλεκτρονικούς υπολογιστές.

Το σενάριο που εκτέλεσαν ζητούσε τα εξής:

- Μετάβαση στη Σύνθετη Αναζήτηση
- Αναζήτηση μοντέλου σχετικού με Hotel
- Άνοιγμα του Business κομματιού του μοντέλου
- Δημιουργία των εξής κριτηρίων
  - Hotel.country
  - Rooms.price
  - Transportation.starting-point
- Εισαγωγή των εξής τιμών στα κριτήρια
  - Hotel.country = Finland

- Transportation.starting-point = Airport με βάρος 70%

- Εκτέλεση του query
- Εμφάνιση των πληροφοριών μίας υπηρεσίας
- Εκτέλεση μίας υπηρεσίας

Σε γενικές γραμμές όλοι οι χρήστες βρήκαν πολύ κατανοητή τη διαδικασία, η οποία έτσι κι αλλιώς είναι αρκετά απλή και μπόρεσαν με πολύ εύκολο τρόπο να βρουν την επιθυμητή υπηρεσία.

Λαμβάνοντας υπόψη και το γεγονός ότι έρχονταν πρώτη φορά σε επαφή με την εφαρμογή ο έλεγχος δείχνει ότι ένας χρήστης που γνωρίζει το περιβάλλον εργασίας του DBE θα μπορέσει να την χρησιμοποιήσει με επιτυχία για αναζήτηση υπηρεσιών.

5.5 Ανακεφαλαίωση

Στο παρόν κεφάλαιο παρουσιάστηκε ο τρόπος με τον οποίο ελέγχθηκε η σταθερότητα και η ευχρηστία του συστήματος που υλοποιήσαμε. Αρχικά περιγράφηκε η μέθοδος της Deborah J. Mayhew στο βιβλίο της "Principles and Guidelines in Software User Interface Design" [26] στην οποία βασίστηκε η διαδικασία που ακολουθήσαμε. Στη συνέχεια περιγράψαμε τις προδιαγραφές που πρέπει να ικανοποιεί η λειτουργία του συστήματος, όλες τις δυνατές καταστάσεις στις οποίες μπορεί να βρεθεί για να ικανοποιούνται οι προδιαγραφές και το γεγονός ότι μετά το πέρας του ελέγχου το σύστημα λειτουργεί σωστά σε όποια κατάσταση και αν βρεθεί. Για τον έλεγχο ευχρηστίας του συστήματος περιγράψαμε τα σενάρια που κλίθηκαν να εκτελέσουν οι χρήστες που δοκίμασαν το σύστημά μας και τα θετικά σχόλια που αποκόμισε η εφαρμογή μας από τη διαδικασία αυτή.

Στο επόμενο κεφάλαιο γίνεται μία ανακεφαλαίωση όλων όσων αναφέρθηκαν στα προηγούμενα κεφάλαια και παρουσιάζονται κάποιες ιδέες για το πως μπορεί να επεκταθεί η παρούσα εργασία.

Κεφάλαιο 6

Ανακεφαλαίωση - μελλοντικές εργασίες

6.1 Εισαγωγή

Στο κεφάλαιο αυτό θα γίνει μία ανακεφαλαίωση των βασικών στοιχείων που παρουσιάστηκαν σε αυτή την εργασία μέσω των προηγούμενων κεφαλαίων και θα παρουσιαστούν κάποιες ιδέες για το πως μπορεί να επεκταθεί η παρούσα εργασία για να καλύψει κάποια επιπλέον θέματα.

6.2 Ανακεφαλαίωση

Το αντικείμενο της εργασίας αυτής είναι ο σχεδιασμός και υλοποίηση γραφικών διεπαφών χρήστη για πρόσβαση σε MOF (MetaObject Facility) [6] βάσεις γνώσεις.

Υλοποιήσαμε δύο γραφικές διεπαφές σε διαφορετικά περιβάλλοντα, στο Eclipse(σαν plug-in) και στο Laszlo (σαν web εφαρμογή), για την κατασκευή

ερωτήσεων εκφρασμένες σε QML (Query Metamodel Language) [7] προς MOF βάσεις γνώσεις, ώστε να μπορεί ένας χρήστης να αποκτήσει πρόσβαση σε αυτές χωρίς να έχει γνώση για τη δομή των μετα-δεδομένων που περιέχουν αλλά ούτε και της γλώσσας ερωτήσεων (QML) που χρησιμοποιείται, αποκρίπτοντάς την πολυπλοκότητά της.

Για την υποστήριξη περισσοτέρων της μίας υλοποίησης το σύστημα σχεδιάστηκε με βάση το MVC (Model-View-Controller) [5] πρότυπο σχεδίασης, ώστε να μπορεί όποιος επιθυμεί να δημιουργήσει μία γραφική διεπαφή στο ήδη υπάρχον σύστημα χωρίς να χρειάζεται να υλοποιήσει κάποιον μηχανισμό για τη διαχείριση των πολύπλοκων MOF πληροφοριών (μετα-μοντέλα, μοντέλα, οντολογίες, δεδομένα), αφού αυτός είναι κοινός για όλες τις υλοποιήσεις. Επίσης θελήσαμε το σύστημά μας να αποτελείται από αυτοδύναμες μονάδες ώστε αν προκύψει μία αλλαγή σε μία μονάδα να μην επηρεάζει τις άλλες.

Για την αναπαράσταση μίας ερώτησης ήταν επιθυμητό να χρησιμοποιούνται δύο διακριτές μονάδες. Στη μία μονάδα περιέχονται τα στοιχεία MOF (μετα-μοντέλο, μοντέλο, ή οντολογία) στα οποία θέλουμε να θέσουμε περιορισμούς (Template) και στην άλλη οι περιορισμοί αυτοί (Query). Ο διαχωρισμός αυτός επιτρέπει την ανεξαρτησία των στοιχείων που έχουμε επιλέξει από τις τιμές των περιορισμών που θέλουμε να πληρούν, ώστε ένα Template να μπορεί να επαναχρησιμοποιηθεί για τη δημιουργία πολλών Queries, ενώ η δομή του Template αποκρύπτει την πολυπλοκότητα των MOF πληροφοριών (μετα-μοντέλα, μοντέλα, οντολογίες) από το χρήστη.

Η υλοποίηση έγινε με τέτοιο τρόπο ώστε ο χρήστης να μπορεί να εκμεταλλευθεί τις (επιθυμητές) δυνατότητες που προσφέρει η QML για τη δημιουργία περιορισμών στο MOF, με γραφικό τρόπο, αποκρύπτοντας όμως την πολυπλοκότητά της. Στην υλοποίηση στο Eclipse, στην οποία ο χρήστης δημιουργεί και διαχειρίζεται templates και queries (στο Laszlo τη δημιουργία template και query για μία ερώτηση την κάνει το σύστημα αδιαφανώς), για την αναπαράσταση των queries (και κατ' επέκταση των ερωτήσεων) ακολουθήσαμε την μέθοδο QbE (Query by Example) [8], που είναι πολύ διαδεδομένη και παρέχει έναν φιλικό προς το χρήστη τρόπο δημιουργίας ερωτήσεων, χωρίς λάθη, που δεν απαιτεί την εξειδικευμένη γνώση κάποιας γλώσσα.

Οι υλοποιήσεις αυτές ελέγχθηκαν από εμάς ως προς τη σταθερότητά τους, ώστε να εξασφαλίσουμε ότι ικανοποιούν τις προδιαγραφές που επιθυμούμε και ότι το

σύστημά μας λειτουργεί σωστά σε όποια κατάσταση και αν βρεθεί. Επίσης ελέγχθηκε η ευχρηστία του συστήματος από χρήστες μέσα από σενάρια που κλίθηκαν να εκτελέσουν και για τις δύο υλοποιήσεις αποκομίζοντας θετικά σχόλια.

6.3 Μελλοντικές εργασίες

Με την παρούσα εργασία σχεδιάστηκαν και υλοποιήθηκαν γραφικές διεπαφές για πρόσβαση σε MOF βάσεις γνώσεις. Παρ' όλο που η εργασία αυτή είναι ολοκληρωμένη και έχει καλύψει πλήρως τις προδιαγραφές τις και τις ανάγκες δημιουργίας της, υπάρχουν μερικά θέματα τα οποία πραγματεύεται η εργασία τα οποία μπορούν να επεκταθούν ώστε να υποστηρίζονται πλήρως ή με καλύτερο τρόπο. Συγκεκριμένα αυτά είναι τα ακόλουθα:

- Αποθήκευση του MOF μετα-μοντέλου, μοντέλου, ή οντολογίας στο template τοπικά ώστε να μη χρειάζεται επικοινωνία με τη βάση γνώση για την επεξεργασία των templates.
- Αποθήκευση των templates στη βάση γνώσης με version μηχανισμούς, δίνοντας τη δυνατότητα στους χρήστες για αναζήτηση templates με συγκεκριμένα χαρακτηριστικά, π.χ. τον τύπο τους, αν αναφέρεται σε MOF μετα-μοντέλο, μοντέλο, ή οντολογία, ή την περιγραφή τους, ή ακόμα και κάποιες λέξεις-κλειδιά που να το περιγράφουν (επεκτείνοντας έτσι το template).
- Δημιουργία και υποστήριξη OCL Expressions μέσω της εφαρμογής που έγινε στην πλατφόρμα του Eclipse.
- Υποστήριξη μεγαλύτερου υποσυνόλου (ή ακόμη και ολόκληρης) της QML στην εφαρμογή της πλατφόρμας Eclipse.

Παράρτημα Α

Οδηγός χρήσης QFSDT

Α.1 DBE Semantic Discovery Perspective

Το perspective στο περιβάλλον του Eclipse είναι ένα σύνολο από views και μίας editor area των οποίων οι θέσεις πάνω στην επιφάνεια εργασίας είναι προκαθορισμένες. Για το γραφικό περιβάλλον του εργαλείου έχει δημιουργηθεί ένα τέτοιο perspective, το DBE Semantic Discovery Perspective, το οποίο αποτελείται από τα εξής:

- Views:
    - Query Navigator, όπου φαίνονται τα templates και τα queries που δημιουργήθηκαν με τη χρήση του QFSDT.
    - Properties View, όπου φαίνονται πληροφορίες για κάθε κριτήριο ενός query και υπάρχει δυνατότητα αλλαγής τιμών.
    - Template Properties View, όπου φαίνονται πληροφορίες σχετικές με το template στο οποίο αντιστοιχεί κάποιο query ή με κάποιο template το οποίο επέλεξε ο χρήστης για επισκόπηση.
    - Results View, όπου φαίνονται τα αποτελέσματα που επιστρέφει η εκτέλεση ενός query.

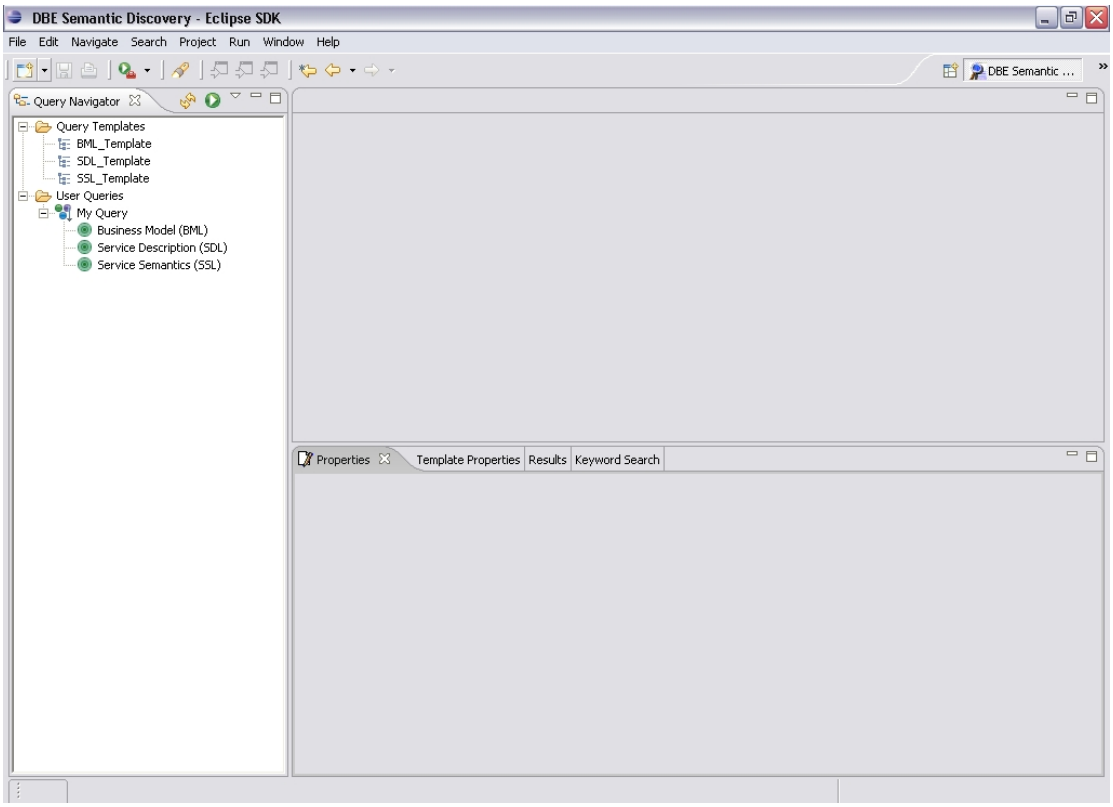
- Keyword Search View, όπου παρέχεται ένας τρόπος γρήγορης αναζήτησης με την εισαγωγή λέξεων-κλειδιών. Τα αποτελέσματα της γρήγορης αναζήτησης επιστρέφονται στο ίδια view.

- Editor area:
  - Στην περιοχή αυτή φαίνονται οι editors με τους οποίους μπορεί κάποιος να δημιουργήσει περιορισμούς στα queries (Για κάθε query ανοίγει ένας editor).

Κάθε view που ανήκει στο εργαλείο μπορεί να κλείσει. Για να ξανανοιχτεί από το DBE Semantic Discovery Perspective, θα πρέπει να επιλεχθεί Window -> Show View -> και ύστερα το View που είναι επιθυμητό να ανοιχθεί.

Α.2 Ξεκινώντας το QFSDT

Την πρώτη φορά που ξεκινάει το QFSDT δημιουργεί templates και queries (βασισμένα στα πρώτα) για να μπορεί ο χρήστης να καταλάβει εύκολα τη συσχέτιση μεταξύ ενός query κι ενός template. Τα templates και τα queries είναι αποθηκευμένα στο τοπικό σύστημα αρχείων και πιο συγκεκριμένα στο runtime workspace του Eclipse. Στην εικόνα 32 παρουσιάζεται τι θα αντικρίσει ο χρήστης την πρώτη φορά που θα ξεκινήσει το QFSDT.



Εικόνα 32: Το Perspective του QFSDT.

A.2.1 Δημιουργώντας ένα template

Για τη δημιουργία ενός νέου template υπάρχουν τρεις τρόποι για να ενεργήσει ο χρήστης:

- να επιλέξει File -> New -> Query Template
- να επιλέξει New template στο pull-down μενού του Query Navigator View
- να κάνει δεξί κλικ σε ένα template και να επιλέξει New template

Επιλέγοντας κάποιο από τα παραπάνω εμφανίζεται ένας wizard. Ο wizard αποτελείται από τέσσερα βήματα από τα οποία τα δύο πρώτα είναι υποχρεωτικά και τα άλλα δύο είναι προαιρετικά.

Σε όλα τα βήματα του wizard υπάρχουν στο πάνω μέρος της σελίδας οδηγίες προς τον χρήστη για τον τρόπο με τον οποίο πρέπει να ενεργήσει ώστε να ολοκληρώσει επιτυχώς το κάθε βήμα. Σε κάθε περίπτωση παράλειψης ή λάθους, στο ίδιο σημείο εμφανίζεται το κατάλληλο μήνυμα για να ενημερώσει το χρήστη ο οποίος θα μπορέσει εύκολα να προχωρήσει σε διορθώσεις.

Συνοπτικά τα βήματα του wizard είναι τα ακόλουθα:

- Βήμα $1^{ο}$: Ορισμός ονόματος, περιγραφής και τύπου του νέου template.

- Βήμα 2^{ο} : Επιλογή των στοιχείων που θα περιέχει το template.
- Βήμα 3^{ο} : Ορισμός συζεύξεων στα στοιχεία του template (προαιρετικό).
- Βήμα 4^{ο} : Ορισμός συνόλου αποτελεσμάτων (προαιρετικό).

Παρακάτω αναλύονται οι λειτουργίες του κάθε βήματος.

A.2.1.1 Βήμα 1^{ο} : Ορισμός ονόματος, περιγραφής και τύπου του νέου template

Στη σελίδα του πρώτου βήματος υπάρχουν πεδία και λίστες επιλογής για συμπλήρωση στοιχείων για τη δημιουργία του template. Η σελίδα αυτή φαίνεται στην εικόνα 33.

Για να δημιουργηθεί ένα νέο template θα πρέπει να εισαχθούν οι παρακάτω πληροφορίες:

- Το όνομα του template,
- Μία περιγραφή (προαιρετικά),
- Ένας τύπος για το template

Είναι επίσης δυνατό να αντιγραφούν τιμές ενός υπάρχοντος template διαλέγοντας το template από τη λίστα με τα διαθέσιμα templates.

Υπάρχουν τρεις τύποι για ένα template. Η λειτουργικότητα που παρέχεται για κάθε τύπο είναι :

- Metamodel

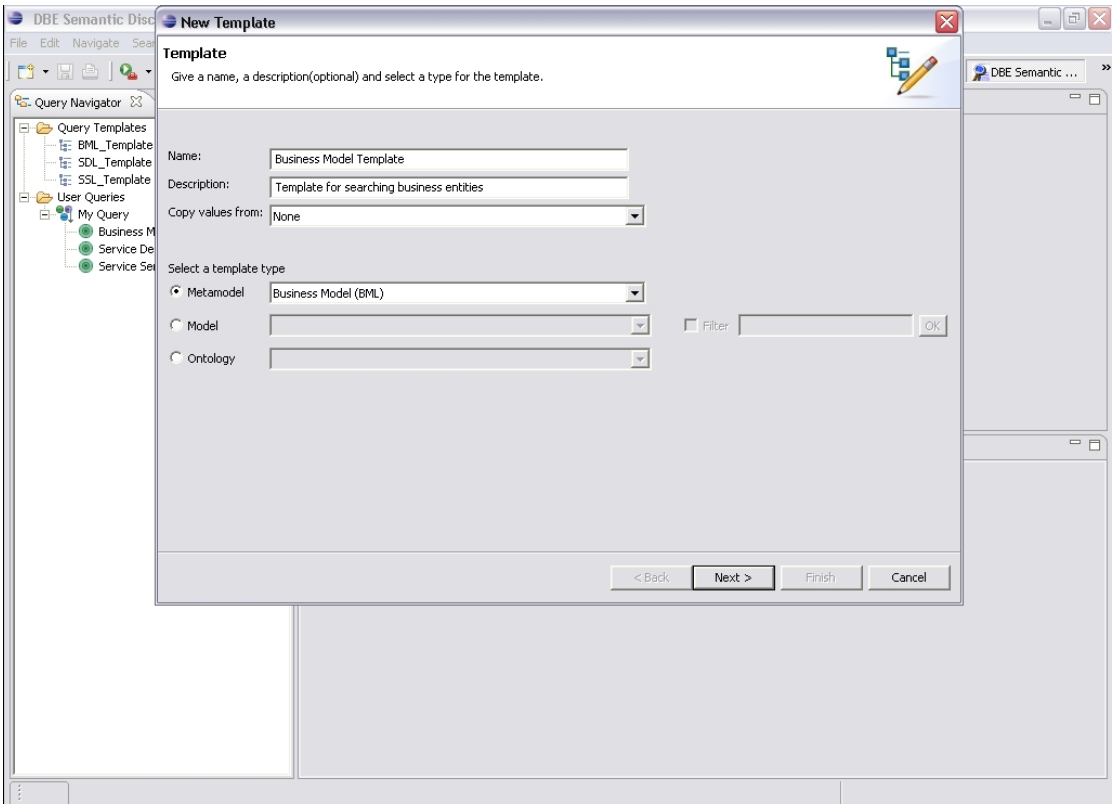
  Επιλέγοντας μετα-μοντέλο (Metamodel) ο χρήστης θα μπορεί να επιλέξει ανάμεσα στα υπάρχοντα μετα-μοντέλα (BML, SSL, SDL) τα οποία είναι ορισμένα από τις προδιαγραφές του DBE.
- Model

  Επιλέγοντας μοντέλο (Model) ο χρήστης θα μπορεί να επιλέξει ανάμεσα στα υπάρχοντα μοντέλα. Κατά την εκκίνηση του wizard, το σύστημα ανακτά όλα τα μοντέλα που υπάρχουν αποθηκευμένα στην βάση γνώσεων στην οποία έχει συνδεθεί το σύστημα. Επίσης ο χρήστης έχει τη δυνατότητα να αναζητήσει μοντέλα χρησιμοποιώντας λέξεις-κλειδιά (επιλέγοντας το checkbox filter δίπλα από τη λίστα των μοντέλων και εισάγοντας τις επιθυμητές λέξεις-κλειδιά) οι οποίες μπορεί να περιγράφουν κάποια από την πληροφορία των μοντέλων.
- Ontology

  Επιλέγοντας οντολογία (Ontology) ο χρήστης θα μπορεί να επιλέξει ανάμεσα στις

υπάρχουσες οντολογίες. Κατά την εκκίνηση του wizard, το σύστημα ανακτά όλες τις οντολογίες που υπάρχουν αποθηκευμένες στην βάση γνώσεων στην οποία έχει συνδεθεί το σύστημα.

Μετά το τέλος της εισαγωγής των στοιχείων θα πρέπει ο χρήστης να πιέσει το κουμπί "Next" για να συνεχίσει με τη δημιουργία του νέου template. Αν για κάποιο λόγο θέλει να διακόψει τη δημιουργία του νέου template μπορεί να πιέσει το κουμπί "Cancel" σε οποιοδήποτε βήμα του wizard.



Εικόνα 33: Ορισμός ονόματος, περιγραφής και τύπου του νέου template.

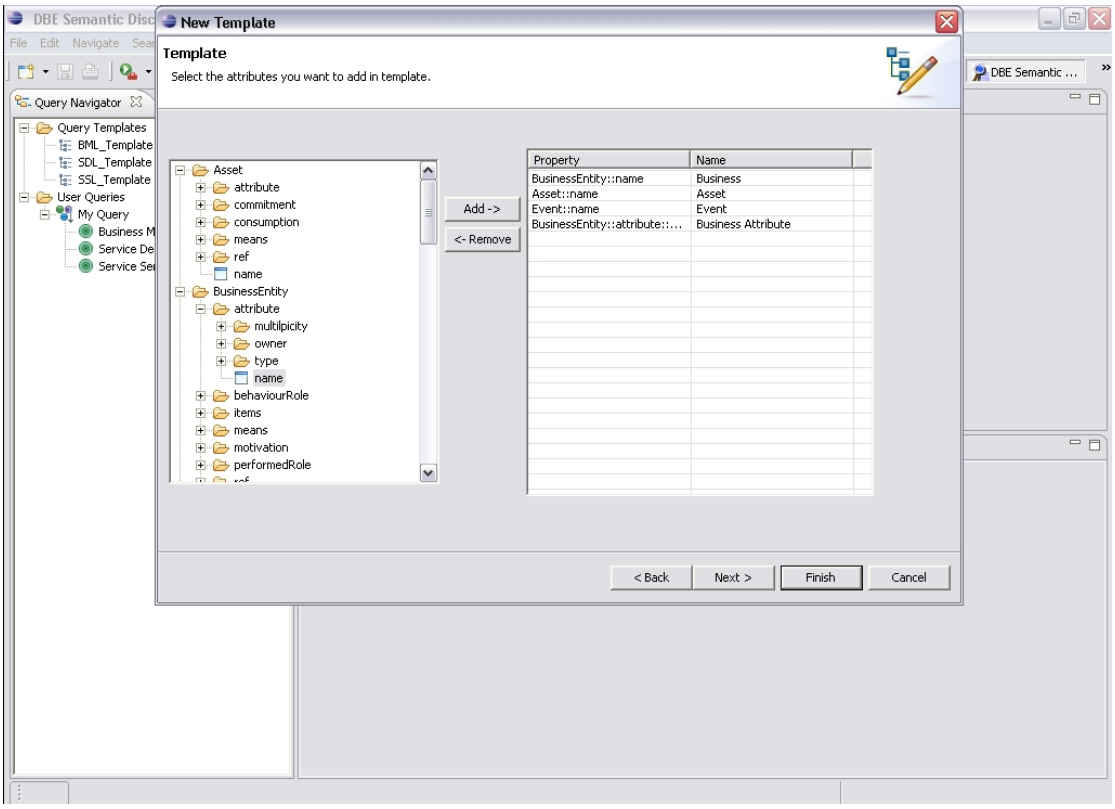
A.2.1.2 Βήμα 2ο : Επιλογή των στοιχείων που θα περιέχει το template

Στη σελίδα του δεύτερου βήματος του wizard υπάρχει ένα δέντρο στην αριστερή πλευρά της σελίδας κι ένας πίνακας με δύο στήλες στη δεξιά πλευρά. Σ'αυτό το βήμα ο χρήστης επιλέγει τα στοιχεία που θέλει να εισάγει στο template από το δέντρο. Το δέντρο αναπαριστά μία ιεραρχική δομή που αναφέρεται στον τύπο (μετα-μοντέλο, μοντέλο ή οντολογία) του template που είχε επιλέξει ο χρήστης στην πρώτη σελίδα του wizard. Τα στοιχεία που μπορούν να επιλεχτούν είναι

συγκεκριμένα και είναι τα φύλλα του δέντρου. Η σελίδα αυτή φαίνεται στην εικόνα 34.

Ο χρήστης μπορεί να επιλέξει το επιθυμητό στοιχείο με διπλό κλικ ή να επιλέξει το επιθυμητό στοιχείο από το δέντρο και να πιέσει το κουμπί "Add". Το επιλεγμένο στοιχείο θα προστεθεί στον πίνακα και πιο συγκεκριμένα στην πρώτη στήλη του. Στη δεύτερη στήλη του πίνακα πρέπει ο χρήστης μπορεί να εισάγει ένα μοναδικό και περιγραφικό (για να είναι φιλικό στον χρήστη) όνομα που θα αντιπροσωπεύει το στοιχείο της πρώτης στήλης (αυτόματα το σύστημα δίνει ένα όνομα από μόνο του). Σε περίπτωση που ο χρήστης θέλει να αφαιρέσει ένα στοιχείο από τον πίνακα, αρκεί να επιλέξει το στοιχείο που είχε εισάγει και να πιέσει το κουμπί "Remove".

Για να συνεχίσει ο χρήστης στο επόμενο βήμα θα πρέπει να πιέσει το κουμπί "Next". Αν πατήσει το κουμπί "Back" θα γυρίσει στο πρώτο βήμα του wizard. Αν ο χρήστης δεν θέλει να εισάγει άλλες πληροφορίες για το template μπορεί να τερματιστεί τον wizard πιέζοντας το κουμπί "Finish", ενώ αν θέλει να ακυρώσει τη διαδικασία δημιουργίας νέου template θα πρέπει να πιέσει το κουμπί "Cancel".



Εικόνα 34: Εισαγωγή στοιχείων στο template.

A.2.1.3 Βήμα 3^{ο} : Ορισμός συζεύξεων στα στοιχεία του template

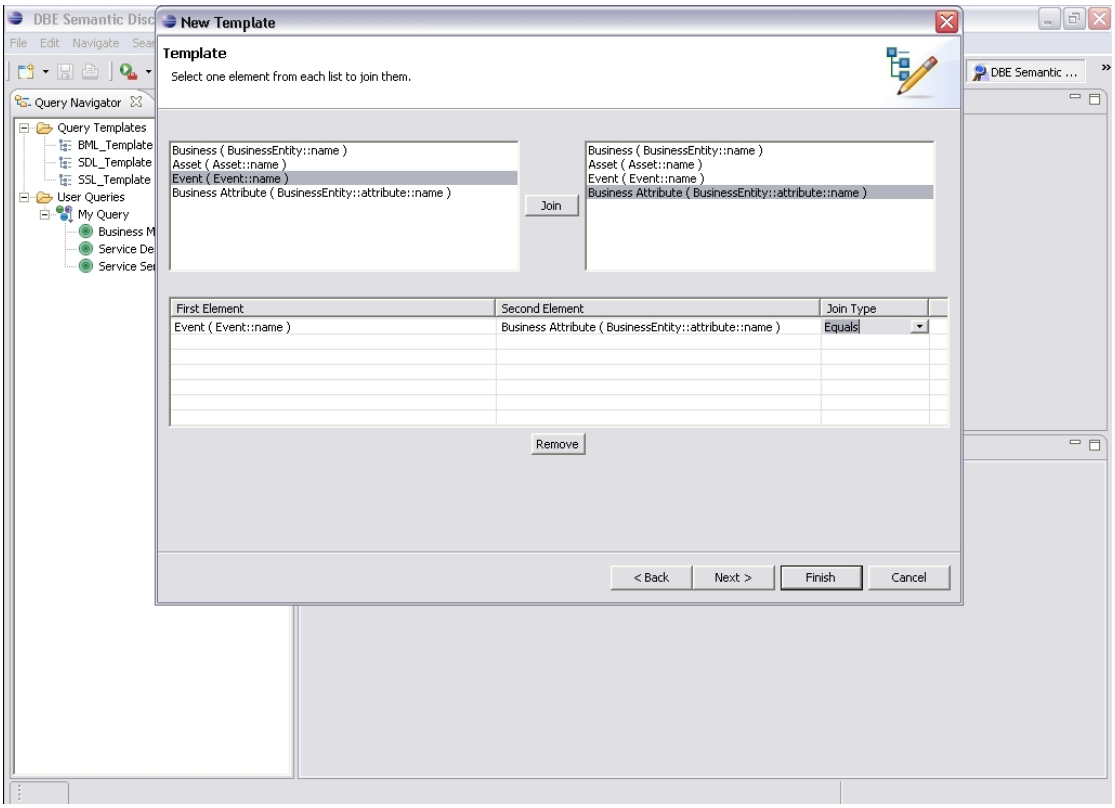
Στη σελίδα του τρίτου βήματος του wizard υπάρχουν τρεις πίνακες. Στο επάνω μέρος της οθόνης είναι δύο πίνακες οι οποίοι είναι πανομοιότυποι και σαν δεδομένα έχουν τα στοιχεία που έχει επιλέξει ο χρήστης στο προηγούμενο βήμα του wizard (δεύτερο βήμα). Η σελίδα αυτή φαίνεται στην εικόνα 35.

Ο χρήστης μπορεί να δημιουργήσει συζεύξεις μεταξύ των στοιχείων των δύο πινάκων. Επιλέγοντας ένα στοιχείο από τον κάθε πίνακα και πιέζοντας το κουμπί "Join" δημιουργείται σύζευξη μεταξύ των στοιχείων. Η σύζευξη προστίθεται σ'έναν τρίτο πίνακα ο οποίος είναι στο κάτω μέρος της σελίδας του τρίτου βήματος του wizard.

Ο τρίτος πίνακας έχει τρεις στήλες. Στις δύο πρώτες φαίνονται τα δύο στοιχεία στα οποία έχει γίνει η σύζευξη από τους δύο πάνω πίνακες. Στην τρίτη στήλη ο χρήστης επιλέξει τον τύπο σύζευξης των δύο στοιχείων. Οι τιμές στις οποίες μπορεί να γίνει η επιλογή του τύπου είναι "Contains" και "Equals" που σημαίνει ότι το δεύτερο στοιχείο θα περιέχεται στις πληροφορίες του πρώτου ή ότι οι τιμές του πρώτου και του δεύτερου στοιχείου θα έχουν ίδιες τιμές αντίστοιχα.

Αν ο χρήστης θέλει να αφαιρέσει μία σύζευξη από αυτές που έχει δημιουργήσει, αρκεί να επιλέξει τη συγκεκριμένη σύζευξη στον τρίτο πίνακα ο οποίος βρίσκεται στο κάτω μέρος της σελίδας του τρίτου βήματος του wizard και να πιέσει το κουμπί "Remove".

Αν ο χρήστης θέλει να τερματίσει τη δημιουργία νέου template αρκεί να πιέσει το κουμπί "Finish". Σε περίπτωση που θέλει να προσθέσει νέα στοιχεία στο template θα πρέπει να πιέσει το κουμπί "Back" για να επιλέξει τα στοιχεία που επιθυμεί στη σελίδα του δεύτερου βήματος του wizard. Για να συνεχίσει να προσθέτει πληροφορίες για το template ο χρήστης θα πιέσει το κουμπί "Next" ενώ αν θέλει να ακυρώσει τη διαδικασία δημιουργίας νέου template θα πρέπει να πιέσει το κουμπί "Cancel".



Εικόνα 35: Ορισμός συζεύξεων στο template.

A.2.1.4 Βήμα 4ο : Ορισμός συνόλου αποτελεσμάτων

Στη σελίδα του τέταρτου βήματος του wizard φαίνεται ένα δέντρο στην αριστερή πλευρά της σελίδας κι ένας πίνακας στη δεξιά πλευρά της σελίδας. Το δέντρο είναι το ίδιο με αυτό που εμφανίζεται στη σελίδα του δεύτερου βήματος του wizard. Η σελίδα αυτή φαίνεται στην εικόνα 36.

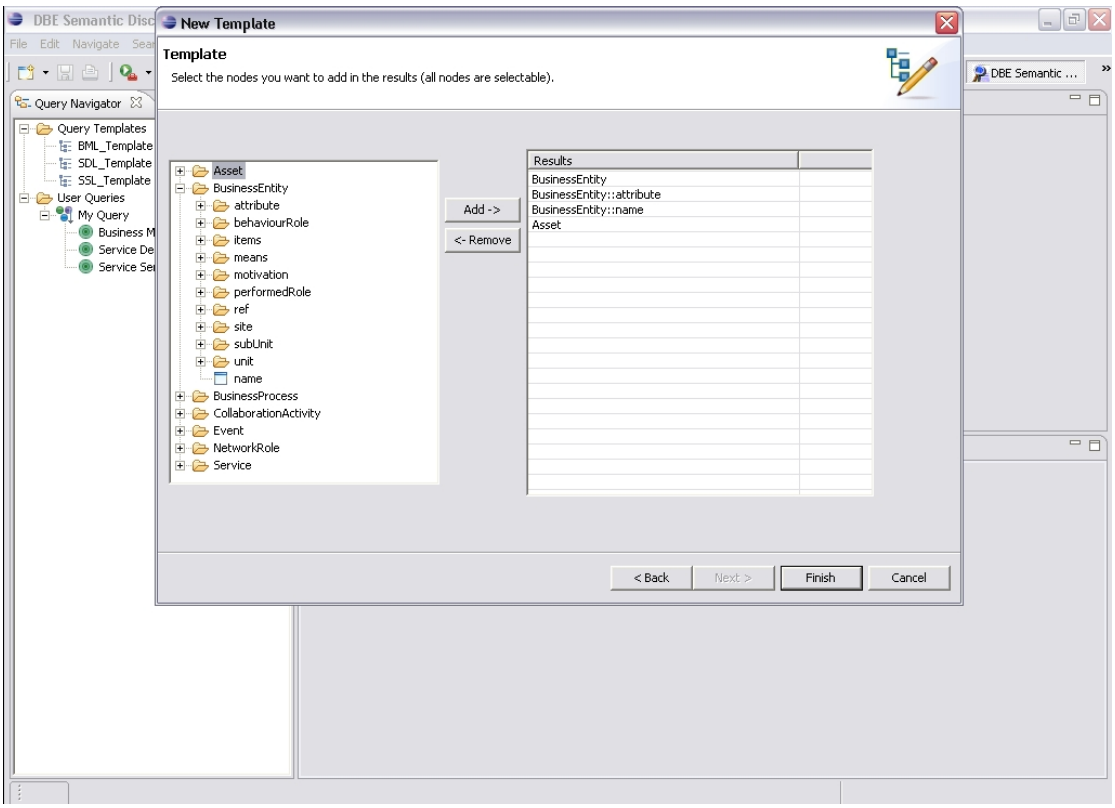
Ο χρήστης μπορεί να δημιουργήσει το δικό του σύνολο αποτελεσμάτων στη περιοχή επισκόπησης των αποτελεσμάτων. Δηλαδή σε περίπτωση που ο χρήστης δεν θέλει να επισκοπήσει ολόκληρο τον όγκο της πληροφορίας που θα επιστρέψει η εκτέλεση του query που θα βασιστεί στο template το οποίο θα δημιουργηθεί μετά το πέρας της διαδικασίας δημιουργίας νέου template, θα δηλώσει ένα σετ από χαρακτηριστικά γνωρίσματα για να μπορεί να πλοηγηθεί σ'αυτά. Κάθε στοιχείο του δέντρο μπορεί να επιλεχθεί σαν μέρος του συνόλου των αποτελεσμάτων και προστίθεται στον πίνακα που φαίνεται στο αριστερό μέρος της σελίδας του τέταρτου βήματος του wizard.

Για να γίνει εισαγωγή ενός στοιχείου από το δέντρο στο σύνολο των αποτελεσμάτων, αρκεί ο χρήστης να κάνει διπλό κλικ στο στοιχείο που θέλει να

εισάγει ή να επιλέξει ένα στοιχείο από το δέντρο και να πιέσει έπειτα το κουμπί "Add".

Αν ο χρήστης θέλει να αφαιρέσει ένα στοιχείο από το σύνολο των αποτελεσμάτων, θα πρέπει να επιλέξει το στοιχείο αυτό στον πίνακα που αναπαριστά το σύνολο των αποτελεσμάτων και μετά να πιέσει το κουμπί "Remove".

Για να τερματιστεί ο wizard και να δημιουργηθεί το νέο template θα πρέπει ο χρήστης να πιέσει το κουμπί "Finish". Αν θέλει να αλλάξει κάτι στο προηγούμενο βήμα του wizard αρκεί να πιέσει το κουμπί "Back" ενώ για να ακυρωθεί η δημιουργία του νέου template ο χρήστης πιέσει το κουμπί "Cancel".



Εικόνα 36: Ορισμός συνόλου αποτελεσμάτων στο template.

Μετά το τέλος της διαδικασίας δημιουργίας νέου template, το σύστημα αποθηκεύει αυτόματα το νέο template στο τοπικό σύστημα αρχείων και ανανεώνει την περιοχή του Query Navigator View όπου φαίνεται πλέον και το νέο template που δημιούργησε ο χρήστης. Το σύστημα θα ρωτήσει τον χρήστη αν θέλει να δημιουργήσει άμεσα και ένα query το οποίο θα βασίζεται στο νέο template.

Α.2.2 Διαγράφοντας ένα template

Για να διαγραφεί ένα template, πρέπει να επιλεχθεί αυτό το template στο Query Navigator View και μετά να γίνει μία από τις ακόλουθες ενέργειες:

- δεξί κλικ και μετά επιλογή του "Delete Template" από το εμφανιζόμενο μενού, ή
- να πιεστεί το πλήκτρο "Delete" στο πληκτρολόγιο

Σε κάθε περίπτωση το σύστημα θα ζητήσει την επιβεβαίωση της διαγραφής.

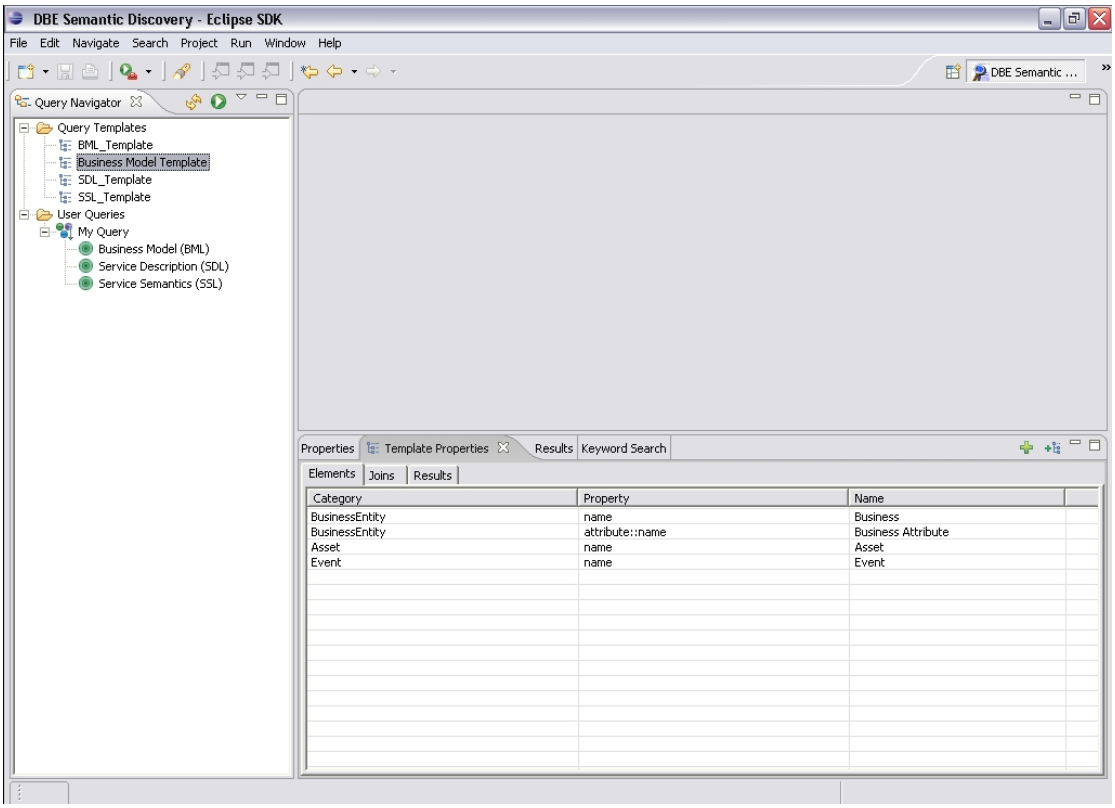
Α.2.3 Επισκόπηση template

Η επισκόπηση ενός template γίνεται στο Template Properties View. Για να δει ο χρήστης τις πληροφορίες που περιέχει ένα template μπορεί να ενεργήσει με τους ακόλουθους τρόπους:

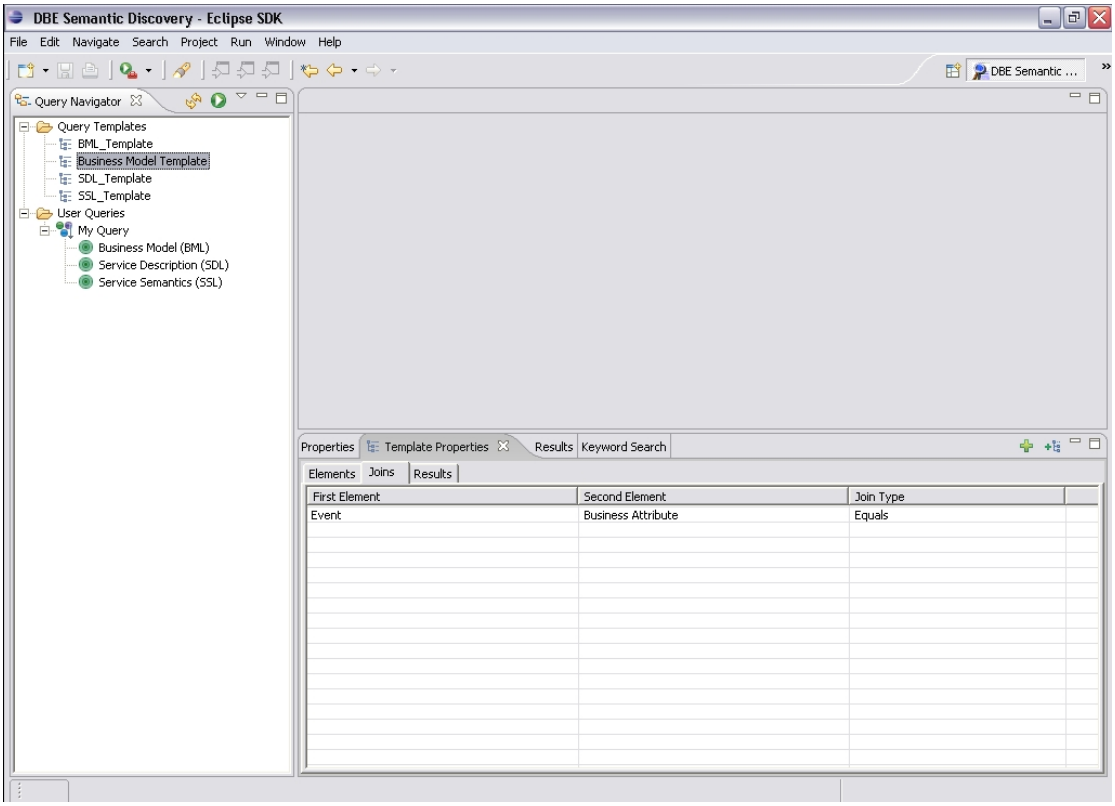
- δεξί κλικ στο template και μετά επιλογή του "Open Template" από το εμφανιζόμενο μενού
- διπλό κλικ στο template
- να πιέσει το πλήκτρο "Enter" στο πληκτρολόγιο όταν θα είναι επιλεγμένο το template που ο χρήστης θέλει να επισκοπήσει
- να επιλέξει το Template Properties View όταν ένα query θα έχει ανοίξει σε έναν editor (για να δει τις πληροφορίες του template στο οποίο είναι βασισμένη το query)

Στο Template Properties View ο χρήστης μπορεί να δει τις παρακάτω πληροφορίες που αφορούν στο template :

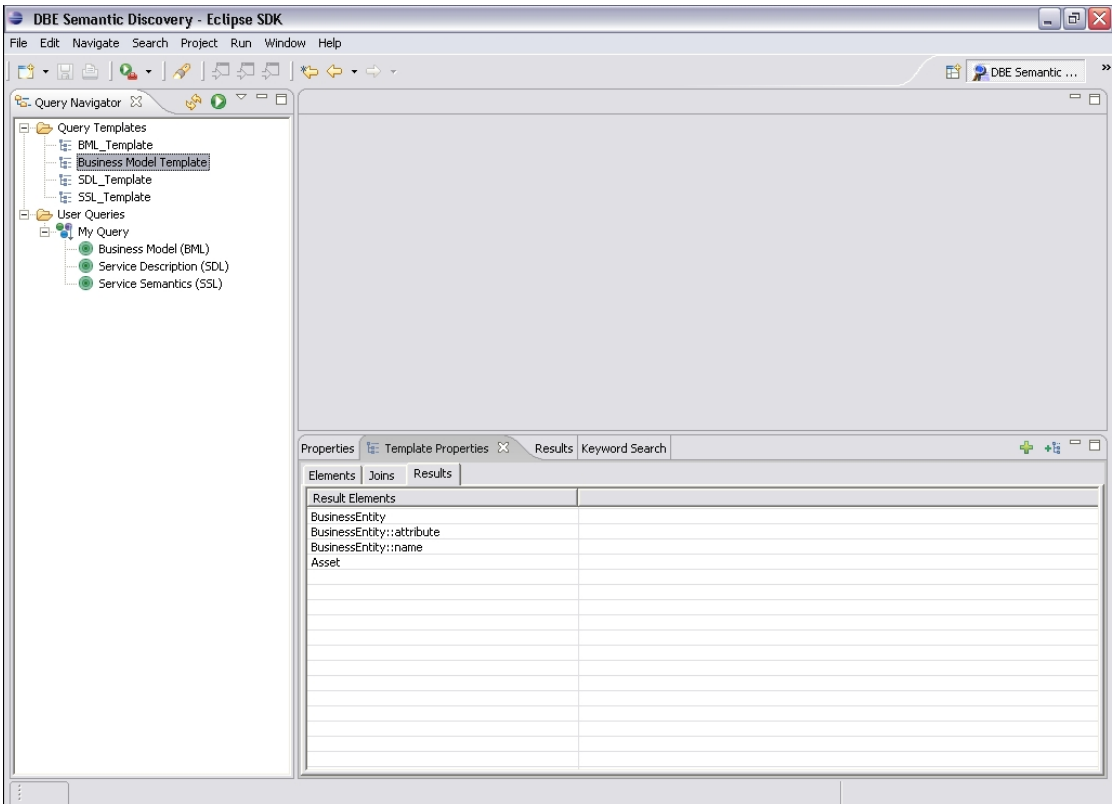
- τα χαρακτηριστικά γνωρίσματα που συνθέτουν το template και το όνομα το οποίο έχει δώσει ο χρήστης που φαίνεται στην εικόνα 37
- τις συζεύξεις που έχει δημιουργήσει σ'αυτά τα χαρακτηριστικά γνωρίσματα ο χρήστης που φαίνεται στην εικόνα 38
- το σύνολο αποτελεσμάτων που έχει ορίσει ο χρήστης για τα queries που βασίζονται σ'αυτό το template που φαίνεται στην εικόνα 39



Εικόνα 37: Επισκόπηση των χαρακτηριστικών γνωρισμάτων που συνθέτουν το template.



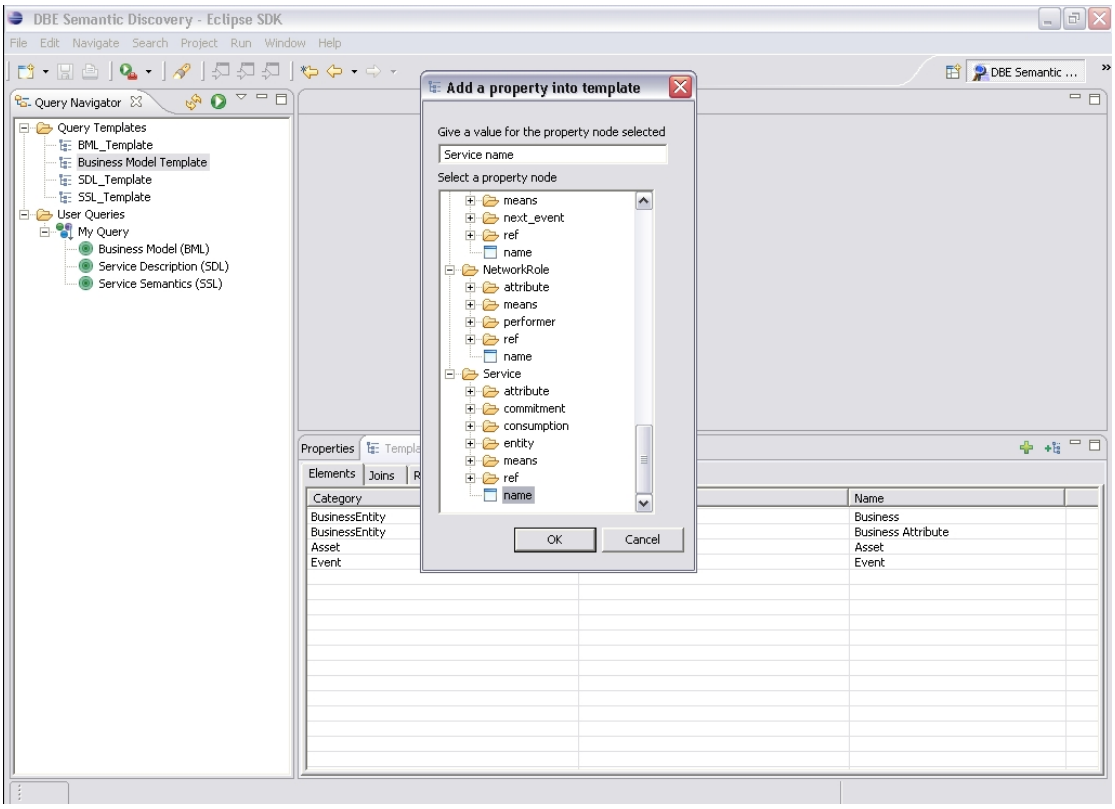
Εικόνα 38: Επισκόπηση των συζεύξεων στα χαρακτηριστικά γνωρίσματα του template.



Εικόνα 39: Επισκόπηση του συνόλου αποτελεσμάτων του template.

Στην γραμμή εργαλείων του Template Properties View ο χρήστης έχει τη δυνατότητα να προσθέσει με γρήγορο τρόπο ένα στοιχείο στο τρέχον template πιέζοντας το κουμπί “Add an element to the template ”. Θα εμφανιστεί ένα παράθυρο, που φαίνεται στην εικόνα 40, στο οποίο φαίνεται ένα δέντρο που αναπαριστά την πληροφορία του τύπο του template από το οποίο ο χρήστης μπορεί να διαλέξει ένα φύλλο (το οποίο αντιπροσωπεύει ένα χαρακτηριστικό γνώρισμα) και να του δώσει ένα όνομα που θα το περιγράφει (το σύστημα δίνει ένα όνομα κατά την επιλογή του στοιχείου από το δέντρο). Σε περίπτωση λανθασμένης εισαγωγής στοιχείων, το σύστημα ενημερώνει τον χρήστη με το κατάλληλο μήνυμα λάθους. Η ενέργεια αυτή είναι δυνατό επίσης να εκτελεστεί και κατά την επεξεργασία ενός query, κάνοντας δεξί κλικ στην περιοχή του editor και επιλέγοντας “Add an element to the template ” από το μενού που εμφανίζεται.

Επίσης ο χρήστης έχει τη δυνατότητα να επεξεργαστεί το template πιέζοντας το κουμπί “Edit elements, joins and results in template ”.



Εικόνα 40: Προσθήκη ενός νέου στοιχείου στο template

A.2.4 Επεξεργασία template

Για να τροποποιηθεί ένα template ο χρήστης κάνει δεξί κλικ επάνω του και επιλέγει "Edit template" από το εμφανιζόμενο μενού. Ύστερα θα εμφανιστεί ένας wizard ο οποίος θα έχει τα βήματα δύο έως τέσσερα του wizard δημιουργίας νέου template με τη διαφορά ότι τα δεδομένα του template που είχαν οριστεί θα φαίνονται στην σελίδα του wizard που τους αντιστοιχεί. Οι αλλαγές που γίνονται στο template επηρεάζουν τα queries που βασίζονται σε αυτό.

A.2.5 Δημιουργώντας ένα query

Ένα query στο QFSDT περιέχει τιμές που είναι οι περιορισμοί για τα χαρακτηριστικά γνωρίσματα του template στο οποίο το query θα αναφέρεται.

Για να δημιουργήσει κάποιος ένα query θα πρέπει να υπάρχει ένα template πάνω στο οποίο θα βασίζεται το query. Αν κάποιο από τα υπάρχοντα template

καλύπτει τον χρήστη τότε μπορεί να γίνει απευθείας το query. Σε αντίθετη περίπτωση θα πρέπει το template να δημιουργηθεί.

Εφόσον υπάρχει το template πάνω στο οποίο μπορεί να δημιουργηθεί το query ο χρήστης μπορεί να ενεργήσει με έξι τρόπους για να ξεκινήσει τη διαδικασία της δημιουργίας του:

- να επιλέξει File -> New -> Query
- να επιλέξει New Query στο pull-down του Query Navigator View
- να κάνει δεξί κλικ σε ένα query (αν υπάρχει) και να επιλέξει New query
- να κάνει δεξί κλικ σε ένα γκρουπ query (αν υπάρχει) και να επιλέξει New query (επιλέγεται αυτόματα το γκρουπ στο οποίο θα ανήκει το query)
- να κάνει δεξί κλικ σε ένα template (αν υπάρχει) και να επιλέξει New query (επιλέγεται αυτόματα το template στο οποίο θα βασίζεται το query)
- μετά το τέλος της δημιουργίας νέου template να επιλέξει να δημιουργήσει ένα query που θα βασίζεται στο template αυτό (επιλέγεται αυτόματα το template στο οποίο θα βασίζεται το query)

Ενεργώντας με μία από τις παραπάνω ενέργειες ο wizard που φαίνεται στην εικόνα 41 εμφανίζεται.

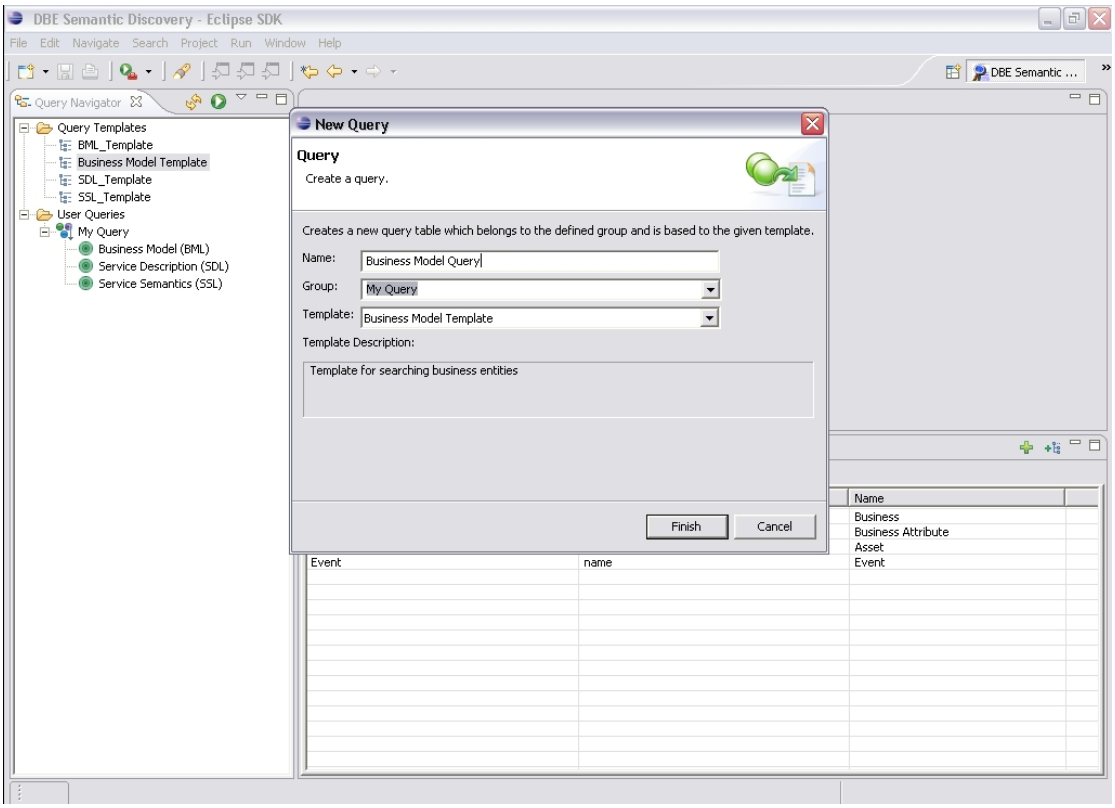
Υπάρχουν στο πάνω μέρος της σελίδας οδηγίες προς τον χρήστη για τον τρόπο με τον οποίο πρέπει να ενεργήσει ώστε να ολοκληρώσει επιτυχώς τον wizard. Σε κάθε περίπτωση παράλειψης ή λάθους, στο ίδιο σημείο εμφανίζεται το κατάλληλο μήνυμα για να ενημερώσει το χρήστη ο οποίος θα μπορέσει εύκολα να προχωρήσει σε διορθώσεις.

Πρέπει να ακολουθηθούν τα παρακάτω βήματα ώστε να δημιουργηθεί μία νέου query:

- Να εισαχθεί ένα όνομα για το query.
- Να εισαχθεί το γκρουπ στο οποίο αυτό το query θα ανήκει, ώστε να υπάρχει κατηγοριοποίηση σε ομάδες (π.χ. το γκουπ να εξαρτάται από το template στο οποίο θα ανήκει το query).
- Να επιλεχθεί ένα template για το query το οποίο θα ορίζει τι θα ψάχνει ο χρήστης χρησιμοποιώντας αυτό το query. Αν δεν υπάρχουν template ή κανένα από τα υπάρχοντα templates δεν μπορεί να χρησιμοποιηθεί, πριν δημιουργηθεί το query θα πρέπει να δημιουργηθεί ένα template.

Πιέζοντας το “Finish” δημιουργείται ένα νέο query.

Για να ακυρωθεί η διαδικασία δημιουργίας νέου query ο χρήστης θα πρέπει να πιέσει το κουμπί “Cancel”.



Εικόνα 41: Δημιουργία νέου query.

Μετά το τέλος της διαδικασίας δημιουργίας νέου query, το σύστημα αποθηκεύει αυτόματα το νέο query στο τοπικό σύστημα αρχείων και ανανεώνει το Query Navigator View, όπου φαίνεται πλέον και το νέο query που δημιούργησε ο χρήστης. Επίσης ανοίγει ένας editor για την επεξεργασία του νέου query.

A.2.6 Επεξεργασία query

Ένα query αναπαριστάται σαν πίνακας. Όταν ανοίγεται ή δημιουργείται ένα query, ανοίγει ένας editor που περιέχει έναν πίνακα. Κάθε κελί του πίνακα μπορεί να πάρει μια τιμή που είναι το κριτήριο που πρέπει να ικανοποιείται από τα αποτελέσματα όταν θα εκτελεστεί το query. Για κάθε κριτήριο μπορεί να οριστεί μια πράξη και ένα βάρος σημασίας. Το βάρος σημασίας διευκρινίζει πόσο σημαντικό είναι το συγκεκριμένο χαρακτηριστικό για τη συνολική ερώτηση. Για παράδειγμα αν ένα χαρακτηριστικό έχει μικρή σημασία, τα αποτελέσματα δεν θα είναι απαραίτητο

να το ικανοποιούν. Έτσι, δίνονται μεγαλύτερα βάρη στα σημαντικά χαρακτηριστικά. Οι διαθέσιμες πράξεις ποικίλουν ανάλογα από τις τιμές που μπορεί να πάρει το χαρακτηριστικό (π.χ. αλφαριθμητική, αριθμητική κλπ.). Για τις αλφαριθμητικές τιμές οι πράξεις είναι:

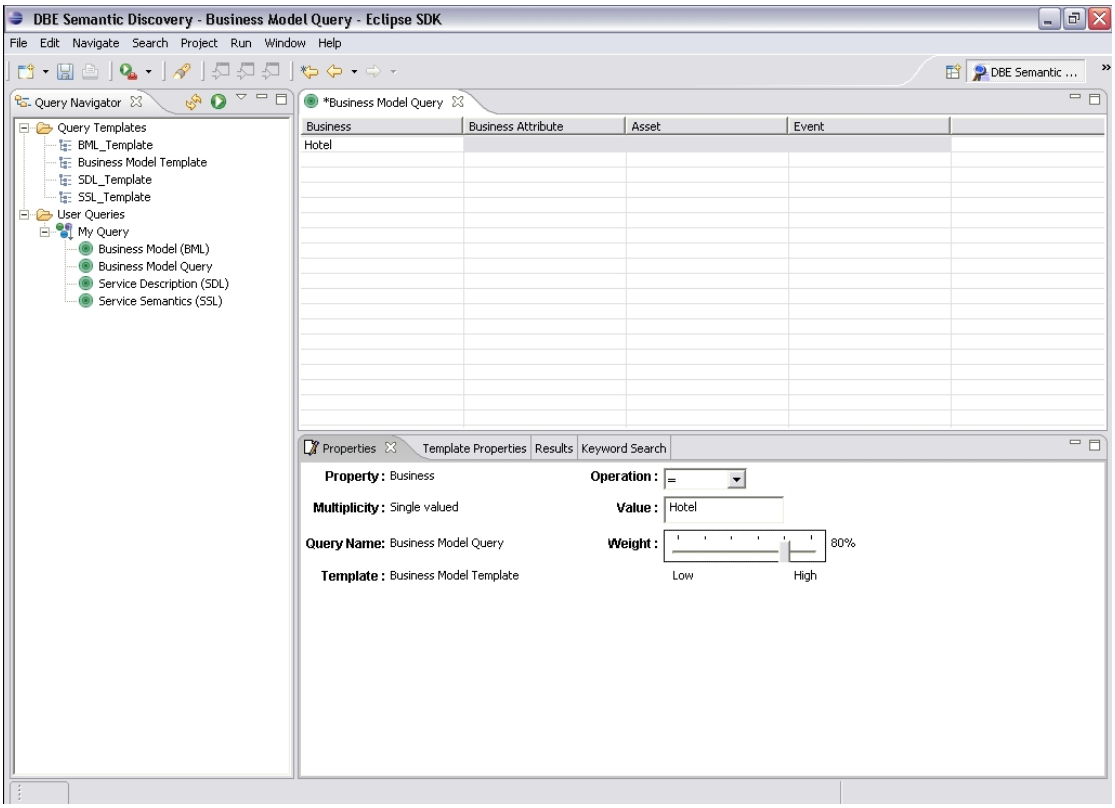
- *ισούται(=)*, η τιμή του χαρακτηριστικού πρέπει να είναι ακριβώς ίδια
- *μοιάζει(like)*, η τιμή του χαρακτηριστικού μπορεί να περιέχει την τιμή. Για παράδειγμα για την τιμή του χαρακτηριστικού "HotelReservation" για ένα μοντέλο υπηρεσιών για ξενοδοχεία θα ταιριάζει με την τιμή "Hotel" με την πράξη μοιάζει.
- *Διαφορετικό(<>),* η τιμή του χαρακτηριστικού πρέπει να είναι διαφορετική.

Όταν ανοίγει ένας editor, μπορούν να εισαχθούν, να επεξεργαστούν και να διαγραφούν οι τιμές των κελιών. Επιλέγοντας ένα κελί στον πίνακα, το Properties View εμφανίζεται στην οθόνη και περιέχει την πληροφορία του τρέχοντος κελιού. Μπορούν να αλλαχθούν η τιμή, η πράξη και το βάρος σημασίας. Για να ξεκινήσει η επεξεργασία, θα πρέπει να γίνει μία από τις ακόλουθες ενέργειες:

- διπλό κλικ στο κελί,
- να πιεστεί το πλήκτρο "F2" στο πληκτρολόγιο, ή
- να πιεστεί το πλήκτρο "Enter" στο πληκτρολόγιο

Να σημειωθεί ότι η πράξη πρέπει να εισαχθεί πριν από την τιμή και αναγνωρίζεται αυτόματα από το σύστημα εφόσον υπάρχει. Αν δεν εισαχθεί καμιά πράξη, τότε η πράξη "=" επιλέγεται αυτόματα.

Η επεξεργασία ενός query φαίνεται στην εικόνα 42.



Εικόνα 42: Επεξεργασία query.

Α.2.7 Αποθηκεύοντας ένα query

Όταν ένας πίνακας ενός query αλλάζει, ένας αστερίσκος εμφανίζεται στον τίτλο του παραθύρου του editor που σημαίνει ότι τα περιεχόμενα αυτού του query έχουν αλλάξει. Οι αλλαγές μπορούν να σωθούν με μία από τις ακόλουθες ενέργειες:

- να επιλεχθεί “File->Save” από το μενού,
- να πιεστεί “Ctrl+S” στο πληκτρολόγιο

Αν υπάρχουν μη αποθηκευμένες αλλαγές σε queries και γίνει απόπειρα να κλείσει το Eclipse, το σύστημα θα ρωτήσει για την αποθήκευση των μη αποθηκευμένων αλλαγών.

Α.2.8 Διαγράφοντας ενός query

Για να διαγραφεί ένα query, πρέπει να επιλεχθεί αυτό το query στο Query Navigator View και μετά να γίνει μία από τις ακόλουθες ενέργειες:

- δεξί κλικ και μετά επιλογή του “Delete Query” από το εμφανιζόμενο μενού ή

- να πιεστεί το πλήκτρο “Delete” στο πληκτρολόγιο

Σε κάθε περίπτωση το σύστημα θα ζητήσει την επιβεβαίωση της διαγραφής.

Α.2.9 Εκτελώντας ένα query

Για να πάρει ο χρήστης αποτελέσματα για ένα query που έχει δημιουργήσει θα πρέπει να το εκτελέσει. Ένα query μπορεί να εκτελεστεί με μια από τις ακόλουθες ενέργειες:

- πιέζοντας το κουμπί “Execute editor query” στη γραμμή εργαλείων του Query Navigator View,
- πιέζοντας το πλήκτρο “F5” στο πληκτρολόγιο στην περιοχή του editor που έχει ανοιχθεί για το query,
- κάνοντας δεξί κλικ πάνω στο query στο Query Navigator View και επιλέγοντας “Execute this query” από το εμφανιζόμενο μενού
- κάνοντας δεξί κλικ στην περιοχή του editor που έχει ανοιχθεί για το query και επιλέγοντας “Execute query” από το εμφανιζόμενο μενού

Αφού εκτελεστεί το query, η περιοχή επισκόπησης των αποτελεσμάτων θα εμφανιστεί, Results View, περιέχοντας τα αποτελέσματα της ερώτησης, τα οποία μπορούν να εξερευνηθούν.

Α.2.10 Εξερευνώντας τα αποτελέσματα

Μετά την εκτέλεση ενός query, γίνεται ανάκτηση των αποτελεσμάτων που ικανοποιούν τα κριτήρια του query. Τα αποτελέσματα προστίθενται σ’έναν πίνακα με τρεις στήλες:

- Το όνομα του αποτελέσματος (Model Name)
- Το ποσοστό σχετικότητας του αποτελέσματος σε σχέση με τα κριτήρια της ερώτησης (Rank)
- Μία περιγραφή για τα περιεχόμενα του αποτελέσματος (Description)

Κάθε λίγα δευτερόλεπτα ο πίνακας των αποτελεσμάτων ανανεώνεται με νέα αποτελέσματα τα οποία προστίθενται στη λίστα των αποτελεσμάτων. Οι ακόλουθες ενέργειες μπορούν να εφαρμοστούν στην περιοχή επισκόπησης των αποτελεσμάτων:

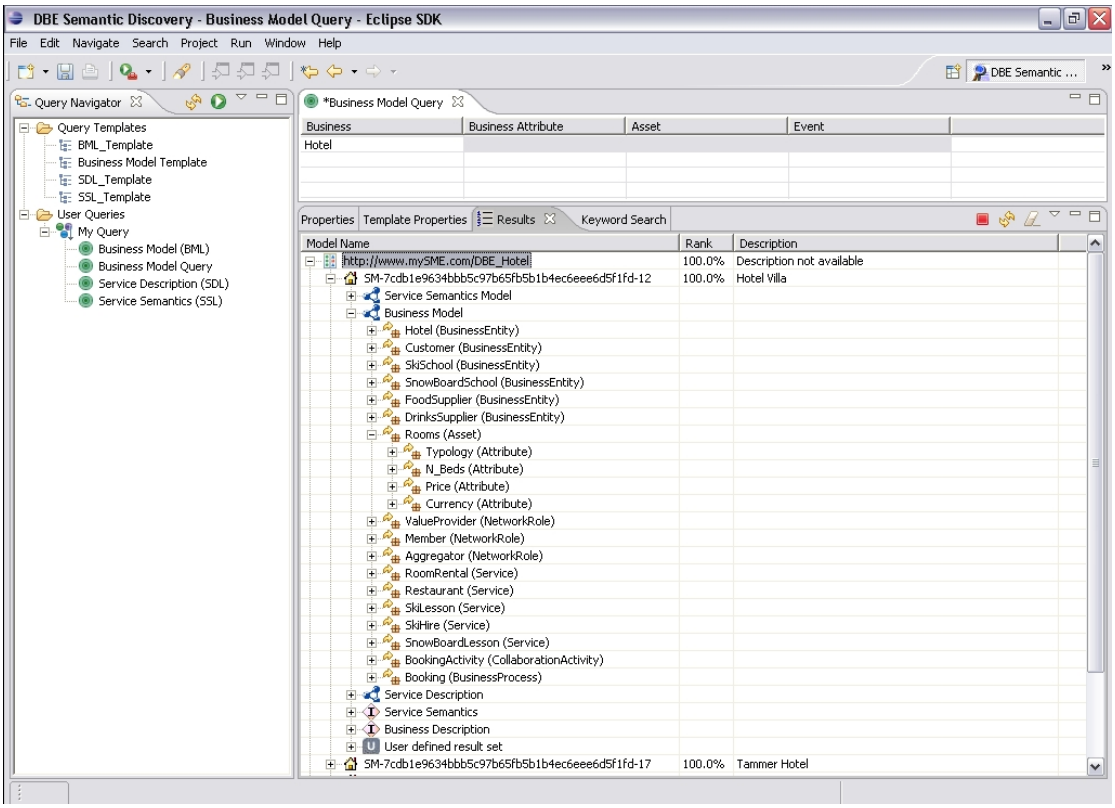
- Σταμάτημα της ανανέωσης των αποτελεσμάτων, πιέζοντας το κουμπί “Stop fetching results”.
- Ανανέωση των αποτελεσμάτων, πιέζοντας το κουμπί “Refresh results”.
- Καθαρισμός των αποτελεσμάτων, πιέζοντας το κουμπί “Clear results”.

Τα αποτελέσματα επιστρέφονται ταξινομημένα με την σχετικότητά τους ως προς τα κριτήρια της ερώτησης (ποσοστό επί τις εκατό). Η σχετικότητα ενός αποτελέσματος φαίνεται στην στήλη Rank.

Ο χρήστης μπορεί να δει περισσότερες πληροφορίες για το αποτέλεσμα πλοηγούμενος στις πληροφορίες που επιστρέφονται. Ανάλογα με τον τύπο του template στο οποίο στηρίζεται το query, θα μπορεί να πλοηγηθεί στους παρακάτω τύπους αποτελεσμάτων :

- Αν το template (και κατ’ επέκταση το query) αναφέρεται σε μοντέλα ή σε οντολογίες (Model , Ontology), τα αποτελέσματα που θα επιστραφούν θα είναι service manifests που ακολουθούν αυτό το μοντέλο ή θα περιέχουν έννοιες από οντολογία. Εξερευνώντας ένα service manifest, ο χρήστης θα έχει τη δυνατότητα να εξερευνήσει την παρακάτω πληροφορία:
  - το μοντέλο υπηρεσιών (Service Semantics Model),
  - το μοντέλο επιχειρήσεων (Business Model),
  - το μοντέλο περιγραφής υπηρεσιών (Service Description),
  - τα δεδομένα υπηρεσιών (Service Semantics),
  - τα δεδομένα επιχειρήσεων (Business Description),
  - το σύνολο των αποτελεσμάτων που έχει οριστεί στο template το οποίο ακολουθεί το query (User defined result set)
- Αν το template (και κατ’ επέκταση το query) αναφέρεται σε μετα-μοντέλο (Metamodel), τα αποτελέσματα που θα επιστραφούν θα είναι μοντέλα. Εξερευνώντας ένα μοντέλο, θα προστεθούν σαν παιδιά αυτού του μοντέλου τα service manifests που ακολουθούν αυτό το μοντέλο. Η εξερεύνηση του service manifest αναφέρθηκε παραπάνω.

Η πλοήγηση στα αποτελέσματα φαίνεται στην εικόνα 43.



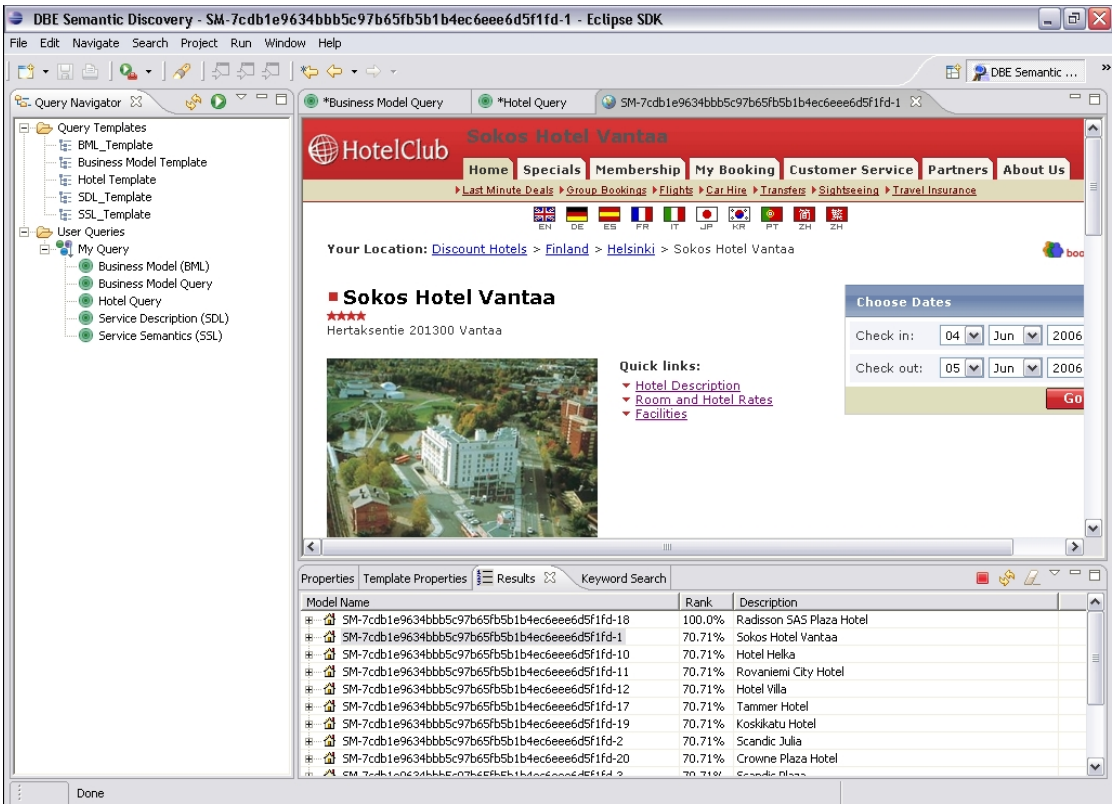
Εικόνα 43: Πλοήγηση στα αποτελέσματα.

Α.2.11 Λειτουργικότητα στα αποτελέσματα

Ανάλογα με τον τύπο του αποτελέσματος, ο χρήστης έχει μια σειρά από λειτουργικότητες στο Result View. Πριν αναλυθούν αυτές οι λειτουργικότητες, θα αναφερθεί ένας μηχανισμός που αναπτύχθηκε για την επικοινωνία του QFSDT με άλλα εργαλεία στο περιβάλλον του DBE Studio.

Έχει δημιουργηθεί μία διεπαφή για να συνδέονται εργαλεία κι επεξεργαστές στο QFSDT ώστε να εκμεταλλεύονται μέρος της λειτουργικότητάς του. Με τη σύνδεση αυτή μπορεί κάποιος να εγγράψει μία ενέργεια στο QFSDT, έτσι ώστε κάνοντας δεξί κλικ σε κάποιο αποτέλεσμα ή καλώντας το pull-down μενού, να εμφανίζεται η ενέργεια αυτή.

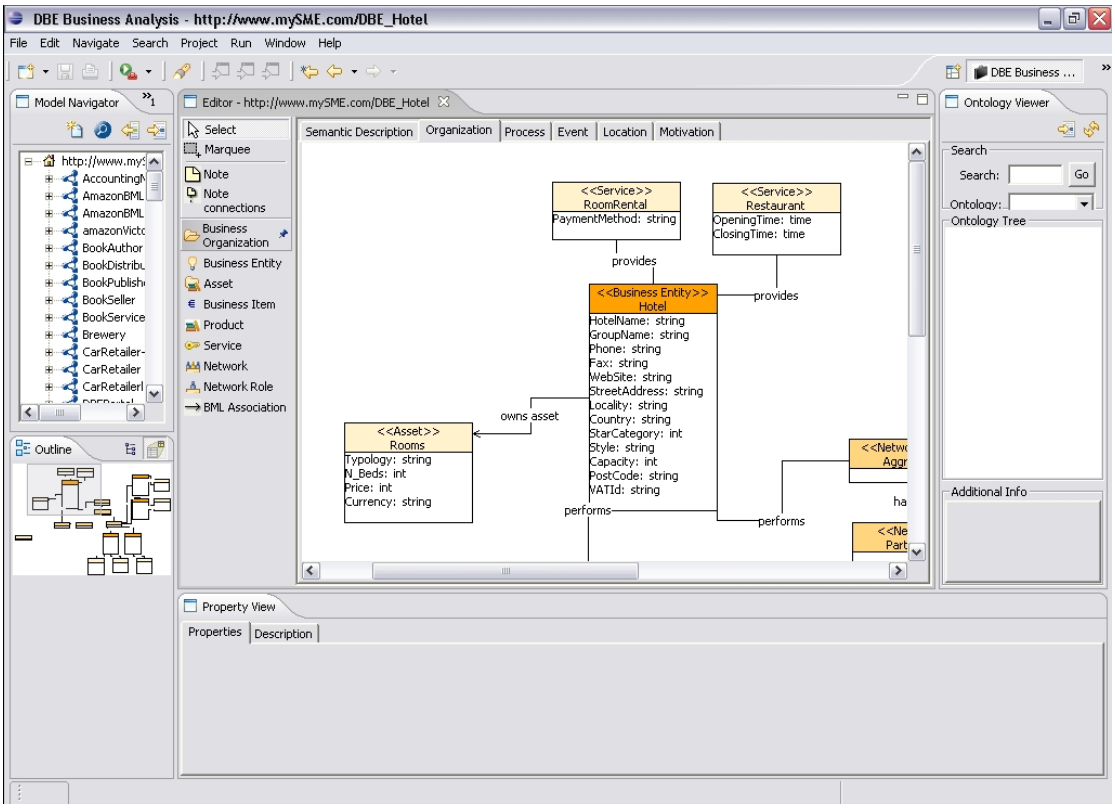
Όταν το QFSDT βρίσκεται σε περιβάλλον service registry, εγγράφει στον εαυτό του μία λειτουργικότητα επίκλησης υπηρεσιών. Όταν ο χρήστης κάνει δεξί κλικ σ'ένα service manifest ή όταν επιλέξει ένα service manifest και καλέσει το pull-down μενού, θα εμφανιστεί μία επιλογή με όνομα "Execute service" η οποία εκτελεί την ενέργεια αυτή, όπως φαίνεται στην εικόνα 44.



Εικόνα 44: Εκτέλεση υπηρεσίας.

Η παραπάνω λειτουργικότητα είναι από τις λειτουργικότητες που παρέχονται από το ίδιο το QFSDT. Άλλη μία λειτουργικότητα για το ίδιο περιβάλλον είναι αυτή στην οποία ο χρήστης επιλέγει ένα μοντέλο και κάνοντας δεξί κλικ ή επιλέγοντας την κατάλληλη ενέργεια από το pull-down μενού, ανοίξει τον wizard δημιουργίας νέου template για το επιλεγμένο μοντέλο.

Ένα παράδειγμα της δυναμικής λειτουργίας της διεπαφής που αναφέρθηκε παραπάνω είναι να βρίσκεται το εργαλείο σε περιβάλλον παραγωγής υπηρεσιών (Service Factory) και ο χρήστης να επιλέξει τις λειτουργίες που προσφέρονται για ένα μοντέλο (είτε κάνοντας δεξί κλικ επάνω σε αυτό ή καλώντας το pull-down μενού της περιοχής επισκόπησης των αποτελεσμάτων). Σε περίπτωση που έχει εγγραφεί στο QFSDT ο BML Editor του DBE Studio, ο χρήστης θα μπορέσει να ανοίξει το μοντέλο με τον BML Editor, όπως φαίνεται στην εικόνα 45.



Εικόνα 45: Άνοιγμα μοντέλου από τον BML Editor που έχει εγγραφεί στο QFSDT.

Α.2.12 Αναζητώντας μοντέλα ή υπηρεσίες χρησιμοποιώντας λέξεις-κλειδιά

Εκτός από τη δημιουργία δομημένων ερωτήσεων για την ανάκτηση μοντέλων και υπηρεσιών, το QFSDT προσφέρει κι έναν τρόπο αναζήτησης με τη χρήση λέξεων-κλειδιών.

Τα αποτελέσματα αυτής της αναζήτησης είναι διαφορετικά ανάλογα με το περιβάλλον στο οποίο βρίσκεται το εργαλείο. Υπάρχουν δύο περιβάλλοντα στα οποία το εργαλείο λειτουργεί και φέρνει ανάλογα αποτελέσματα:

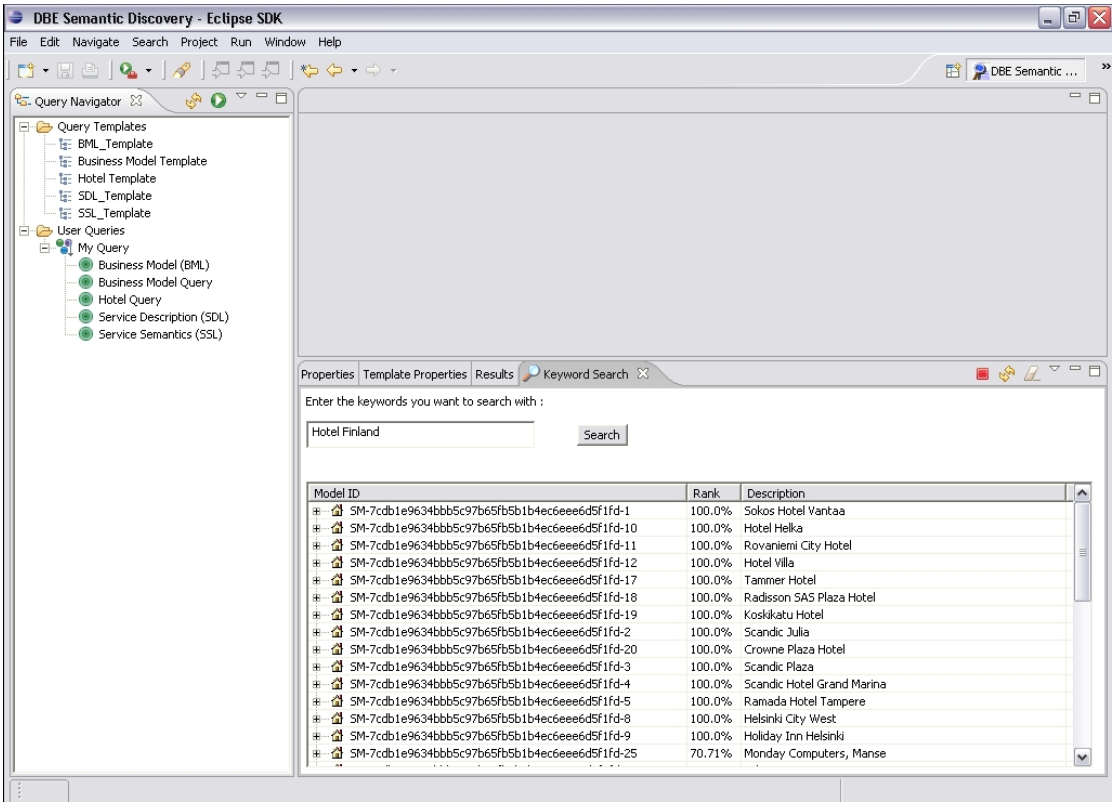
- Περιβάλλον εκτέλεσης υπηρεσιών (Service Execution Environment). Στο περιβάλλον αυτό τα αποτελέσματα που επιστρέφονται είναι service manifests.
- Περιβάλλον παραγωγής υπηρεσιών (Service Factory Environment). Στο περιβάλλον αυτό τα αποτελέσματα που επιστρέφονται είναι μοντέλα.

Τα αποτελέσματα της αναζήτησης με λέξεις-κλειδιά δεν προστίθενται στο Results View όπως γίνεται στην περίπτωση των δομημένων ερωτήσεων. Προστίθενται σε έναν πίνακα που βρίσκεται στο Keyword Search View. Οι λειτουργίες που γίνονται και οι δυνατότητες που έχει ο χρήστης στον πίνακα των

αποτελεσμάτων του Quick Search View είναι οι ίδιες με αυτές του πίνακα του Results View. Η μόνη διαφορά είναι ότι στα αποτελέσματα του πίνακα αποτελεσμάτων του Results View υπάρχουν και τα σύνολα αποτελεσμάτων που έχει δηλώσει ο χρήστης στη διαδικασία δημιουργίας template.

Τέλος, ανάλογα με το περιβάλλον στο οποίο λειτουργεί το εργαλείο αλλάζει το Keyword Search View. Πιο συγκεκριμένα, όταν το εργαλείο βρίσκεται σε περιβάλλον παραγωγής υπηρεσιών (Service Factory) εμφανίζεται ένα σύνολο από κουμπιά από τα οποία ο χρήστης μπορεί να επιλέξει σε ποια κατηγορία μοντέλων θέλει να κάνει αναζήτηση.

Το keyword Search View φαίνεται στην εικόνα 46.



Εικόνα 46: Keyword αναζήτηση.

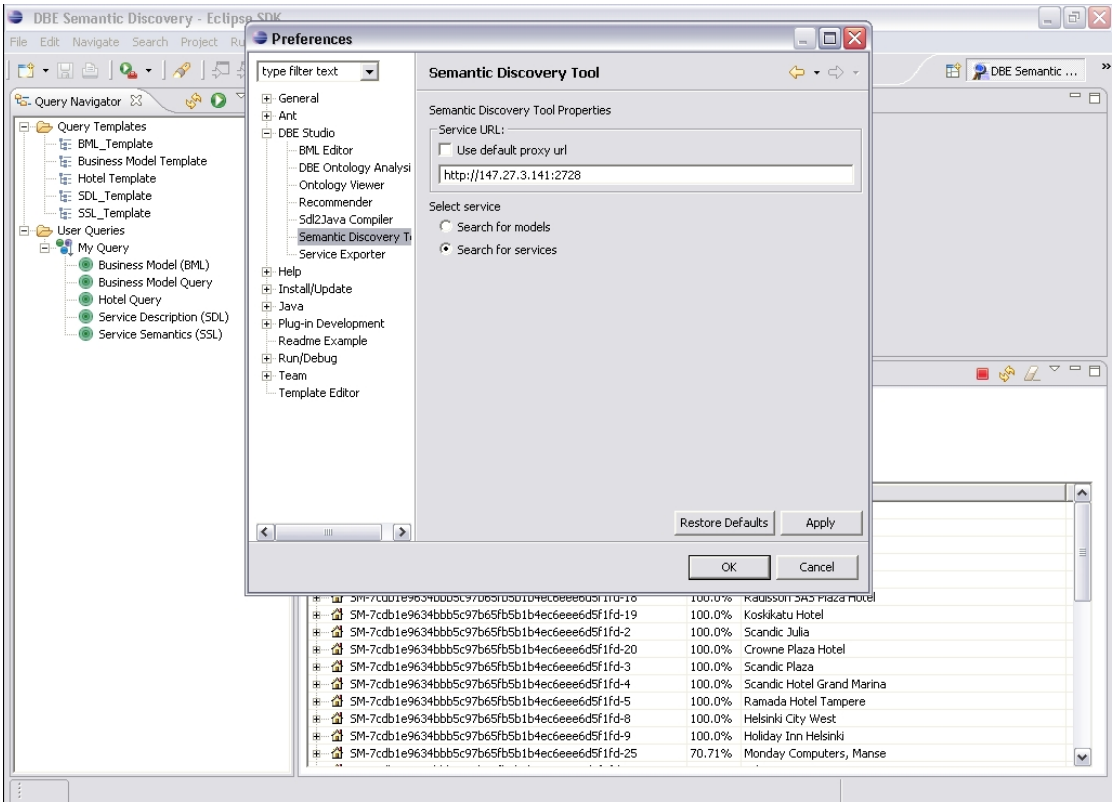
Α.2.13 Δήλωση προτιμήσεων

Το QFSDT προσφέρει λειτουργικότητα η οποία απαιτεί σύνδεση σε έναν servEnt. Η προεπιλογή είναι η σύνδεση με τον servEnt να γίνεται τοπικά στο port 2728 και αυτή η παράμετρος αλλάζει αν κάποιος επιθυμεί να συνδεθεί σε άλλον

servEnt. Επίσης μπορεί να επιλεχθεί η default τιμή που υπάρχει για το DBE Studio.Το περιβάλλον στο οποίο τρέχει το εργαλείο μπορεί να τροποποιηθεί επιλέγοντας:

- Search for models (Service Factory Environment)
- Search for services (Service Execution environment)

Αυτές οι τιμές αλλάζουν στη σελίδα των προτιμήσεων που φαίνεται στην εικόνα 47 και είναι προσβάσιμη επιλέγοντας Window -> Preferences ->DBE Studio->Semantic Discovery Tool από το κύριο μενού του Eclipse.



Εικόνα 47: Δήλωση προτιμήσεων.

Παράρτημα Β

Οδηγός χρήσης Portal

Β.1 DBE Service Discovery Portal

Το Portal αποτελείται από τρεις σελίδες:

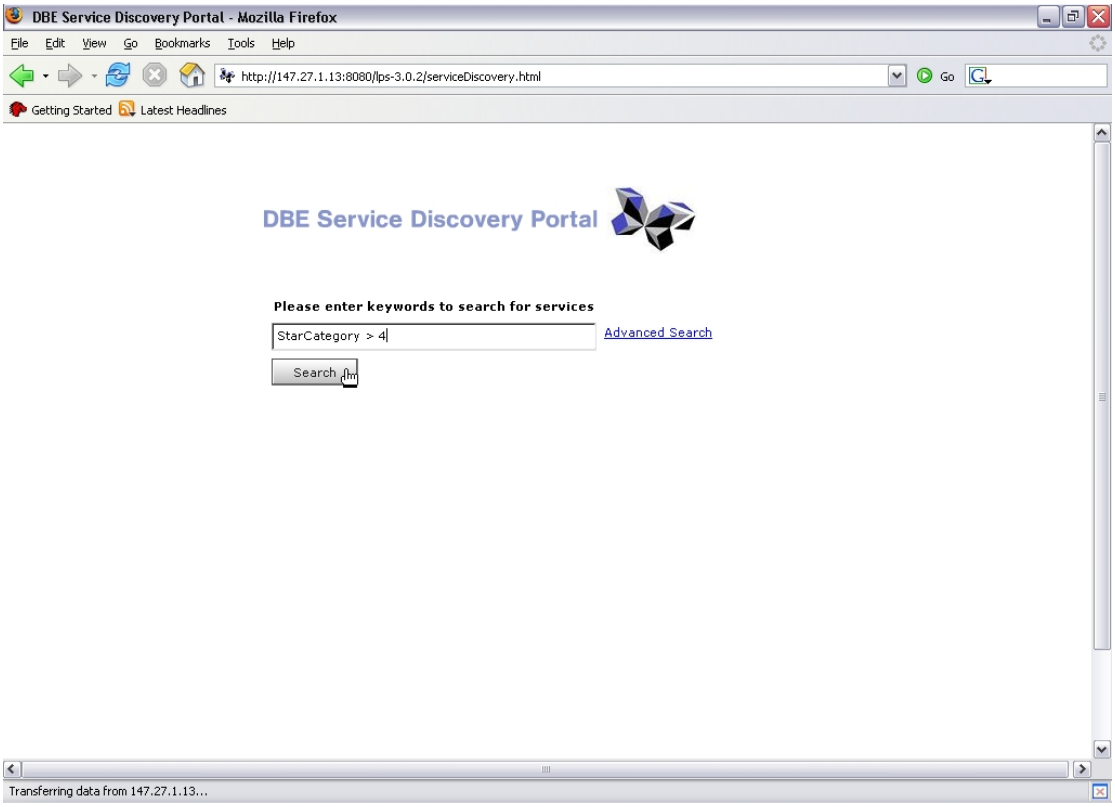
- Την αρχική σελίδα, την οποία βλέπει ο χρήστης όταν συνδέεται στο site που βρίσκεται η εφαρμογή
- Τη σελίδα της προχωρημένης αναζήτησης, στην οποία ο χρήστης μπορεί να δημιουργήσει δομημένες ερωτήσεις.
- Τη σελίδα των αποτελεσμάτων, στην οποία ο χρήστης μπορεί να κάνει επισκόπηση των αποτελεσμάτων.

Σε κάθε σελίδα (εκτός της αρχικής) το λογότυπο είναι ενεργός σύνδεσμος προς την αρχική σελίδα.

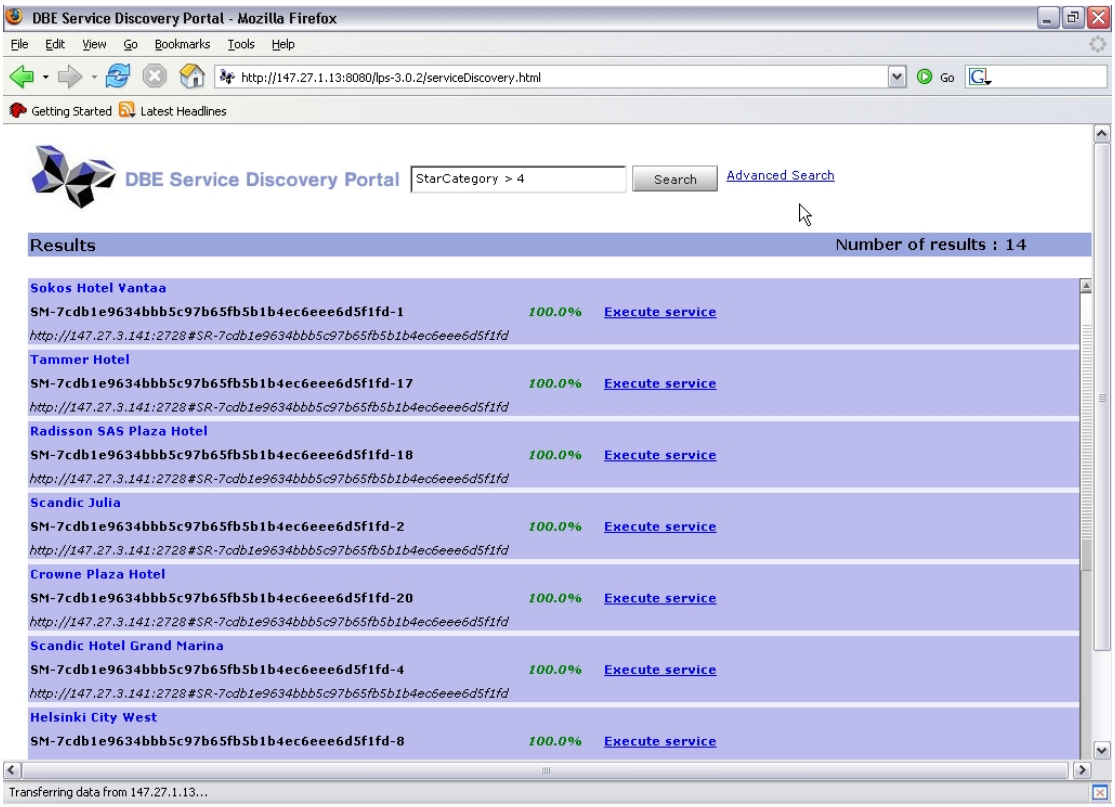
Β.1.2 Χρησιμοποιώντας λέξεις-κλειδιά για αναζήτηση υπηρεσιών

Τόσο στην αρχική σελίδα όσο και στη σελίδα των αποτελεσμάτων υπάρχει ένα πεδίο στο οποίο ο χρήστης έχει τη δυνατότητα εισαγωγής λέξεων-κλειδιών, όπως φαίνεται στην εικόνα 48. Αν το σύστημα εντοπίσει υπηρεσίες που να περιέχουν στις πληροφορίες τους κάποιες από τις λέξεις που εισήγαγε ο χρήστης τότε θα προστεθούν

αυτές οι υπηρεσίες στα αποτελέσματα. Αυτές θα είναι ταξινομημένες κατά το βαθμό σχετικότητας των λέξεων-κλειδιών με τις πληροφορίες που εντοπίστηκαν στις πληροφορίες των υπηρεσιών, όπως φαίνεται στην εικόνα 49.



Εικόνα 48: Keyword αναζήτηση.



Εικόνα 49: Αποτελέσματα αναζήτησης.

B.1.3 Δημιουργώντας δομημένες ερωτήσεις για αναζήτηση υπηρεσιών

Αν ο χρήστης επιθυμεί να αναζητήσει συγκεκριμένες πληροφορίες για κάποια υπηρεσία, τότε θα πρέπει να δημιουργήσει δομημένες ερωτήσεις. Καταρχάς θα πρέπει να μεταβεί στη σελίδα της προχωρημένης αναζήτησης είτε από την αρχική σελίδα ή από τη σελίδα των αποτελεσμάτων κάνοντας κλικ στο σύνδεσμο “Advanced Search”.

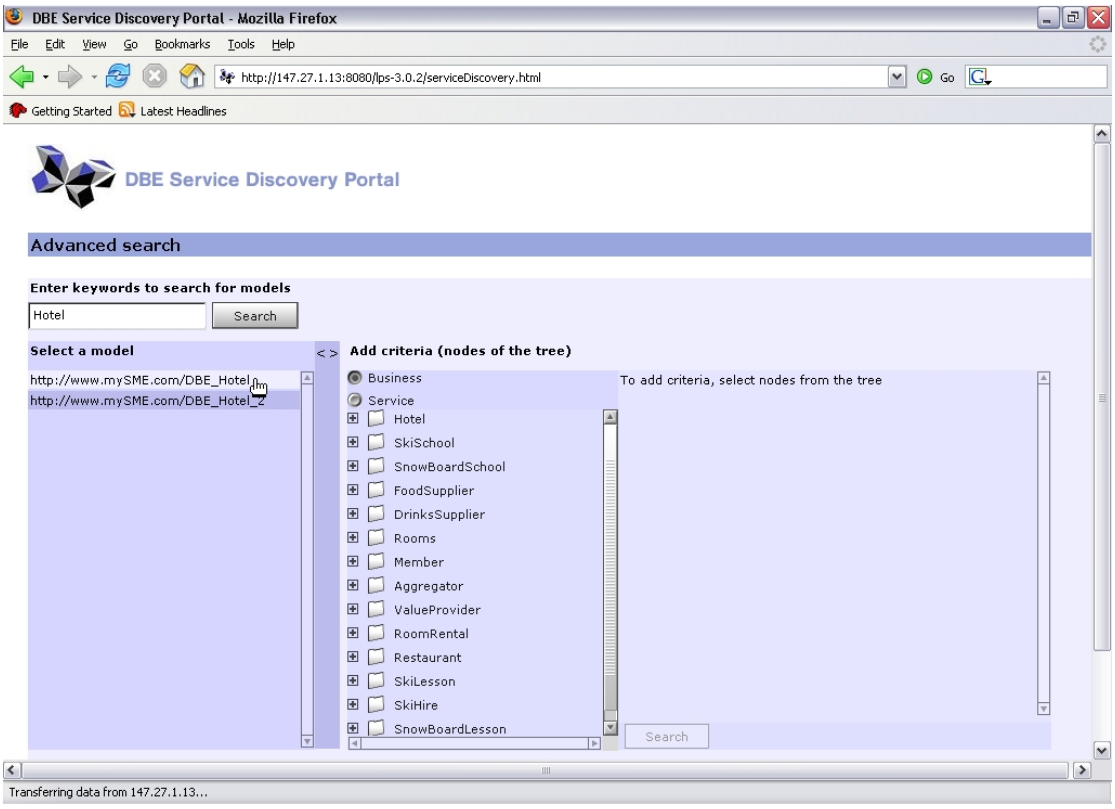
Θα εμφανιστεί η σελίδα της προχωρημένης αναζήτησης στης οποίας το πάνω μέρος είναι το λογότυπο του προγράμματος DBE. Στα αριστερά φαίνεται ένα πεδίο στο οποίο ο χρήστης εισάγει λέξεις-κλειδιά για την αναζήτηση μοντέλων και κάτω από το πεδίο υπάρχει μία λίστα στην οποία προστίθενται τα μοντέλα που ικανοποιούν την προηγούμενη αναζήτηση. Στο κέντρο εμφανίζεται το μοντέλο πάνω στο οποίο ο χρήστης διάλεξε να δημιουργήσει μία δομημένη ερώτηση. Στα αριστερά φαίνονται τα κριτήρια τα οποία δημιούργησε ο χρήστης για τη δομημένη ερώτηση.

Όπως αναφέραμε στη σελίδα ο χρήστης εισάγει λέξεις-κλειδιά ώστε να ανακτήσει κάποια μοντέλα που να περιέχουν στις πληροφορίες τους κάποιες από αυτές τις λέξεις. Μετά ο χρήστης επιλέγει κάποιο από τα μοντέλα που ανακτήθηκαν

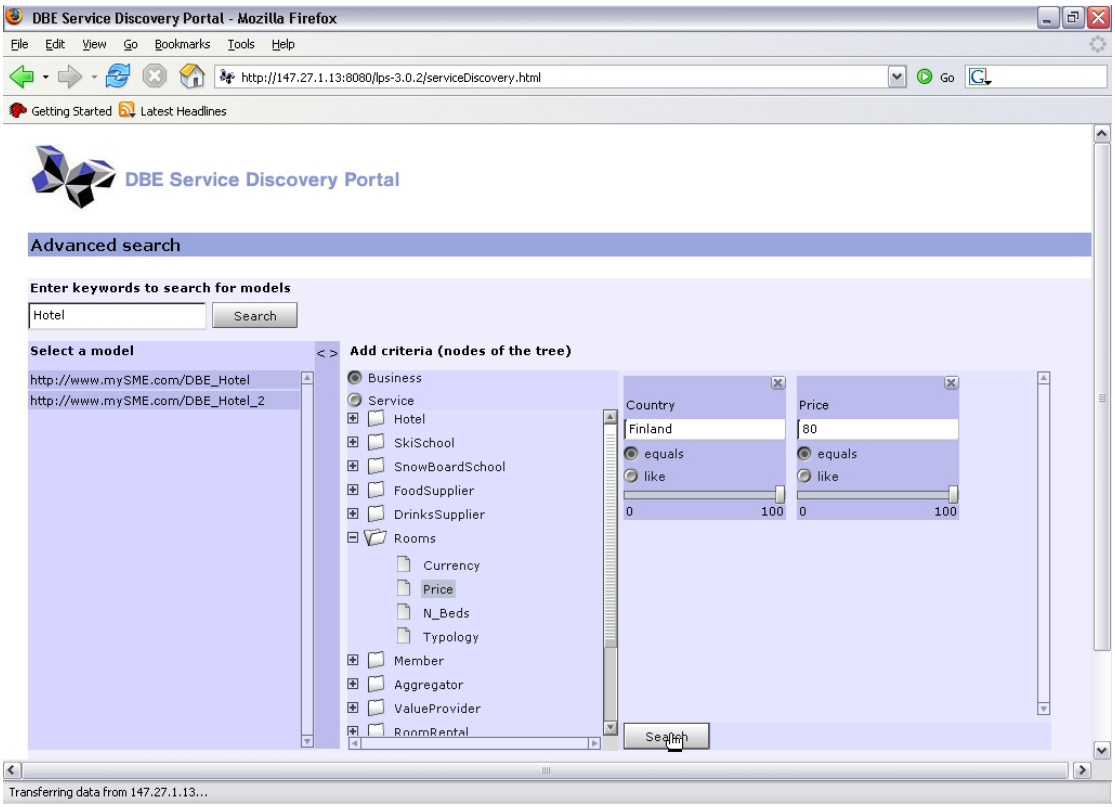
και στο κέντρο της οθόνης εμφανίζονται οι πληροφορίες του μοντέλου σε δεντρική αναπαράσταση, όπως φαίνεται στην εικόνα 50. Ο χρήστης μπορεί να επιλέξει οποιοδήποτε από τα στοιχεία του δέντρου:

- Αν επιλέξει κόμβο ο οποίος δεν είναι φύλλο του δέντρου, τότε θα πλοηγηθεί μέσα στις πληροφορίες του στοιχείου που επέλεξε.
- Αν ο κόμβος είναι φύλλο, τότε στα δεξιά της οθόνης θα εμφανιστεί ένα κουτί που αναπαριστά ένα κριτήριο.

Μετά την επιλογή των κριτηρίων που επιθυμεί για την ερώτηση, μπορεί να δώσει τιμή, πράξη και βάρος σημασίας για κάθε ένα κριτήριο και μετά μπορεί να ξεκινήσει την αναζήτηση με βάση τη δομημένη ερώτηση πιέζοντας το κουμπί "Search", όπως φαίνεται στην εικόνα 51.



Εικόνα 50: Επιλογή μοντέλου για άνοιγμα.



Εικόνα 51: Ανάθεση τιμών στα κριτήρια.

B.1.4 Επισκοπώντας τις πληροφορίες των αποτελεσμάτων

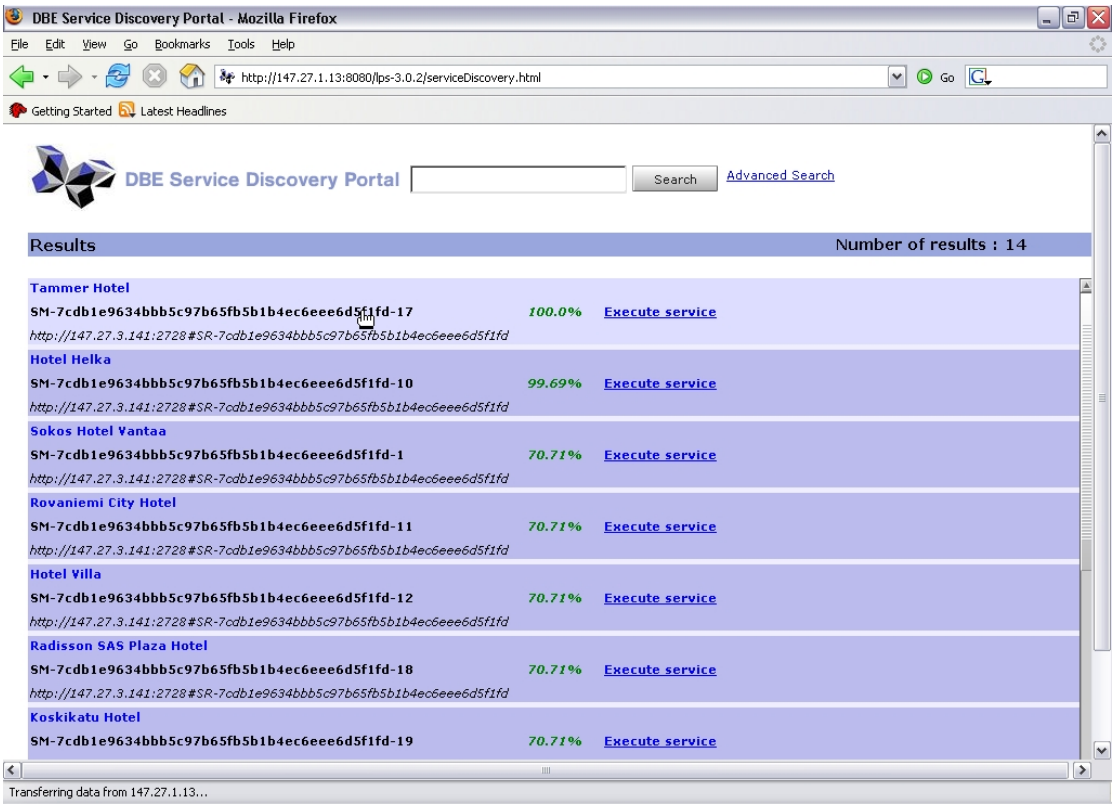
Είτε γίνει αναζήτηση με λέξεις-κλειδιά ή με τη δημιουργία δομημένης ερώτησης, τα αποτελέσματα έχουν την ίδια μορφή και εμφανίζονται στη σελίδα των αποτελεσμάτων. Κάθε αποτέλεσμα έχει τις εξής πληροφορίες :

- Μια περιγραφή της υπηρεσίας
- Το service manifest id της υπηρεσίας
- Το id του κόμβου από τον οποίο ανακτήθηκε η υπηρεσία
- Ένας σύνδεσμος για την εκτέλεση της υπηρεσίας

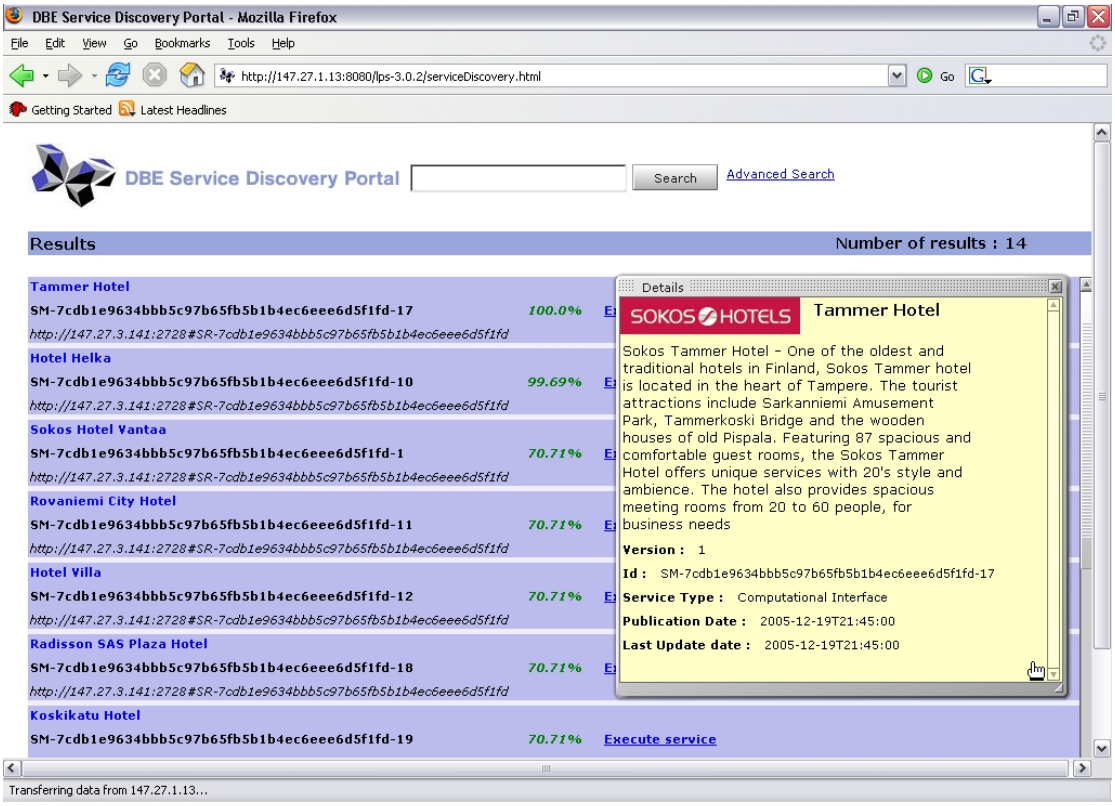
Οι πληροφορίες αυτές φαίνονται στην εικόνα 52.

Για περισσότερες πληροφορίες ο χρήστης μπορεί να κάνει κλικ επάνω στην περιοχή που αντιπροσωπεύει ένα αποτέλεσμα. Ένα παράθυρο θα εμφανιστεί με πληροφορίες που αφορούν στην υπηρεσία, όπως φαίνεται στην εικόνα 53.

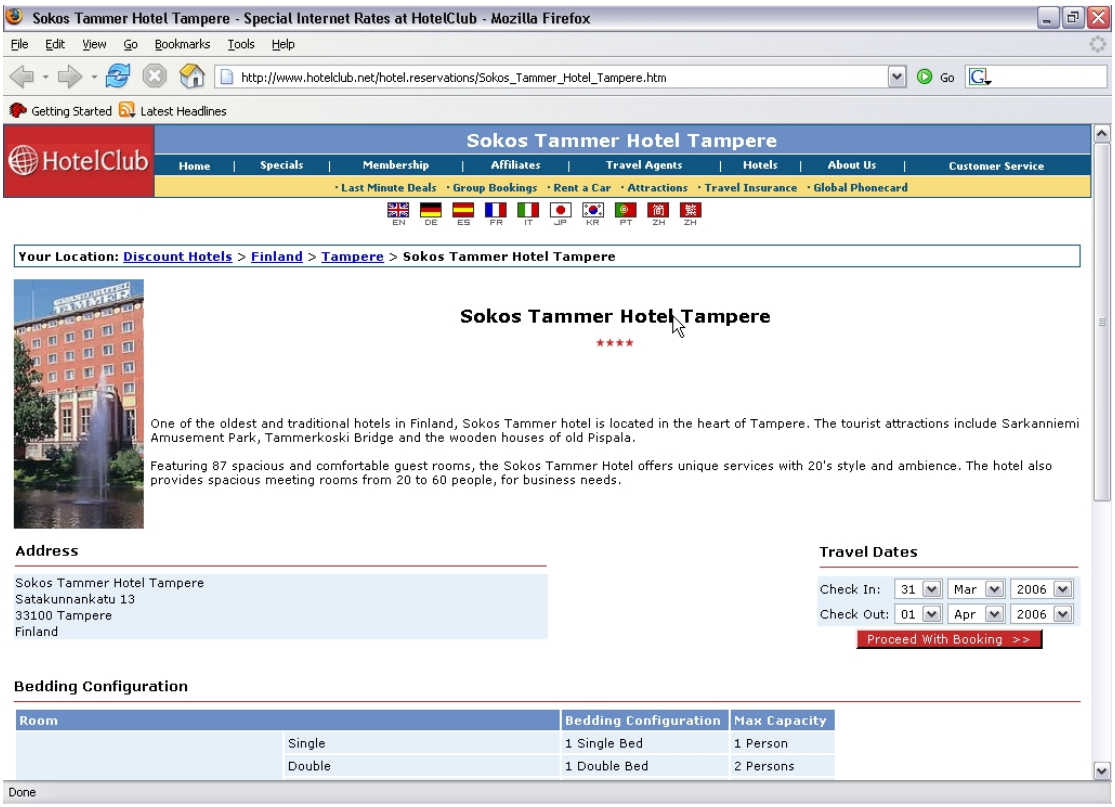
Αν ο χρήστης επιλέξει να εκτελέσει μια υπηρεσία επιλέγοντας τον σύνδεσμο “Execute Service”, τότε (εφόσον είναι διαθέσιμη η υπηρεσία) θα ανοίξει ένας νέος Web Browser στον οποίο θα μπορεί ο χρήστης να αλληλεπιδράσει με αυτήν. Ένα τέτοιο παράδειγμα φαίνεται στην εικόνα 54.



Εικόνα 52: Επισκόπηση αποτελεσμάτων.



Εικόνα 53: Λεπτομέρειες υπηρεσίας-αποτελέσματος.



Εικόνα 54: Αλληλεπίδραση με υπηρεσία.

Βιβλιογραφία

[1] DBE (Digital Business Ecosystem)
http://www.digital-ecosystem.org

[2] DBE Studio
http://dbestudio.sourceforge.net

[3] SourceForge DBE project
http://sourceforge.net/projects/dbestudio

[4]] Object Management Group – OMG, "*MDA (Model Driven Architecture)*", 2002
http://www.omg.org/mda

[5] Trygve Reenskaug "*MODELS - VIEWS - CONTROLLERS*" (MVC), 1979
http://heim.ifi.uio.no/~trygver/1979/mvc-2/1979-12-MVC.pdf

[6] Object Management Group – OMG, "*MetaObject (MOF) Facility Specification*", 2002
http://www.omg.org/mof

[7] TUC, DBE Deliverable, D17.1 – "Recommender": https://dbe.digital-ecosystem.net/ servlets/ProjectDocumentList?folderID =16& expandFolder=16&folderID=0, March 2005

[8] M. M. Zloof. "*Query by example: A database language*" , 1977
http://www.research.ibm.com/journal/sj/164/ibmsj1604C.pdf

[9] NetBeans "*MetaData Repository - MDR*" , 2003
http://mdr.netbeans.org

[10] Object Management Group – OMG, Inc.250 First Ave. Suite 100 Needham, MA 02494, U.S.A.
http://www.omg.org/

[11] Object Management Group – OMG "*XML MetaData Interchange - XMI*" , 2001
http://www.omg.org/technology/documents/formal/xmi.htm

[12] Sun Developer Network "*Java MetaData Interface – JMI*" , 2002
http://java.sun.com/products/jmi

[13] DBE Knowledge Representation Models TUC, DBE Deliverable, D14.1
https://dbe.digital-ecosystem.net/servlets/ProjectDocumentList ?folderID=16&expandFolder=16&folder ID=0, May 2005

[14] Object Management Group – OMG "Object Constraint Language 2.0 – OCL 2.0" , 2005
http://www.omg.org/docs/ptc/05-06-06.pdf

[15] IBM Corporation, "*Eclipse Platform Technical Overview*", 2003
http://www.eclipse.org

[16] OpenLaszlo
http://www.openlaszlo.org

[17] Extensible Markup Language (XML) 1.0 (Third Edition), 2004
http://www.w3.org/XML

[18] Brendan Eich, C. Rand Mckinney "*JavaScript*" , 1996
http://hepunx.rl.ac.uk/~adye/jsspec11/jsrefspe.htm

[19] Macromedia Flash
http://www.macromedia.com/software/flash/flashpro

[20] Leonidas Fegaras "*Voodoo: a Visual Object-Oriented Database language for ODMG OQL*" , 1999

[21] Object Data Management Group (ODMG) - OQL (Object Query Language)
http://www.odmg.org

[22] IBM Rational ClearQuest Client for Eclipse
http://www-128.ibm.com/developerworks/rational/library/04/r-3089/

[23] Microsoft Access
http://office.microsoft.com/en-us/FX010857911033.aspx

[24] XQbE (XQuery by Example)
DANIELE BRAGA, ALESSANDRO CAMPI, and STEFANO CERI, Politecnico di Milano "XQBE (XQuery By Example): A Visual Interface to the Standard XML Query Language" , 2005

[25] Alistair Cockburn "Writing Effective Use Cases"

[26] Deborah J. Mayhew "Principles and Guidelines in Software User Interface Design"

[27] NetBeans Organization
http://mdr.netbeans.org

[28] OWL (Ontology Web Language)
http://www.w3.org/TR/owl-features/