ΠΑΝΕΠΙΣΤΗΜΙΟ ΣΤΕΡΕΑΣ ΕΛΛΑΔΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗΝ ΒΙΟΙΑΤΡΙΚΗ

Εξέλιξη Ιατρικών Οντολογιων

Κουτονιάς Παναγιώτης

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υπεύθυνος
Βαρλάμης Ηρακλής

ΠΑΝΕΠΙΣΤΗΜΙΟ ΣΤΕΡΕΑΣ ΕΛΛΑΔΟΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΪΑΤΡΙΚΗ

ΤΙΤΛΟΣ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ: Εξέλιξη ιατρικών οντολογιών

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΗΡΑΚΛΗΣ ΒΑΡΛΑΜΗΣ

ΟΝΟΜΑ ΦΟΙΤΗΤΗ: ΠΑΝΑΓΙΩΤΗΣ ΚΟΥΤΟΝΙΑΣ Α.Μ.29

Λαμία, Μάρτιος 2009

Περιεχόμενα

Εικόνες

Περίληψη

Η παρούσα πτυχιακή προσπαθεί να αξιοποιήσει την επιστήμη της πληροφορικής για να συγκεντρώσει και να οργανώσει γνώσεις από την επιστήμη της βιοιατρικής. Κύριος στόχος της πτυχιακής είναι να κατανοήσει τη λειτουργία και τη δομή των οντολογιών και να προτείνει μια εφαρμογή που θα επεκτείνει ημι-αυτόματα μία οντολογία.

Η εκρηκτική ανάπτυξη που έχουν γνωρίσει τα τελευταία χρόνια τα πληροφοριακά συστήματα και η ανάλογη διόγκωση της αποθηκευμένης πληροφορίας, έχουν δημιουργήσει τα τελευταία χρόνια ένα σημαντικό πρόβλημα στην ανάκτηση των δεδομένων και κυρίως στην επαναχρησιμοποίησή τους από διαφορετικές εφαρμογές με αποτέλεσμα όλη η αποθηκευμένη γνώση να μην είναι πρακτικά προσβάσιμη.

Η σημασιολογία έρχεται να καλύψει ένα σημαντικό κενό σε ένα πληροφοριακό σύστημα προσδίδοντας σημασιολογικό περιεχόμενο κυρίως σε δεδομένα που είναι αποθηκευμένα σε φυσική γλώσσα ενώ οι οντολογίες είναι το εργαλείο που κάνει δυνατή τη μοντελοποίηση των σχέσεων ανάμεσα στους διάφορους τύπους δεδομένων.

Στα πλαίσια αυτής της εργασίας μελετώνται οι βασικές αρχές της σημασιολογίας και της δόμησης οντολογιών και εφαρμόζονται στο πεδίο της ιατρικής. Με τις γνώσεις που απέκτησα από την ανάλυση της θεωρίας των οντολογιών και με παράληλλη εξοικείωση με την wikipedia, εγινε εφικτή η σχεδίαση και η ανάπτυξη της εφαρμογής.

Λέξεις Κλειδιά: Σημασιολογικός Ιστός, μεταδεδομένα, οντολογία, γλώσσες αναπαράστασης οντολογιών, εξέλιξη και εργαλεία ανάπτυξης οντολογιών, ελληνικό και αγγλικό Wikipedia, java

Ευχαριστίες

Υπεύθυνος καθηγητής στην εκπόνηση της πτυχιακής είναι ο κ. Η. Βαρλάμης στον οποίο οφείλω ιδιαίτερες ευχαριστίες για την ανάθεση αυτής, και για την δυνατότητα που μου δόθηκε μέσω της συγκεκριμένης διπλωματικής να αποκτήσω πολύτιμες εμπειρίες. Επιπλέον, θα ήθελα να τον ευχαριστήσω για την υπομονή, την υποστήριξη, την καθοδήγηση που μου παρείχε και την διαρκή ενθάρρυνση και την εμπιστοσύνη που επέδειξε στην προσπάθεια μου.

Τέλος, θέλω να ευχαριστήσω την οικογένειά μου για την συμπαράσταση και εμπιστοσύνη που μου έδειξε και συνεχίζει να μου δείχνει όλα αυτά τα χρονιά.

1 Εισαγωγή

1.1 Αντικείμενο της πτυχιακής

Στο συγκεκριμένο σημείο κρίνεται σκόπιμο να παρουσιαστεί η δομή της παρούσας διπλωματικής εργασίας, ώστε να γίνει πιο εύκολη η περαιτέρω ανάλυση που θα ακολουθήσει.

Η συγκεκριμένη εργασία αποτελείται από δύο στάδια. Στο πρώτο στάδιο θα πρέπει να γίνει κατανοητή η θεωρία πίσω από τις οντολογίες και στο δεύτερο στάδιο θα υλοποιηθεί ένα παράδειγμα πάνω στην εξέλιξη μιας οντολογίας.

Αναφορικά με το **πρώτο μέρος** πρέπει να αναφερθεί ότι ολοκληρώνεται με τη συγγραφή ενός κειμένου που συνοψίζει:

- τι είναι οντολογία,
- ποιες οι βασικές έννοιές των οντολογιών,
- ποιες γλώσσες και ποια εργαλεία χρησιμοποιούνται για την αναπαράσταση και την οργάνωση των οντολογιών,
- ποια τα προβλήματα του χώρου (εξέλιξη οντολογιών, σύνθεση οντολογιών, ευθυγράμμιση οντολογιών κ.ά.).

Επίσης εστιάζει στο θέμα της εξέλιξης των οντολογιών (ontology evolution) και τα προβλήματα που προκύπτουν καθώς επίσης εξηγεί τι σημαίνει εξέλιξη, που εμφανίζεται και ποιες μορφές και ποιες μεθοδολογίες χρησιμοποιούνται για να απεικονίσουν την εξέλιξη.

Στο **δεύτερο μέρος** της εργασίας όπως προαναφέρθηκε θα υλοποιηθεί ένα παράδειγμα πάνω στην εξέλιξη μιας οντολογίας. Τα βήματα στο συγκεκριμένο κομμάτι που ακολουθεί είναι τα εξής:

- χρησιμοποιείται μία οντολογία βιοπληροφορικής (σε XML μορφή)
- δημιουργείται μια εφαρμογή που θα την επεκτείνει ημι-αυτόματα ως εξής: για κάθε όρο της οντολογίας θα ψάχνει τη Wikipedia, αν βρει σχετική σελίδα, θα εξάγει τις υπερσυνδέσεις της σελίδας αυτής. Οι συνδεδεμένοι όροι θα αναζητούνται στην οντολογία:

  α) αν υπάρχουν θα προτείνεται στο χρήστη να γίνει μια σύνδεση,

β) αν δεν υπάρχουν θα προτείνονται στο χρήστη να τους προσθέσει στην οντολογία.

1.2 Οργάνωση Πτυχιακής

Το **πρώτο μέρος** αποτελείται από τις εξής ενότητες:

- Ανάπτυξη Σημασιολογικού Ιστού
- Τι είναι οντολογία
- Μορφή οντολογίας
- Κατηγορίες οντολογιών
- Βασικές έννοιες οντολογιών
- Πως χτίζεται μία οντολογία
- Γλώσσες αναπαράστασης οντολογίας
- Εργαλεία αναπαράστασης και οργάνωσης οντολογιών
- Κατασκευή οντολογίας
- Η έννοια εξέλιξης οντολογιών
- Εργαλεία δημιουργίας οντολογιών
- Ontology editors

Ενώ το **δεύτερο μέρος** αποτελείται από:

- Παρουσίαση του λογισμικού που χρησιμοποιήθηκε
- Προετοιμασία περιβάλλοντος
- Δημιουργία νέου project
- Ανάλυση της θεωρίας του project

2 Παγκόσμιος Ιστός

2.1 Ο παγκόσμιος ιστός σήμερα

Ο Παγκόσμιος Ιστός είναι ένα σύνολο από ιστοσελίδες και διασυνδέσεις και ένας τρόπος αναπαράστασης της πληροφορίας όπου η αναπαράσταση και ο σχεδιασμός γίνεται από τους developers. Οι χρήστες πρέπει να ερμηνεύσουν την πληροφορία ενώ οι υπολογιστές είναι υπεύθυνοι μόνο για την παρουσίαση και δεν μπορούν να επεξεργαστούν την πληροφορία.

Στα τελευταία χρόνια ο Ιστός αντιμετώπισε δύο επαναστατικές αλλαγές που στοχεύουν στο να το μετασχηματίσουν από μια στατική συλλογή εγγράφων σε έναν ευφυή και δυναμικό περιβάλλον ολοκλήρωσης στοιχείων. Κατ'αρχάς, η τεχνολογία της υπηρεσίας ιστού επέτρεψε ομοιόμορφη πρόσβαση υπηρεσιών μέσω του πρότυπου ιστού σε τμήματα λογισμικού που ανήκουν σε διαφορετικές πλατφόρμες και γράφονται σε διαφορετικές γλώσσες προγραμματισμού. Κατά συνέπεια, τμήματα λογισμικού παρέχουν ποικίλες λειτουργίες που κυμαίνονται από την μετατροπή του νομίσματος στην κράτηση πτήσης ή το βιβλίο που αγοράζει και που είναι πλέον προσιτές μέσω ενός συνόλου προτύπων και πρωτοκόλλων Ιστού. Φυσικά, η πραγματική αξία των υπηρεσιών Ιστού είναι στη σύνθεση τους η οποία τους επιτρέπει τη δημιουργία νέων και σύνθετων λειτουργιών από τις υπάρχουσες υπηρεσίες.

2.2 Σημασιολογικός Ιστός

«Το πρώτο βήμα είναι να δομηθούν τα δεδομένα στο Διαδίκτυο σε μια μορφή που οι μηχανές να μπορούν φυσικά να τα καταλάβουν ή να τα μετατρέψουν σε αυτήν. Αυτή είναι η απαρχή του τι αποκαλώ Semantic Web, ένα δίκτυο δεδομένων που μπορούν να επεξεργαστούν άμεσα ή έμμεσα οι μηχανές»

Tim Berners Lee, "Weaving the Web", (Berners Lee, Fischetti 2001)

Το Semantic Web (Berners Lee et al 2000) προτείνει μία νέα μορφή στο περιεχόμενο του Web του οποίου η σημασία θα είναι κατανοητή στους υπολογιστές . Θα οδηγήσει σε μία επανάσταση νέων τεχνολογιών και δυνατοτήτων για τους υπολογιστές. Η υλοποίηση αυτού του οράματος είναι πολύ δύσκολη προς το παρόν. Η αρχή γίνεται

προσθέτοντας σημασιολογικό σχολιασμό (semantic annotation) στις πηγές περιεχομένου.

Το Διαδίκτυο και ο Παγκόσμιος Ιστός ήταν μια επανάσταση για την τεχνολογία της πληροφόρησης αλλά και για την καθημερινή ζωή του ανθρώπου. Πρόκειται για μια προσπάθεια που κατόρθωσε να ξεπεράσει τους γεωγραφικούς φραγμούς και να συνδέσει όλο τον πλανήτη σε ένα κοινό δίκτυο διακίνησης της πληροφορίας. Είναι προφανές ότι αυτή η πληροφορία, αφού παράγεται και δημοσιεύεται πρακτικά από οποιονδήποτε, δεν έχει μία ενιαία μορφή.

Επίσης δεν υπάρχει ένας ενιαίος τρόπος οργάνωσης της ώστε να μπορεί ο καθένας να την ανακτά. Σημαντικό ρόλο σε αυτό έπαιξε το γεγονός ότι οι πιο σύγχρονες μορφές περιεχομένου του ιστού έχουν διαμορφωθεί έτσι ώστε να είναι κατανοητές μόνο από τους ανθρώπους. Αυτό έχει ως αποτέλεσμα να είναι ελάχιστα κατανοητές από τους Η/Υ.

Ο σημασιολογικός Ιστός στοχεύει να παράσχει γλώσσες έκφρασης ικανές να περιγράψουν την πληροφορία σε μορφές κατανοητές από μηχανές. Αυτό θα προσφέρει την δυνατότητα για μια καλύτερη οργάνωση και ταξινόμηση της πληροφορίας ώστε πλέον, με αυτοματοποιημένο τρόπο μέσα από το ίδιο το σύστημα, να μπορεί να ανακτάται με μεγαλύτερη ακρίβεια και ποιότητα αποτελέσματος, αφού θα υπάρχει 'συνεννόηση' ανθρώπου-μηχανής. Ο σημασιολογικός ιστός είναι μια ερευνητική προσπάθεια για την καλύτερη οργάνωση του περιεχομένου στον ιστό. Αυτό θα γίνει με την χρήση και καλύτερη αξιοποίηση των ήδη υπαρχόντων εργαλείων και δυνατοτήτων, αλλά και με την ανάπτυξη νέων. Το αρχικό όραμά του περιλάμβανε σαφώς κάτι περισσότερο από την ανάκτηση HTML σελίδων από τους υπολογιστές που αποτελούν το Διαδίκτυο.

Η κύρια γλώσσα έκφρασης της δομής της πληροφορίας στον ιστό είναι η XML (eXtensible Markup Language). Όμως δεν καταφέρνει να αποτελέσει μια απόλυτη λύση για την περιγραφή της σημασίας της πληροφορίας στο σημασιολογικό ιστό. Αυτό που καταφέρνει να κάνει είναι να παρουσιάσει κάποιες σημασιολογικές ιδιότητες μέσα από τη συντακτική δομή της. Αυτό εξαρτάται όμως από τον ορισμό κειμένου που δίνει κάποιο DTD (Document Type Definition) για το συγκεκριμένο αρχείο. Επομένως, παρόλο που μπορεί κανείς να εξάγει κάποιες σημασιολογικές έννοιες από τις ετικέτες XML, η σημασιολογική έννοια κάθε ετικέτας δεν ορίζεται έξω από ένα DTD.

Για την ανάπτυξη λοιπόν ενός σημασιολογικού ιστού είναι απαραίτητο οι πόροι στον ιστό να αναπαρίστανται ή να σχολιάζονται με δομημένες περιγραφές του περιεχομένου και των σχέσεων τους, κατανοητές από Η/Υ. Αυτό είναι εφικτό με την χρήση λεξιλογίων και κατασκευασμάτων που δηλώνονται ρητά σε μια οντολογία περιοχής.

Ένας σημαντικός περιορισμός της τεχνολογίας υπηρεσιών Ιστού είναι ότι η εύρεση και η σύνθεση των υπηρεσιών απαιτούν ακόμα χειρωνακτική προσπάθεια. Αυτό επιφέρει ένα σοβαρό φορτίο με τον αυξανόμενο αριθμό υπηρεσιών Ιστού. Για να εξετάσουν αυτό το πρόβλημα, σημασιολογικοί ερευνητές Ιστού προώθησαν την ιδέα της αύξησης των υπηρεσιών Ιστού με μια σημασιολογική περιγραφή της λειτουργίας τους που θα μπορούσε να διευκολύνει την ανακάλυψη και ολοκλήρωση. Πιο συγκεκριμένα, υπηρεσίες Ιστού περιγράφονται από έννοιες που παρέχονται από μία οντολογία περιοχών. Η οικοδόμηση αυτών των περιγραφών είναι ένας προκλητικός στόχος.

2.3 Απαιτήσεις για το σημασιολογικό Ιστό

Ο Tim Berners Lee έχει ένα δισυπόστατο όραμα για το μέλλον του Διαδικτύου. Το πρώτο μέρος είναι να γίνει το Διαδίκτυο ένα πιο «συνεργάσιμο» μέσο. Το δεύτερο μέρος είναι να κατασταθεί το Διαδίκτυο κατανοητό, και έτσι επεξεργάσιμο, από τις μηχανές. Δυστυχώς, τέτοιες σχέσεις μεταξύ των πόρων του Διαδικτύου δεν συμπεριλαμβάνονται στην παρούσα δομή του Ιστού. Η τεχνολογία που δύναται να συλλάβει τέτοιες σχέσεις καλείται Resource Description Framework (RDF). Το βασικό σημείο που πρέπει να κατανοηθεί είναι ότι το αρχικό όραμα για το Διαδίκτυο κάλυπτε και πρόσθετα μετα-στοιχεία (meta-data) πέρα και πίσω από αυτό που θεωρούμε σήμερα το World Wide Web. Αυτά τα πρόσθετα μετα-στοιχεία απαιτείται οι μηχανές να είναι σε θέση να επεξεργαστούν, προκειμένου να λειτουργούν σε ένα επίπεδο σημασιολογικής και νοήμονος λειτουργίας στο Διαδίκτυο.

Είναι φανερό ότι στον σημασιολογικό ιστό τον πιο σημαντικό ρόλο θα τον παίζουν οι οντολογίες. Πρόκληση αποτελεί η ανάπτυξη στον τομέα των οντολογιών, γλωσσών παρουσίασης οντολογιών, μεθόδων εκμάθησης οντολογιών και συστημάτων οντολογιών για βιβλιοθήκες, που θα διαχειρίζονται, θα υιοθετούν και θα προτυποποιούν οντολογίες. Επίσης προσθήκη σημασιολογικών markup σχολιασμών

στις web pages. Η σημασιολογία δίνεται χρησιμοποιώντας οντολογίες. Μπορούμε τώρα να συνθέσουμε έναν νέο ορισμό του Semantic Web: ένας επεξεργαζόμενος από μηχανές Ιστός (machine-processable Web) έξυπνων δεδομένων. Επιπλέον, μπορούμε περαιτέρω να ορίσουμε τα έξυπνα δεδομένα ως στοιχεία που είναι ανεξάρτητα από την εκάστοτε εφαρμογή, ικανά να ανασυντεθούν, ταξινομημένα ιεραρχικά και μέρος ενός μεγαλύτερου «οικοσυστήματος» πληροφοριών που αποκαλούμε οντολογία. Η κοινοπραξία World Wide Web (W3Consortium) έχει καθιερώσει μια δραστηριότητα (που αποτελείται από διάφορες ομάδες) που αφιερώνεται στην εφαρμογή του οράματος του Semantic Web.

2.4 Η δομή του Σημασιολογικού Ιστού;

Το Semantic Web αναμένεται να επιλύσει διάφορα βασικά προβλήματα που αντιμετωπίζουν οι τρέχουσες αρχιτεκτονικές τεχνολογίες πληροφοριών.

Η αρχιτεκτονική του σημασιολογικού ιστού συνίσταται στα παρακάτω τρία επίπεδα:

1) ***The metadata layer (Το επίπεδο μεταδεδομένων):*** Το μοντέλο δεδομένων σε αυτό το επίπεδο περιέχει κυρίως τις έννοιες πόροι και ιδιότητες. Η γλώσσα RDF (Resource Description Framework) είναι το επικρατέστερο μοντέλο δεδομένων για αυτό το επίπεδο.
2) ***The schema layer (το επίπεδο σχημάτων):*** Σε αυτό το επίπεδο εισάγονται γλώσσες οντολογιών για τον ιστό που ορίζουν ιεραρχικές περιγραφές εννοιών και ιδιοτήτων. Το RDFS (Resource Description Framework Schema) είναι το επικρατέστερο υποψήφιο σχήμα για αυτό το επίπεδο.
3) ***The logical layer (το λογικό επίπεδο):*** Που αποτελείται από πιο ισχυρές γλώσσες οντολογιών. Αυτές οι γλώσσες προσφέρουν ένα μεγαλύτερο σύνολο από διαμορφωμένες αρχές που μπορούν να χαρτογραφηθούν στις γνωστές εκφραστικές λογικές περιγραφής. Εδώ η OIL και η DAML-OIL ήταν οι δύο επικρατέστερες γλώσσες. Πλέον ο W3C προτείνει την OWL Web Ontology Language ως επίσημη γλώσσα οντολογιών.

2.5 Υπερφόρτωση Πληροφοριών (Information Overloading)

Η υπερφόρτωση πληροφοριών είναι το προφανέστερο πρόβλημα που χρήζει λύσης και οι τεχνολόγοι εμπειρογνώμονες έχουν προειδοποιήσει ήδη εδώ και 50 έτη. Είναι φυσικό, ότι αυτό το πρόβλημα έχει οπωσδήποτε διογκωθεί με τη διάδοση του Διαδικτύου, του ηλεκτρονικού ταχυδρομείου, και τώρα του στιγμιαίου μηνύματος. Δυστυχώς, η προκατάληψή μας υπέρ της παραγωγής και κατά της επαναχρησιμοποίησης της γνώσης έχει αφήσει αυτό το πρόβλημα εκκρεμές έως ότου έχει λάβει τελικά μια τραγική αναλογία.

Μια υπενθύμιση της αποτυχίας μας να σημειώσουμε πρόοδο σε αυτό το ζήτημα είναι και η προειδοποίηση του Vannevar Bush (Bush, 1945) όταν είπε: «Τώρα πια δημιουργείται ένα βουνό ανάπτυξης στην έρευνα. Ωστόσο είναι ολοένα αυξανόμενη η πεποίθηση ότι αδυνατούμε να σκαρφαλώσουμε καθώς η εξειδίκευση επεκτείνεται. Ο ερευνητής τρικλίζετε από τις διαπιστώσεις και τα συμπεράσματα χιλιάδων άλλων εργαζομένων - συμπεράσματα που δεν μπορεί να βρει το χρόνο να αντιληφθεί και πολύ λιγότερο να τα θυμηθεί, όπως εμφανίζονται. Παρ'όλα αυτά η εξειδίκευση γίνεται όλο και πιο απαραίτητη για την πρόοδο, και η προσπάθεια που οφείλει να καταβάλει κανείς προκειμένου να γεφυρώσει το χάσμα μεταξύ των απόψεων είναι ακολούθως υπερφυσική».

2.6 Ο ρόλος των μεταδεδομένων

Ένας έξυπνος και αρκετά απλός ορισμός του όρου μεταδεδομένα (metadata) είναι «δομημένα δεδομένα για τα δεδομένα». Ο όρος αυτός χρησιμοποιείται μόνο τα τελευταία δεκαπέντε (15) χρόνια και είναι γνωστός και κατανοητός από το πληθυσμό που ασχολείται με τον Παγκόσμιο Ιστό. Ο παραπάνω πολύ γενικός ορισμός περιλαμβάνει ένα απεριόριστο φάσμα δυνατοτήτων το οποίο εκτείνεται από την χειροκίνητη γραπτή περιγραφή μιας πηγής πληροφοριών ως την αυτόματη εξαγωγή μεταδεδομένων η οποία μπορεί να είναι χρήσιμη μόνο σε εφαρμογές λογισμικού.

Ο όρος μεταδεδομένα, κατά γράμμα «δεδομένα για τα δεδομένα», είναι ένας ευρέως διαδεδομένος όρος τον οποίο αντιλαμβάνονται και κατανοούν ποικιλοτρόπως

οι διάφορες επαγγελματικές κοινότητες οι οποίες δημιουργούν, περιγράφουν, συντηρούν και χρησιμοποιούν πληροφοριακά συστήματα και πηγές πληροφοριών. Καθώς αυτές οι κοινότητες συγκεντρώνονται ώστε να γίνει πραγματικότητα η «Εποχή της Πληροφορίας», είναι απαραίτητο να κατανοήσουμε τους διαφορετικούς ρόλους που διαδραματίζουν οι διαφορετικοί τύποι μεταδεδομένων στην ανάπτυξη αποτελεσματικών και επίσημων συστημάτων πληροφοριών πολιτισμικής κληρονομιάς και διατήρησης αρχείου / εγγραφών. Μέχρι τα μέσα της δεκαετίας του '90, ο όρος μεταδεδομένα χρησιμοποιούταν από κοινότητες οι οποίες ασχολούνταν με τη διαχείριση γεωγραφικών δεδομένων, με τη διαχείριση δεδομένων και τον σχεδιασμό πληροφοριακών συστημάτων (η χρήση ήταν απολύτως συνυφασμένη με τις ανάγκες της κάθε κοινότητας). Για αυτές τις κοινότητες, τα μεταδεδομένα αναφέρονταν σε ένα τμήμα της βιομηχανίας ή σε πειθαρχημένα πρότυπα καθώς και σε πρόσθετη εσωτερική ή εξωτερική τεκμηρίωση και άλλες απαιτούμενες πληροφορίες για την αναγνώριση, την αναπαράσταση, την τεχνική διαχείριση, την επίδοση και την χρήση των δεδομένων που εμπεριέχονται σε ένα πληροφοριακό σύστημα.

Το γνωστικό αντικείμενο είναι οτιδήποτε στο οποίο μπορεί ένας άνθρωπος ή ένα σύστημα να απευθυνθεί και να το διαχειριστεί ως διακριτή οντότητα. Ένα τέτοιο αντικείμενο μπορεί να αποτελείται από ένα μόνο απλό αντικείμενο ή να αποτελεί το συνάθροισμα πολλών απλών αντικειμένων. Γενικά, όλα τα γνωστικά αντικείμενα, ανεξάρτητα από τη φυσική και πνευματική υπόστασή τους, διαθέτουν τρία χαρακτηριστικά τα οποία είναι δυνατόν να απεικονιστούν στα μεταδεδομένα:

- ***Content (περιεχόμενο)*** το οποίο σχετίζεται με το περιεχόμενο του αντικειμένου γνώσης και είναι εγγενές στοιχείο ενός γνωστικού αντικειμένου
- ***Context (πλαίσιο χρήσης)*** που υποδεικνύει ποιος, τι, που, γιατί και πως δημιουργεί, μετατρέπει ή χρησιμοποιεί το αντικείμενο αυτό και είναι εξωτερικό στοιχείο του αντικειμένου εσωτερικά ή μεταξύ των αντικειμένων και μπορεί να είναι είτε εξωτερικό ή εγγενές στοιχείο του γνωστικού αντικειμένου

Συνεπώς , κάνουν σαφή την πληροφορία που είναι αόριστη " making explicit the implicit" και εκθέτουν την πληροφορία για αναζήτηση, επεξεργασία και ενοποίηση.

3 Οντολογίες

3.1 Οντολογίες

Σε αυτό το μέρος αποσαφηνίζεται ο όρος Οντολογία (Menzel, 2003) περνώντας από τη φιλοσοφία στην έννοια που παίρνει στην επιστήμη της Τεχνολογίας της Πληροφορίας (Information Technology). Παρουσιάζονται τα βασικά στοιχεία μιας οντολογίας, τα επίπεδα στα οποία συνήθως αναλύεται καθώς και η διαδικασία που ακολουθείται για το σχεδιασμό της με ιδιαίτερη αναφορά στα κριτήρια σχεδίασης και την αξιολόγηση της. Ακολουθεί μια συνοπτική αναφορά των πιο γνωστών γλωσσών περιγραφής οντολογιών με επίκεντρο τις Web-based γλώσσες και των εργαλείων ανάπτυξης τους με επίκεντρο το Protege 2000. Στη συνέχεια δίνεται μια εικόνα των εφαρμογών στις οποίες είναι χρήσιμες οι οντολογίες.

Η ανάπτυξη μιας οντολογίας είναι μια καινοτόμος εφαρμογή, που έρχεται να καλύψει κενά προηγούμενων μοντελοποιήσεων, στον τομέα αναπαράστασης γνώσης. Επομένως, ο κάθε σχεδιαστής -προγραμματιστής θα πρέπει να αναπτύσσει μια καινούρια οντολογία έχοντας κατά νου ότι αυτή θα πρέπει:

Να διαμοιράζει την κοινή αντίληψη της δομής της πληροφορίας: Είναι ίσως ο πιο σημαντικός στόχος στην ανάπτυξη οποιασδήποτε οντολογίας. Για παράδειγμα, αν υποθέσουμε ότι διάφοροι δικτυακοί τόποι περιέχουν πληροφορία, διαμοιράζονται και δημοσιεύουν υλικό που βασίζεται πάνω στην ίδια οντολογία, τότε οι πράκτορες μπορούν να ανακτήσουν και να συνθέσουν πληροφορία από όλες αυτές τις διαφορετικές πηγές.

Να δίνει τη δυνατότητα επαναχρησιμοποίηση της πληροφορίας του τομέα: Μια από τις κινητήριες δυνάμεις που βρίσκονται πίσω από τις τρέχουσες εξελίξεις στην έρευνα για τις οντολογίες. Εάν μια ομάδα ερευνητών αναπτύσσει μια οντολογία, κάποιες άλλες ομάδες μπορούν απλά να την επαναχρησιμοποιήσουν στο δικό τους αντικείμενο. Επιπρόσθετα, εάν χρειαστεί να αναπτυχθεί μια ευρύτερη οντολογία, αυτό μπορεί να γίνει με τη συνένωση ήδη υπαρχόντων επιμέρους οντολογιών.

Να δημιουργεί ρητές υποθέσεις για κάθε γνωστικό τομέα, έτσι ώστε αν χρειαστεί να αλλάξει κάποια από τις υποθέσεις, εφόσον αλλάξει η γνώση που έχουμε για τον τομέα αυτό, να μπορεί να υλοποιηθεί πολύ εύκολα η αλλαγή αυτή.

Να διαχωρίζει τη γνώση του κάθε τομέα από τη λειτουργική γνώση: Μπορεί, για παράδειγμα, να γίνει η περιγραφή μιας εργασίας για τη διενέργεια μιας διεπιχειρησιακής συναλλαγής ως μια ακολουθία διάφορων επιχειρηματικών βημάτων, και να υλοποιηθεί και ένα πρόγραμμα το οποίο κάνει τη διαδικασία της διεπιχειρησιακής συναλλαγής ανεξάρτητα από τα εκάστοτε βήματα.

Να δίνει τη δυνατότητα στο χρήστη να αναλύει της γνώση του τομέα: Αυτό μπορεί να γίνει πολύ εύκολα σε περίπτωση που είναι διαθέσιμες ξεκάθαρες προδιαγραφές για την ορολογία που χρησιμοποιείται.

3.2 Η σημασία των οντολογιών στη Βιοπληροφορική

Περίπου δέκα έτη πριν οι οντολογίες ήταν σχεδόν άγνωστες έννοιες στη βιοπληροφορική. Το πλήθος των δεδομένων όμως, είναι αυτό που δημιουργεί προβλήματα στην έρευνα των βιολογικών φαινομένων και καθιστά απαραίτητη την χρήση οντολογιών, μιας και το πλήθος πηγών και υπηρεσιών προσφέρουν πληροφορίες για τα ίδια θέματα, έτσι δημιουργούνται σοβαρά προβλήματα στην ανεύρεση, εξαγωγή, αναπαράσταση, ερμηνεία και συντήρηση της πληροφορίας.

Το ερώτημα, λοιπόν, που τίθεται με ευκολία στο σημείο αυτό είναι γιατί γίνεται χρήση οντολογιών στη βιοπληροφορική; Και η απάντηση ότι: υπάρχουν διαφορές ανάμεσα στις υπάρχουσες βάσεις βιολογικών δεδομένων, δηλαδή διαφορές ορολογίας και σημασιολογικές διαφορές και οι οντολογίες χρειάζονται για να γεφυρώσουν τις διαφορές αυτές.

Οι ονομασίες των βιολογικών αντικειμένων, διαφέρουν μεταξύ των βάσεων δεδομένων και δεν ακολουθούν ένα ενοποιημένο σχέδιο επίσης οι σημαντικές βιολογικές έννοιες που είναι θεμελιώδεις στη μοριακή βιολογία είναι διφορούμενες

Οι Οντολογίες συνενώνουν δύο ουσιώδη συστατικά:

- διευκολύνουν την επεξεργασία της πληροφορίας από τον Η/Υ και
- επιτρέπουν τη σύνδεση του περιεχομένου το οποίο επεξεργάζεται μηχανικά, με τη σημασία που του δίνουν οι άνθρωποι βασιζόμενοι σε κοινά αποδεκτή ορολογία.

3.3 Η οντολογία στη φιλοσοφία

Η οντολογία ως κλάδος της φιλοσοφίας είναι η επιστήμη που μελετά την ύπαρξη, τα είδη και τις δομές των αντικειμένων, τις ιδιότητες, τα γεγονότα, τις διαδικασίες και τις σχέσεις σε κάθε πεδίο της πραγματικότητας. Ο όρος «οντολογία» χρησιμοποιείται συχνά από τους φιλοσόφους ως συνώνυμο της μεταφυσικής (δηλαδή του τι υπάρχει μετά τη «φυσική», με την έννοια που χρησιμοποίησαν οι πρώτοι μαθητές του Αριστοτέλη για να αναφερθούν σε αυτό που ο ίδιος αποκαλούσε «πρώτη φιλοσοφία»).

"*Η μεταφυσική μελέτη της φύσης της ζωής και της ύπαρξης*" - Αριστοτέλης

Η ρεαλιστική ή περιγραφική οντολογία, που είναι ο τομέας με τον οποίο ασχολείται αυτή η εργασία, δεν αναζητεί μια εξήγηση αλλά μια περιγραφή της πραγματικότητας μέσα από την οποία με την εξαντλητική κατηγοριοποίηση οντοτήτων καταφέρνει να δώσει απαντήσεις σε ερωτήματα όπως «ποιες κλάσεις οντοτήτων είναι απαραίτητες ώστε να περιγράψουμε και να εξηγήσουμε όλα όσα συμβαίνουν στο σύμπαν;». Ο όρος «εξαντλητική» αναφέρεται στο ότι όλοι οι τύποι οντοτήτων πρέπει να εμπίπτουν στην υπάρχουσα κατηγοριοποίηση, συμπεριλαμβανομένων και των σχέσεων που συνδέουν τις οντότητες μεταξύ τους. Ακριβώς αυτό το χαρακτηριστικό τη διαχωρίζει από τη σχολή του «reductionism» σύμφωνα με την οποία το ίδιο έργο είναι δυνατό με τη χρήση ενός ελαχιστοποιημένου βασικού συνόλου οντοτήτων.

Η ιστορία της οντολογίας χαρακτηρίζεται από τον συμβιβασμό ανάμεσα στην «παραγωγικότητα» (generativity) από τη μια και την «περιγραφικότητα» από την άλλη. Με τον όρο «παραγωγικότητα» νοείται η δύναμη μιας οντολογίας να δημιουργεί νέες κατηγορίες και κατά συνέπεια εξαντλεί το πεδίο εφαρμογής της. «Περιγραφικότητα» ονομάζεται η ικανότητα μιας οντολογίας να αναπαριστά την πραγματικότητα. Αυτά τα δύο είναι τα κύρια χαρακτηριστικά μιας οντολογίας σε κάθε της μορφή, είτε αυτή είναι φιλοσοφική είτε όχι.

Η έννοια της οντολογίας έχει υιοθετηθεί από την Τεχνητή Νοημοσύνη και σημαίνει

"*μια διαμοιρασμένη και κοινή κατανόηση κάποιου τομέα, η οποία μπορεί να ανταλλαγεί μεταξύ ανθρώπων και συστημάτων εφαρμογών*"

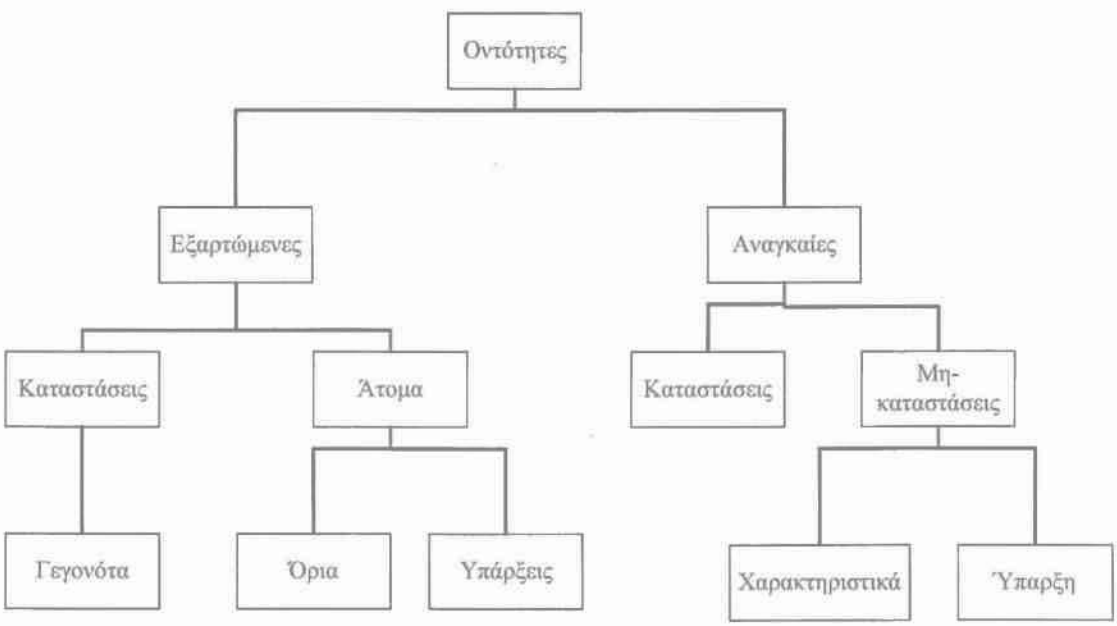
επίσης

Μια οντολογία είναι μια τυπική (formal), κατηγορηματική (explicit) προδιαγραφή μιας διαμοιρασμένης (shared) εννοιολογικής αναπαράστασης (conceptualization) – (Gruber, 1993)

Ο πιο αποδεκτός ορισμός μιας οντολογίας είναι ο παραπάνω ορισμός του Gruber σύμφωνα με τον οποίο μια οντολογία είναι «μια επίσημη προδιαγραφή μιας σύλληψης» μέσα σε μια συγκεκριμένη περιοχή. Είναι μια ιεραρχημένη περιγραφή ενός συνόλου από έννοιες (είναι μια ιεραρχία), ένα σύνολο από ιδιότητες και οι μεταξύ τους σχέσεις καθώς κι ένα σύνολο από συμπερασματικούς κανόνες. Αφού οι οντολογίες εξελίσσονται με την πάροδο του χρόνου, είναι σημαντικό να βρούμε έναν τρόπο να ενημερώνουμε και να εξελίσσουμε τις παρούσες οντολογίες

3.4 Ταξινομία

Μια οντολογία είναι σε μια πρώτη προσέγγιση μια συλλογή κατηγοριών στην οποία κάθε οντότητα είναι ένας κόμβος τοποθετημένος σε ένα ιεραρχικό δέντρο. Αυτή η λογική στηρίζεται στη σκέψη του Αριστοτέλη καθώς και των μεσαιωνικών φιλοσόφων και επαναφέρθηκε στον χώρο της σύγχρονης οντολογίας από τον Chisholm (1996) ο οποίος δίνει το εξής δέντρο κατηγοριών(εικόνα 3.1) :



Εικόνα 3.1 Δέντρο του Chisholm

Επί της αρχής, κάθε οντότητα μπορεί να ενταχθεί στο παραπάνω δέντρο το οποίο μπορεί να επεκταθεί σε οποιονδήποτε βαθμό ώστε να καλυφθεί το υπό μελέτη πεδίο. Βέβαια το παραπάνω παράδειγμα είναι ιδιαίτερα γενικό αφού επιχειρεί να θεμελιώσει κάθε πιθανή οντολογία. Στην πράξη όμως οι οντολογίες έχουν έναν κατά περίπτωση χαρακτήρα (ad-hoc) αφού αυτός κρίνεται ικανός για τη δόμηση μιας ταξινομίας ικανής να περιγράψει την ιεραρχία ενός συστήματος.

Άλλα κριτήρια που πρέπει να ικανοποιεί μια ταξινομία ώστε να ικανοποιεί της αρχές της οντολογίας έχουν να κάνουν με το κατά πόσο αυτή είναι καλοσχηματισμένη (well-formedness) ή -σε γλωσσολογικούς όρους- ποια είναι η ορθότητα της γραμματικής της. Ως προς αυτή την ιδιότητα μπορούν να τεθούνε οι εξής αρχές:

- Η ταξινομία πρέπει να έχει τη μορφή δέντρου με τη μαθηματική έννοια του όρου. Αυτό σημαίνει πως πρέπει να είναι ένα συνδεδεμένος γράφος χωρίς κλειστά μονοπάτια (κύκλους). Οι σε υψηλότερο επίπεδο κόμβοι πρέπει να αναπαριστούν οντότητες με μεγαλύτερη γενικότητα και οι θυγατρικοί τους κόμβοι να είναι υποκατηγορίες τους. Κατά συνέπεια, ισχύουν οι αρχές τις κληρονομικότητας με την αυστηρή έννοια αφού απαγορεύεται μια οντότητα να έχει παραπάνω από μια μητρική. Αν και εκ πρώτης όψεως αυτό το χαρακτηριστικό φαίνεται να είναι περιοριστικό αφού στην πράξη μπορεί να είναι επιθυμητή η ένταξη ενός αντικειμένου σε δύο κατηγορίες (π.χ. ένα κόκκινο αυτοκίνητο στην κατηγορία «αυτοκίνητα» και στην «αντικείμενα με κόκκινο χρώμα») αυτό το εμπόδιο μπορεί να υπερπηδηθεί με τη χρήση σχέσεων άλλου τύπου πέραν αυτής της κληρονομικότητας ανάμεσα σε κατηγορίες αντικειμένων. Με άλλα λόγια, πέρα από την «Is-a relationship» που δομεί τη βασική μορφή του δέντρου μπορούν από αυτό να προκύψουν άλλοι γράφοι (συνδεδεμένοι ή ασύνδετοι) με τον ορισμό άλλων σχέσεων οι οποίοι με τη σειρά τους είναι διαφορετικές «ταξινομίες». Η χρήση των εισαγωγικών γίνεται γιατί μια ταξινομία με την αυστηρή έννοια είναι το ιεραρχικό δέντρο κληρονομικότητας που περιγράψαμε.
- Μία ταξινομία πρέπει να έχει στη βάση της «φύλλα» δηλαδή κατηγορίες χωρίς υποκατηγορίες. Αν και η δομή δέντρου εγγυάται κάτι

τέτοιο ο κανόνας αυτός προστίθεται για να τονίσει την ανάγκη καθορισμού των στοιχειωδών κατηγοριών που δομούν μια υπερκατηγορία ώστε να διασφαλίζεται ότι κάθε αντικείμενο μπορεί να ενταχθεί στο δέντρο ή με άλλα λόγια ότι η κατηγοριοποίηση είναι όντως «εξαντλητική».

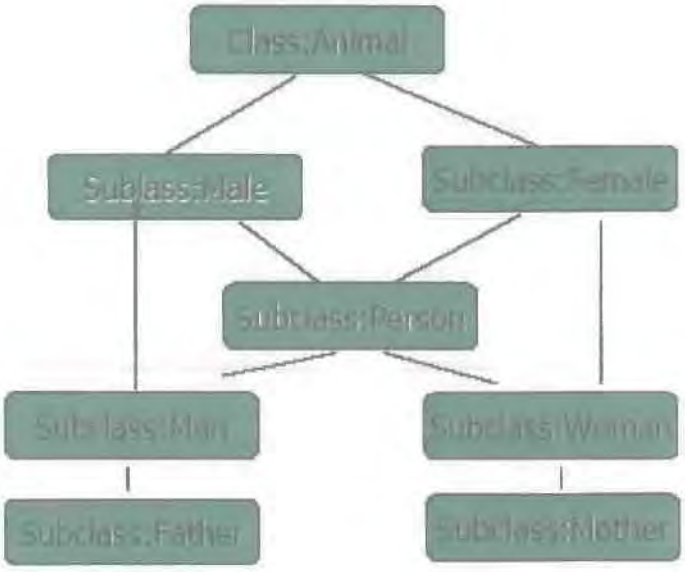
- Η ταξινομία πρέπει να είναι «ενοποιημένη» δηλαδή το δέντρο της σε όρους γράφου να είναι συνδεδεμένο με μία και μόνο κατηγορία να βρίσκεται στην κορυφή και όλες τις άλλες να είναι υποκατηγορίες της. Αν αυτό το κριτήριο δεν ικανοποιείται και υπάρχουν περισσότερες από μία ρίζες τότε προκύπτουν δύο διαφορετικές ταξονομίες.
- Αν μία ταξινομία δομηθεί με αυτούς τους όρους τότε για τη ρίζα της μπορεί να υιοθετηθεί ο όρος οντότητα υπό την έννοια ότι όλες οι υποκατηγορίες εκφράζουν οντότητες.

3.5 *Οντολογία*

Ο όρος οντολογία χρησιμοποιείται όλο και περισσότερο τα τελευταία χρόνια στο πεδίο των επιστημών των υπολογιστών και της πληροφορίας. Είναι ιδιαίτερα δημοφιλής στα πεδία της διαχείρισης γνώσης (knowledge engineering and management), της επεξεργασίας φυσικής γλώσσας (natural language processing), της διασύνδεσης πληροφοριακών συστημάτων (cooperative information systems), της τεχνητής νοημοσύνης (artificial intelligence) καθώς και αλλού. Στο χώρο των πληροφοριακών συστημάτων, σύμφωνα με τον Barry Smith (Smith 2008), «οντολογία είναι ένα λογισμικό (ή φορμαλιστική γλώσσα) σχεδιασμένο βάσει ενός συγκεκριμένου συνόλου χρήσεων και υπολογιστικών περιβαλλόντων και επί ενός συγκεκριμένου πλαισίου».

Στον τομέα της πληροφορίας ο αυξανόμενος όγκος και πολυπλοκότητα των δεδομένων σήμανε και την αυξημένη δυσκολία στη διαχείριση τους καθώς και την επαναχρησιμοποίηση τους από άλλα προγράμματα. Αν οι εφαρμογές θεωρηθούν ως αναπαραστάσεις διαδικασιών και οι διάφορες δομές δεδομένων αναπαραστάσεις αντικειμένων και λαμβάνοντας υπόψιν ότι για τη δημιουργία αποτελεσματικών αναπαραστάσεων είναι σημαντικό να είναι

γνωστό το νόημά τους συμπεραίνουμε ότι δεν αρκεί να γνωρίζουμε μόνο τα αντικείμενα μιας αναπαράστασης αλλά και τις ιδιότητές τους, τις σχέσεις τους καθώς και τις διαδικασίες στις οποίες μπορούν να εμπλακούν. Η απάντηση σε αυτήν την απαίτηση δόθηκε με τη δημιουργία «δηλωτικών» (declarative) αναπαραστάσεων των διαδικασιών μέσω «εννοιολογικών παραστάσεων» (conceptual schemes) που επιτρέπουν την επαναχρησιμοποίηση δεδομένων (data) και διαδικασιών (procedures) ανάμεσα σε προγράμματα/ εφαρμογές.



Εικόνα 3.2 Παράδειγμα οντολογίας

Μια οντολογία μπορεί να πάρει διάφορες μορφές(π.χ εικόνα 3.2 και 3.3) αλλά οπωσδήποτε θα περιλαμβάνει ένα λεξιλόγιο όρων και κάποιας μορφής προδιαγραφές για τη σημασία τους. Σχετικά με τον βαθμό της τυπικότητας της αναπαράστασης μιας οντολογίας αυτή μπορεί να είναι:

- *Άτυπη* (informal), εκφρασμένη σε μια φυσική γλώσσα.
- *Ημι-άτυπη* (semi-informal): για παράδειγμα διατυπωμένη σε ένα περιορισμένο και δομημένο υποσύνολο κάποιας φυσικής γλώσσας.
- *Ημι-τυπική* (semi-formal): διατυπωμένη σε μια τεχνητή και αυστηρά ορισμένη γλώσσα.
- *Αυστηρά τυπική* (rigorously formal): ορισμοί όρων με αυστηρή σημασιολογία, θεωρήματα και αποδείξεις ιδιοτήτων όπως η ορθότητα (soundness) και η πληρότητα (completeness).

```
class-def animal
%plants are a class that is disjoint from animals
class-def plant subclass-of NOT animal
%it is necessary but not sufficient for a tree to be a plant:
class-def tree subclass-of plant
%branches are PART OF trees
class-def branch
   slot-constraint is-part-of has-value tree
%it is necessary and sufficient for a carnivore to be an
   animal:
class-def defined carnivore subclass-of animal
   slot-constraints eats value-type animal
%herbivores eat only plants OR part of plants
class-def defined herbivore subclass-of animal
   slot-constraint eats value-type plant OR
   (slot-constraint is-part-of has-value plant)
```

Εικόνα 3.3 Παράδειγμα οντολογίας

Οι οντολογίες χωρίζονται σε ορισμένες κατηγορίες. Μερικές χαρακτηριστικές κατηγορίες οντολογιών είναι οι ακόλουθες:

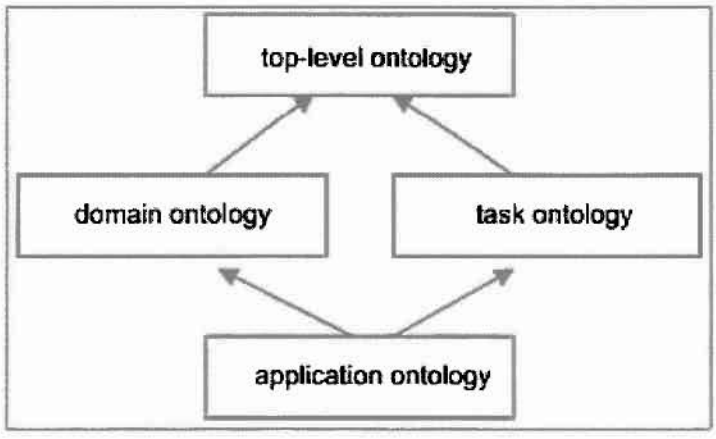
- *Οντολογίες πεδίου ορισμού* (domain ontologies): αναπαριστούν γνώση γύρω από ένα συγκεκριμένο πεδίο (π.χ. ιατρική, ηλεκτρονικά κ.λ.π.).
- *Οντολογίες μεταδεδομένων* (metadata ontologies): παρέχουν ένα λεξιλόγιο για την περιγραφή του περιεχομένου ηλεκτρονικά διαθέσιμης πληροφορίας.
- *Γενικές* ή *κοινές οντολογίες* (generic or common sense ontologies): στοχεύουν στο να αποτυπώσουν γενική γνώση γύρω από τον κόσμο, παρέχοντας βασικές έννοιες όπως ο χρόνος, ο χώρος, τα συμβάντα, κ.λ.π.
- *Οντολογίες αναπαράστασης* (representational ontologies): παρέχουν οντότητες αναπαράστασης χωρίς να προσδιορίζουν τη συγκεκριμένο αναπαριστούν - π.χ. Frame Ontology (Gruber 1993): ορίζει έννοιες όπως frames, slots, slot constraints κ.λ.π.
- *Οντολογίες μεθοδολογίας* ή *εργασιών* (method or task ontologies): παρέχουν όρους που αναφέρονται σε συγκεκριμένες εργασίες (π.χ. διάγνωση κ.λ.π.)

Η λέξη "οντολογία" έχει χρησιμοποιηθεί για να περιγράψει τεχνήματα διαφορετικών βαθμών δομής από απλές ταξινομήσεις (όπως η ιεραρχία του Yahoo), σχήματα μεταδεδομένων ως λογικές θεωρίες. Παρόλα αυτά οι περισσότερες εφαρμογές στην πράξη (π.χ. το Semantic Web) χρειάζονται οντολογίες με έναν

σημαντικό βαθμό δομής και οι οντολογίες αυτές πρέπει να διευκρινίζουν τις περιγραφές για τα ακόλουθα είδη εννοιών:

- Κλάσεις (classes): Έννοιες που σχετίζονται με ένα πεδίο ή κάποιες εργασίες, οι οποίες είναι συνήθως οργανωμένες σε κάποιο ταξινομικό σύστημα, σε μια οντολογία που αφορά το νοσοκομείο: ο 'ιατρός' και ο 'ασθενής' αποτελούν δύο κλάσεις.
- Σχέσεις (relations): Ένας τύπος αλληλεπίδρασης μεταξύ εννοιών ενός πεδίου όπως subclass-of, is-a
- Συναρτήσεις (functions) μια ειδική περίπτωση σχέσης στην οποία το ν-οστό στοιχείο της σχέσης προσδιορίζεται μοναδικά από τα ν-1 προηγούμενα στοιχεία. Παράδειγμα: Η χρέωση του ασθενή μπορεί να προσδιορίζεται σαν συνάρτηση της προηγούμενης επίσκεψης καθώς και της ιατρικής κάλυψης που τυχών να έχει ο ασθενής
- Αξιώματα (axioms) αναπαριστούν προτάσεις που είναι πάντα αληθείς. Παράδειγμα: αν ο Α είναι ο ασθενής τότε μπορεί να νοσηλευτεί στο επιλεγόμενο νοσοκομείο Ν.
- Στιγμιότυπα (instances) αναπαριστούν συγκεκριμένα στοιχεία. Παράδειγμα: ο ασθενής με το όνομα Νίκος είναι ένα στιγμιότυπο της κλάσης 'ασθενής'.

Σύμφωνα με τον Guarino τα επίπεδα στα οποία συνίσταται μια οντολογία είναι όπως απεικονίζονται στην εικόνα 3.4 (Guarino, 1994):



Εικόνα 3.4 Επίπεδα οντολογίας κατά Guarino

- Ανώτερο επίπεδο (top level): Περιγράφει πολύ γενικές έννοιες (όπως ο χώρος, ο χρόνος, το γεγονός), οι οποίες είναι ανεξάρτητες από ένα συγκεκριμένο πρόβλημα ή περιοχή γνώσης. Είναι λογική η ύπαρξη ενοποιημένης οντολογίας υψηλού επιπέδου για μεγάλες κοινότητες χρηστών.
- Επίπεδο «πεδίου γνώσης» (domain): Περιγράφει λεξιλόγιο σχετικό με ένα γενικό πεδίο γνώσης εξειδικεύοντας τις έννοιες που εισήχθησαν στην οντολογία υψηλού επιπέδου.
- Επίπεδο δραστηριοτήτων (task): Περιγράφει λεξιλόγιο σχετικό με μια γενική δραστηριότητα/ εργασία έτσι ώστε να εξειδικεύσει την οντολογία υψηλού επιπέδου.
- Επίπεδο εφαρμογής (application): Είναι οι πιο συγκεκριμένες οντολογίες. Οι έννοιες σε οντολογίες εφαρμογής συχνά ανταποκρίνονται σε ρόλους οντοτήτων της περιοχής γνώσης καθώς εκτελούν συγκεκριμένη δραστηριότητα.

Μία βάση δεδομένων είναι ένα σύνολο από πίνακες και τις μεταξύ τους σχέσεις. Μία οντολογία περιέχει συντακτικά και σημασιολογικά πλουσιότερη πληροφορία από τις βάσεις δεδομένων. Η πληροφορία που περιγράφεται από μία οντολογία περιέχει ημί-δομημένο κείμενο σε φυσική γλώσσα και όχι πληροφορία ενός σχεσιακού πίνακα. Μία οντολογία πρέπει να είναι δικτυακής αρχιτεκτονικής γιατί χρησιμοποιείται για το διαμοιρασμό της πληροφορίας.

3.6 Χρήσεις των οντολογιών στη επιστήμη της πληροφορίας

Ο αριθμός των εφαρμογών των οντολογιών στον χώρο τον πληροφοριακών συστημάτων συνεχώς αυξάνεται και βασικός λόγος γι' αυτό είναι το γεγονός ότι η εστίαση μιας οντολογίας στην ταξινόμηση και στον περιορισμό των επιτρεπόμενων ταξονομιών και ορισμών έχει αποδειχθεί χρήσιμη με τρόπους που δεν είχαν αρχικά προβλεφθεί. Η αυτοματοποίηση απαιτεί υψηλό βαθμό ακρίβειας στην περιγραφή των διαδικασιών της και η οντολογία είναι ο κατάλληλος μηχανισμός για την επίτευξη αυτού.

Συχνά η οντολογία (με την έννοια της επιστήμης) απαντάται με άλλους όρους. Έτσι υπάρχουν τα «εννοιολογικά σχήματα» (conceptual schemas) στη σχεδίαση βάσεων δεδομένων, τα «μοντέλα πεδίων εφαρμογής» (models of application domains) στο software engineering ή τα «μοντέλα κλάσεων» (class models) στην αντικειμενοστραφή σχεδίαση λογισμικού. Οι σχεδιαστές μεγάλων βάσεων δεδομένων χρησιμοποιούν αυξανόμενα οντολογικές μεθόδους ως τμήμα της προσπάθειάς τους να επιβάλλουν περιορισμούς στα δεδομένα με τέτοιο τρόπο που σύνολα δεδομένων από διαφορετικές πηγές θα αποδοθούν με συμβατό μεταξύ τους τρόπο εξαρχής. Οντολογικές μέθοδοι χρησιμοποιούνται επίσης στον φορμαλισμό των προτύπων σε επίπεδο μεταδεδομένων (που είναι άλλωστε το αντικείμενο μελέτης αυτής της εργασίας) όπου στόχος είναι η με συστηματικό τρόπο παροχή πληροφοριών σχετικών με τα δεδομένα.

4 Χτίσιμο και γλώσσες οντολογιών

4.1 Πως χτίζεται μια οντολογία

Αφού αναπτύχθηκαν τα πλεονεκτήματα που προσφέρει μια οντολογία έχει ενδιαφέρον η μελέτη της δόμησης μίας στην πράξη. Εύκολα μπορεί να ανακαλύψει κανείς και να γράψει έναν μοναδικό ορισμό για κάθε έννοια. Αυτός ο ορισμός πρέπει να είναι αρκετά ακριβής ώστε να κάνει διακριτή την κάθε έννοια από όλες τις άλλες έννοιες στην οντολογία, και πρέπει να είναι αρκετά λεπτομερείς ώστε να παρέχει μια σαφή κατανόηση της έννοιάς του. Πρέπει να είναι κανείς κατηγορηματικός όταν οι υποκατηγορίες δεν συνδέονται. Αυτό βοηθά πολύ κατά την έρευνα μέσω της ιεραρχίας των υποκατηγοριών. Πρέπει να καθορίζεται η πλήρης συνδετικότητα των εννοιών. Με αυτό τον τρόπο όλες οι έννοιες καθορίζονται με συνέπεια σε σχέση με την αναφερόμενη οντολογία ενώ το αντίθετο θα μπορούσε να προκαλέσει συγκρουόμενες ή επικαλυπτόμενες έννοιες. Χρησιμοποιείται στην αρχή της οντολογίας μία έννοια μόνο. Αυτή η έννοια μπορεί να είναι επιλεγμένη αρκετά γενικά ώστε να περιλάβει την ποικιλία των σχετικών εννοιών. Προστίθεται γνώση υποβάθρου για κάθε έννοια για να εκφραστούν οι σχετικές ιδιότητες. Προστίθενται links, λέξεις κλειδιά, βάσεις δεδομένων, ώστε με αυτόν τον τρόπο να γίνεται η σύνδεση της οντολογία με μια εφαρμογή.

Για να αναπτυχθεί μια νέα οντολογία προκειμένου να αναπαραστήσει επαρκώς έναν τομέα γνώσης προτείνονται τα παρακάτω βήματα:

1) Εύρεση του σκοπού της οντολογίας
2) Σύλληψη: Προσδιορισμός των βασικών εννοιών και των μεταξύ τους σχέσεων. Παραγωγή σαφών προδιαγραφούν σε μορφή κειμένου αυτών των εννοιών και των μεταξύ τους σχέσεων. Συμφωνία για τους όρους με τους οποίους θα αναφερόμαστε στις έννοιες και σχέσεις.
3) Κωδικοποίηση: ρητή αναπαράσταση της σύλληψης του προηγούμενου σταδίου σε μια τυπική γλώσσα.
    α) Ορισμός διάφορων αξιωμάτων
    β) Εισαγωγή στιγμιότυπων

γ) Κωδικοποίηση της οντολογίας σε μία συγκεκριμένη γλώσσα

δ) Αξιολόγηση

ε) Τεκμηρίωση

4) Ενοποίηση υπαρχουσών οντολογιών: Θα πρέπει να χρησιμοποιηθούν υπάρχουσες οντολογίες (ή τμήματα αυτών); Γενικά δύσκολο πρόβλημα.
5) Αξιολόγηση: «έκφραση τεχνικών κρίσεων σχετικά με τις οντολογίες, το σχετιζόμενο με αυτές περιβάλλον λογισμικού, και τη τεκμηρίωση σε σχέση με ένα πλαίσιο αναφοράς...» (Gomez-Perez, 2004).
6) Τεκμηρίωση: όλες οι σημαντικές παραδοχές τόσο αναφορικά με τις βασικές έννοιες που ορίζονται στην οντολογία όσο και με τα βασικά δομικά στοιχεία που χρησιμοποιήθηκαν για την έκφραση αυτών των εννοιών στην οντολογία, πρέπει να τεκμηριωθούν.

4.2 Προβλήματα που μπορεί να εμφανιστούν

Το κύριο πρόβλημα του Σημασιολογικού Ιστού είναι η έλλειψη σημασιολογίας δηλαδή η έννοια ενός περιεχομένου του Ιστού δεν είναι προσιτή από τη μηχανή

π.χ ένα πρόβλημα της HTML είναι ότι το κείμενο εμφανίζεται στον αναγνώστη σε προσιτή μορφή ενώ μία μηχανή αναζήτησης μπορεί να αντιμετωπίσει ορισμένα προβλήματα.

Προβλήματα σε σχέση με τις οντολογίες

- Η χειροκίνητη δημιουργία οντολογιών είναι δαπανηρή και χρονοβόρα
- Απαιτούνται τρόποι αυτόματης περιγραφής περιεχομένου
- Υπάρχει έλλειψη μηχανών εξαγωγής συμπερασμάτων
- Δεν υπάρχει μία standard web ontology language
- Πρέπει να ξεκαθαριστεί το γιατί να κατασκευαστεί η οντολογία και ποιες είναι οι πιθανές χρήσεις της.
- Χρήσιμο είναι επίσης να προσδιοριστεί ποιοι πρόκειται να είναι οι πιθανοί χρήστες της οντολογίας.

Η λειτουργική αρχιτεκτονική του σημασιολογικού ιστού έχει τρία επίπεδα, το επίπεδο μεταδεδομένων, το επίπεδο σχήματος και το λογικό επίπεδο. Αυτή τη στιγμή το επικρατέστερο μοντέλο δεδομένων είναι το RDF για το επίπεδο μεταδεδομένων. Παρόλο που το RDF μοντέλο δεδομένων θεωρείται αρκετό για τον ορισμό και την

χρήση μεταδεδομένων, δεν έχουν οριστεί ακόμα οι σημασιολογικές έννοιες για τις δηλώσεις που ορίζουν δηλώσεις. Το RDFS είναι το πιο δημοφιλές σχήμα για το επίπεδο σχήματος. Όμως το RDFS δεν διαθέτει επίσημα semantics και μια πρόταση είναι να οριστεί μια αρχιτεκτονική μεταδεδομένων για το RDFS παρόμοια με αυτή του UML (Universal Modeling Language) κι έτσι να οριστούν επίσημα semantics. Αυτή η προσέγγιση, παρόλο που είναι επίσημη, είναι πολύπλοκη. Παρόλο που το RDFS έχει κατηγορηθεί για μια σημασιολογική σύγχυση, η χρήση κάποιων επίσημων semantics δεν είναι απίθανη. Για να εξομαλυνθεί η διαδικασία της ανάπτυξης του σημασιολογικού ιστού, πρέπει να λυθεί το παραπάνω πρόβλημα.

4.3 Διαχείριση οντολογιών: Συνδυασμός συσχέτιση και προβλήματα

Η επαναχρησιμοποίηση ο συνδυασμός και η ενοποίηση διαφορετικών οντολογιών (δηλαδή συνένωση σε 1 οντολογία ή χωριστά) γίνεται με την εξής διαδικασία ενοποίησης οντολογιών:

- Εύρεση των σημείων εκείνων στις οντολογίες που έχουν κοινά χαρακτηριστικά.
- Συσχέτιση των εννοιών εκείνων που είναι σημασιολογικά κοντά, μέσω της ισοδυναμίας και των σχέσεων υπαγωγής.
- Έλεγχος της συνοχής και της συνάφειας του αποτελέσματος
- Ευθυγράμμιση εννοιών:είναι δύσκολη διαδικασία η οποία επιφέρει αλλαγή σε μία τουλάχιστον από τις οντολογίες με αποτέλεσμα νέα έκδοση της οντολογίας. Σε περίπτωση που οι οντολογίες δεν αναπαριστώνται στην ίδια γλώσσα αναγκαία η μετάφρασή τους.

Κατά την διαδικασία ενοποίησης οντολογιών μπορεί να δημιουργηθούν προβλήματα σε διάφορα επίπεδα της οντολογίας όπως: Στο επίπεδο της γλώσσας ή το επίπεδο μετα-μοντέλου (language or meta-model level) χρησιμοποιούνται τα πρωταρχικά στοιχεία μιας γλώσσας προκειμένου να οριστεί και να προσδιοριστεί μια οντολογία (αναντιστοιχίες ανάμεσα στους διαφορετικούς μηχανισμούς που ορίζουν τις κλάσεις, τις σχέσεις κλπ.).

Στο επίπεδο οντολογίας ή το επίπεδο μοντέλου (ontology or model level) ενυπάρχει η πραγματική οντολογία ενός τομέα (η διαφορά στον τρόπο με τον οποίο

μοντελοποιείται το εκάστοτε πεδίο ορισμού). Καθώς και αναντιστοιχίες στο επίπεδο της γλώσσας στο συντακτικό, στη λογική αναπαράσταση, στη σημασιολογία πρωταρχικών στοιχείων, στην εκφραστικότητα της γλώσσας.

Τέλος μπορεί να δημιουργηθούν αναντιστοιχίες στο επίπεδο οντολογίας στην εμβέλεια και στην κάλυψη μοντέλου και διαβάθμιση. Παράδειγμα: σε κάποια περιγραφή μιας έννοιας να έχουμε αναπαράσταση συνώνυμων και ομώνυμων όρων σε διαφορετικά πεδία (αναντιστοιχίες ορολογίας).

4.4 Κριτήρια σχεδιασμού

Για να σχεδιαστεί και να αναπτυχθεί σωστά μια οντολογία χρειάζονται «αντικειμενικά» κριτήρια σχεδίασης. Αυτά τα κριτήρια έχουν να κάνουν κυρίως με τις απαιτήσεις της ίδιας της ίδιας της οντολογίας (τι θέλουμε να αναπαραστήσουμε;) και ως τέτοια προτείνονται:

Σαφήνεια: Η οντολογία θα πρέπει να διαθέτει αντικειμενικά ορισμένες έννοιες. Όταν αυτό είναι δυνατό θα πρέπει να προτιμάται ο ολικός ορισμός μιας έννοιας παρά η χρήση της ως αφηρημένη, ενώ απαιτείται όλες οι έννοιες να συνοδεύονται από περιγραφή σε φυσική γλώσσα.

Συνοχή: Τα αξιώματα της οντολογίας πρέπει να είναι λογικά συνεπή, ενώ συνοχή οφείλει να διαθέτει και κάθε ορισμός έννοιας σε φυσική γλώσσα. Σε περίπτωση που μια πρόταση που μπορεί να προέλθει από βασικό αξίωμα, έρχεται σε αντίθεση με την τεκμηρίωση ή ένα παράδειγμα της φυσικής γλώσσας τότε η συνοχή της οντολογίας καταλύεται.

Επεκτασιμότητα: Μια οντολογία πρέπει να έχει όσο το δυνατόν μεγαλύτερες δυνατότητες επεκτασιμότητας, στόχος που επιτυγχάνεται όταν προσφέρει ένα εννοιολογικό θεμέλιο για να εξυπηρετήσει περαιτέρω τους μελλοντικούς στόχους της. Το βασικό λεξιλόγιο της οφείλει να είναι τέτοιο ώστε να μπορεί να στηρίξει τον ορισμό νέων εννοιών, πιο εξειδικευμένων κατά περίπτωση χωρίς να χρειάζεται αναθεώρηση των αρχικών όρων.

Ελάχιστη «προκατάληψη» κωδικοποίησης: Η οντολογία πρέπει να δομηθεί σε επίπεδο γνώσης και να παραμείνει κατά το δυνατόν ανεξάρτητη από μια ειδικευμένη κωδικοποίηση σε επίπεδο συμβόλων ώστε να μπορεί η κοινή γνώση να εφαρμοστεί σε διαφορετικά συστήματα ή μοντέλα αναπαράστασης.

Ελάχιστη οντολογική δέσμευση: Μια οντολογία πρέπει να απαιτεί την ελάχιστη οντολογική δέσμευση επαρκή βέβαια για να στηρίξει και να μοντελοποιήσει τη διαμοιραζόμενη γνώση. Κατά αυτόν τον τρόπο, δίνεται η δυνατότητα στις οργανωτικές δομές της να την εξειδικεύσουν και να την αρχικοποιούν κατά περίπτωση. Εφόσον η οντολογική δέσμευση βασίζεται στον σαφή ορισμό του λεξιλογίου μιας οντολογίας, αρκεί να ορίζονται οι βασικοί κανόνες που χρειάζεται η μοντελοποίηση της κοινής γνώσης.

4.5 Δυσκολίες στην οικοδόμηση μιας οντολογίας

Υπάρχουν διάφορες δυσκολίες που συναντάμε κατά την οικοδόμηση μιας οντολογίας. Μερικές δυσκολίες είναι έμφυτες στη διαδικασία οικοδόμησης της οντολογίας, ενώ άλλες προκύπτουν από το πεδίο εφαρμογής π.χ :

- Ελλιπή οντολογικά στοιχεία όπως ελλιπείς κατηγορίες και ελλιπείς ιδιότητες /σχέσεις
- Στις σχέσεις μεταξύ των εννοιών 1:1 εναντίον 1: Many και 1:Many εναντίον Many:Many
- Υπέρ-ορισμένα όπως περιττά οντολογικά στοιχεία και λεπτομέρειες που δεν είναι σχετικές με την έννοια

4.6 Μερικές Εφαρμογές των Οντολογιών

Σε αυτό το κεφάλαιο παρουσιάζονται μερικές εφαρμογές των οντολογιών όπως Επικοινωνία μεταξύ ανθρώπων και οργανισμών: Παρέχουν ολοκληρωμένο πλαίσιο εννοιών και ορολογίας μεταξύ ανθρώπων με διαφορετικές ανάγκες και οπτικές γωνίες στα πλαίσια ενός οργανισμού. Διευκολύνουν την επικοινωνία των ανθρώπων στα πλαίσια του οργανισμού.

Δια-λειτουργικότητα (inter-operability) μεταξύ συστημάτων: Διάφοροι χρήστες χρειάζεται να ανταλλάσσουν δεδομένα ή χρησιμοποιούν διαφορετικά πακέτα λογισμικού. Χρήση οντολογιών για την υποστήριξη μετάφρασης μεταξύ διαφορετικών γλωσσών και αναπαραστάσεων.

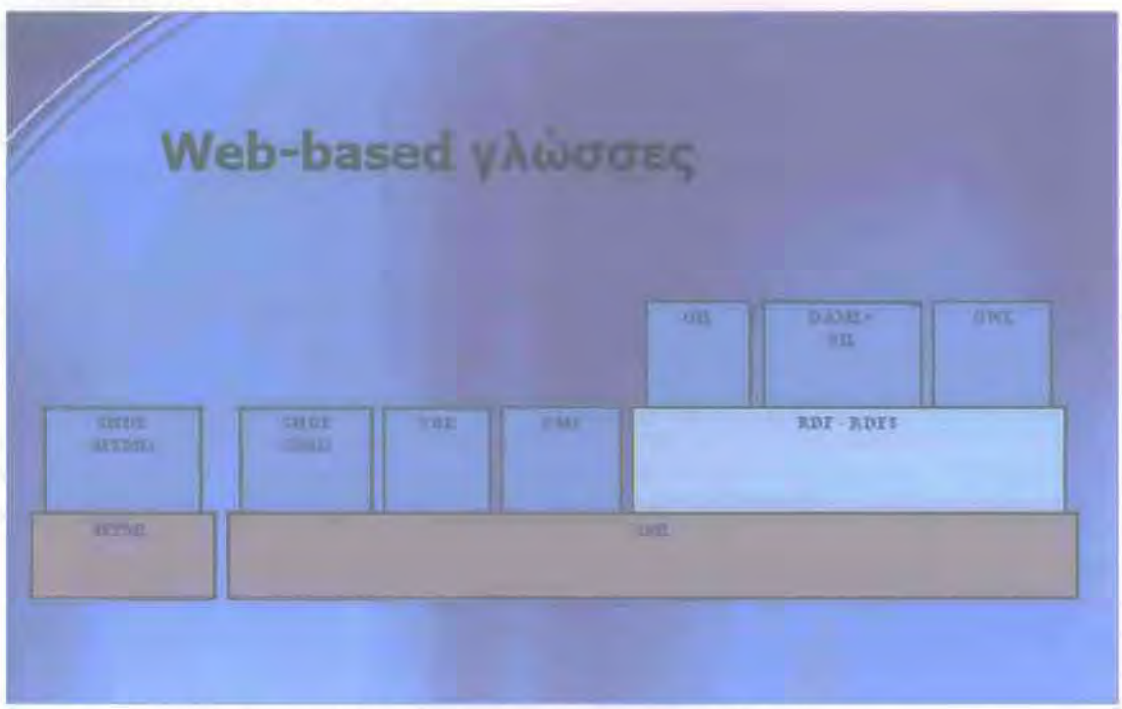
Μηχανική Συστημάτων (systems engineering): Προδιαγραφές επαναχρησιμοποίηση τμημάτων και αξιοπιστία.

4.7 Γλώσσες αναπαράστασης οντολογίας

Η γλώσσα οντολογίας είναι η γλώσσα που χρησιμοποιείται για να την αντιπροσωπεύσει. Οι πιο δημοφιλείς γλώσσες οντολογίας για το σημασιολογικό Ιστό είναι RDF και OWL. Οι σημασιολογικές γλώσσες οντολογίας Ιστού μπορούν να χωριστούν σε δύο μέρη: τα λογικά και πρόσθετος-λογικά. Το λογικό μέρος βασικά αποτελείται από διάφορα λογικά αξιώματα, τα οποία διαμορφώνουν την κατηγορία (έννοια) ορισμοί, ορισμοί ιδιοκτησίας (σχέση), ορισμοί περίπτωσης, κ.λπ.

Το *πρόσθετος-λογικό* μέρος της γλώσσας αποτελείται από τις μη λειτουργικές ιδιότητες (π.χ. όνομα συντακτών, ημερομηνία δημιουργιών, σχόλια φυσικής γλώσσας, πολύγλωσσα) και άλλες πρόσθετες-λογικές δηλώσεις, όπως οι δηλώσεις namespace. Οι μη λειτουργικές ιδιότητες είναι χαρακτηριστικές μόνο για τον ανθρώπινο αναγνώστη, ενώ πολλές από τις άλλες πρόσθετος-λογικές δηλώσεις είναι μηχανό-επεξεργάσιμες.

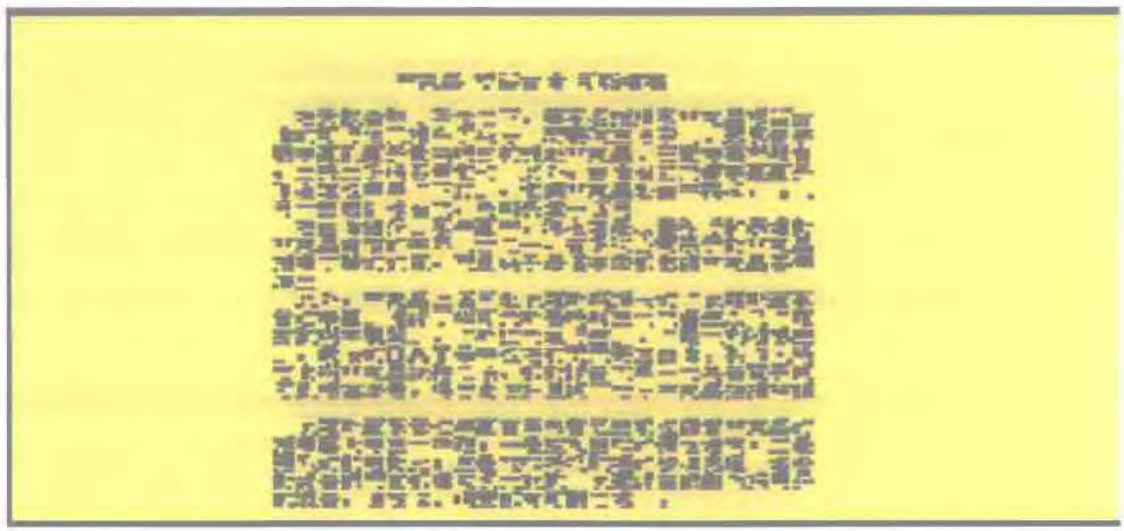
Η γλώσσα οντολογίας είναι η γλώσσα που χρησιμοποιείται για να αντιπροσωπεύσει την οντολογία



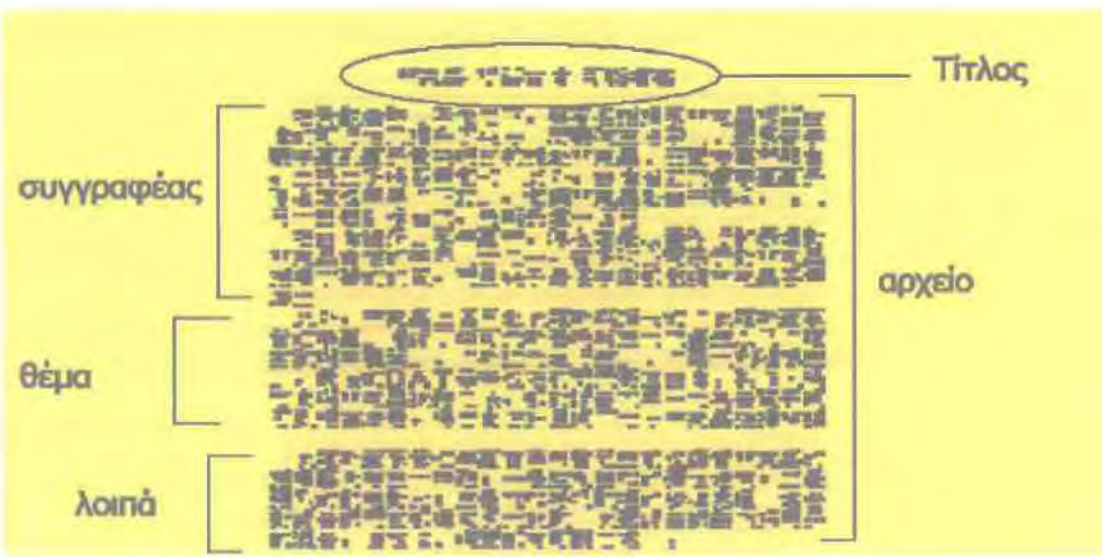
Εικόνα 4.1 Web-based γλώσσες αναπαράστασης οντολογιών

Οι γλώσσες οντολογιών διαφέρουν στη Σύνταξη, στην Ορολογία (Class-concept Instance-object και Slot-property) στην εκφραστικότητα π.χ κάτι που μπορούμε να εκφράσουμε σε μία γλώσσα δεν μπορούμε σε μία άλλη και στην σημασιολογία δηλαδή μια ίδια δήλωση μπορεί να σημαίνει διαφορετικά πράγματα σε διαφορετικές γλώσσες.

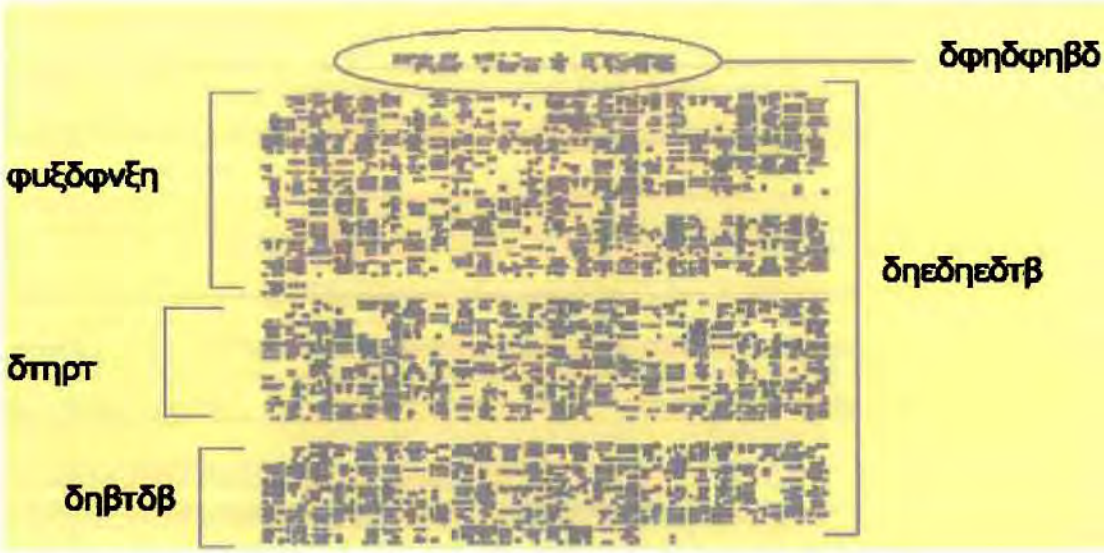
4.8 XML(eXtensible Markup Language)



Εικόνα 4.2Τι βλέπουν οι υπολογιστές



Εικόνα 4.3 Ετικέτες για αναπαράσταση γνώσης



Εικόνα 4.4 XML νόημα μόνο στους ανθρώπους

Η XML (eXtensible Markup Language) είναι ένα απλό και ιδιαίτερα ευέλικτο πρότυπο κειμένου που προήλθε από την Standard Generalized Markup Language (SGML). Αρχικά αναπτύχθηκε για καλύψει τις ανάγκες ευρείας κλίμακας δημοσίευσης δεδομένων αλλά παίζει όλο και μεγαλύτερο ρόλο στην ανταλλαγή ενός μεγάλου εύρους δεδομένων στο διαδίκτυο αλλά και αλλού. Η XML είναι ένας τρόπος για περιγραφή δεδομένων και ένα αρχείο XML μπορεί να περιέχει και τα δεδομένα που περιγράφει με λογική παρόμοια αυτής των βάσεων δεδομένων. Γλώσσες βασιζόμενες στην XML επιτρέπουν σε προγράμματα να τροποποιούν και να ελέγχουν κείμενα γραμμένα στις γλώσσες αυτές χωρίς να έχουν γνώση της μορφής τους.

Η XML προσφέρει τρόπους βασιζόμενους σε απλό κείμενο για την περιγραφή και εφαρμογή μιας δενδρικής μορφής στα δεδομένα. Είναι μια δηλωτική γλώσσα, δηλαδή τα δεδομένα συνοδεύουν δηλώσεις ή μεταδεδομένα που τα αφορούν. Μια δήλωση έχει την εξής μορφή:

<name attribute="value">content</name>

Αυτή η στοιχειώδης δήλωση ονομάζεται element. Ένα element έχει ένα όνομα (name), πιθανώς κάποια χαρακτηριστικά (attributes) και ένα περιεχόμενο (content). Το περιεχόμενο περικλείεται από δύο ετικέτες (tags) που έχουν συγκεκριμένη δομή. Κύριο χαρακτηριστικό είναι η δυνατότητα δόμησης δενδρικών δομών με την εξής μορφή:

<element>
<child1/>
<child2/>
</element>

Η δυνατότητες της επισύναψης μεταδεδομένων και της δημιουργίας δενδρικών δομών σε συνάρτηση με το γεγονός ότι μπορούν να περιγραφούν οποιαδήποτε δεδομένα κάνει την XML ένα δυνατό εργαλείο για τη δόμηση πιο εκφραστικών δηλωτικών γλωσσών. Ακριβώς αυτά τα χαρακτηριστικά εκμεταλλεύεται και η RDF που είναι το υπόστρωμα της OWL καθώς και ένα χρήσιμο συμπλήρωμά της.

4.9 RDF(Resource Description Framework)

Η Resource Description Framework είναι μια γλώσσα που αναπαριστά πληροφορίες για πηγές (resources) στον Παγκόσμιο Ιστό (World Wide Web). Βασικός σκοπός της είναι ειδικότερα η αναπαράσταση μεταδεδομένων για πηγές του Ιστού όπως ο τίτλος, ο συγγραφέας κλπ. αλλά με επέκταση της έννοιας resource μπορεί να χρησιμοποιηθεί και για πράγματα που μπορούν να περιγραφούν με μοναδικό τρόπο. Η χρήση της RDF συνίσταται σε περιπτώσεις όπου η πληροφορία πρέπει να είναι επεξεργάσιμη από εφαρμογές παρά όταν απλά προβάλλεται σε ανθρώπους. Προσφέρει ένα κοινό πλαίσιο για έκφραση της πληροφορίας ώστε αυτή να είναι ανταλλάξιμη μεταξύ εφαρμογών χωρίς απώλεια νοήματος και κατά συνέπεια πλήρως επαναχρησιμοποιήσιμη.

Ας υποτεθεί ότι κάποιος επιθυμεί να κάνει τη δήλωση ότι μια συγκεκριμένη ιστοσελίδα έχει δημιουργηθεί από κάποιον με το όνομα John Smith. Σε φυσική γλώσσα η δήλωση αυτή θα είχε την ακόλουθη μορφή: ***http://www.example.org/index.html έχει δημιουργό του οποίου η τιμή είναι John Smith***

Κομμάτια της παραπάνω δήλωσης έχουν τονιστεί ώστε να δειχθεί ότι για να περιγραφούν οι ιδιότητες από κάτι πρέπει να υπάρχουν τρόποι ώστε να κατονομαστεί ένα σύνολο πραγμάτων όπως:

Το αντικείμενο που περιγράφει η πρόταση (στο παράδειγμα την ιστοσελίδα http://www.example.org/index.html)

- Μια συγκεκριμένη ιδιότητα αυτού του αντικειμένου (δημιουργός)
- Την τιμή της ιδιότητας αυτής

Στην παραπάνω περίπτωση, το Uniform Resource Locator (URL) της σελίδας χρησιμοποιείται για την ταυτοποίηση, η λέξη «δημιουργός» χρησιμοποιείται για να ονομαστεί η ιδιότητα και αντίστοιχα οι λέξεις «John Smith» για να εκφράσουν την τιμή της.

Άλλες ιδιότητες της ιστοσελίδας μπορούν να εκφραστούν με παρόμοιες προτάσεις σε φυσική γλώσσα:

http://www.example.org/index.html έχει ημερομηνία δημιουργίας της οποίας η τιμή είναι 16 Αυγούστου 1999

http://www.example.org/index.html έχει γλώσσα της οποίας η τιμή είναι αγγλικά

Η RDF στηρίζεται στο σκεπτικό ότι τα υπό περιγραφή πράγματα έχουν ιδιότητες (properties) οι οποίες έχουν τιμές και ότι οι περιγραφές αυτές μπορούν να γίνουν με δηλώσεις σαν τις παραπάνω. Σε όρους της RDF (που είναι αυτοί της κατηγορηματικής αναπαράστασης στη γλωσσολογία) το πράγμα περί του οποίου γίνεται η δήλωση ονομάζεται υποκείμενο (subject) η ιδιότητα που προσδίδεται στο υποκείμενο είναι το κατηγόρημα (predicate) και η τιμή της ιδιότητας αυτής είναι το αντικείμενο (object). Η χρησιμοποιούμενη ορολογία είναι αντίστοιχη με αυτή της γλωσσολογίας. Έχει ιδιαίτερο ενδιαφέρον η μελέτη ενός γλωσσολογικού ορισμού του κατηγορήματος:

«Κατηγόρημα είναι η λέξη ή οι λέξεις που φανερώνουν εκείνο που λέγεται μέσα στην πρόταση για το υποκείμενο»

Υπό τη γλωσσολογική σκοπιά δηλαδή η έννοια του κατηγορούμενου θα έπρεπε να συμπεριλαμβάνει και το αντικείμενο της δήλωσής. Αυτή η διαφορά βέβαια δεν έχει κάποια ιδιαίτερη σημασία γιατί είναι ορολογικού χαρακτήρα.

Επιπλέον θέλω να επισημάνω ότι υπάρχουν κάποιους περιορισμούς της RDF. Αυτοί είναι οι εξής:

- Πολύ αδύναμη σημασιολογική αναπαράσταση,
- δεν περιγράφει καλά το νόημα της πληροφορίας και
- δεν διαθέτει μηχανή συμπερασματολογίας.

4.10 DAML+OIL

Στην DAML+OIL (DAML 2009), κάθε έννοια αντιστοιχεί σε μια κλάση και θεμέλιος λίθος της είναι η δυνατότητα περιγραφής μιας ταξονομίας με τη δημιουργία υποκλάσεων. Σε επόμενο στάδιο είναι δυνατή η περιγραφή σχέσεων μεταξύ των κλάσεων καθώς και η προσάρτηση ιδιοτήτων σε αυτές. Με αυτά τα εργαλεία είναι δυνατή η περιγραφή συνεκτικών οντολογιών που είναι παράλληλα επεξεργάσιμες από υπολογιστή και μπορούν να χρησιμοποιηθούν για reasoning.

Δημιουργήθηκε από την US Defense Advanced Research project agency (DARPA) σε συνεργασία με την EU committee on agent markup languages. Έχουν δημιουργηθεί πολλές οντολογίες με την DAML+OIL και είναι διαθέσιμες σε όλους και βασίζεται στην RDF Schema. Πρόγονος της OWL (standard web ontology language)

4.11 OWL(Web Ontology Language)

Η OWL (Dean & Schreiber, 2004) ως πρότυπο ανταλλαγής έχει σχεδιαστεί ώστε να μπορεί να κωδικοποιεί ένα μεγάλο εύρος οντολογιών. Με την ευελιξία της είναι δυνατή η σταδιακή δόμηση μιας οντολογίας και η σύνθεση διαφορετικών οντολογιών κάτι που απλοποιεί σημαντικά το έργο της περιγραφής ενός πεδίου εφαρμογής. Ιδανική όταν πρέπει οι πληροφορίες που περιλαμβάνονται σε έγγραφα, να επεξεργαστούν από εφαρμογές, σε αντιδιαστολή με τις περιπτώσεις όπου το περιεχόμενο πρέπει να παρουσιαστεί μόνο σε ανθρώπους. Παρέχει περισσότερες ευκολίες για νοηματική και σημασιολογική έκφραση από τις XML, RDF και RDF(S) και, συνεπώς, η OWL υπερβαίνει αυτές τις γλώσσες χρησιμοποιώντας τη δυνατότητά της να αναπαριστά το αναγνώσιμο από μηχανή περιεχόμενο στον Ιστό.

4.12 Άλλες Γλώσσες Περιγραφής Οντολογιών

KIF

Είναι τα αρχικά του Knowledge Interchange Format και είναι δημιούργημα του πανεπιστήμιου του Stanford υπό την καθοδήγηση του Mike Genesereth (Genesereth, 1991). Αν και δεν δημιουργήθηκε για την εξυπηρέτηση οντολογικών εφαρμογών είναι ένα σημαντικό βήμα στην ιστορία της αναπαράστασης και ανταλλαγής γνώσης.

Στηρίζεται στην άλγεβρα πρώτης τάξης (First-order predicate calculus, FOPC ή First-order logic, FOL) που είναι ένα σύστημα μαθηματικής λογικής επέκταση της προτασιακής άλγεβρας (propositional ή sentential logic) που αποτελείται από ένα σύνολο μορφολογικών κανόνων, κανόνων μετασχηματισμού και αξιωμάτων και αλγοριθμικά μπορεί να παράγει θεωρήματα καθώς και να αποφασίζει αν μια πρόταση είναι αφενός σύμφωνη με τους κανόνες και τα αξιώματα και αφετέρου, εάν είναι ορθή, κατά πόσο είναι θεώρημα (προκύπτει από υπάρχοντα αξιώματα) ή είναι αξίωμα (είναι πρωτογενής κανόνας). Η KIF είναι μια γλώσσα για την αναπαράσταση γνώσης που έχει σαν στόχο την ανάπτυξη ενός εκφραστικού, ευέλικτου και κατανοητού από τον άνθρωπο και τον υπολογιστή μέσου για την ανταλλαγή γνωσιακών βάσεων. Η ύπαρξη μιας τέτοια γλώσσας σημαίνει ότι κάθε πληροφοριακό σύστημα, υπό την προϋπόθεση ότι η σύνταξή του είναι μετατρέψιμη σε αυτή της KIF, είναι σε θέση να διαχειρίζεται εσωτερικά δεδομένα με τον δικό του τρόπο με την εγγύηση όμως ότι τα αποτελέσματα των δράσεών του θα είναι συμβατά με αυτά άλλων συστημάτων που με τη σειρά τους είναι συμβατά με την KIF. Η γλώσσα έχει τρία βασικά χαρακτηριστικά: α) ένα δεδομένο σετ περιγραφικών σημασιολογικών κανόνων β) λογική περιεκτικότητα, δηλαδή όλες τις εκφραστικές δυνατότητες της FOPC γ) την δυνατότητα να αναπαριστά αναπαραστάσεις ή με άλλα λόγια να εκφράζει γνώση για τη γνώση. Στην KIF υπάρχει ένα σύνολο αντικειμένων, που αποτελούν το πεδίο εφαρμογής (π.χ. φυσικά πρόσωπα), ένα σύνολο ιδιοτήτων που προσδίδονται στα αντικείμενα και τους παρέχουν σημασιολογικό περιεχόμενο καθώς και ένα σύνολο σχέσεων μεταξύ των αντικειμένων που παρέχουν το οντολογικό πλαίσιο. Έτσι με την KIF και με χρήση της FOPC δίδεται η δυνατότητα να περιγραφούν αντικείμενα με τον απαραίτητο φορμαλισμό για την εξαγωγή λογικών συμπερασμάτων στηριζόμενων σε ένα σύνολο ορισμένων αξιωμάτων (π.χ. τα θηλαστικά είναι ζώα, ο ελέφαντας είναι θηλαστικό όπου η σχέση «είναι» εκφράζει ένα αξίωμα

ανάμεσα σε ιδιότητες και με χρήση της FOPC προκύπτει ότι ο ελέφαντας είναι θηλαστικό).

ONTOLINGUA

Με βάση την KIF, το Stanford Research Institute ανέπτυξε μια πιο χρηστική γλώσσα για την αναπαράσταση οντολογιών με το όνομα Ontolingua. Η Ontolingua έχει δομηθεί με βάση την KIF 3.0 αλλά έχει έναν ιδιαίτερο στόχο. Ενώ πρόθεση της KIF είναι να παρέχει μια διεπαφή ανάμεσα σε συστήματα αναπαράστασης γνώσης, στόχος της Ontolingua είναι να παρέχει μια διεπαφή ανάμεσα σε οντολογίες. Προσφέρει ένα περιβάλλον και ένα σύνολο εργαλείων σχεδιασμένων να επιτρέψουν σε ετερογενείς οντολογίες να συνενωθούν υπό μια ενιαία πλατφόρμα δια μέσω της μετάφρασης σε μια κοινή γλώσσα. Η Ontolingua προσθέτει στον αρχικό κορμό της KIF χαρακτηριστικά που επιτρέπουν την αναπαράσταση δομικών συνιστωσών μιας οντολογίας. Πάνω απ' όλα προσφέρει μια επέκταση στη μεταχείριση των σχέσεων (με μια σχεδόν αποκλειστική εστίαση στις δυαδικές σχέσεις νοούμενων ως συνόλων δυαδικών λιστών), καθώς και την εισαγωγή της έννοιας της κλάσης. Οι κλάσεις ορίζονται σαν μοναδικές σχέσεις (δηλαδή σαν σύνολο λιστών με μήκος ένα) και τα μέλη των λιστών είναι απαραιτήτως «άτομα» που αποκαλούνται instances. Επιπροσθέτως, η Ontolingua συμπεριλαμβάνει την έννοια του Individual-Thing (κλάσεις αντικειμένων που δεν είναι σύνολα αλλά στοιχεία ενός συνόλου).

5 Εργαλεία Οντολογιών

5.1 Εργαλεία ανάπτυξης Οντολογιών

Τα πιο γνωστά λογισμικά εργαλεία που χρησιμοποιούνται για την ανάπτυξη οντολογιών (Ribiere and Charlton, 2000) είναι: OilEd (http://kaon.semanticweb.org/) από το University of Manchester, OntoSaurus (http://www.isi.edu/isd/ontosaurus.html) από το ISI (Information Sciences Institute) του USC (University of Southern California), KAON (http://kaon.semanticweb.org/) και OntoEdit (http://ontoserver.aifb.unikarlsruhe.de/ontoedit/) και από το Universitat Karlsruhue, Protege2000 (http://protege.stanford.edu/) από το SMI (Stanford Medical Informatics) του Stanford University κ.α.

Εργαλεία αναπαράστασης και οργάνωσης οντολογιών

Ontology Development Tools:

Ontology Editors and browsers

Ontology graphical edition

Ontology Translators

Ontology library management

Ontology merge and alignment tools

Ontology based annotation tools

Ontology querying tools and inference engines

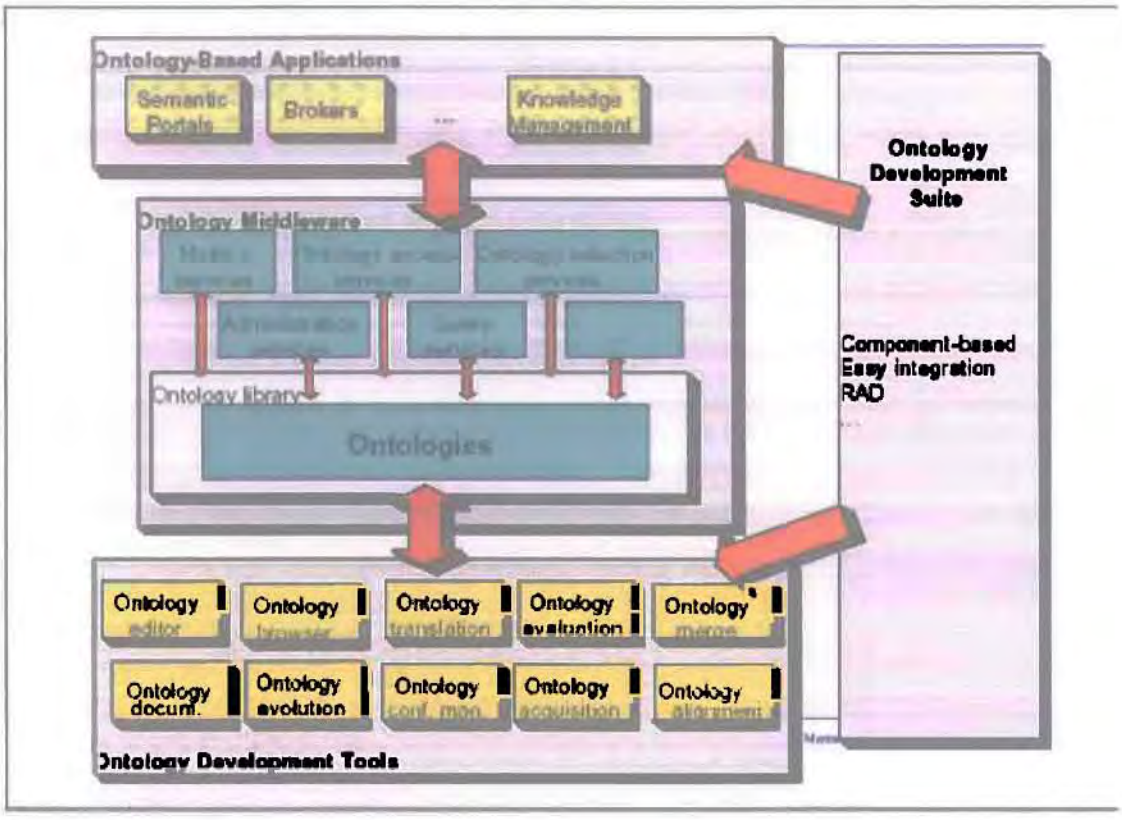
Ontology learning tools

Ontology documentation

Ontology population

Ontology evaluation

Ontology Evolution



Εικόνα 5.1 Εργαλεία οντολογιών

Ontology management (διαχείριση οντολογίας) είναι το σύνολο μεθόδων και τεχνικών που είναι απαραίτητο να χρησιμοποιήσουν αποτελεσματικά τις πολλαπλάσιε παραλλαγές των οντολογιών από ενδεχομένως διαφορετικές πηγές για τους διαφορετικούς στόχους.

Ontology modification (τροποποίηση οντολογίας) προσαρμόζεται όταν ένα σύστημα διαχείρισης οντολογίας επιτρέπει τις αλλαγές στην οντολογία που είναι σε χρήση, χωρίς εξέταση της συνέπειας.

Ontology versioning (Υποστήριξη πολλαπλών εκδόσεων) Ύπαρξη πολλαπλών διαφορετικών παραλλαγών μιας οντολογίας σε κάποιο περιβάλλον.

Παραλλαγές :

α) τροποποιήσεις σε μια ήδη υπάρχουσα οντολογία.

β) διαφορετικές εκδόσεις οντολογιών του ίδιου τομέα

ανεξάρτητα ανεπτυγμένες.

Η δυνατότητα να χειριζόμαστε αλλαγές στις οντολογίες με τη δημιουργία και διαχείριση διαφορετικών παραλλαγών. Δημιουργία μιας μεθοδολογίας που θα

διακρίνει και θα αναγνωρίζει εκδόσεις, διαδικασίες για ενημερώσεις και αλλαγές στις οντολογίες, ώστε να παρακολουθούνται οι σχέσεις ανάμεσα στις διάφορες εκδόσεις.

Αιτίες που οδηγούν σε αλλαγές στις οντολογίες:

- *Αλλαγές στον τομέα*: Ένα παράδειγμα τέτοιου είδους αλλαγής είναι η συγχώνευση (merging) δύο τμημάτων κάποιου πανεπιστημίου (πρόκειται για μία αλλαγή στον πραγματικό κόσμο, η οποία επιβάλει να τροποποιείται (αυτόματα) και η οντολογία που περιγράφει τον τομέα αυτό).
- *Αλλαγές στη διαμοιρασμένη εννοιολογική αναπαράσταση*: Μία διαμοιρασμένη εννοιολογική αναπαράσταση ενός τομέα είναι δυνατόν να αλλάξει και εξαιτίας της προοπτικής της χρήσης της. Διαφορετικές εργασίες μπορούν να υποδηλώσουν διαφορετικές οπτικές γωνίες σε ένα τομέα και συνεπώς μία διαφορετική αντίληψη. Όταν μια οντολογία προσαρμόζεται για μια νέα εργασία ή ένα νέο πεδίο, οι αλλαγές που γίνονται απεικονίζουν μεταβολές στην αναπαράσταση.
- *Αλλαγές στις προδιαγραφές*: Μία αλλαγή στον προσδιορισμό (προδιαγραφές) μιας οντολογίας είναι ένα είδος μετάφρασης π.χ. μία αλλαγή στον τρόπο με τον οποίο μία αναπαράσταση καταγράφεται επίσημα.

Οι διάφορες αλλαγές στις οντολογίες επιφέρουν κάποιες επιπτώσεις. Παρακάτω υπάρχουν μερικές επιπτώσεις αλλαγών

Ασυμβατότητες:

α) ως προς τα δεδομένα (π.χ. ιστοσελίδες). Όταν μια οντολογία μεταβάλλεται, τα δεδομένα αυτά μπορεί να λάβουν διαφορετική ερμηνεία ή να χρησιμοποιηθούν άγνωστοι όροι.

β) ως προς άλλες οντολογίες, οι οποίες χρησιμοποιούν την τροποποιημένη οντολογία. Αλλαγές στην αρχική οντολογία θα μπορούσαν να επηρεάσουν και την τελική.

γ) ως προς ορισμένες εφαρμογές που χρησιμοποιούν μια οντολογία δυσλειτουργία.

Κατά την ανάπτυξη των παραπάνω βημάτων θα πρέπει να ελέγχεται κατά πόσο η αναπτυσσόμενη οντολογία πληρεί συγκεκριμένα κριτήρια σχεδιασμού. Επίσης θα πρέπει και μετά τη σχεδίαση και να αξιολογείται, ελέγχοντας αν ανταποκρίνεται σε μια σειρά από κριτήρια (ερωτήματα) ικανότητας:

Ontology mediation Η μεσολάβηση οντολογίας είναι η διαδικασία εύρεσης των διαφορών μεταξύ ετερογενών οντολογιών προκειμένου να επιτευχθεί η λειτουργικότητα μεταξύ των στοιχείων. Αυτό περιλαμβάνει την ανακάλυψη και προδιαγραφή *των χαρτογραφήσεων οντολογίας*, καθώς επίσης και τη χρήση αυτών

των χαρτογραφήσεων για ορισμένους στόχους, όπως το ξαναγράψιμο ερώτησης και ο μετασχηματισμός περίπτωσης.

Ontology Mapping Η οντολογία που χαρτογραφεί, μια χαρτογράφηση οντολογίας είναι μια (δηλωτική) προδιαγραφή της σημασιολογικής επικάλυψης μεταξύ δύο οντολογιών. Αυτή η χαρτογράφηση μπορεί να είναι μονόδρομη (injective) ή διπλής κατεύθυνσης (bijective)

Mapping Pattern Σχέδιο χαρτογράφησης, αν και δεν χρησιμοποιείται συχνά στις τρέχουσες προσεγγίσεις στη μεσολάβηση οντολογίας, τα σχέδια μπορούν να διαδραματίσουν έναν σημαντικό ρόλο στην προδιαγραφή των χαρτογραφήσεων οντολογίας, επειδή έχουν τη δυνατότητα να καταστήσουν τις χαρτογραφήσεις με πιο συνοπτικό, καλύτερα κατανοητό και με μικρότερο αριθμό λαθών. Ένα *σχέδιο χαρτογράφησης* μπορεί να φανεί σαν πρότυπο για τις χαρτογραφήσεις που εμφανίζονται πολύ συχνά. Τα σχέδια μπορούν να κυμανθούν σε πολύ απλά (π.χ. μια χαρτογράφηση μεταξύ μιας έννοιας και μιας σχέσης) σε πολύ σύνθετο, στο οποίο το σχέδιο συλλαμβάνει τις περιεκτικές υποδομές των οντολογιών, οι οποίες είναι σχετικός με έναν ορισμένο τρόπο.

Ontology Merging (Noy & Musen 2000) Οντολογία που συγχωνεύει δημιουργώντας μια νέα οντολογία από δύο ή περισσότερες οντολογίες, η νέα οντολογία θα ενοποιήσει και θα αντικαταστήσει τις αρχικές.

Ontology Aligning Οντολογία που ευθυγραμμίζει φέρνοντας τις οντολογίες στην αμοιβαία συμφωνία Εδώ, οι οντολογίες κρατιούνται χωριστά, αλλά τουλάχιστον μια από τις αρχικές οντολογίες προσαρμόζεται έτσι ώστε η σύλληψη και το λεξιλόγιο να ταιριάζουν με στα επικαλύπτοντας μέρη των οντολογιών.

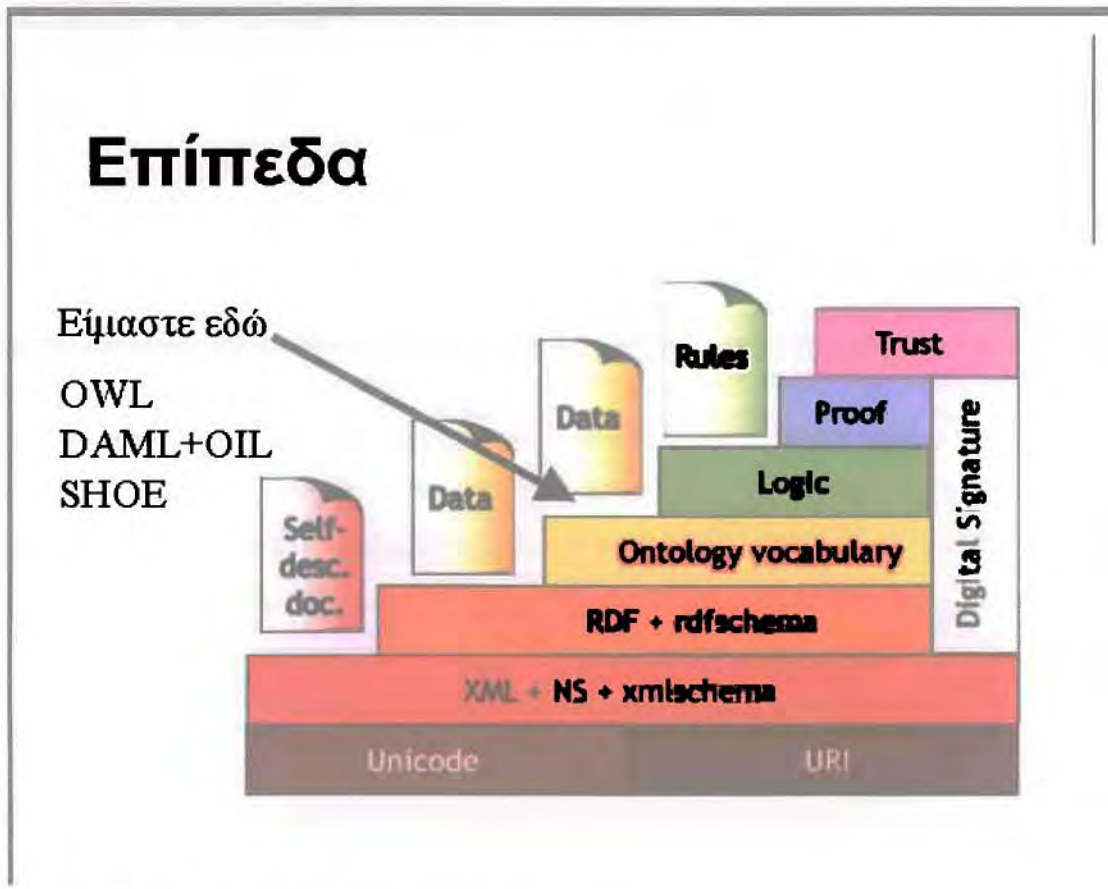
Ontology-level Mismatches οι κακοί συνδυασμοί στο επίπεδο οντολογίας περιλαμβάνουν τους λάθος συνδυασμούς κατά την έννοια ή κωδικοποίηση των εννοιών στις διαφορετικές οντολογίες. Αυτοί οι κακοί συνδυασμοί εμφανίζονται όταν χρησιμοποιούνται διαφορετικά παραδείγματα για την ερμηνεία της ίδιας έννοια.

One-to-one integration vs. Global integration (Noy, Klein 2003): Σε οποιοδήποτε σύστημα ολοκλήρωσης στοιχείων (ή οντολογίας) είναι ενδιαφέρων να φανεί πως διαφορετικές οντολογίες συσχετίζονται πραγματικά η μια με την άλλην. Διακρίνουμε δύο ευδιάκριτες περιπτώσεις. Στην πρώτη περίπτωση, υπάρχει μια ένα προς ένα σχέση(One-to-one integration) μεταξύ των οντολογιών, δηλ. κάθε ζευγάρι των οντολογιών που είναι ενσωματωμένο έχει μια χαρτογράφηση μεταξύ τους, ενώ

στη δεύτερη περίπτωση (Global integration), η ολοκλήρωση πραγματοποιείται μέσω μιας σφαιρικής οντολογίας, η οποία χαρτογραφείται σε όλες τις τοπικές οντολογίες.

5.2 Η έννοια της εξέλιξης οντολογιών

Η εξέλιξη οντολογίας ορίζεται ως η έγκαιρη προσαρμογή της οντολογίας στις συνεχώς μεταβαλλόμενες απαιτήσεις και τη συνεχή εξάπλωση των αλλαγών στα εξαρτώμενα δημιουργήματα.



Εικόνα 5.2 Που βρισκόμαστε σήμερα

Η σημασιολογική πληροφορία που είναι συσχετισμένη με πηγές στο διαδίκτυο (ως μεταδεδομένα) . Πρέπει να ερμηνευτεί και να συνδυαστεί από τις μηχανές παραγωγή λογικής. Μία πλατφόρμα για μηχανές εύρεσης και έξυπνους πράκτορες. Το σημερινό διαδίκτυο πρέπει να εξελιχθεί έτσι ώστε η πληροφορία να μπορεί να επεξεργαστεί από ανθρώπους και μηχανές. Ο σημασιολογικός ιστός δεν αντικαθιστά το σημερινό παγκόσμιο ιστό. Το διαδίκτυο έχει ήδη δισεκατομμύρια σελίδες.

Πρέπει όμως όλες οι σελίδες να είναι σημασιολογικές?

Ο σημασιολογικός ιστός κάθεται δίπλα στο σημερινό παγκόσμιο ιστό.

Εξέλιξη και διατήρηση οντολογιών γίνετε με τη μέθοδο OntoClean. Η δημιουργία οντολογιών είναι μία επαναληπτική και δυναμική διαδικασία. Οι οντολογίες πρέπει να έχουν τη δυνατότητα να εξελιχθούν για πολλούς λόγους: Οι ανάγκες των εφαρμογών και των χρηστών αλλάζουν. Το σύστημα μπορεί να βελτιωθεί. Η δημιουργία των οντολογιών είναι δαπανηρή και η εξέλιξη τους είναι ακόμα πιο δαπανηρή. Σκοπός (σε κατανεμημένα περιβάλλοντα) είναι η αύξηση της γνώσης σε κάθε κόμβο ενός ανοιχτού κατανεμημένου συστήματος, με την απόκτηση περιγραφών πηγών από τις οντολογίες των άλλων κόμβων.

Περιλαμβάνει 6 στάδια:

1) εντοπισμός αλλαγών
2) αναπαράσταση αλλαγών
3) σημασιολογία αλλαγών
4) διάδοση αλλαγών
5) υλοποίηση των αλλαγών
6) επικύρωση αλλαγών

Το OntoView είναι ένα εργαλείο, το οποίο παρέχει υποστήριξη για την εξέλιξη οντολογιών που είναι διαθέσιμες on-line στο διαδίκτυο. Η κύρια λειτουργία του στηρίζεται στην προσπάθεια του συστήματος να βοηθήσει το χρήστη να διαχειρίζεται τις εκάστοτε αλλαγές που παρουσιάζονται στις οντολογίες, αλλά κυρίως επιδιώκει να επιτύχει τη διαλειτουργικότητα μεταξύ των διαφόρων εκδόσεων μιας οντολογίας, με το να συγκρίνει τις διάφορες εκδόσεις και να τονίζει τις διαφορές μεταξύ τους. Στη συνέχεια, επιτρέπει στο χρήστη να προσδιορίζει την εννοιολογική σχέση ανάμεσα στις διαφορετικές εκδόσεις των εννοιών. Επιπρόσθετα, το σύστημα λειτουργεί ως αποθηκευτικό μέσο για εκδόσεις οντολογιών, παρέχοντας μια διαφανή διεπαφή για την επικοινωνία μεταξύ τυχαίων εκδόσεων των οντολογιών.

5.3 Έρευνα υπαρχόντων βιβλιοθηκών οντολογιών

Παρακάτω, ακολουθεί μια παρουσίαση ορισμένων βιβλιοθηκών οντολογιών(εικόνα 5.3):

Εικόνα 5.3 Παρουσίαση ορισμένων βιβλιοθηκών οντολογιών

1. WebOnto: Το WebOnto είναι μια βιβλιοθήκη οντολογιών, η οποία αναπτύχθηκε από το Knowledge Media Institute του Open University (UK). Σχεδιάστηκε για να υποστηρίξει τη συνεργατική δημιουργία, ανάγνωση και επεξεργασία των οντολογιών. Παρέχει μια διεπαφή άμεσου χειρισμού, η οποία εκθέτει οντολογικές εκφράσεις και επιπλέον ένα εργαλείο "συνομιλίας" με οντολογίες (ontology discussion tool), το οποίο ονομάζεται Tadzebao και είναι ικανό να υποστηρίζει ταυτόχρονα και σύγχρονες και ασύγχρονες "συνομιλίες". Οι οντολογίες που αποθηκεύονται στο WebOnto δεν είναι ταξινομημένες σύμφωνα με κάποιες υπάρχουσες κατηγορίες, αλλά διαιρούνται σε μικρές μονάδες. Προσδιορίζονται μοναδικά με το όνομά τους και απεικονίζονται σε γραφικό περιβάλλον. Εκφράζονται με τη γλώσσα αναπαράστασης OCML, η οποία υποστηρίζει εξαγωγή συμπερασμάτων βάσει κανόνων. Το σύστημα δεν υποστηρίζει πολλαπλές εκδόσεις.

2. OntoLingua: Το OntoLingua αναπτύχθηκε στις αρχές της δεκαετίας του '90 στο Knowledge Systems Laboratory του Stanford University. Αποτελείται από έναν εξυπηρετητή και μια γλώσσα αναπαράστασης. Ο εξυπηρετητής παρέχει ένα αποθετήριο οντολογιών (βιβλιοθήκη οντολογιών), προκειμένου να βοηθήσει τους χρήστες να δημιουργούν νέες οντολογίες και να τροποποιούν και να βελτιώνουν τις ήδη υπάρχουσες συνεργατικά. Η οντολογία που αποθηκεύεται στον εξυπηρετητή

είναι δυνατόν να μετατραπεί σε διαφορετικές διατάξεις. Οι οντολογίες που αποθηκεύονται στο OntoLingua δεν ταξινομούνται σύμφωνα με κάποιες πρότυπες κατηγορίες αλλά οργανώνονται με βάση τη θεωρία του πλέγματος (lattice theory), κατά την οποία κάθε οντολογία συμπεριλαμβάνει τις οντολογίες εκείνες, κάτω από τις οποίες βρίσκεται αυτή τοποθετημένη με εσοχή. Κάθε οντολογία έχει ένα μοναδικό όνομα που τη διακρίνει από τις άλλες οντολογίες. Το Ontolingua δεν παρέχει τη δυνατότητα υποστήριξης πολλαπλών εκδόσεων.

3. DAML: Η βιβλιοθήκη οντολογιών DAML είναι μέρος του DARPA Agent Markup Language (DAML) Program, το οποίο ξεκίνησε επισήμως τον Αύγουστο του 2000. Στόχος αυτής της προσπάθειας είναι η ανάπτυξη μιας γλώσσας και ορισμένων εργαλείων που θα διευκολύνουν την έννοια του Σημασιολογικού Ιστού. Η βιβλιοθήκη οντολογιών περιλαμβάνει έναν κατάλογο, ο οποίος είναι διαθέσιμος σε XML, HTML και DAML μορφές. Οι οντολογίες ταξινομούνται σύμφωνα με το Open Directory Category (www.dmoz.org) και προσδιορίζονται μοναδικά σύμφωνα με τους αναγνωριστές του και το URI. Το σύστημα δε διαθέτει λειτουργίες υποστήριξης πολλαπλών εκδόσεων. Επίσης, δεν υπάρχουν ιδιαίτερες δυνατότητες αναζήτησης, ούτε ιδιαίτερες λειτουργίες επεξεργασίας και συλλογιστικής
υποστήριξης.

4. Shoe: Η Shoe (Simple HTML Ontology Extensions) αναπτύχθηκε από το University of Maryland (USA) και είναι η πρώτη γλώσσα για τη σημασιολογία του Ιστού που αναπτύχθηκε ως γλώσσα σήμανσης. Η βιβλιοθήκη περιλαμβάνει λίστες οντολογιών, οι οποίες ευρετηριάζονται αλφαβητικά και ταξινομούνται ως προς την εξάρτηση της οντολογίας, με την αυστηρή δομή δέντρου. Κάθε οντολογία αναγνωρίζεται μοναδικά από το όνομά της. Το σχήμα παραλλαγών της Shoe είναι απαραίτητο για το χειρισμό διαφόρων ειδών αναθεωρήσεων. Διατηρεί κάθε έκδοση της οντολογίας ως μία χωριστή ιστοσελίδα και ένα στιγμιότυπο πρέπει να δηλώνει με ποια έκδοση συμφωνεί. Γι' αυτό το λόγο, οι πηγές δεδομένων μπορούν να αναβαθμιστούν σε μια νέα οντολογία. Η SHOE ήταν το πρώτο σύστημα που εστίασε στο πρόβλημα της εξέλιξης και συνέχειας των οντολογιών.

5. Vub Ontology Server: Ο εξυπηρετητής οντολογιών VUB, που αναπτύχθηκε από το Vrije Universiteit Brussels, συνδέει την τεχνολογία οντολογιών με τη σημασιολογία των βάσεων δεδομένων, εγκαθιστώντας τεχνικές από τις βάσεις

δεδομένων για τη διαχείριση και την κατανόηση των οντολογιών. Το σύστημα διαχείρισης βάσεων δεδομένων (DBMS) είναι εξοπλισμένο με ποικίλες συντακτικές δομές και δίνει τη δυνατότητα στα σχήματα βάσεων να παρουσιάζουν αντικείμενα, ταξινομίες, περιορισμούς ακεραιότητας, και κανόνες παραγωγής. Στην τρέχουσα φάση ανάπτυξης του εξυπηρετητή δεν υποστηρίζεται η δυνατότητα υποστήριξης πολλαπλών εκδόσεων και συλλογισμού. Οι οντολογίες περιγράφονται σε XML. Τέλος, δεν υιοθετείται κάποια πρότυπη οντολογία στον εξυπηρετητή.

5.4 Εργαλεία ανάπτυξης οντολογιών

Η υποστήριξη των διαφόρων εργαλείων για οντολογίες είναι διαθέσιμη για όλα τα στάδια του κύκλου ζωής μιας οντολογίας. Πολλά εργαλεία προσφέρονται ακόμα ως ερευνητικά πρωτότυπα, αλλά υπάρχουν και πολλά εμπορικά (είναι συχνά εμπορικές εκδόσεις των άμεσων ερευνητικών αντίστοιχών τους). Η υποστήριξη των προτύπων RDF, OWL κ.λ.π. κερδίζει έδαφος συνεχώς. Εντούτοις, λόγω της ύπαρξης διαφορετικών μορφών του RDF(S), τα εργαλεία οντολογιών δεν έχουν ακόμα καλή διαλειτουργικότητα.

Σήμερα, τα εργαλεία ανάπτυξης οντολογιών απαιτούν από τους χρήστες τους να είναι εκπαιδευμένοι στα αντικείμενα της αναπαράστασης γνώσης (knowledge representation) και της κατηγορηματικής λογικής (predicate logic). Η υποστήριξη για ημιαυτοματισμούς (π.χ. εξαγωγή όρων) είναι ακόμη στο στάδιο της ωρίμανσης. Η υποστήριξη για τη συνεργατική συγγραφή (collaborative authoring) και ανάπτυξη είναι ακόμα αδύνατη.

5.5 Κύκλος ζωής οντολογιών και υποστήριξη μέσω εργαλείων

Ο κύκλος ζωής μιας οντολογίας (εικόνα 5.4) εκτείνεται από τη δημιουργία έως στην εξέλιξη όπως φαίνεται παρακάτω

Εικόνα 5.4 Κύκλος ζωής οντολογιών

Η υποστήριξη μέσω εργαλείων είναι διαθέσιμη για όλα τα στάδια του κύκλου ζωής :

Δημιουργία - Μπορεί να ξεκινήσει από το μηδέν, χρησιμοποιώντας ένα εργαλείο για δημιουργία και επεξεργασία δομών κλάσεων. Εντούτοις, υπάρχει πολλή διαθέσιμη βοήθεια σε αυτό το στάδιο :

- Μπορεί να χρησιμοποιηθεί η εξόρυξη κειμένων (text mining) για να εξάγει ορολογία από κείμενα, παρέχοντας έτσι ένα σημείο εκκίνησης για τη δημιουργία οντολογιών.
- Πολλές φορές, η πληροφορία της οντολογίας είναι διαθέσιμη σε κληροδοτημένη μορφή (legacy form), όπως είναι τα σχήματα βάσεων δεδομένων, οι κατάλογοι προϊόντων, και οι χρυσοί οδηγοί. Πολλοί από τους πρόσφατους επεξεργαστές οντολογιών μπορούν να κάνουν εισαγωγή σχημάτων από βάσεις δεδομένων και άλλων κληροδοτημένων μορφών.
- Είναι επίσης πιθανό να επαναχρησιμοποιηθούν, γενικά ή εν μέρει, οντολογίες που έχουν ήδη αναπτυχθεί στη δημιουργία μιας νέας οντολογίας. Αυτό έχει ως πλεονέκτημα τη δυνατότητα να γίνει εκμετάλλευση και επέκταση λεπτομερούς εργασίας που έχει ήδη γίνει από έναν άλλο μηχανικό οντολογιών.

Εισαγωγή εννοιών (δημιουργία πληθυσμού) - Αναφέρεται στη διαδικασία δημιουργίας στιγμιότυπων των εννοιών μιας οντολογίας, και στη σύνδεση τους με εξωτερικές πηγές :

- Οι συνηθισμένες ιστοσελίδες μπορεί να είναι μια καλή πηγή πληροφοριών – υπάρχουν πολλά εργαλεία δημιουργίας πληθυσμού τα οποία είναι βασισμένα στο σχολιασμό (annotation) που περιέχουν οι ίδιες οι ιστοσελίδες.
- Κληροδοτημένες πηγές στιγμιότυπων είναι επίσης συχνά διαθέσιμες : κατάλογοι προϊόντων, λίστες εξαρτημάτων, λευκές σελίδες (white pages), πίνακες βάσεων δεδομένων, κ.λ.π. μπορούν όλα να εξορυχτούν κατά τη δημιουργία πληθυσμού μιας οντολογίας. Η δημιουργία του πληθυσμού μπορεί να γίνει είτε χειροκίνητα είτε με ημιαυτόματο τρόπο. Ο ημιαυτόματος τρόπος συστήνεται ανεπιφύλακτα όταν υπάρχει μεγάλος αριθμός και όγκος πηγών γνώσης.

Ανάπτυξη - Υπάρχουν πολλοί τρόποι ανάπτυξης μιας οντολογίας αφού δημιουργηθεί και αποκτήσει πληθυσμό :

- Η οντολογία παρέχει ένα φυσικό ευρετήριο των στιγμιότυπων που περιγράφονται σε αυτή, και ως εκ τούτου μπορεί να χρησιμοποιηθεί ως βοηθός πλοήγησης κατά τη διάρκεια περιήγησης των στιγμιότυπων.
- Περισσότερο πολύπλοκες μέθοδοι, όπως η μέθοδος case-based reasoning (CBR), μπορούν να χρησιμοποιήσουν την οντολογία για να τροφοδοτήσουν με μέτρα ομοιότητας (similarity measures) μια case-based ανάκτηση.
- Οι DAML+OIL και OWL έχουν δυνατότητες έκφρασης αξιωμάτων και περιορισμών στις έννοιες της οντολογίας - ως εκ τούτου μπορούν να χρησιμοποιηθούν ισχυρές μηχανές logical reasoning για να βγάλουν συμπεράσματα για τα στιγμιότυπα μιας οντολογίας.

Η σημασιολογική ολοκλήρωση σε διάφορες εφαρμογές είναι πιθανώς το γρηγορότερα εξελισσόμενο πεδίο της ανάπτυξης για τα σύστημα βασισμένα σε οντολογίες.

Επικύρωση, Εξέλιξη και Διατήρηση - Οι οντολογίες, όπως οποιοδήποτε άλλο συστατικό ενός σύνθετου συστήματος, είναι απαραίτητο να αλλάζουν καθώς το περιβάλλον τους αλλάζει. Μερικές αλλαγές μπορεί να γίνουν απλά για να καλύψουν κάποια σφάλματα ή παραλείψεις της αρχικής οντολογίας, άλλες για να προσαρμοστεί η οντολογία σε μια αλλαγή του περιβάλλοντος. Υπάρχουν πολλοί τρόποι με τους οποίους μια οντολογία μπορεί να επικυρωθεί προκειμένου να βελτιωθεί και να εξελιχθεί - οι αποτελεσματικότερες κριτικές είναι βασισμένες σε αυστηρά επίσημη σημασιολογία του τι σημαίνει η δομή κλάσης :

- Έχουν αναπτυχθεί εκτεταμένα logical frameworks που υποστηρίζουν αυτό το είδος reasoning, τα οποία ονομάζονται "Description Logics".
- Μερικά προηγμένα εργαλεία χρησιμοποιούν αυτοματοποιημένες μηχανές περιγραφικής λογικής για να καθορίσουν πότε μια οντολογία έχει αντιφάσεις, ή πότε μια συγκεκριμένη έννοια σε μια οντολογία μπορεί να ταξινομηθεί διαφορετικά, σύμφωνα με την περιγραφή της και την περιγραφή άλλων εννοιών.
- Αυτές οι κριτικές μπορούν να χρησιμοποιηθούν για να προσδιορίσουν τις ελλείψεις στη γνώση που αναπαρίσταται στην οντολογία, ή μπορούν να χρησιμοποιηθούν για να τροποποιήσουν αυτόματα την οντολογία, ενοποιώντας τις πληροφορίες που περιλαμβάνονται μέσα σε αυτή.

Η συντήρηση μιας οντολογίας μπορεί να απαιτήσει τη συγχώνευση οντολογιών διαφόρων προελεύσεων. Όταν συμβαίνει αυτό τότε είναι σημαντική η υποστήριξη μέσω εργαλείων :

- Μερικά εργαλεία παρέχουν ανθρωποκεντρικές δυνατότητες έρευνας μέσω των οντολογιών για παρόμοιες έννοιες (συνήθως στο όνομα), και πρόνοια για τη συγχώνευση των εννοιών.
- Άλλα εκτελούν πιο πολύπλοκο ταίριασμα, βασιζόμενα σε κοινά στιγμιότυπα ή πρότυπα συσχετιζόμενων εννοιών.

5.6 Εργαλεία οντολογιών

Αφού αναλύσαμε τα στάδια του κύκλου ζωής μιας οντολογίας, είναι θεμιτό να εξετάσουμε τα διαθέσιμα εργαλεία για οντολογίες. Τα εργαλεία που κυκλοφορούν χρησιμοποιούνται σε ένα ή περισσότερα στάδια του κύκλου ζωής της οντολογίας (πολλές φορές και σε όλα).

Πιο κάτω παρουσιάζεται ένας κατάλογος αντιπροσωπευτικών εργαλείων διαθέσιμων σήμερα που χρησιμοποιούνται στο στάδιο της ανάπτυξης, και μια συνοπτική περιγραφή των δυνατοτήτων τους. Έχει υπάρξει μια σημαντική αύξηση του αριθμού προϊόντων για οντολογίες. Δεν αναφέρονται όλα τα διαθέσιμα εργαλεία για ανάπτυξη οντολογιών. Στη δημιουργία αυτού του πίνακα επιλέχτηκαν τα εργαλεία που :

- Υποστηρίζουν όλα ή μερικά από τα πρότυπα RDF(S) – DAML+OIL/OWL.

- Έχουν ισχυρό τεχνικό όραμα για λύσεις βασισμένες σε οντολογίες.
- Είναι συμπαγή και έτοιμα για χρήση.

Εργαλείο: Orenge.

Δυνατότητες: Εκτελεί αναζήτηση σε φυσική γλώσσα με χρήση οντολογιών. Υποστηρίζει : Concept-based αναζήτηση με χρήση ολοκληρωμένων μοντέλων οντολογιών. Επιτρέπει την ανάπτυξη συσχετίσεων μεταξύ των εννοιών και την αναζήτηση με βάση προηγούμενες περιπτώσεις. (case-based retrieval). Το Orenge είναι μία πλήρης πλατφόρμα για τη δημιουργία όλων των απαιτούμενων υποσυνόλων μίας οντολογίας (έννοιες, συσχετίσεις, αξιώματα).

Επιπλέον επιτρέπει και προσαρμογή προηγούμενης στην αντίστοιχη περίπτωση (adaptation).

Κατασκευαστής: Empolis

Σχόλια:Υποστηρίζει επίσης και δημιουργία οντολογιών. Συμβατό με XML και θεματικούς χάρτες.

Εργαλείο: Freedom Enterprise Semantic Platform

Δυνατότητες: Κατηγοριοποίηση και αναζήτηση με χρήση οντολογιών. Συνάθροιση και κανονικοποίηση του περιεχομένου από μια ευρεία ποικιλία πηγών. Το ESP είναι μια πλατφόρμα για σημασιολογική ολοκλήρωση ετερογενούς περιεχομένου συμπεριλαμβανομένων των μέσων και των επιχειρηματικών βάσεων δεδομένων. Υποστηρίζει : Automated Concept Tagger, αναζήτηση με πλοήγηση, Concept-based αναζήτηση.

Κατασκευαστής: Semagix

Σχόλια:Υποστηρίζει επίσης τη δημιουργία και τη δημιουργία πληθυσμού οντολογιών. XMLbased.

Εργαλείο: OntoBroker

Δυνατότητες: Παρέχει framework για επεξεργασία κανόνων που οργανώνονται από μια οντολογία. Υποστηρίζει : Βοηθό Σχεδίασης Προϊόντος.

Κατασκευαστής: Ontoprise

Σχόλια:Συμβατό με RDF, DAML+OIL.

Εργαλείο: Semantic Miner

Δυνατότητες: Κατασκευάζει σημασιολογικά έγκυρες ερωτήσεις από ερωτήσεις σε φυσική γλώσσα.

Κατασκευαστής: Ontoprise

Σχόλια:Συμβατό με RDF, DAML+OIL.

Εργαλείο: OntoOffice

Δυνατότητες: Παράδοση περιεχομένου βασισμένου σε οντολογίες. Υποστηρίζει : Contextaware retriever.

Κατασκευαστής: Ontoprise

Σχόλια: Συνεργάζεται με το MS Office. Συμβατό με RDF, DAML+OIL.

Εργαλείο: Coherence

Δυνατότητες: Διαχείριση επιχειρησιακών δεδομένων βασισμένη σε οντολογίες. Εστιάζει στα δομημένα δεδομένα. Υποστηρίζει : Enterprise Data Model Manager.

Κατασκευαστής: Unicorn Solutions

Σχόλια: Υποστήριξη RDFS/DAML+OIL. Επίσης υποστηρίζει και δημιουργία οντολογιών.

Εργαλείο: Contextia

Δυνατότητες: Επιχειρησιακά δεδομένα και ολοκλήρωση εφαρμογών. Αποδέχεται διάφορα είδη εισόδων για απεικόνιση και μοντελοποίηση, συμπεριλαμβανομένων των σχημάτων XML, των εγγενών σχημάτων (native schemas), των πινάκων βάσεων δεδομένων, και των οριοθετημένων αρχείων. Μπορεί να λειτουργήσει αυτόνομα ή σε συνεργασία με υπάρχουσες υποδομές IT - όπως EAI, B2Bi, διαχείριση επιχειρησιακής διαδικασίας, μεσίτες μηνυμάτων και off-the-shelf συνδέσμους. Μπορεί να χειριστεί μετασχηματισμούς πληροφοριών που περιλαμβάνουν σύνθετα στοιχείων δεδομένων, φωλιασμένες δομές και ασυμβίβαστη ή συγκρουόμενη σημασιολογία (semantics). Υποστηρίζει : Semantic Data Integrator, Semantic Application Integration.

Κατασκευαστής: Modulant.

Σχόλια: Δημιουργία οντολογιών με την υποστήριξη του FirstStep XG που περιλαμβάνει το Contextia. Μερική υποστήριξη για κριτικές το μοντέλο ISO 10303

χρησιμοποιείται για την επικύρωση. Υποστηρίζει XML, έτοιμο για υπηρεσίες Ιστού και υποστήριξη SOAP.

Εργαλείο: Cerebra Server

Δυνατότητες: Σημασιολογική ολοκλήρωση επιχειρησιακών δεδομένων και εφαρμογών. Αυτή είναι μια εμπορική έκδοση της σημασιολογικής μηχανής OilED. Υποστηρίζει : Semantic Data Integrator, Semantic Form Generator, Results Classifier, Knowledge Pulse.

Κατασκευαστής: Network Inference

Σχόλια: Υποστηρίζει επίσης τη δημιουργία πληθυσμού. Υποστηρίζει τη συντήρηση και την εξέλιξη οντολογιών με την προσθήκη των νέων αξιωμάτων/ σχέσεων χωρίς διακοπή του συστήματος. Υποστηρίζει διεπαφές RDF, DAML+OIL, OWL και SOAP.

Εργαλείο: Tucana KnowledgeStore

Δυνατότητες: Κατανεμημένη βάση δεδομένων σχεδιασμένη ειδικά για μεταδεδομένα και διαχείριση μεταδεδομένων. Η βάση δεδομένων έχει σχεδιαστεί για ανάκτηση μεταδεδομένων με εξαιρετικά γρήγορα επίπεδα απόδοσης διατηρώντας τη μόνιμη ακεραιότητα και την ασφαλή πρόσβαση.

Κατασκευαστής: Plugged In Software

Σχόλια: Υποστηρίζει επίσης τη δημιουργία και τη δημιουργία πληθυσμού με το Tucana Metadata Extractor™. Υποστηρίζει RDF, SOAP, COM και διεπαφές Java.

5.7 Protégé

Είναι ένα κατεξοχήν εργαλείο για ανάπτυξη οντολογιών και για διαχείριση εφαρμογών βασισμένων στη γνώση. Το εργαλείο αποτελείται από μια open source προσέγγιση της ακαδημαϊκής κοινότητας του πανεπιστημίου του Stanford (http://protege. stanford. edu) βασισμένη σε Java με δυνατότητες επέκτασης μέσω plug ins και με μια πολύ ισχυρή κοινότητα από developers που περιλαμβάνει ακαδημαϊκούς, κυβερνητικά μέλη και στελέχη επιχειρήσεων και εξυπηρετεί σε εφαρμογές διαχείρησης γνώσης σε πολλούς και διαφορετικούς τομείς έρευνας. Το Protégé χρησιμοποιήθηκε αρχικά για το migration. Συγκεκριμένα έγινε χρήση του OntoBase Plug in το οποίο επιτρέπει να συνδεθείς με βάση δεδομένων Oracle και να επιλέξεις τα στοιχεία της βάσης που επιθυμεί κανείς να κάνει import.

5.8 Ontology Editors

Οι ontology editors (Gómez-Pérez et al, 2004) είναι εργαλεία που επιτρέπουν στους χρήστες να χειριστούν οπτικά τις οντολογίες , αξιολογούμε τους ontology editors από την άποψη των απαιτήσεων για την εξέλιξη οντολογίας. Επιλέγουμε τρεις συντάκτες οντολογίας που χρησιμοποιούνται πολύ συχνά στη σημασιολογική κοινότητα Ιστού,

Protege είναι ένας γραφικός και διαλογικός σχεδιασμός οντολογίας και απόκτηση γνώσεων από το περιβάλλον που αναπτύσσεται από το Stanford Medical Informatics group (SMI) στο πανεπιστήμιο του Στάνφορντ. Το Protege - OWL επεκτείνει την πλατφόρμα Protege σε έναν συντάκτη οντολογίας για την OWL

OntoEdit υποστηρίζει ένα περιβάλλον εφαρμοσμένης μηχανικής οντολογίας ανάπτυξης και συντήρησης οντολογιών με γραφικά μέσα. Η OntoEdit είναι χτισμένη πάνω σε ένα ισχυρό εσωτερικό πρότυπο οντολογίας

OilED έχει αναπτυχθεί από το πανεπιστήμιο του Μάντσεστερ. Είναι ένας απλός ontology editor , ο οποίος επιτρέπει στο χρήστη να χτίσει οντολογίες χρησιμοποιώντας το OIL και το OWL. Αρχικά δεν προορίζονται ως πλήρες περιβάλλον ανάπτυξης οντολογίας. Όλοι οι συντάκτες επιτρέπουν στοιχειώδες αλλαγές στις οντολογίες. Οι περισσότερες από τις υπάρχουσες οντολογίες δεν περιλαμβάνουν σύνθετες αλλαγές. Μόνο ο OntoEdit παρέχει την υποστήριξη για μερικές σύνθετες αλλαγές.

6 Εφαρμογή εξέλιξης οντολογίας με την εγκυκλοπαίδεια Wikipedia μέσω java

Σκοπός του δεύτερου μέρους αυτής της πτυχιακής εργασίας είναι να αναπτυχθεί λογισμικό που θα αναλύει ημι-αυτόματα μία οντολογία η οποία θα βρίσκεται σε XML αρχείο αποθηκευμένο κάπου στον υπολογιστή ενός χρήστη. Η οντολογία αυτή θα αποτελείται από έναν ορισμένο αριθμό λέξεων όπου για κάθε μια από τις λέξεις αυτές θα ψάχνει συνώνυμες ή παρόμοιες λέξεις στο Wikipedia που δεν υπάρχουν στην οντολογία και θα προτείνει στο χρήστη να τις προσθέσει.

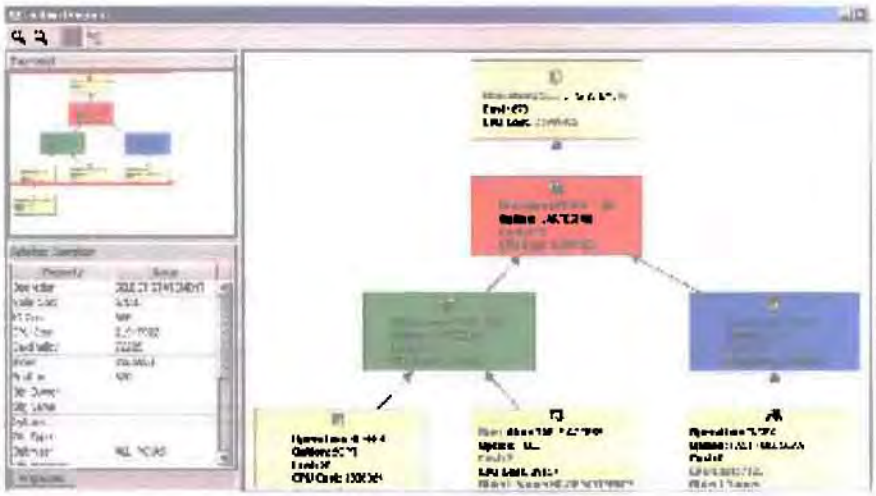
Η γλώσσα υλοποίησης της παραπάνω εργασίας θα είναι η Java. Οι δυνατότητες της εφαρμογής δεν θα είναι ιδιαίτερα αναπτυγμένες καθώς το λογισμικό θα έχει δυνατότητα να διαχειρίζεται οντολογίες με λεξικό ορισμένων δεκάδων λέξεων.

Η εφαρμογή θα είναι μορφής Application θα εμφανίζει στο χρήστη τη μια λίστα με συνώνυμες και παρόμοιες λέξεις. Λέξεις που δεν υπάρχουν στην οντολογία, αλλά υπάρχουν στο Wikipedia.

6.1 Παρουσίαση λογισμικού

6.1.1 JGraph

Το JGraph [1] είναι γραμμένο 100% σε Java που ακολουθεί το MVC [2] (Model-View-Controller) Architectural Pattern που συναντάται στη βιβλιοθήκη της Java. Σκοπός της βιβλιοθήκης αυτής είναι η δημιουργία γραφημάτων. Περιέχει δυνατότητες για ζουμ, για drag & drop, layering, moving/sizing και πολλές άλλες. Τέλος είναι «ανοιχτού κώδικα» και διατίθεται με την άδεια χρήσης λογισμικού ανοιχτού κώδικα LGPL [3] (Lesser Gnu Public License).



Εικόνα 6.1 Γράφημα με τη χρήση του JGraph

Το λογισμικό αυτό μπορεί να χρησιμοποιηθεί για να δημιουργηθούν γραφήματα (εικόνα 6.1) που μέσα από εικόνες να εμφανίζουν τις λέξεις που αποτελούν την οντολογία καθώς και τις συνώνυμες ή παρόμοιες λέξεις. Μάλιστα συγκεκριμένα χρώματα θα μπορούσαν να χρησιμοποιηθούν για τις συνώνυμες ή αντίθετες λέξεις. Η τελική χρήση της βιβλιοθήκης αυτής θα παρουσιαστεί στη συνέχεια.

6.1.2 Apache HTTPClient

Για την εργασία αυτή θα χρησιμοποιηθεί σαν πηγή πληροφοριών το Wikipedia. Επειδή δεν υπάρχουν συγκεκριμένα API [4] (Application Programming Interface) για την επικοινωνία με το Wikipedia θα χρησιμοποιηθούν απλά HTTP requests στο server του Wikipedia.

Η Java περιέχει στο πακέτο java.net βασικές εντολές για χρήση του πρωτοκόλλου HTTP. Όμως δεν περιέχει πλήρη υποστήριξη για όλα τα features του HTTP που ίσως φανούν απαραίτητα κατά τη διάρκεια της υλοποίησης της εργασίας αυτής. Το HttpClient [5] αποτελεί μια υλοποίηση ανοιχτού κώδικα από το Apache Foundation [6] που προσπαθεί να καλύψει το κενό του java.net παρέχοντας μια βιβλιοθήκη που υλοποιεί πλήρως τα HTTP standards και recommendations.

Το HttpClient έχει σχεδιαστεί με σκοπό να υποστηρίξει πλήρως το πρωτόκολλο HTTP και είναι ιδιαιτέρως χρήσιμο σε οποιαδήποτε εφαρμογή Java που

θέλει να κάνει εξειδικευμένη χρήση του πρωτοκόλλου αυτού. Παρακάτω παρουσιάζονται ορισμένα από τα χαρακτηριστικά της βιβλιοθήκης αυτής:

- Java implementation του HTTP 1.0 και 1.1
- Πλήρη υποστήριξη όλως των μεθόδων (GET, POST, PUT, DELETE, HEAD, OPTIONS, και TRACE) μέσα από ένα καλοσχεδιασμένο αντικειμενοστρεφούς framework.
- Υποστήριξη κωδικοποίησης HTTPS (HTTP μέσα από SSL).
- Χρήση HTTP proxies.
- Connection management για χρήση σε multi-threaded applications. Μπορεί δηλαδή η εφαρμογή να κάνει ταυτόχρονες παράλληλες κλήσεις σε σελίδες HTTP. Αναγνωρίζει μάλιστα και κλείνει συνδέσεις που αποτυγχάνουν.
- Αυτόματη διαχείριση Cookies.
- Εύκολη άμεση πρόσβαση στα headers που στέλνει ο server.
- Όλα τα παραπάνω κάνουν τη χρήση της βιβλιοθήκης αυτής απαραίτητη για την εφαρμογή που ζητείται από την εργασία αυτή και περισσότερες λεπτομέρειες θα παρουσιαστούν στη συνέχεια.

6.1.3 Apache Log4J

Επειδή η εφαρμογή που θα αναπτύξουμε θα εμφανίζει διάφορα μηνύματα κατά το debugging phase, σε περίπτωση που δεν υπάρχει σύνδεση στο διαδίκτυο ή όταν το XML μήνυμα δεν είναι valid αποφασίστηκε να χρησιμοποιηθεί η βιβλιοθήκη Log4J [7] της Apache για τη διαχείριση των logs.

Σκοπός της βιβλιοθήκης αυτής είναι η παροχή μηχανισμών για να μπορεί ο προγραμματιστείς να τοποθετεί κώδικα που να δημιουργεί Logs διαφόρων επιπέδων ανάμεσα σε FATAL, CRITICAL, HIGH, LOW, WARNING. Επειδή το «κόστος» του logging όσον αφορά τη μνήμη και τον επεξεργαστή ορισμένες φορές είναι σχεδόν δυσβάστακτος, το Log4J επιτρέπει στον προγραμματιστή να το κάνει enable/disable on demand.

Οι παράμετροι του Log4J μπορούν να τοποθετηθούν μέσα από ένα απλό configuration αρχείο. Έτσι δεν είναι αναγκαία καμία αλλαγή στον κώδικα της εφαρμογής για να γίνει κάποια αλλαγή σχετική με το logging.

Η καλύτερη υλοποίηση για logging στην Java τη στιγμή αυτή είναι το Log4J και για αυτό θα χρησιμοποιηθεί έστω και σε μικρό βαθμό στην εργασία αυτή. Ένας ακόμα λόγος για τη χρήση της βιβλιοθήκης αυτής είναι ότι η εφαρμογή Log4J είναι ανοιχτού κώδικα και διανείμετε δωρεάν.

6.1.4 Apache Commons

Τρέχοντας για πρώτη φορά την εφαρμογή που αναπτύσσαμε εμφανίστηκαν τα πρώτα μηνύματα λάθους (εικόνα 6.2). Το πρώτα αναφέρει την έλλειψη του org.apache.commons.logging. Η βιβλιοθήκη commons-logging.jar εξαρτάται της βιβλιοθήκης HttpClient. Δηλαδή για να χρησιμοποιήσει κανείς το HttpClient πρέπει οπωσδήποτε να προσθέσει στο classpath τη βιβλιοθήκη commons-loggging.jar [8]

Εικόνα 6.2 μήνυμα λάθους που εμφανιζόταν

Εφόσον προστέθηκε η απαραίτητη βιβλιοθήκη η εφαρμογή άρχισε να εμφανίζει σειρά άλλων μηνυμάτων (εικόνα 6.3). Αυτή τη φορά ανακοίνωνε την έλλειψη της βιβλιοθήκης apache.commons.codec.
Παρακάτω ακολουθεί το μήνυμα λάθους που εμφανιζόταν:

Εικόνα 6.3 μήνυμα λάθους

Αφού τοποθετηθεί η βιβλιοθήκη codec στο classpath, ο HttpClient μπορούσε να τρέξει κανονικά. Έτσι η codec αποτελεί το δεύτερο dependency της HttpClient. Πρέπει όμως να σημειωθεί ότι η codec αποδείχθηκε πραγματικά χρήσιμη με τις λειτουργίες που υλοποιεί. Λεπτομέρειες για την περαιτέρω χρήση της θα παρουσιαστούν στη συνέχεια της πτυχιακής.

6.1.5 XML Parser

Η εργασία αυτή ζητά την εισαγωγή δεδομένων μέσω XML αρχείων. Για να μπορέσει η εφαρμογή μας να επεξεργαστεί τα αρχεία αυτά, να κατανοήσει τη δομή τους και να μπορέσει να εξάγει χρήσιμες πληροφορίες από αυτά θα χρειαστεί η χρήση ενός XML Parser.

Υπάρχουν αρκετές υλοποιήσεις XML parser σε Java διαθέσιμες. Οι απαιτήσεις όμως της εφαρμογής της εργασίας αυτής δεν είναι αυξημένες για να χρησιμοποιηθεί κάποια ειδικευμένη υλοποίηση. Η υλοποίηση που περιέχεται μέσα στο JDK επαρκεί για τις ανάγκες.

Η Java περιέχει τρείς διαφορετικές υλοποιήσεις για χειρισμό XML μηνυμάτων.

- SAX (Simple API for XML) Το SAX αποτελεί μια απλοποιημένη υλοποίηση για XML handling.
- StaX (Streaming API for XML) Το StaX παρουσιάστηκε για πρώτη φορά στην version 6.0 της Java και είναι ανώτερο από το SAX και το DOM.
- Java Architecture for XML Binding (JAXB)

6.1.6 Apache ANT

Για να μπορεί η εφαρμογή να πακεταριστεί θα πρέπει όλα τα .class μετά το compilation να μπουν σε ένα αρχείο .JAR Για να γίνει αυτό πρέπει πρώτα από όλα να δημιουργηθεί ένα MANIFEST αρχείο.

Το παραπάνω αρχείο περιγράφει που θα βρεθεί η κύρια κλάση της εφαρμογής και ποιές βοηθητικές βιβλιοθήκες χρησιμοποιούνται και απαιτούνται για να τρέξει η εφαρμογή (όλες αυτές που έχουν παρουσιαστεί).

Εφόσον δημιουργηθεί το MANIFEST αρχείο, ένα ακόμα θέμα είναι αν η εφαρμογή χρησιμοποιεί ορισμένες εικόνες, θα πρέπει και αυτές να τοποθετηθούν στο .JAR

Για να αυτοματοποιηθεί όμως η παραπάνω διαδικασία κατασκευής του αρχείου .JAR αποφασίστηκε να χρησιμοποιηθεί ένα ant script file. Ονομάζεται build.xml και βρίσκεται στο root κατάλογο του project.

Το ANT αποτελεί ένα open source project του Apache Foundation και είναι ένα εργαλείο που βοηθάει στην αυτοματοποίηση της διαδικασίας BUILD και packaging εφαρμογών Java.

Για να χρησιμοποιήσει κανείς το ANT αφού το εγκαταστήσει και το τοποθετήσει στο PATH του λειτουργικού συστήματος πρέπει να δημιουργήσει ένα αρχείο XML, το build.xml. Η υλοποίηση του αρχείου αυτού θα παρουσιαστεί στη συνέχεια.

6.2 Προετοιμασία περιβάλλοντος ανάπτυξης

Στο κεφάλαιο αυτό θα παρουσιαστούν λεπτομέρειες για την προετοιμασία του περιβάλλοντος, την εγκατάσταση του απαραίτητου λογισμικού και τις ρυθμίσεις των επιμέρους πακέτων λογισμικού του συστήματος.

6.2.1 Εγκατάσταση JDK 1.6

Η εφαρμογή της εργασίας αυτής θα υλοποιηθεί σε Java σαν application. Για αυτό και οι βιβλιοθήκες που επιλέχθηκαν είναι υλοποιημένες σε Java. Η έκδοση της Java που θα χρησιμοποιηθεί είναι η πλέον πρόσφατη. Κάνουμε λοιπόν download το JDK 1.6.

Επισκεπτόμαστε τη σελίδα: http://java.sun.com/javase/downloads/index.jsp όπου βλέπουμε ότι το πλέον πρόσφατο είναι το JDK 6 Update 7, οπότε επιλέγουμε Download και κατόπιν την πλατφόρμα που θα «στήσουμε» το περιβάλλον μας. Συμφωνούμε στην άδεια χρήσης και πατάμε το κουμπί Continue (εικόνα 6.4).

Εικόνα 6.4 Πλατφόρμα και γλώσσα που επιλέγουμε

Τέλος κάνουμε κλικ στο όνομα του αρχείου που μας προτείνεται για να ξεκινήσουμε τη λήψη του JDK 1.6 (εικόνα 6.5)

Εικόνα 6.5 Λήψη αρχείου

Αφού κατέβει το JDK προχωράμε στην εγκατάστασή του και στη ρύθμιση του συστήματος. Για την εγκατάσταση του JDK κάνουμε διπλό κλικ στο executable. Η διαδικασία είναι πολύ απλή. Δεχόμαστε την άδεια χρήσης του λογισμικού από την Sun Microsystems πατώντας 'I Accept License' και κατόπιν πατάμε 'Next ->' μέχρι να εγκατασταθεί το JDK. Τελικά πατάμε 'Finish'. Στο βήμα αυτό εγκαθιστάμε και τις δύο εκδόσεις του JDK.

Η διαδικασία εγκαθιστά και το JDK και το JRE. Το JRE είναι το Runtime Environment. Χρειάζεται για αποκτήσει ο υπολογιστής δυνατότητες να «τρέχει» εφαρμογές Java είτε αυτές είναι applet είτε εφαρμογές. Το JDK από την άλλη είναι απαραίτητο για να αποκτήσει δυνατότητες ο υπολογιστής να κάνει compile ή ακριβέστερα να κάνει interpret αρχεία με κώδικα (αρχεία με κατάληξη .java) σε αρχεία με κατάληξη .class (που είναι η interpreted μορφή των Java κλάσεων).

By default η διαδρομή στην οποία θα εγκατασταθεί σε Windows λειτουργικό σύστημα είναι η C:\Program Files\Java\JDK_version και η C:\Program Files\Java\JRE_version

Με το πέρας της εγκατάστασης θα πρέπει να βρίσκονται οι παρακάτω φάκελοι στη διαδρομή C:\Program Files\Java (εικόνα 6.6)

Εικόνα 6.6 Οι φάκελοι που υπάρχουν μετά την εγκατάσταση

6.2.2 Εγκατάσταση του Eclipse

Για την ανάπτυξη της εφαρμογής θα χρησιμοποιηθεί το Eclipse IDE [9]. Το Eclipse αποτελεί μια πλατφόρμα ανάπτυξης λογισμικού. Είναι ανοιχτού κώδικα και ξεκίνησε από μια προσπάθεια της IBM να δημιουργήσει ένα εργαλείο με ανοιχτή φιλοσοφία που να βελτιώνουν και να προσθέτουν δυνατότητες.

Τη στιγμή αυτή για τη συγγραφή κώδικα Java θεωρείται ένα από τα πιο χρήσιμα εργαλεία μαζί με το NetBeans [10], πλατφόρμα με αντίστοιχες δυνατότητες που υλοποιείται με την καθοδήγηση της Sun Microsystems.

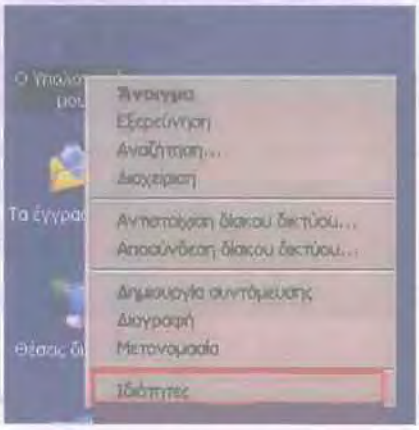
Οι δυνατότητες των πλατφόρμων αυτών είναι εκατοντάδες και δε θα παρουσιαστούν στην εργασία αυτή. Για την εγκατάσταση του Eclipse πρέπει να κάνουμε λήψη από τη σελίδα του Eclipse. Τη στιγμή αυτή το Ganymede, ή έκδοση του Eclipse 3.4.1 είναι η πλέον πρόσφατη.

Αφού κατεβάσουμε το eclipse δεν υπάρχει ανάγκη εγκατάστασης. Απλά αποσυμπιέζουμε το περιεχόμενο του project σε ένα φάκελο και τρέχει με διπλό κλικ στο /bin/eclipse.exe.

Την πρώτη φορά που τρέχει το Eclipse ζητάει να ορίσουμε το φάκελο μέσα στον οποίο θα τοποθετηθεί ο κώδικας και τα αρχεία των εφαρμογών που θα αναπτύξουμε. Καλό είναι να ορίσουμε το Workspace φάκελο σε διαφορετικό από αυτό που τοποθετήσαμε την πλατφόρμα Eclipse. Έτσι αν αποσυμπιέσαμε το Eclipse στο C:\Program Files\Eclipse3.4.1 θα τοποθετήσουμε το Workspace στο C:\Workspace. Έτσι όταν μελλοντικά επιθυμούμε να αναβαθμίσουμε το Eclipse, ο φάκελος που περιέχει όλες τις εφαρμογές που έχουμε αναπτύξει θα παραμείνει ανέπαφος και λειτουργικός. Το μόνο που θα έχουμε να κάνουμε είναι να διαγράψουμε το φάκελο C:\Program Files\Eclipse3.4.1 και να αποσυμπιέσουμε μια νεότερη έκδοση του Eclipse σε νέο φάκελο.

6.2.3 Εγκατάσταση του ANT

Αφού αποσυμπιέσουμε το apache-ant-1.7.1-bin.zip σε έναν φάκελο θα χρειαστεί να ρυθμίσουμε ορισμένες μεταβλητές του περιβάλλοντος για να είναι πλήρως λειτουργικά τόσο το JDK όσο και το ANT (εικόνα 6.7). Πατώντας δεξί κλικ στο 'Ο Υπολογιστής μου' προχωράμε στις 'Ιδιότητες'.

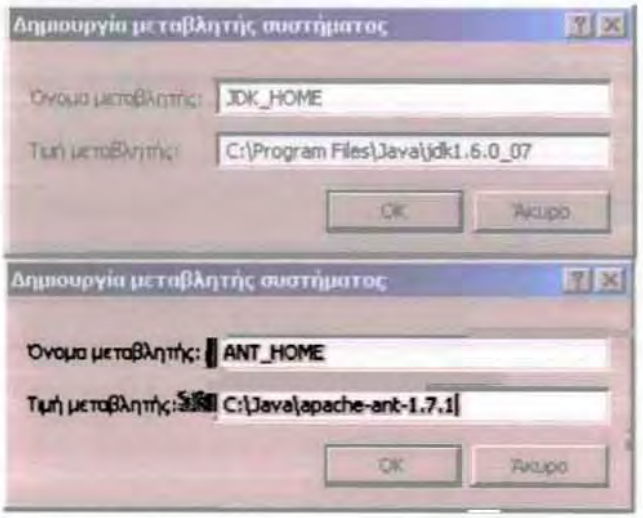


Εικόνα 6.7 Ρύθμιση μεταβλητών περιβάλλοντος

Στην καρτέλα 'Για προχωρημένους' επιλέγουμε 'Μεταβλητές περιβάλλοντος' (εικόνα 6.8)

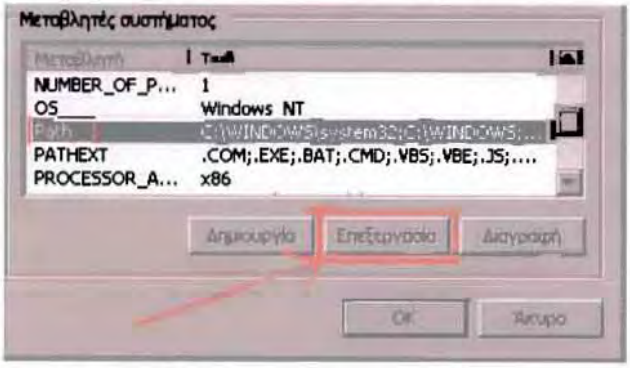
Εικόνα 6.8 Καρτέλα 'για προχωρημένους'

Επιλέγουμε 'Δημιουργία' και δημιουργούμε την πρώτη εγγραφή με όνομα μεταβλητής JAVA_HOME και τιμή μεταβλητής τη διαδρομή που εγκαταστήσαμε το JDK (εικόνα 6.9). Στην προκειμένη περίπτωση το C:\Program Files\Java\jdk1.6.0_07. Επαναλαμβάνουμε για τις υπόλοιπες μεταβλητές.



Εικόνα 6.9 Δημιουργία μεταβλητής συστήματος

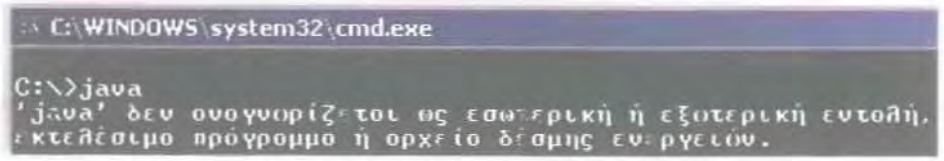
Εφόσον έχουμε δημιουργήσει όλες τις απαραίτητες μεταβλητές πρέπει να κάνουμε επεξεργασία του Path (εικόνα 6.10).



Εικόνα 6.10 Επεξεργασία στο Path

Προσθέτουμε λοιπόν στο Path τις διαδρομές των executables ;%ANT_HOME%\bin;%JAVA_HOME%\bin

Πρωτού ορίσουμε τις μεταβλητές αυτές, το σύστημα δεν γνωρίζει την τοποθεσία του κάθε πακέτου λογισμικού όπως φαίνεται στο επόμενο παράδειγμα (εικόνα 6.11).



Εικόνα 6.11 Γραμμή εντολών

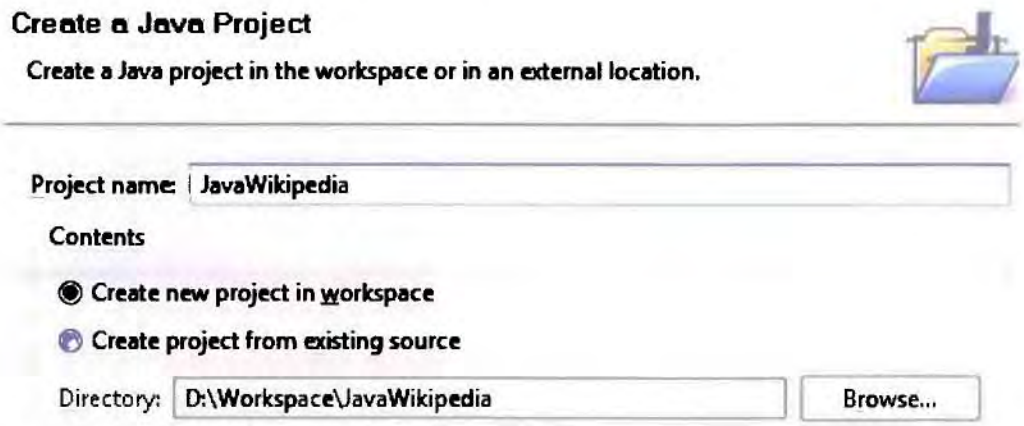
Μετά την τοποθετηθούν των διαδρομών στο PATH οι εντολές εκτελούνται κανονικά (εικόνα 6.12).

Εικόνα 6.12 Γραμμή εντολών

7 Δημιουργία νέου project

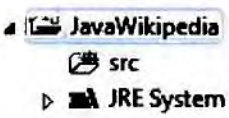
Αφού ανοίξουμε το Eclipse για πρώτη φορά και επιλέξουμε το WorkSpace (τον φάκελο στον οποίο θα αποθηκεύονται τα αρχεία τύπου Java και γενικότερα όλα τα αρχεία του κάθε μας Project) προχωράμε στη δημιουργία ενός νέου project για την εργασία αυτή (εικόνα 7.1).

Εικόνα 7.1: File → New → Java Project



Εικόνα 7.2 Δημιουργία νέου project

Ονοματίζουμε το νέο μας Project 'JavaWikipedia'και πατάμε Finish (εικόνα 7.2).



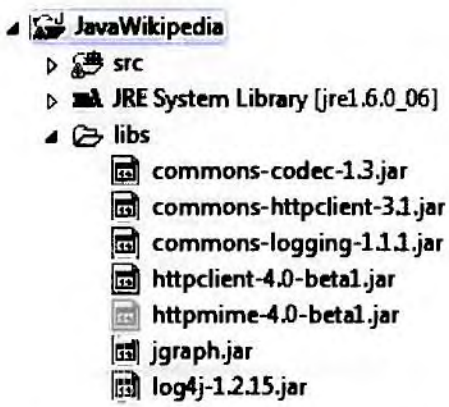
Εικόνα 7.3: Οι φάκελοι που περιέχει το νέο project

Το επόμενο βήμα είναι να δημιουργήσουμε ένα φάκελο με το όνομα libs μέσα στον οποίο θα τοποθετήσουμε ορισμένες βιβλιοθήκες της Java που θα χρησιμοποιήσουμε στη συνέχεια για την ανάπτυξη της εφαρμογής μας. Δημιουργούμε λοιπόν το νέο αυτό φάκελο και τοποθετούμε τις βιβλιοθήκες που αναφέραμε νωρίτερα στον φάκελο αυτό. Τα απαραίτητα αρχεία των βιβλιοθηκών που επιλέξαμε είναι τα:

- Jgraph.jar από το jgraph-5.12.1.0-lgpl.jar της βιβλιοθήκης JGraph
- log4j-1.2.15 από το Log4J project.

- commons-httpclient-3.1.jar και commons-codec-1.3.jar από το codec project της Apache.
- httpmime-4.0-beta1.jar και httpclient-4.0-beta1.jar από το HttpClient project της Apache.

Αφού τοποθετήσουμε τα παραπάνω αρχεία σε νέο φάκελο με το όνομα libs και πατήσουμε στο Eclipse το F5 για να κάνει Refresh βλέπουμε τα αρχεία μας στο φάκελο (εικόνα 7.4).

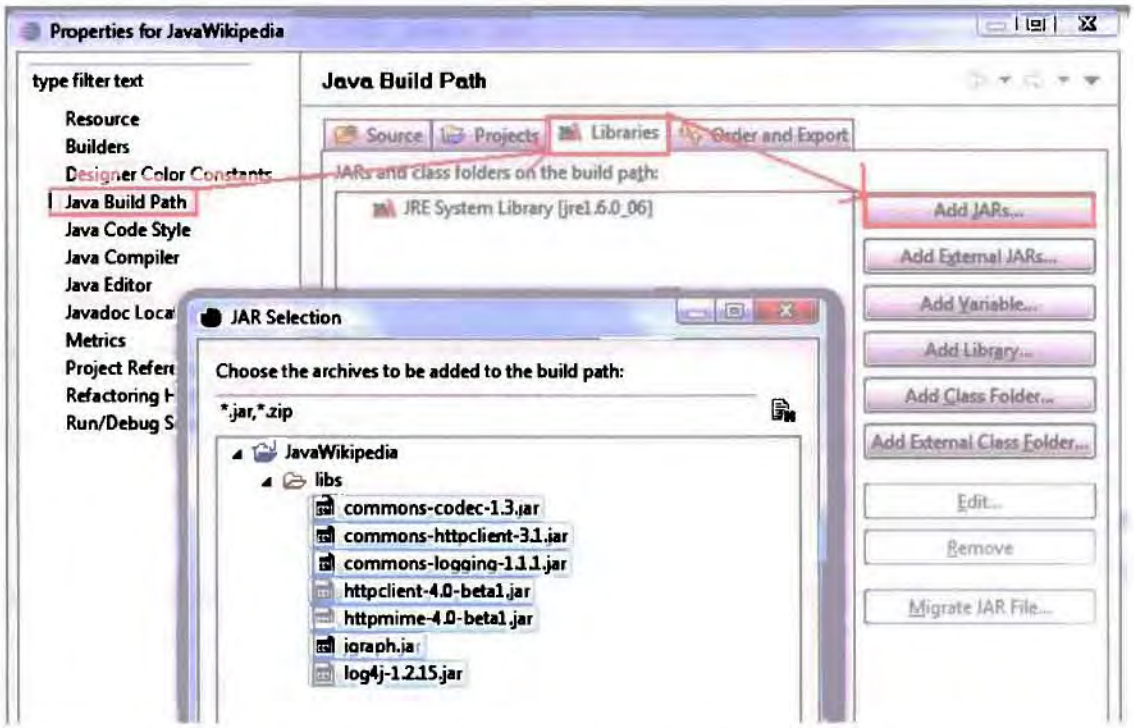


Εικόνα 7.4 Τα αρχεία που υπάρχουν στο libs

Στη συνέχεια πρέπει να ενημερώσουμε το project μας για την ύπαρξη των αρχείων αυτών ώστε να τα τοποθετήσει στο classpath της εφαρμογής μας (εικόνα 7.5). Για να το κάνουμε αυτό πατάμε Alt+Enter ή από το μενού Project → Properties:

Εικόνα 7.5 Project → Properties

Στη σελίδα που εμφανίζεται επιλέγουμε Java Build Path και στην καρτέλα Libraries πατάμε Add JARs... όπου επιλέγουμε τα JAR αρχεία, τα αρχεία των βιβλιοθηκών που θα χρησιμοποιήσουμε στη συνέχεια (εικόνα 7.6).



Εικόνα 7.6 Τα αρχεία των βιβλιοθηκών που θα χρησιμοποιήσουμε

7.1Σχετικές τεχνολογίες

7.1.1 Unicode και URI

Το πρότυπο *Unicode* [11] είναι ένα σύστημα κωδικοποίησης χαρακτήρων που σχεδιάστηκε για να υποστηρίζει την ανταλλαγή, επεξεργασία και εμφάνιση κειμένων σε οποιαδήποτε γλώσσα του κόσμου. Το *unicode* αποτελεί βασικό συστατικό και διεθνές στάνταρ όλων των σύγχρονων πρωτοκόλλων στις τεχνολογίες πληροφορικής και τηλεπικοινωνιών, καθώς παρέχει μία κοινή και ενιαία αρχιτεκτονική για την κωδικοποίηση περισσότερων από 96.000 χαρακτήρων. Δημιουργήθηκε για να αντικαταστήσει αποτελεσματικά το προηγούμενο στάνταρ το ANSI ASCII.

Στο ίδιο επίπεδο με το unicode, βρίσκεται επίσης και το *URI* (*Uniform Resource Identifier*) [12], το πρότυπο για την αναγνώριση και τον εντοπισμό ενός ονόματος ή μιας πηγής σε ένα δίκτυο, με τη χρήση μιας προκαθορισμένης μορφής απόδοσης ονομάτων.

7.1.2 XML - Ελληνικό Wikipedia

Ο βασικός λόγος δημιουργίας του XML (eXtensible Markup Language) είναι να μεταφέρει και να αποθηκεύει δεδομένα συνδέοντας τα με κατάλληλες πληροφορίες. Μέσα σε ένα XML μπορούμε να φτιάξουμε την δομή που εμείς θέλουμε και χρειαζόμαστε. Μοιάζει με την HTML, με τη διαφορά ότι τα "tags" που χρησιμοποιούμε δεν είναι κάποια στάνταρ, αλλά μπορούμε να τα ορίσουμε όπως εμείς θέλουμε. Στην ουσία δεν είναι κάτι ιδιαίτερο, παρά ένα απλό κείμενο. Δεν χρειάζεται κάποιο συγκριμένο λογισμικό για να το διαβάσουμε.

Ας δούμε ένα παράδειγμα XML.

```
<dictionary>
  <verb>τρέχω</verb>
  <verb>είμαι</verb>
  <proverb>δοκιμή</proverb>
  <proverb>έκταση</proverb>
</ dictionary>
```

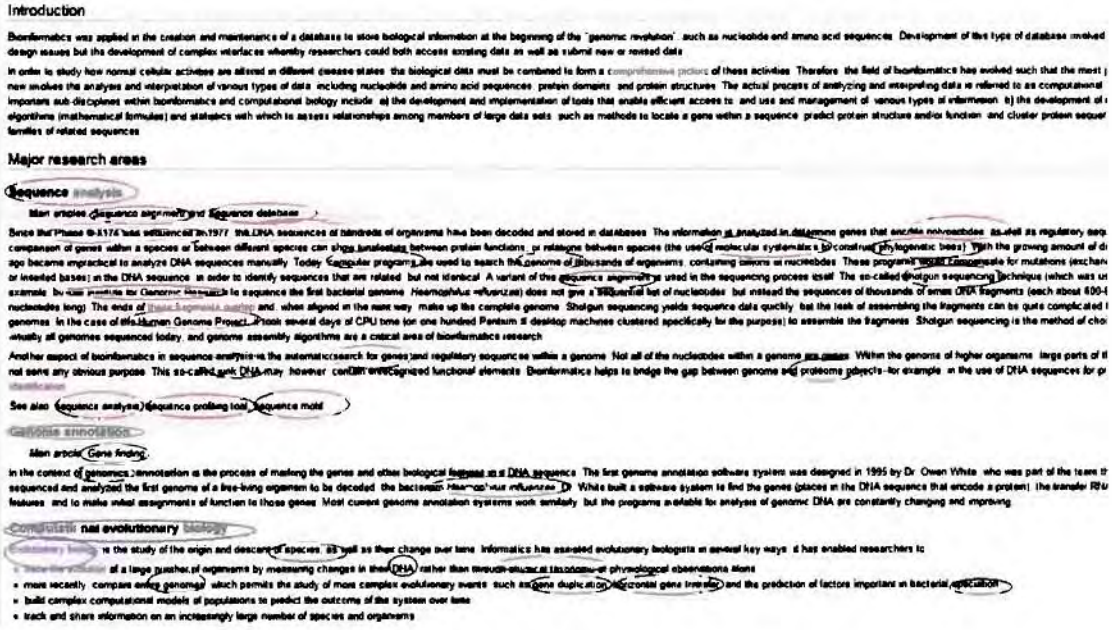
Το παραπάνω είναι αρκετά περιγραφικό και από μόνο του. Τα tags όπως <dictionary>, <verb>, <proverb> που χρησιμοποιούνται δεν είναι κάποια XML standard, αλλά τα θέσαμε εμείς στο παράδειγμά μας. Το <dictionary> είναι "*root element*", ενώ τα <verb> και το <proverb> είναι "*child element*s". Για να κλείσει ένα XML πρέπει να το κλείσουμε και εμείς με το κάθε "*child element*" και κατόπιν το "*root element*" </dictionary> τα οποία είχαμε ξεκινήσει ποιο πριν.

Θα πρέπει να προσέξουμε ότι κάθε XML element που ξεκινάμε θα πρέπει να το κλείνουμε . Επίσης πρέπει να δοθεί προσοχή επειδή είναι case sensitive. Μέσα σε κάθε element δεν υπάρχει κανένας περιορισμός να ξεκινήσουμε και να κλείσουμε κάποιο άλλο element όπως φαίνετε και στο παράδειγμά μας.

7.1.3 XML - Αγγλικό Wikipedia

Η δεύτερη περίπτωση της εργασίας αυτής αφορά το Αγγλικό Wikipedia και την ανάλυση επιστημονικών όρων. Τα tags: verb και proverb δεν επαρκούν για την αποθήκευση σε μορφή XML των δεδομένων που συλλέγονται σε μορφή που να έχει κάποιο νόημα.

Παίρνοντας σαν παράδειγμα τον όρο Bioinformatics στο Αγγλικό Wikipedia παραθέτουμε την εικόνα 7.7.



Introduction

Major research areas

Sequence analysis

Gene finding

Computational evolutionary biology

Εικόνα 7.7 Όρος Bioinformatics στο Αγγλικό Wikipedia

Μπορούμε λοιπόν να θεωρήσουμε ως κύριο tag το dictionary όπως κάναμε και στην ελληνική έκδοση του parser. Στη συνέχεια και για να ομαδοποιήσουμε τα δεδομένα που συλλέγουμε ορίζουμε ένα νέο tag με την ονομασία class όπου θα τοποθετούμε την ομάδα σαν id του tag αυτού. Κατόπιν όσα links υπάρχουν μέσα σε αυτή την κατηγορία θα έχουν το tag instance.

Έτσι η μορφή του XML αρχείου θα είναι:

```
<dictionary>
 <class id= "Sequence analysis">
    <instance>Sequence_alignment</instance>
    <instance>Sequence_database</instance>
    <instance>Phi-X174_phage</instance>
    <instance>Sequencing</instance>
    <instance>DNA_sequence</instance>
    <instance>Polypeptides</instance>
    <instance>Species</instance>
    <instance>Molecular_systematics</instance>
    <instance>Phylogenetic_tree</instance>
    <instance>Computer_program</instance>
    <instance>Genome</instance>
    <instance>Nucleotide</instance>
    ....
 </class>
 .....
</dictionary>
```

Οι κύριες ομαδοποιήσεις θα βασίζονται στο class και η τιμή θα βρίσκεται στο id, και οι δευτερεύον θα βασίζονται στο instance.

7.1.4 Wikipedia Usage - Ελληνικό

Ας δούμε πως χρησιμοποιώντας το Wikipedia μπορούμε να εντοπίσουμε συνώνυμες λέξεις. Το καθεαυτό Wikipedia δεν μπορεί να δώσει τέτοιες πληροφορίες καθώς δεν σχεδιάστηκε για να χειρίζεται αυτού του είδους το περιεχόμενο. Αντιθέτως το wiktionary που είναι ένα sister-project του Wikipedia περιέχει αυτές ακριβώς τις πληροφορίες.

Χρησιμοποιώντας το http://el.wiktionary.org και κάνοντας μια αναζήτηση π.χ. για τη λέξη «γαλήνη» βλέπουμε ότι μας επιστρέφει την ετυμολογία της λέξης, την προφορά της, λεκτική ανάλυση (Ουσιαστικό) και τέλος συγγενικές λέξεις και μεταφράσεις σε διάφορες γλώσσες του κόσμου (εικόνα 7.8).

Πίνακας περιεχομένων [Απόκρυψη]

1 Ελληνικά (el)
- 1.1 Ετυμολογία
- 1.2 Προφορά
- 1.3 Ουσιαστικό
  - 1.3.1 Συγγενικές λέξεις
  - 1.3.2 Μεταφράσεις

Εικόνα 7.8 Πινακας περιεχομένων wikipedia

Στην λεκτική ανάλυση παρατηρούμε ότι μας επιστρέφει συνώνυμα και της πρώτης και της μεταφορικής σημασίας της λέξης. Μπορούμε λοιπόν να χρησιμοποιήσουμε την υπηρεσία αυτή για εντοπίσουμε συνώνυμες λέξεις (εικόνα 7.9)

Ουσιαστικό

γαλήνη *θηλυκό μόνο στον ενικό*

1. η ήρεμη κατάσταση της θάλασσας ή της ατμόσφαιρας, γενικά
   - *συνώνυμα:* ηρεμία, κάλμα, μπουνάτσα, νηνεμία
   - *αντώνυμα:* τρικυμία, φουρτούνα
2. (*μεταφορικά*) η ψυχική και πνευματική ηρεμία
   - *συνώνυμα:* αταραξία, ηρεμία, πραότητα
   - *αντώνυμα:* αναστάτωση, ταραχή

Εικόνα 7.9 Εντοπισμός συνώνυμων λέξεων

Ορισμένες λέξεις όπως για παράδειγμα η λέξη «περισκόπιο» δεν έχουν ούτε ορισμό ούτε συνώνυμες λέξεις. Οπότε η εφαρμογή μας πρέπει να προσέχει και να εντοπίζει τις περιπτώσεις αυτές (εικόνα 7.10).

Ουσιαστικό

περισκόπιο *ουδέτερο*

- *Λείπει ο ορισμός (ή οι ορισμοί) αυτής της λέξης. (Προσθέστε τον/τους!)*

Εικόνα 7.10 Όταν δεν υπάρχουν συνώνυμες λέξεις

Υπάρχουν μάλιστα ορισμένες περιπτώσεις που το wiktionary δεν γνωρίζει καν την ύπαρξη των λέξεων και επιστρέφει τη σελίδα που βλέπουμε στη συνέχεια (εικόνα 7.11).

Εικόνα 7.11 Λέξεις που το wikipedia δεν τις γνωρίζει

Στο επόμενο κεφάλαιο θα παρουσιάσουμε πως η εφαρμογή μας θα αξιοποιήσει όλες τις παραπάνω πληροφορίες για να εμφανίσει στο χρήστη συνώνυμες λέξεις.

7.1.5 Wikipedia Usage - Αγγλικό

Στην έκδοση για το Ελληνικό wikipedia ενδιαφερόμασταν κυρίως για απλούς συντακτικούς όρους. Το parsing της κάθε σελίδας βασιζόταν σε λέξεις κλειδιά μέσω των οποίων εξάγαμε τα απαραίτητα δεδομένα. Στην περίπτωση του Αγγλικού Wikipedia εξάγουμε κατηγοριοποιημένα τα υπάρχοντα links. Ο πιο άμεσος τρόπος για να το υλοποιήσουμε αυτό και εφόσον μελετήσαμε τη μορφή του κειμένου HTML που παρέχει ο ιστότοπος ήταν να ανακαλύπτουμε τα σημεία εκείνα στο έγγραφο που περιείχαν το κάτωθι HTML κώδικα:

<span class=\"mw-headline\

Μέσα στο παραπάνω span υπάρχουν οι κατηγορίες τις οποίες αναπαριστάμε με το tag class. Παρόμοιες τεχνικές με αυτές της έκδοσης για το ελληνικού Wikipedia χρησιμοποιήθηκαν όπως filtering out σε άσχετα δεδομένα. Παρακάτω παρουσιάζουμε ορισμένα από τα φίλτρα αυτά.

```
if ( !( lemmaLink.contains("Category:")|
lemmaLink.contains("Portal:")|
lemmaLink.contains("Special:")|
lemmaLink.contains("Image:")|
lemmaLink.contains("$1") |
```

```
lemmaLink.contains("See                                    Also") |
lemmaLink.contains("Wikipedia:") |
lemmaLink.contains("Help:") ) ) {
```

Τα παραπάνω φίλτρα απομακρύνουν άσχετες πληροφορίες.

Επιπλέον, για τον διαχωρισμό του τύπου των στοιχείων που συλλέγουμε μια νέα κλάση υλοποιήθηκε η SmallWord.java και έγιναν εκτεταμένες τροποποιήσεις στον υπάρχον κώδικα, και φυσικά απομακρύνθηκε η χρήση του URL encoding.

7.2 Ανάπτυξη Εφαρμογής

Στο παρόν κεφάλαιο θα παρουσιαστεί η ανάπτυξη της εφαρμογής.

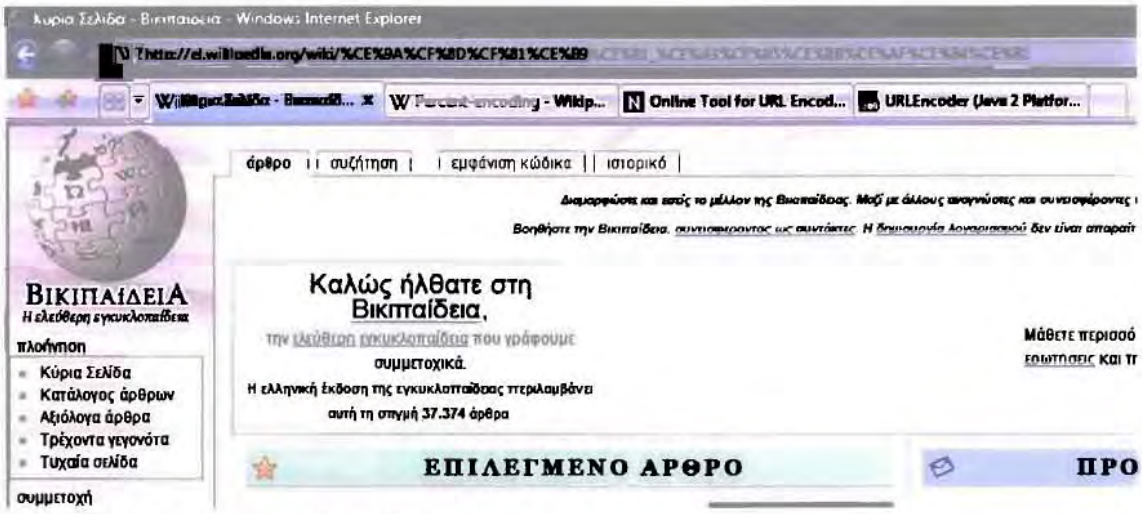
7.2.1 Percent Encoding

Ένα από τα πρώτα προβλήματα που παρουσιάστηκαν κατά την ανάπτυξη της εφαρμογής της εργασίας αυτής είναι το γεγονός ότι τα ξενόγλωσσα (μη αγγλικά) Wikipedia, όπως το Ελληνικό Βικιπαίδια χρησιμοποιούν το γνωστό Percent Encoding.

Το specification των URL ορίζει σαν επιτρεπτούς χαρακτήρες μόνο ένα μέρος της ASCI κωδικοποίησης [ASCII reference] "...Only alphanumerics [0-9a-zA-Z], the special characters "$-_.+!*'()," [not including the quotes - ed], and reserved characters used for their reserved purposes may be used unencoded within a URL." [RFC 1738 - http://www.rfc-editor.org/rfc/rfc1738.txt]

Έτσι όλοι οι άλλοι χαρακτήρες για να χρησιμοποιηθούν ως URL πρέπει να κωδικοποιηθούν. Η κωδικοποίηση γίνεται με τη χρήση του χαρακτήρα % συν δύο δεκαεξαδικούς χαρακτήρες. Οι δύο χαρακτήρες αυτοί μαζί με το σύμβολο % δίνουν το τελικό σύμβολο. Έτσι ο χαρακτήρας του κενό (space) κωδικοποιείται ως %20 κτλ.

Όπως όλοι οι φυλλομετρητές, έτσι και ο Internet Explorer κωδικοποιεί τα URL με το Percent Encoding. Μάλιστα αυτό εμφανίζεται και στην οθόνη του χρήστη (εικόνα 7.12).



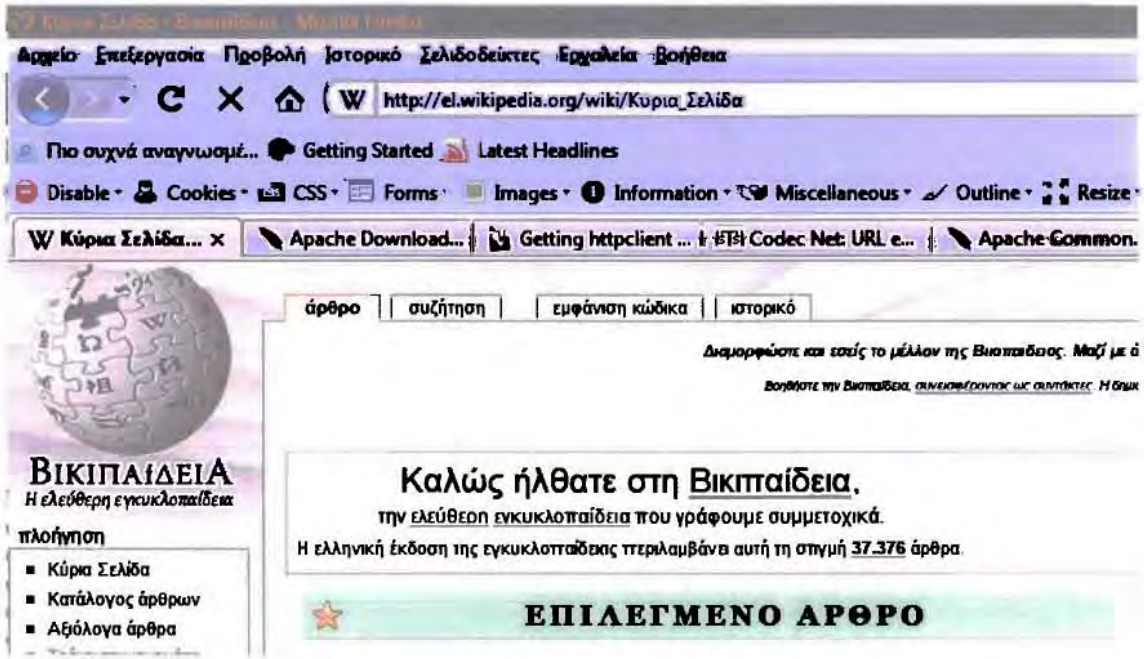
Εικόνα 7.12 Κύρια σελίδα Βικιπαίδεια

Και το URL με Percent Encoding:

http://el.wikipedia.org/wiki/%CE%9A%CF%8D%CF%81%CE%

%CE%A3%CE%B5%CE%BB%CE%AF%CE%B4%CE%B1

Ο Firefox από την άλλη, ναι μεν χρησιμοποιεί το URL Encoding αλλά το κάνει χωρίς να το εμφανίζει στον τελικό χρήστη (εικόνα 7.13).



Εικόνα 7.13 Κύρια σελίδα Βικιπαίδεια

Όπως όλες οι εφαρμογές που χρησιμοποιούν URLs έτσι και η δική μας θα χρειαστεί έναν τρόπο για να αποκωδικοποιεί και κωδικοποιεί URL από τη μια μορφή στην άλλη. Ευτυχώς το πακέτο org.apache.commons.codec.net.URLCodec περιέχει μεθόδους για αυτόν το σκοπό.

```
private URLCodec encoder = new URLCodec("UTF-8");
```

```
try {
    encoder.decode(text);
} catch (DecoderException e1) {
    e1.toString();
}
```

Ο παραπάνω κώδικας είναι αρκετός για τη μετατροπή από UTF-8 Percent Encoding σε Ελληνικά κείμενα. Σαν proof of concept υλοποιήσαμε τον παρακάτω κώδικα:

```
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import javax.swing.JTextField;

import org.apache.commons.codec.*;
import org.apache.commons.codec.net.*;



public class UrlDecoding {

    // Variables
```

```
    private JTextArea result_area = new
JTextArea(3,100);

    private JTextField encoded_URL = new JTextField(60);

    private JButton decode_button = new
JButton("Decode");

    private URLCodec encoder = new URLCodec("UTF-8");


    /**
     * Constructor
     * @param args
     */
    public UrlDecoding() {


        // 1. Build User Interface

        JFrame main_frame = new JFrame("Percent URL
Decoding - Koutonias Panagiotis");

        JPanel main_panel = new JPanel();

        main_panel.setLayout(new BorderLayout(5,5));


        //    Create user-input panel

        JPanel user_input_panel = new JPanel();

        user_input_panel.add(new JLabel("Εισάγετε το
percent encoded URL:"), BorderLayout.WEST);

        user_input_panel.add(encoded_URL,
BorderLayout.CENTER);

        user_input_panel.add(decode_button,
BorderLayout.EAST);


        //      Create the result panels with a text-
area
```

```
            JPanel result_panel = new JPanel();

     result_panel.setBorder(BorderFactory.createTitledBor
der("Εδώ θα εμφανιστεί το decoded URL"));

            result_panel.setLayout(new BorderLayout());

            result_panel.add(result_area,
BorderLayout.CENTER);

            main_panel.add(user_input_panel,
BorderLayout.NORTH);


     main_panel.add(result_panel,BorderLayout.CENTER);


     // 2. Add action-listener to the button

     decode_button.addActionListener(new  ActionListener()
{

       public void actionPerformed(ActionEvent e) {

            result_area.setText("");

            try {

     result_area.setText(encoder.decode(encoded_URL.getTe
xt()));

            } catch (DecoderException e1) {

                 result_area.setText("EXCEPTION:      "      +
e1.toString());

            }

       }

     });


            main_frame.getContentPane().add(main_panel);

            main_frame.pack();

            main_frame.setVisible(true);

     }
```

```
    public static void main(String args[]) {
        new UrlDecoding();
    }

//%CE%A6%CE%B5%CE%B2%CF%81%CE%BF%CF%85%CE%B1%CF%81%CE%AF%
CE%BF%CF%85

}
```

Ο παραπάνω κώδικας εμφανίζει μια φόρμα όπου ο χρήστης τοποθετεί το Percent Encoded URL και πατώντας το κουμπί Decode του εμφανίζεται το decoded URL (εικόνα 7.14).

Εικόνα 7.14 Φόρμα που ο χρήστης τοποθετεί το Percent Encoded URL και πατώντας το κουμπί Decode του εμφανίζεται το decoded URL

Αφού λύθηκε το ζήτημα του Percent Encoding ήρθε η ώρα να αναπτύξουμε τη συνέχεια της εφαρμογής μας.

7.2.2 XML Handling Ελληνικού Wikipedia

Ας δούμε ένα απλό XML αρχείο που περιέχει μια μικρή οντολογία με δύο ρήματα και δύο ουσιαστικά.

```
<dictionary>
  <verb>τρέχω</verb>
  <verb>γράφω</verb>
  <proverb>δοκιμή</proverb>
```

<proverb>εκταση</proverb>
</dictionary>

Αυτό που θέλουμε σε πρώτη φάση είναι να κάνουμε parse το παραπάνω αρχείο και να εξάγουμε τα δεδομένα. Θα χρησιμοποιήσουμε τον SAX parser.

```
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Vector;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;

import org.xml.sax.helpers.DefaultHandler;

/**
 * A toolkit class to provide XML reading & writing
 * functions.
 * @author Panagiotis Koutoniaw
 */
public class XMLToolkit extends DefaultHandler{
```

```
    /** Private variables. Two vectors to hold data an a
temp-string **/
    private Vector<String> verbs = new Vector<String>();
    private     Vector<String>     proverbs     =     new
Vector<String>();
    private String tempVal;


    /**
     * Constructor
     */
    public XMLToolkit(){
    }


    /**
     * @return Returns the vector holding all verbs
     */
    public Vector<String> getVerbs() {
        return verbs;
    }


    /**
     * @return Returns the vector holding all proverbs
     */
    public Vector<String> getProverbs() {
        return proverbs;
    }


    /**
```

```
     * This method parses a string representing a valid
XML file
     *  and  places  verbs  and  proverbs  to  the
corresponding
     * Vectors
     * @param xml_string the XML file as a string
     */
    private void parseDocument(String xml_string) {
        //get a factory
        SAXParserFactory                spf                =
SAXParserFactory.newInstance();
        try {
            //get a new instance of parser
            SAXParser sp = spf.newSAXParser();

            //parse  the  file  and  also  register  this
class for call backs
            InputStream          is          =          new
ByteArrayInputStream(xml_string.getBytes("UTF-8"));
            //InputStream      i_stream      =      new
BufferedInputStream(XML_FILE);
            sp.parse(is,this);

        }catch(SAXException se) {
            se.printStackTrace();
        }catch(ParserConfigurationException pce) {
            pce.printStackTrace();
        }catch (IOException ie) {
            ie.printStackTrace();
        }
```

```
    }

    /**
     * Prints data after XML parsing
     */
    private void printData(){
        int i;
        for (i = 0; i < verbs.size(); i++)
            System.out.println("Verb      :      "      +
verbs.elementAt(i));
        for (i = 0; i < proverbs.size(); i++)
            System.out.println("Proveb     :      "      +
proverbs.elementAt(i));
    }

    /**
     * Event Handler
     */
    public void startElement(String uri, String
localName, String qName, Attributes attributes) throws
SAXException {
        //reset
        tempVal = "";
        if(qName.equalsIgnoreCase("dictionary")) {
            System.out.println("Valid document");
            //attributes.getValue("type"));
        }
    }
```

```
    /**
     * Helper function
     */
    public void characters(char[] ch, int start, int
length) throws SAXException {
        tempVal = new String(ch,start,length);
    }

    /**
     * parses child-elements
     */
    public void endElement(String uri, String localName,
String qName) throws SAXException {
        if(qName.equalsIgnoreCase("verb")) {
            verbs.addElement(tempVal);
        }else if (qName.equalsIgnoreCase("Proverb")) {
            proverbs.addElement(tempVal);
        }
    }

    /**
     * Method to generate an XML file. Takes as input
two vectors. One holding verbs
     * and one holding proverbs. A boolean 'print_out'
specifies whether the generated
     * XML will be printed to the screen.
     *
     * @param in_verbs The vector holding verbs
     * @param in_proverbs The vector holding proverbs
```

```
     * @param print_out A boolean specifying if the XML
will be printed to the screen
     * @return a String
     */
    public String createXML(Vector<String> in_verbs,
Vector<String> in_proverbs, boolean print_out) {
        String str = "<dictionary>";
        if (print_out)
            System.out.println(str);

        int i;
        for (i = 0; i< in_verbs.size(); i++) {
            String temp = "<verb>" +
in_verbs.elementAt(i) + "</verb>";
            if (print_out)
                System.out.println(temp);
            str += temp;
        }
        for (i = 0; i< in_proverbs.size(); i++) {
            String temp = "<proverb>" +
in_proverbs.elementAt(i) + "</proverb>";
            if (print_out)
                System.out.println(temp);
            str += temp;
        }

        String temp = "</dictionary>";
        if (print_out)
            System.out.println(temp);
```

```
            str += temp;
            return str;
        }

        /**
         * A testing method
         */
        private void testXMLCreation() {
            Vector<String> testing_verbs = new
Vector<String>();
            testing_verbs.addElement("κοιτάζω");
            testing_verbs.addElement("παρατηρώ");
            testing_verbs.addElement("περπατάω");
            testing_verbs.addElement("μιλάω");
            Vector<String> testing_proverbs = new
Vector<String>();
            testing_proverbs.addElement("ποδήλατο");
            testing_proverbs.addElement("αμάξι");
            testing_proverbs.addElement("όχημα");
            testing_proverbs.addElement("υπολογιστής");
            testing_proverbs.addElement("τραπέζι");
            createXML(testing_verbs, testing_proverbs,
true);
        }

        /**
         * The starting point of this application
         * @param args
```

```
 */
public static void main(String[] args){
    XMLToolkit spe = new XMLToolkit();
    //spe.testXMLCreation();
    spe.parseDocument(XML_FILE);
    spe.printData();
}

public static String XML_FILE =
"<dictionary><verb>τρεχω</verb><verb>είμαι</verb><proverb
>δοκιμή</proverb><proverb>έκταση</proverb></dictionary>";
}
```

Η υλοποίηση βασίζεται στη χρήση δύο (2) Vectors του verbs και του proverbs. Μια μέθοδος, η parseDocument παίρνει σαν είσοδο το XML αρχείο σαν String. Επειδή το String αυτό περιέχει Ελληνικούς χαρακτήρες και επειδή ο SAX parser δέχεται inputstreams μετατρέπουμε το String με την εξής εντολή:
InputStream is = new ByteArrayInputStream(xml_string.getBytes("UTF-8"));

Ο κώδικας μας αφού ελέγξει ότι είναι valid XML (ακολουθεί τον κανόνα των tags) κοιτάζει για tags verb και proverb. Όταν εντοπίσει τέτοιο tag τοποθετεί το περιεχόμενό του στον αντίστοιχο Vector.

Η μέθοδος printData εκτυπώνει στην οθόνη τα verbs και proverbs για να ελέγξουμε ότι λειτουργεί αποτελεσματικά.

Valid document

Verb : τρέχω

Verb : γράφω

Proveb : δοκιμή

Proveb : έκταση

Εφόσον ελέγχθηκε η παραπάνω μέθοδος δημιουργήθηκε μια άλλη με το όνομα createXML που δέχεται σαν είσοδο δύο Vectors (ο πρώτος με αριθμό από

verbs και ο δεύτερος με αριθμό από proverbs) και μια Boolean τιμή και επιστρέφει το XML αρχείο που δημιουργεί βάση των δεδομένων εισόδου.

Η Boolean μεταβλητή καθορίζει αν το XML μήνυμα που θα δημιουργήσει θα εμφανιστεί στην οθόνη του χρήστη. Μια μέθοδος με το όνομα testXMLCreation περνάει δοκιμαστικά δεδομένα για να βεβαιωθούμε για το σωστό της υλοποίησης.

<dictionary>

<verb>κοιτάζω</verb>

<verb>παρατηρώ</verb>

<verb>περπατάω</verb>

<verb>μιλάω</verb>

<proverb>ποδήλατο</proverb>

<proverb>αμάξι</proverb>

<proverb>όχημα</proverb>

<proverb>υπολογιστής</proverb>

<proverb>τραπέζι</proverb>

</dictionary>

Όλος ο κώδικας ο σχετικός με το XML handling (reading και creation τοποθετήθηκε στο αρχείο XMLToolkit.java και θα χρησιμοποιηθεί στη συνέχεια από διάφορα μέρη της εφαρμογής.

7.2.3 XML Handling Ελληνικού Wikipedia

```
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Vector;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;

import org.xml.sax.helpers.DefaultHandler;

/**
 * A toolkit class to provide XML reading & writing
 * functions.
 * @author Panagiotis Koutonias
 */
public class XMLToolkit extends DefaultHandler{

    /** Private variables. Two vectors to hold data an a
temp-string **/
    private Vector<SmallWord> verbs = new
Vector<SmallWord>();
    //private Vector<String> proverbs = new
Vector<String>();
```

```
    private String tempVal;


    /**
     * Constructor
     */
    public XMLToolkit(){
    }


    /**
     * Constructor
     */
    public XMLToolkit(Vector<SmallWord> in_verbs) { //,
Vector<String> in_provebs){
        this.verbs = in_verbs;
        //this.proverbs = in_provebs;
    }


    /**
     * @return Returns the vector holding all verbs
     */
    public Vector<SmallWord> getVerbs() {
        return verbs;
    }


    /**
     * @return Returns the vector holding all proverbs
     *
    public Vector<String> getProverbs() {
```

```
        return proverbs;
    }

    /**
     * This method parses a string representing a valid
XML file
     * and places verbs and proverbs to the
corresponding
     * Vectors
     * @param xml_string the XML file as a string
     */
    public void parseDocument(String xml_string) {
        //get a factory
        SAXParserFactory spf =
SAXParserFactory.newInstance();
        try {
            //get a new instance of parser
            SAXParser sp = spf.newSAXParser();

            //parse the file and also register this
class for call backs
            InputStream is = new
ByteArrayInputStream(xml_string.getBytes("UTF-8"));
            //InputStream i_stream = new
BufferedInputStream(XML_FILE);
            sp.parse(is,this);

        }catch(SAXException se) {
            se.printStackTrace();
        }catch(ParserConfigurationException pce) {
```

```
            pce.printStackTrace();
        }catch (IOException ie) {
            ie.printStackTrace();
        }
    }

    /**
     * Prints data after XML parsing
     */
    private void printData(){
    }

    /**
     * Event Handler
     */
    public void startElement(String uri, String
localName, String qName, Attributes attributes) throws
SAXException {
        //reset
        tempVal = "";
        if(qName.equalsIgnoreCase("dictionary")) {
            System.out.println("Valid document");
            //attributes.getValue("type"));
        }
    }

    /**
     * Helper function
```

```
     */
    public void characters(char[] ch, int start, int
length) throws SAXException {
        tempVal = new String(ch,start,length);
    }

    /**
     * parses child-elements
     */
    public void endElement(String uri, String localName,
String qName) throws SAXException {
        if(qName.equalsIgnoreCase("verb")) {
            verbs.addElement(new
SmallWord(tempVal,"?"));
        //}else if (qName.equalsIgnoreCase("Proverb"))
{
        //   proverbs.addElement(tempVal);
        }
    }

    /**
     * Method to generate an XML file. Takes as input
two vectors. One holding verbs
     * and one holding proverbs. A boolean 'print_out'
specifies whether the generated
     * XML will be printed to the screen.
     *
     * @param in_verbs The vector holding verbs
     * @param in_proverbs The vector holding proverbs
```

```
     * @param print_out A boolean specifying if the XML
will be printed to the screen
     * @return a String
     */
    public String createXML(Vector<SmallWord> in_verbs,
boolean print_out) {

        boolean virgin = true;

        String str = "<dictionary>\n";
        if (print_out)
            System.out.println(str);

        int i;
        for (i = 0; i< in_verbs.size(); i++) {
            String temp = "";
            if
(in_verbs.elementAt(i).getType().equals("class")) {
                if (!virgin)
                    temp = "  </class>\n";
                temp += "    <class id= \"" +
in_verbs.elementAt(i).getWord() + "\">\n";
                virgin = false;
            }
            else
                temp = "              <instance>" +
in_verbs.elementAt(i).getWord() + "</instance>\n";

            if (print_out)
```

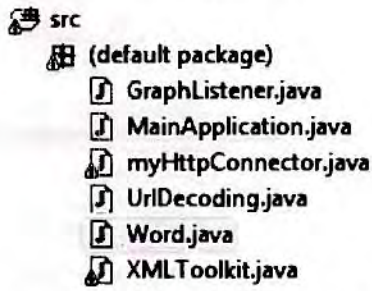
```
                    System.out.println(temp);
                str += temp;
            }
            /*
            for (i = 0; i< in_proverbs.size(); i++) {
                String temp = "<proverb>" +
in_proverbs.elementAt(i) + "</proverb>\n";
                if (print_out)
                    System.out.println(temp);
                str += temp;
            }
            */

            String temp = "  </class>\n";
            if (print_out)
                System.out.println(temp);
            str += temp;

            temp = "</dictionary>\n";
            if (print_out)
                System.out.println(temp);
            str += temp;
            return str;
        }

        /**
         * A testing method
         */
```

```
    private void testXMLCreation() {


    }


    /**
     * The starting point of this application
     * @param args
     */
    public static void main(String[] args){
         XMLToolkit spe = new XMLToolkit();
         spe.testXMLCreation();
         //spe.parseDocument(XML_FILE);
    //   spe.printData();
    }

}
```

7.2.4 Κύρια εφαρμογή

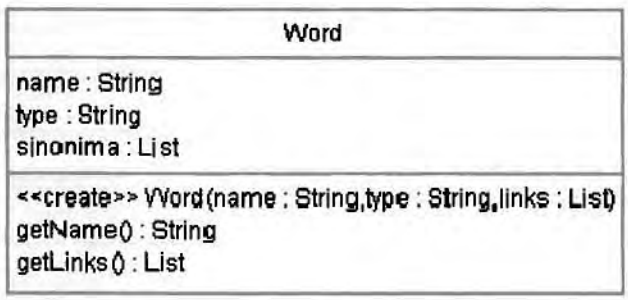
Σκοπός της κύριας εφαρμογής είναι να παρέχει ένα γραφικό περιβάλλον μέσα από το οποίο ο χρήστης θα μπορεί να τοποθετεί το XML αρχείο και να πατά κάποιο κουμπί για να χρησιμοποιεί τις πληροφορίες αυτές και να αναζητά δεδομένα. Επίσης θα χρειαστεί κάποιο τρόπο να απεικονίσει τα δεδομένα από το Wikipedia/Wiktionary Τέλος θα πρέπει να του εμφανίζεται το τελικό XML αρχείο με τις νέες προσθήκες.

Μέχρι τώρα έχουμε υλοποιήσει μηχανισμούς για XML handling αλλά και για το Percent endoding που χρησιμοποιείται σε URLs. Επειδή ο κώδικας της εφαρμογής (εικόνα 7.15) είναι μεγάλος θα παρουσιαστούν ορισμένα μόνο χαρακτηριστικά του στο κεφάλαιο αυτό, κάποια μέρη που να δείχνουν τη σωστή λειτουργία του και θα παρουσιαστεί η μεθοδολογία που χρησιμοποιήθηκε.



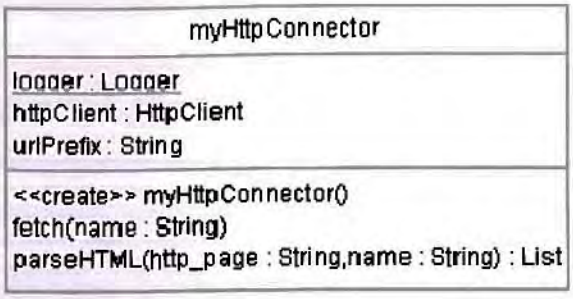
Εικόνα 7.15 Οι κλάσεις της εφαρμογής

Μέχρι τώρα έχουν παρουσιαστεί αναλυτικά τα αρχεία XMLToolkit.java και UrlDecoding.java. Το επόμενο αρχείο που θα παρουσιαστεί είναι το Word.java . Για απλοποίηση θα χρησιμοποιηθούν UML διαγράμματα (εικόνα 7.16).



Εικόνα 7.16 Πληροφορίες σχετικές με τη λέξη

Για κάθε λέξη στο XML αρχείο θα δημιουργείται ένα αντικείμενο τύπου Word. Το αντικείμενο αυτό θα κρατά πληροφορίες σχετικές με τη λέξη αυτή, καθώς και όλες τα links στο Wikipedia/Wiktionary με τις συνώνυμες με αυτή λέξεις (εικόνα 7.17).



Εικόνα 7.17 Επιστροφή τύπου Word που θα περιέχει όλες τις συνώνυμες σε αυτή λέξεις

Η κλάσση myHttpConnector.java έχει τρείς μεταβλητές και δύο μεθόδους. Η μεταβλητή urlPrefix περιέχει το http://el.wiktionary.org/wiki/, η μεταβλητή httpClient αποτελεί ένα instance της κλάσσης της βιβλιοθήκης HttpClient που παρουσιάστηκε στο κεφάλαιο 2.2

Η κύρια εργασία του παραπάνω αρχείου είναι να δέχεται μια λέξη από το XML αρχείο και να επιστρέφει ένα αντικείμενο τύπου Word που θα περιέχει όλες τις συνώνυμες σε αυτήν λέξεις. Αυτό το επιτυγχάνει με τις δύο μεθόδους fetch και parseHTML που είναι ιδιαιτέρως σημαντικές για την εφαρμογή μας.

Η μέθοδος fetch παίρνει σαν είσοδο μια απλή λέξη π.χ. τη λέξη «δοκιμή». Την μετατρέπει σε Percent encoding, δηλαδή σε :

%CE%94%CE%BF%CE%BA%CE%B9%CE%BC%CE%AE

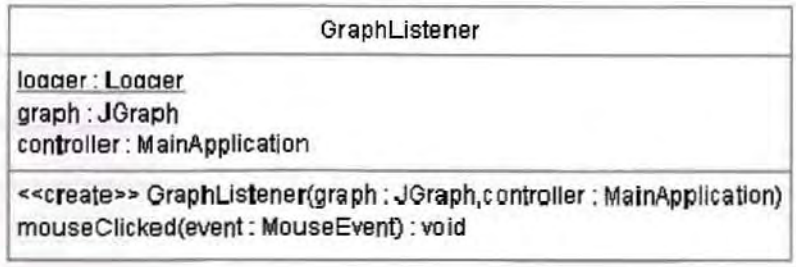
και προσθέτει στην αρχή της το url του wiktionary.

http://el.wiktionary.org/wiki/%CE%94%CE%BF%CE%BA%CE%B9%CE%BC%CE%AE

Κατόπιν καλεί τη μέθοδο GET για να κατεβάσει τη σελίδα από το wiktionary.

Αν ο υπολογιστής που τρέχει την εφαρμογή έχει σύνδεση στο internet τότε η σελίδα κατεβαίνει και καλείται η μέθοδος parseHTML. Η μέθοδος αυτή περιέχει κώδικα που καταλαβαίνει κάνοντας parse όλα τα links, δηλαδή όλες τις χρήσιμες πληροφορίες που περιέχει η σελίδα. Καταλαβαίνει αν υπάρχουν πληροφορίες, και εξάγει με ποσοστό 80% επιτυχία μόνο τις συνώνυμες λέξεις.

Τέλος δημιουργεί και επιστρέφει αντικείμενο τύπου Word με τις κατάλληλες μεταβλητές ώστε να εμφανιστούν στην οθόνη του χρήστη οι συνώνυμες λέξεις της οντολογίας από το XML.



Εικόνα 7.18 Εμφανίζει στην οθόνη του χρήστη τις συνώνυμες λέξεις

Η κλάσση GraphListener.java υλοποιήθηκε για να μπορεί ο χρήστης να έχει διάδραση με την οπτικοποιημένη αναπαράσταση των δεδομένων που λαμβάνονται από το Wiktionary (εικόνα 7.18).

Όπως είπαμε και νωρίτερα, η εφαρμογή χρησιμοποιεί τη βιβλιοθήκη JGraph για να οπτικοποιήσει τα δεδομένα που λαμβάνει. Η κάθε λέξη της οντολογίας αναπαριστάται με ένα τετράγωνο πλαίσιο και υπάρχουν γραμμές προς άλλα τετράγωνα πλαίσια που αναπαριστούν τις συνώνυμες λέξεις της εφαρμογής. Αν θελήσουμε να αναπτύξουμε κώδικα ώστε ο χρήστης με δεξί ή αριστερό κλικ πάνω στα πλαίσια αυτά να περιηγείται μέσα στο wiktionary από λέξη σε λέξη με τον ίδιο τρόπο που κινείται στις HTML σελίδες (κάνοντας απλά κλικ πάνω τους) η GraphListener «ακούει» mouse-clicks και περιέχει τον απαραίτητο κώδικα για να υλοποιηθεί οποιαδήποτε μελλοντική λειτουργία.

Σε πρώτη φάση υλοποιήθηκε ακριβώς αυτό. Δηλαδή να μπορεί ο χρήστης πατώντας πάνω σε ένα πλαίσιο, να βλέπει επόμενη ομάδα συνωνύμων λέξεων. Πολλές ενδιαφέρουσες ιδέες θα μπορούσαν να υλοποιηθούν εδώ. Θα μπορούσε για παράδειγμα να εμφανιζόταν ένα menu με τις επιλογές «Άνοιξε το φυλλομετρητή» ή «Κάνε edit τη σελίδα του Wiktionary» οι οποίες θα μετάφεραν το χρήστη στις αντίστοιχες σελίδες.

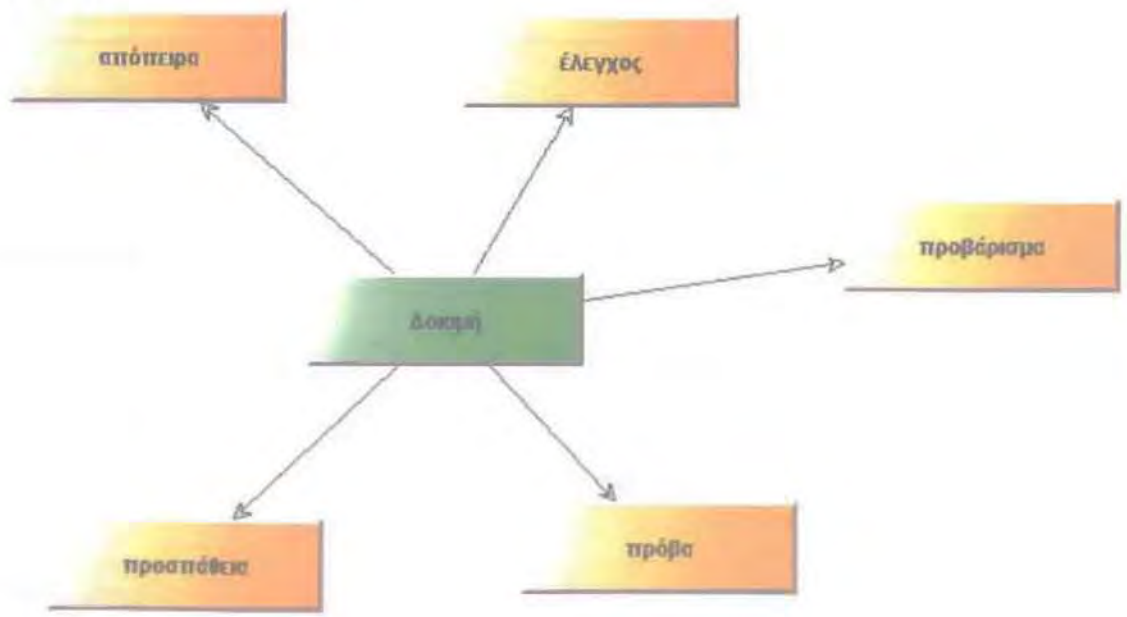
Γενικότερα ότι σχετικό με τη χρήση των Graphs μπορεί να τοποθετηθεί στην κλάση αυτή καθώς κάνει extend τον MouseAdapter και υλοποιεί μεθόδους για mouse-clicks.

MainApplication
fetcher : myHttpConnector fetchedWords : Map width : double height : double space : double frame : JFrame model : GraphModel graph : JGraph cells : ArrayList
<<create>> MainApplication() main(args : String[]) : void fetchWord(name : String) : Word display(word : Word) : void createCell(name : String,x : double,y : double) : DefaultGraphCell createEdge(source : DefaultGraphCell,target : DefaultGraphCell) : DefaultGraphCell

Εικόνα 7.19 Δημιουργία πλαισίων που θα περιέχουν τις λέξεις της οντολογίας

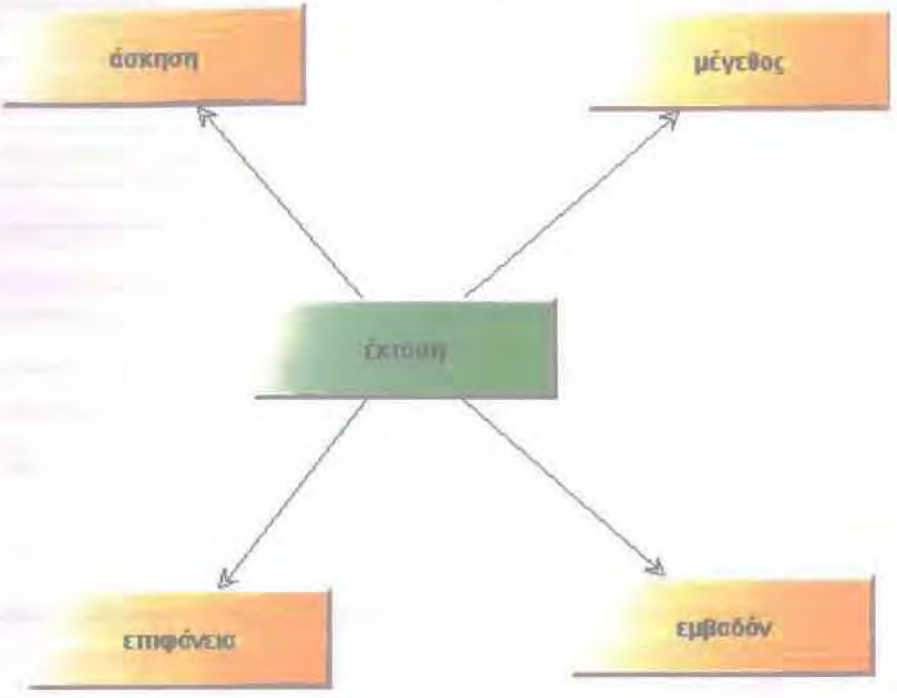
Η τελευταία κλάση της εφαρμογής μας είναι η MainApplication.java Μέσα σε αυτήν υπάρχει κώδικας για την αρχικοποίηση της εφαρμογής και για τη δημιουργία του User Interface (εικόνα 7.19).

Οι μέθοδοι createCell και createEdge περιέχουν JGraph κώδικα για τη δημιουργία πλαισίων που θα περιέχουν τις λέξεις της οντολογίας καθώς και τις συνώνυμές τους. Σε κάθε περίπτωση πρωτού εμφανιστεί μια λέξη μετατρέπεται από URL Encoding στην κανονική της εκδοχή γιατί θέλουμε να είναι κατανοητή από το χρήστη



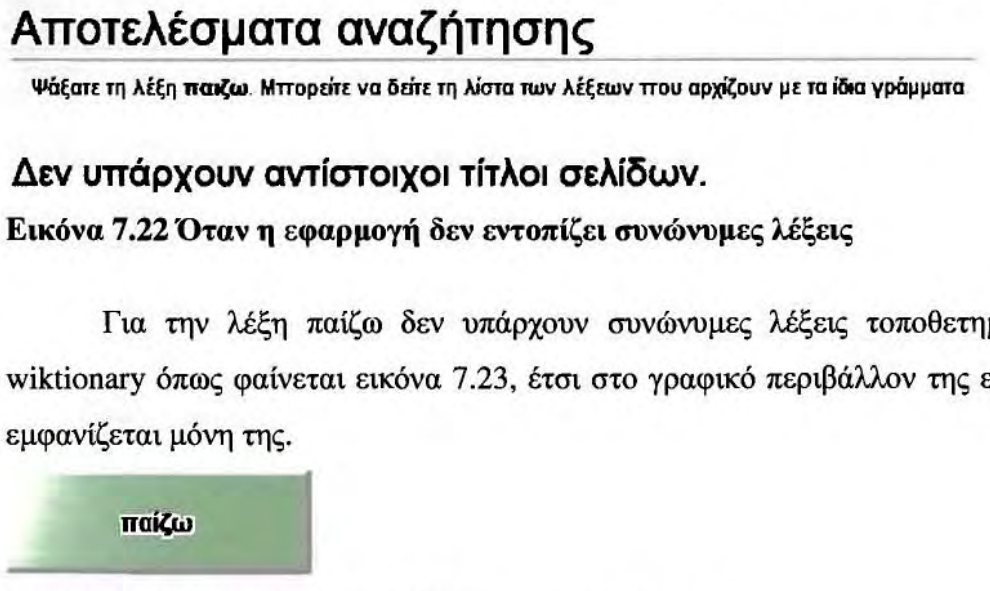
Εικόνα 7.20 Ορισμένα screenshots από την εφαρμογή

Στην εικόνα 7.20 φαίνονται για τη λέξη δοκιμή (πράσινο χρώμα) όλες οι συνώνυμες τις λέξεις (με πορτοκαλί χρώμα).



Εικόνα 7.21 Ορισμένα screenshots από την εφαρμογή

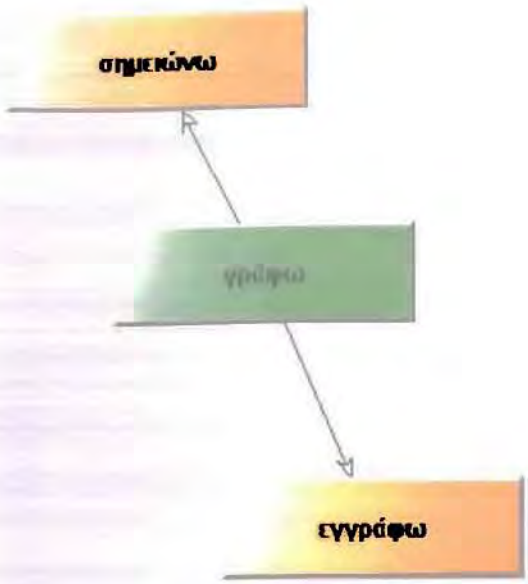
Στην εικόνα 7.21 φαίνονται για τη λέξη έκταση όλες οι συνώνυμες τις λέξεις.



Εικόνα 7.22 Όταν η εφαρμογή δεν εντοπίζει συνώνυμες λέξεις

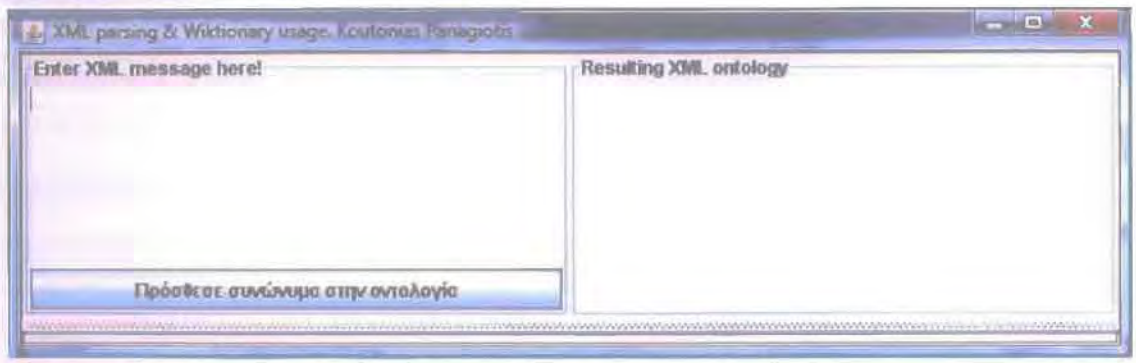
Για την λέξη παίζω δεν υπάρχουν συνώνυμες λέξεις τοποθετημένες στο wiktionary όπως φαίνεται εικόνα 7.23, έτσι στο γραφικό περιβάλλον της εφαρμογής εμφανίζεται μόνη της.

Εικόνα 7.23 Ορισμένα screenshots από την εφαρμογή



Εικόνα 7.24 Ορισμένα screenshots από την εφαρμογή

Αφού εμφανίσαμε ορισμένα παραδείγματα τα οποία λειτουργούν με τον επιθυμητό τρόπο και βεβαιωθήκαμε για το σωστό της υλοποίησης θα παρουσιάσουμε το σύνολο του γραφικού περιβάλλοντος.



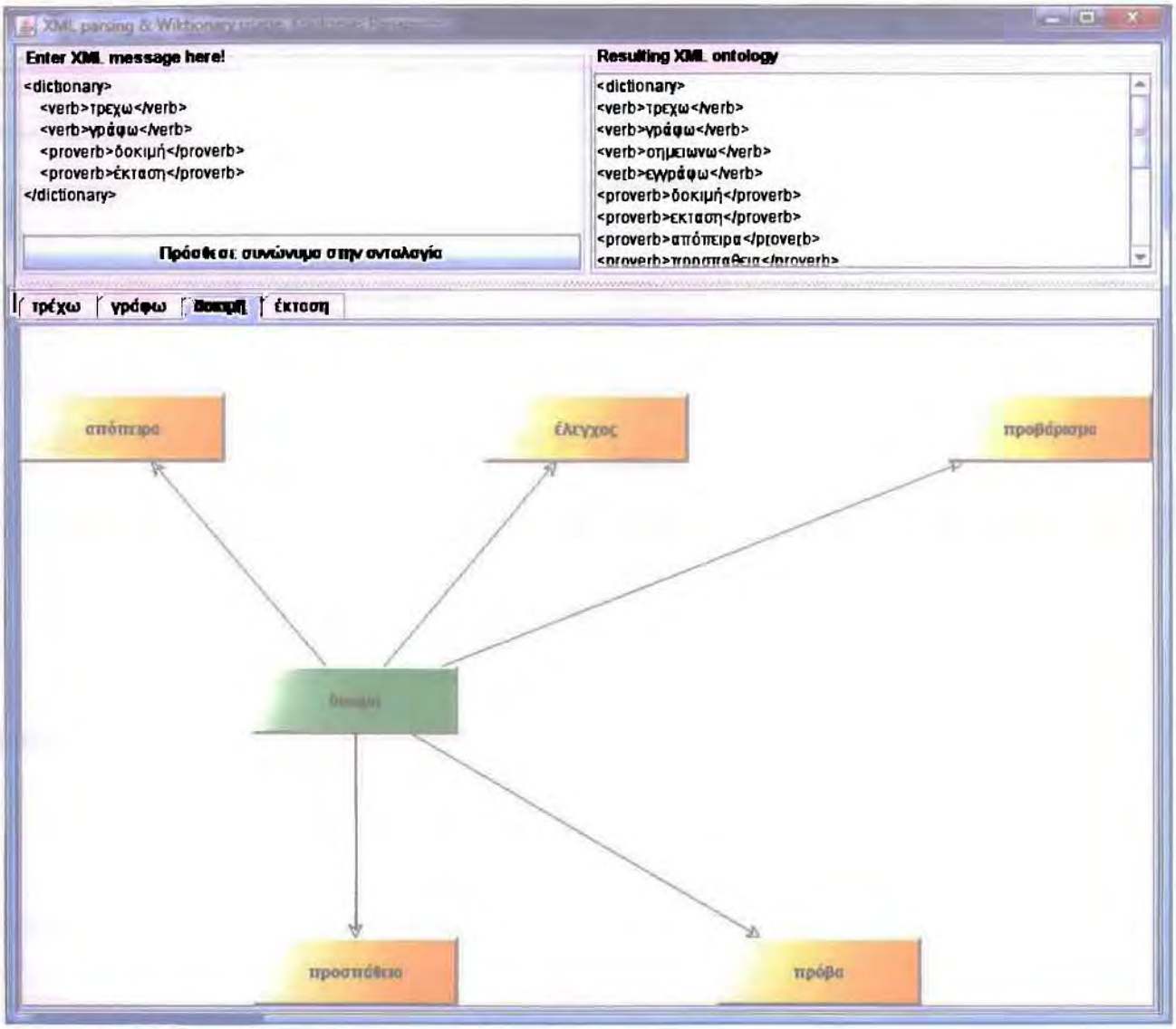
Εικόνα 7.25 Ορισμένα screenshots από την εφαρμογή

Με το που τρέχουμε την εφαρμογή μας εμφανίζεται το παραπάνω γραφικό περιβάλλον (εικόνα 7.25). Σε αυτό ο χρήστης πρέπει να τοποθετήσει στο αριστερό μέρος το XML αρχείο της οντολογίας του και κατόπιν να πατήσει το κουμπί «Πρόσθεσε συνώνυμα στην οντολογία».

Όταν κάνουμε το παραπάνω τότε:

α) Στο δεξί μέρος θα εμφανιστεί η οντολογία σαν XML μήνυμα εμπλουτισμένη με τις συνώνυμες λέξεις που ανακαλύφθηκαν.

β) Στο κάτω μέρος θα προστεθούν καρτέλες (tabs), μια για κάθε λέξη της αρχικής οντολογίας με το γράφημα της.

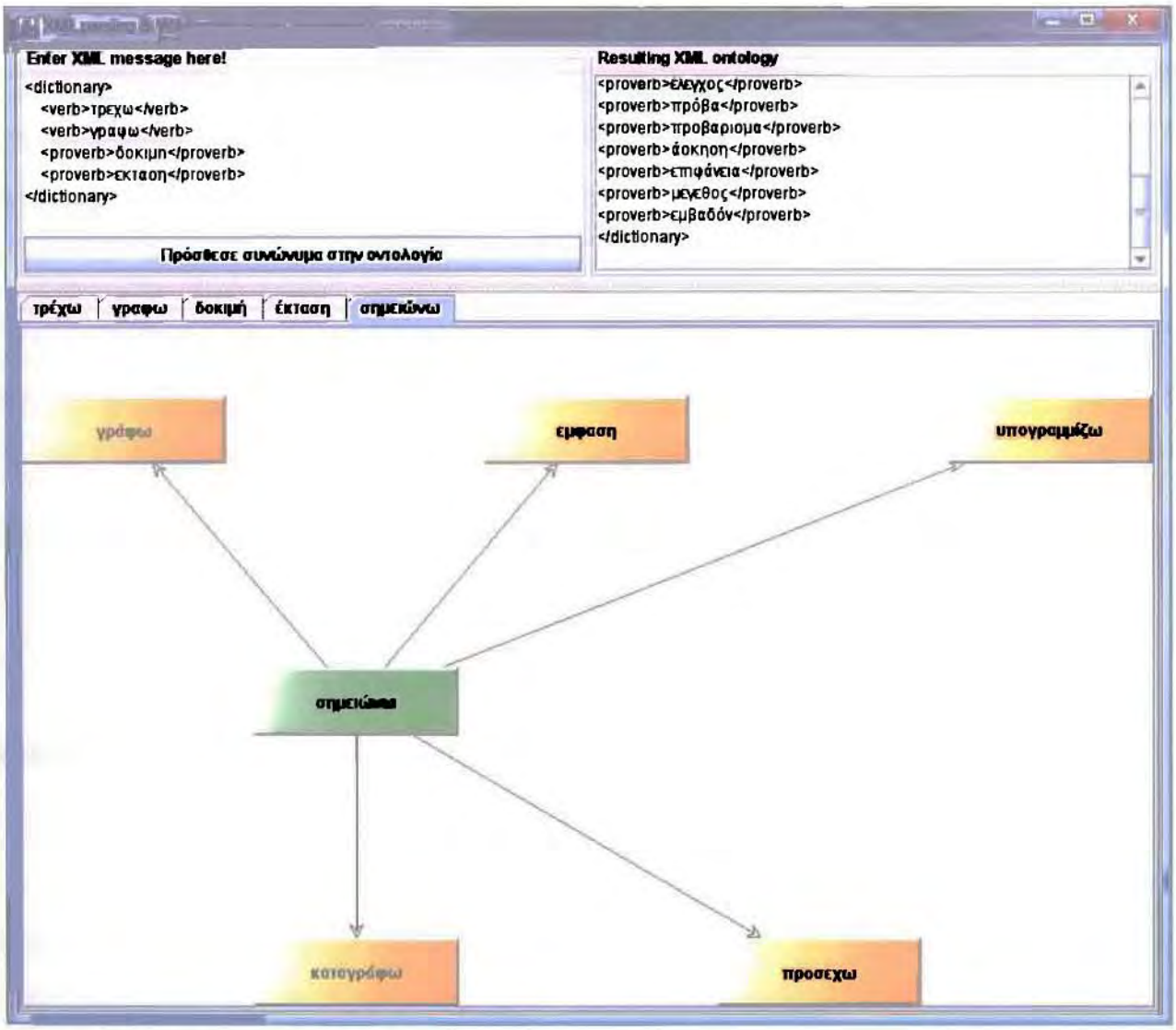


Εικόνα 7.26 Ορισμένα screenshots από την εφαρμογή

Ένα τελευταίο κομμάτι της εφαρμογής που δείχνει πόσο επεκτάσιμα έχει σχεδιαστεί είναι ότι αν ο χρήστης πατήσει με το ποντίκι πάνω σε μια από τις πορτοκαλί λέξεις, τότε γίνεται νέα αναζήτηση στο wiktionary για τη λέξη αυτή.

Μια νέα καρτέλα προστίθεται με το όνομα της λέξης αυτής, και η καρτέλα αυτή περιέχει τις συνώνυμες λέξεις αυτής που ο χρήστης πάτησε με το ποντίκι του. Αυτό υλοποιήθηκε στο GraphListener.java

Στο παράδειγμα που ακολουθεί το XML αρχείο είναι το ίδιο με το παραπάνω. Η μόνη διαφορά είναι ότι πατήθηκε το κουτί που περιείχε τη λέξη «σημειώνω» που είναι συνώνυμη της λέξης «γράφω» (εικόνα 7.27).



Εικόνα 7.27 Ορισμένα screenshots από την εφαρμογή

Η έξτρα αυτή λειτουργικότητα δεν προσθέτει τις νέες συνώνυμες λέξεις στο τελικό XML αρχείο. Υπάρχει μόνο για να διευκολύνει το χρήστη να εντοπίσει νέες συνώνυμες λέξεις.

Ο τελικός κώδικας της εφαρμογής είναι λιγότερο από1000 γραμμές. Μετρώντας με κατάλληλο εργαλείο οι «καθαρές» γραμμές κώδικα (χωρίς κενές γραμμές, και χωρίς τα σχόλια) είναι 463. Η επιτυχία που τόσα πολλά επιτεύχθηκαν με τόσο λίγο κώδικα οφείλεται στη σωστή χρήση των βιβλιοθηκών ανοιχτού κώδικα (εικόνα 7.28).

◢ Total Lines of Code	463
◢ src	463
◢ (default package)	463
MainApplication.java	152
XMLToolkit.java	106
myHttpConnector.java	93
UrlDecoding.java	57
GraphListener.java	31
Word.java	24

Εικόνα 7.28 Γραμμές κώδικα

7.2.5 ANT TASK

Για να πακεταριστεί η εφαρμογή μας σε ένα αρχείο JAR και να είναι πιο εύκολη η εκτέλεση της καθώς η εντολή που χρειάζεται αυτή τη στιγμή είναι μακροσκελής εξαιτίας των πολλών βιβλιοθηκών που χρησιμοποιεί θα δημιουργήσουμε ένα ant task που αυτοματοποιεί τα πάντα.

Πρώτα από όλα χρειάζεται ένα Manifest αρχείο:

Manifest-Version: 2.0
Main-Class: MainApplication
Class-Path: commons-codec-1.3.jar commons-httpclient-3.1.jar commons-logging-1.1.1.jar httpclient-4.0-beta1.jar httpmime-4.0-beta1.jar jgraph.jar log4j-1.2.15.jar
Created-By: 1.6 (Sun Microsystems Inc.)
Specification-Title: @Name@
Specification-Version: @version@
Specification-Vendor: @vendor@
Implementation-Vendor: Panagiotis Koutonias
Implementation-Version: @version@
Implementation-Title: WiktionaryApp

Μετά το Manifest.txt χρειάζεται να δημιουργήσουμε ένα XML αρχείο που αποτελείται από tasks, κάθε ένα από τα οποία κάνει μια συγκεκριμένη εργασία. Δημιουργήθηκε λοιπόν ένα build.xml (κώδικας στο APPENDIX) αρχείο με τα κάτωθι tasks (εικόνα 7.29):

TASK NAME	Περιγραφή – Επεξήγηση
Variables	Αρχικοποιεί μεταβλητές που θα χρησιμοποιηθούν στη συνέχεια της διαδικασίας του build, όπως βοηθητικές βιβλιοθήκες και τοποθεσία φακέλων

Init	Δημιουργεί το φάκελο build/classes όπου θα τοποθετηθούν τα .class (τα compiled αρχεία), το φάκελο build/jars όπου θα τοποθετηθούν τα .JAR αρχεία
Compile	Κάνει compile τα αρχεία .java της εφαρμογής
jar	Πακετάρει τα αρχεία .class το Manifest.txt σε ένα αρχείο WiktionaryApp.jar
Clean	Σβήνει τους φακέλους /build/classes και /build/jars «καθαρίζοντας» το περιβάλλον για δημιουργία νέας έκδοσης της εφαρμογής
Run	Εκτελεί την εφαρμογή

Εικόνα 7.29 build.xml αρχείο

Τέλος πρέπει να ειπωθεί ότι ορισμένα tasks εξαρτούνται (depend) από άλλα tasks. Δηλαδή για να τρέξει το jar-task το οποίο εξαρτάται από το init-task και το compile-task, τρέχουν πρώτα τα απαραίτητα task και κατόπιν το jar-task.

Εφόσον τρέξουμε την εντολή

ant jar

από κονσόλα MS-DOS θα δημιουργηθεί φάκελος /build/jars με τα κάτωθι αρχεία (εικόνα 7.30)

Εικόνα 7.30 Τρέχωντας την εντολή ant jar

Το WiktionaryApp.jar περιέχει το δικό μας κώδικα και το Manifest αρχείο. Για να τρέξουμε την εφαρμογή μας πηγαίνουμε σε κονσόλα DOS και πληκτρολογούμε

```
java -jar -D"file.encoding=UTF-8" WiktionaryApp.jar
```

8 Επίλογος

8.1 Μελλοντικές επεκτάσεις

Σε αυτό το σημείο, είμαστε σε θέση να κάνουμε μια πιο ολοκληρωμένη και συνολική αποτίμηση της πτυχιακής επισημαίνοντας τυχών μελλοντικές επεκτάσεις. Μπορούμε να ισχυριστούμε ότι οι γενικοί στόχοι που είχαμε θέση εκπληρώθηκαν σε ικανοποιητικό βαθμό.

Μια πρώτη παρατήρηση που έχουμε να κάνουμε είναι πως η εφαρμογή μας επεξεργαζεται ένα μικρό πλήθος λέξεων. Επιπλέον πρέπει να φτιαχτεί αντίστοιχος parser από το ελληνικό στο αγγλικό καθώς και κώδικας για να διαβάζει αγγλικές σελίδες.

Επίσης αν αλλάξει κάτι στη δομή του wikipedia τότε θα υπάρξει πρόβλημα και στην εφαρμογή μας, μιας και δημιουργήθηκε βασισμένο στη συγκεκριμένη δομή που υπήρχε εκείνη τη χρονική περίοδο που έγινε η συγγραφή της πτυχιακής.

Berners-Lee T., Hendler J., and Lassila O., (2001). The semantic web. Scientific American, 2001(5), available at http://www.sciam.com/2001/0501issue/0501berners-lee.html

Berners-Lee T., Fischetti Mark, (2000), Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web, HarperCollins Publishers, ISBN-13: 9780062515872.

Bush V., (1945), As we may think, The Atlantic Magazine, July 1945, available at: http://www.theatlantic.com/doc/194507/bush

Chisholm R. M., (1996), A Realistic Theory of Categories—An Essay on Ontology, 1 ed: Cambridge University Press.

DAML+OIL Language, (2009), DAML project. Avalaible at http://www.daml.org/

Dean M. and Schreiber G., (2004). *OWL Web Ontology Language Reference*.. W3C Recommendation 10 February 2004.

Genesereth M. R., (1991), Knowledge Interchange Format. In proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning 1991: 599-600

Gómez-Pérez A. , Angele J., Fernandez-Lopez M., Christophides V., Stutt A., Sure Y. , et al, (2002). A survey on ontology tools. OntoWeb deliverable 1.3, Universidad Politecnia de Madrid.

Gómez-Pérez A. (2004), Ontology Evaluation. Handbook on Ontologies. 251-274, Steffen Staab, Rudi Studer (Eds.), Springer

Gruber T. R. (1993). Toward principles for the design of ontologies used for knowledge sharing. Originally in N. Guarino and R. Poli, (Eds.), International Workshop on Formal Ontology, Padova, Italy. Revised August 1993. Published in International Journal of Human-Computer Studies, Volume 43 , Issue 5-6 Nov./Dec. 1995, Pages: 907-928, special issue on the role of formal ontology in the information technology.

Guarino N. (1994). The Ontological Level. In R. Casati, B. Smith and G. White (eds.), Philosophy and the Cognitive Science. Holder-Pichler-Tempsky, Vienna: 443-456.

Lassila O. and Swick. R. R. (1999) Resource description framework (RDF) model and syntax specification. W3c recommendation, W3C,. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

Menzel C., (2003), Ontology Theory, in J. Euzenat, A. Gomez-Perez, N. Guarino, and H. Stuckenschmidt (eds.), Ontologies and Semantic Interoperability, CEUR Workshop Proceedings, vol. 64 (2003).

Muskens R., Clark J. and Steve DeRose, (2005), "Natural Language Semantics". XML path language (XPath) version 1.0. Recommendation 16 November 1999, W3C, 1999.

Noy N. F., Klein M., (2003) Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems.*

Noy N.F. and Musen. M.A. (2000) Prompt: Algorithm and tool for automated ontology merging and alignment. In *Proc. 17th Natl. Conf. On Artificial Intelligence (AAAI2000)*, Austin, Texas, USA.

Ribière M. and Charlton P, (2000), Comparison of ontology languages: Ontology Overview from Motorola Labs with a comparison of ontology languages, available at (http://www.fipa.org/docs/input/f-in-00045/f-in-00045.pdf).

Roman D., Lausen H., and Keller U.. (2004) Web service modelling ontology standard (WSMO-standard). Working Draft D2v0.2, WSMO,

Smith B., Ontology (Science), in Eschenbach C. and Gruninger M. (eds.), Formal Ontology in Information Systems. Proceedings of FOIS 2008, Amsterdam/New York: ISO Press, 21–35.

Πηγές Internet

1. JGraph,

http://www.jgraph.com/

2. Model-View-Controller architecture

http://en.wikipedia.org/wiki/Model-view-controller

3. GNU Lesser General Public License,

http://www.gnu.org/copyleft/lesser.html

4. API (Application Programming Interface),

http://en.wikipedia.org/wiki/API

5. HttpClient,

http://hc.apache.org/httpcomponents-client/index.html

6. Apache Foundation,

http://www.apache.org/

7. ApacheLog4J,

http://logging.apache.org/log4j/1.2/index.html

8. ApacheCommonsLogging,

http://commons.apache.org/logging/

9. Eclipse IDE,

http://www.eclipse.org/

10. NetBeans IDE,

http://www.netbeans.org/

11. Unicode,
http://unicode.org/

12. URI,
http://en.wikipedia.org/wiki/Uniform_Resource_Identifier

APPENDIX I – Αποτελέσματα για τη σελίδα BioInformatics

<dictionary>
<instance>Bioinformatics</instance>
<instance>National_Center_for_Biotechnology_Information</instance>
<instance>Human_genome</instance>
<instance>Information_technology</instance>
<instance>Molecular_biology</instance>
<instance>Paulien_Hogeweg</instance>
<instance>DNA</instance>
<instance>DNA</instance>
<instance>Data_mining</instance>
<instance>Machine_learning</instance>
<instance>Sequence_alignment</instance>
<instance>Gene_finding</instance>
<instance>Genome_assembly</instance>
<instance>Protein_structural_alignment</instance>
<instance>Protein_structure_prediction</instance>
<instance>Gene_expression</instance>
<instance>Protein-protein_interactions</instance>
<instance>Evolution</instance>
<class id= Introduction>
</class>
<class id= Major research areas>
</class>
<class id= Sequence analysis>
<instance>Sequence_alignment</instance>
<instance>Sequence_database</instance>
<instance>Phi-X174_phage</instance>
<instance>Sequencing</instance>
<instance>DNA_sequence</instance>
<instance>Polypeptides</instance>
<instance>Species</instance>
<instance>Molecular_systematics</instance>
<instance>Phylogenetic_tree</instance>
<instance>Computer_program</instance>
<instance>Genome</instance>
<instance>Nucleotide</instance>
<instance>Sequence_alignment</instance>
<instance>Shotgun_sequencing</instance>
<instance>The_Institute_for_Genomic_Research</instance>
<instance>Human_Genome_Project</instance>
<instance>Gene_fmding</instance>
<instance>Junk_DNA</instance>
<instance>Proteome</instance>
<instance>Sequence_analysis</instance>
<instance>Sequence_profiling_tool</instance>
<instance>Sequence_motif</instance>

</class>
<class id= Genome annotation>
<instance>Gene_finding</instance>
<instance>Genomics</instance>
<instance>Haemophilus_influenzae</instance>
</class>
<class id= Computational evolutionary biology>
<instance>Evolutionary_biology</instance>
<instance>Species</instance>
<instance>DNA</instance>
<instance>Genomes</instance>
<instance>Gene_duplication</instance>
<instance>Horizontal_gene_transfer</instance>
<instance>Speciation</instance>
<instance>Evolutionary_tree</instance>
<instance>Computer_science</instance>
<instance>Genetic_algorithm</instance>
</class>
<class id= Measuring biodiversity>
<instance>Biodiversity</instance>
<instance>Biosphere</instance>
<instance>Earth</instance>
<instance>Species</instance>
<instance>Population</instance>
<instance>Habitat_(ecology)</instance>
<instance>Computer_software</instance>
<instance>Agriculture</instance>
<instance>Conservation_ecology</instance>
<instance>DNA</instance>
<instance>Genome</instance>
<instance>Endangered_species</instance>
<instance>In_silico</instance>
</class>
<class id= Analysis of gene expression>
<instance>Gene_expression</instance>
<instance>MRNA</instance>
<instance>DNA_microarray</instance>
<instance>Expressed_sequence_tag</instance>
<instance>Serial_analysis_of_gene_expression</instance>
<instance>Signal_(information_theory)</instance>
<instance>Noise</instance>
<instance>Epithelial</instance>
</class>
<class id= Analysis of regulation>
<instance>Hormone</instance>
<instance>Protein</instance>
<instance>Sequence_motif</instance>
<instance>Microarray</instance>
<instance>Cell_cycle</instance>
<instance>Cluster_analysis</instance>

<instance>Regulatory_elements</instance>
</class>
<class id= Analysis of protein expression>
<instance>Protein_microarray</instance>
<instance>Mass_spectrometry</instance>
</class>
<class id= Analysis of mutations in cancer>
<instance>Point_mutation</instance>
<instance>Gene</instance>
<instance>Cancer</instance>
<instance>Human_genome</instance>
<instance>Germline</instance>
<instance>Oligonucleotide</instance>
<instance>Comparative_genomic_hybridization</instance>
<instance>Single_nucleotide_polymorphism</instance>
<instance>Terabyte</instance>
<instance>Noise</instance>
<instance>Hidden_Markov_model</instance>
<instance>Copy_number_variation</instance>
</class>
<class id= Prediction of protein structure>
<instance>Protein_structure_prediction</instance>
<instance>Amino_acid</instance>
<instance>Primary_structure</instance>
<instance>Bovine_spongiform_encephalopathy</instance>
<instance>Mad_Cow_Disease</instance>
<instance>Prion</instance>
<instance>Secondary_structure</instance>
<instance>Tertiary_structure</instance>
<instance>Quaternary_structure</instance>
<instance>Homology_(biology)</instance>
<instance>Leghemoglobin</instance>
<instance>Structural_motif</instance>
<instance>Structural_domain</instance>
</class>
<class id= Comparative genomics>
<instance>Comparative_genomics</instance>
<instance>Genes</instance>
<instance>Endosymbiosis</instance>
<instance>Heuristics</instance>
<instance>Approximation_algorithms</instance>
<instance>Markov_Chain_Monte_Carlo</instance>
<instance>Bayesian_analysis</instance>
</class>
<class id= Modeling biological systems>
<instance>Systems_biology</instance>
<instance>Computer_simulation</instance>
<instance>Cell_(biology)</instance>
<instance>Metabolic_network</instance>
<instance>Enzyme</instance>

<instance>Metabolism</instance>
<instance>Signal_transduction</instance>
<instance>Gene_regulatory_network</instance>
<instance>Artificial_life</instance>
</class>
<class id= High-throughput image analysis>
<instance>Accuracy</instance>
<instance>Objectivity_(science)</instance>
<instance>Diagnostics</instance>
<instance>High-content_screening</instance>
<instance>Morphometrics</instance>
<instance>Gene_mapping</instance>
<instance>Sulston_score</instance>
</class>
<class id= Protein-protein docking>
<instance>Protein-protein_docking</instance>
<instance>X-ray_crystallography</instance>
<instance>Protein_nuclear_magnetic_resonance_spectroscopy</instance>
<instance>Protein-protein_interaction</instance>
<instance>Protein-protein_docking</instance>
</class>
<class id= Software and tools>
<instance>List_of_bioinformatics_companies</instance>
<instance>BLAST</instance>
<instance>National_Center_for_Biotechnology_Information</instance>
<instance>List_of_sequence_alignment_software</instance>
</class>
<class id= Web services in bioinformatics>
<instance>SOAP</instance>
<instance>REST</instance>
<instance>European_Bioinformatics_Institute</instance>
<instance>Sequence_alignment_software</instance>
<instance>Multiple_sequence_alignment</instance>
<instance>Bioinformatics#Sequence_analysis</instance>
<instance>Service-oriented</instance>
<instance>Bioinformatics_workflow_management_systems</instance>
</class>
<class id= See also>
</class>
<class id= Related topics>
<instance>Biocybernetics</instance>
<instance>Bioinformatics_companies</instance>
<instance>Biologically-inspired_computing</instance>
<instance>Biomedical_informatics</instance>
<instance>Computational_biology</instance>
<instance>Computational_biomodeling</instance>
<instance>Computational_genomics</instance>
<instance>DNA_sequencing_theory</instance>
<instance>Dot_plot_(bioinformatics)</instance>
<instance>Dry_Lab</instance>

```
    <instance>Margaret_Oakley_Dayhoff</instance>
    <instance>Metabolic_network_modelling</instance>
    <instance>Molecular_modelling</instance>
    <instance>Morphometrics</instance>
    <instance>Natural_computation</instance>
    <instance>Pharmaceutical_company</instance>
    <instance>Protein-protein_interaction_prediction</instance>
    <instance>List_of_numerical_analysis_software</instance>
    <instance>List_of_scientific_journals_in_Bioinformatics</instance>
  </class>
  <class id= Related fields>
    <instance>Applied_mathematics</instance>
    <instance>Artificial_intelligence</instance>
    <instance>Biology</instance>
    <instance>Cheminformatics</instance>
    <instance>Clinomics</instance>
    <instance>Computational_biology</instance>
    <instance>Computational_epigenetics</instance>
    <instance>Computational_science</instance>
    <instance>Computer_science</instance>
    <instance>Cybernetics</instance>
    <instance>Ecoinformatics</instance>
    <instance>Genomics</instance>
    <instance>Informatics</instance>
    <instance>Information_theory</instance>
    <instance>Mathematical_biology</instance>
    <instance>Neuroinformatics</instance>
    <instance>Proteomics</instance>
    <instance>Scientific_computing</instance>
    <instance>Statistics</instance>
    <instance>Structural_biology</instance>
    <instance>Systems_biology</instance>
    <instance>Theoretical_biology</instance>
    <instance>Veterinary_informatics</instance>
  </class>
</dictionary>
```