Master in Information Systems

Πανεπιστήμιο Μακεδονίας

Διπλωματική Εργασία

« Χρήση των μοντέλων OWL και OWL-S σε εφαρμογές ηλεκτρονικής διακυβέρνησης»

2008

Κουϊρουκίδης Νικόλαος
mis0628@uom.gr

Την αφιερώνω στους γονείς μου

και στον αδερφό μου

Περιεχόμενα

Αναπαράσταση γνώσης στο σημασιολογικό ιστο.....4

Οι γλώσσες RDF και RDF-S…….5

Σημασιολογία των RDF και RDF-S…….14

Εκφραστική αναπαράσταση με την OWL…….15

Εισαγωγή στις περιγραφικές λογικές.....27

Υπηρεσίες διαδικτύου(soap /wsdl/ uddi).....41

Υπηρεσίες οντολογίας OWL-S…….....55

Υλοποίηση περίπτωσης χρήσης σεναρίου έκδοσης άδειας οδήγησης.....61

Reasoning……71

Γλώσσες κανόνων….....88

Datalog……88

RuleML……..90

SWRL…….....95

XRML……98

SRML……..105

R2ML……106

URML……...107

ARML……110

BRML……111

WRL……..111

RIF……112

1. Αναπαράσταση Γνώσης στο Σημασιολογικό Ιστό

Ο Σημασιολογικός Ιστός (Semantic Web) αποτελεί μια επέκταση του σημερινού Ιστού. Η επέκταση αυτή έχει ως σκοπό την αυτοματοποίηση των λειτουργιών και των εφαρμογών του διαδικτύου, όπως είναι οι μηχανές αναζήτησης και οι πράκτορες (agents). Η αυτοματοποίηση αυτή μπορεί να επιτευχθεί μόνον εφόσον η γνώση και η πληροφορία που υπάρχει αποθηκευμένη και δημοσιευμένη αυτή τη στιγμή στο σημερινό Παγκόσμιο Ιστό αποκτήσει τυπικό νόημα (formal meaning) και σημασιολογία (semantics) και δομηθεί με ένα τέτοιο τρόπο ώστε να γίνεται κατανοητή από τις μηχανές που την επεξεργάζονται (machine understandable). Από τη στιγμή που η πληροφορία θα είναι δομημένη με έναν σημασιολογικά πλούσιο τρόπο θα ενισχύεται ο διαμοιρασμός (sharing) και η επαναχρησιμοποίησή της (reusability) επιτυγχάνοντας τη *διαλειτουργικότητα (interoperability)* και *συνδεσιμότητα (interconnectivity) ετερογενών (heterogeneous)* συστημάτων και των εφαρμογών.

Προκειμένου η γνώση και η πληροφορία να περιγραφεί με έναν τυπικό (formal) τρόπο ο οποίος θα δηλώνει τη σημασία της πρέπει να χρησιμοποιήσουμε γλώσσες αναπαράστασης γνώσης. Προκειμένου όμως να χρησιμοποιήσουμε τέτοιες τεχνολογίες στο διαδίκτυο θα πρέπει να αναθεωρήσουμε και να τροποποιήσουμε κάποια από τα συστατικά τους. Πιο συγκεκριμένα, όπως είναι γνωστό, στο σημερινό Ιστό κατά ένα πολύ μεγάλο ποσοστό η πληροφορία δομείται με τη χρήση της γλώσσας XML. Έτσι λοιπόν αφενός στο Σημασιολογικό Ιστό πρέπει να περιγράψουμε γνώση με τη χρήση κάποιας γλώσσας αναπαράστασης γνώσης αλλά αφετέρου η σύνταξη της γλώσσας που θα χρησιμοποιήσουμε θα πρέπει να βασίζεται στη γλώσσα XML. Για το λόγο αυτό η W3C, η οποία είναι ο οργανισμός που ασχολείται με την ανάπτυξη και προτυποποίηση τεχνολογιών για τον Παγκόσμιο Ιστό, έχει αναπτύξει δύο γλώσσες αναπαράστασης γνώσης. Οι γλώσσες αυτές είναι η RDF(S) και η OWL.

Για ποιο λόγο όμως χρειαζόμαστε δύο γλώσσες αναπαράστασης γνώσης; Η απάντηση στο ερώτημα αυτό δίνεται αν κοιτάξουμε την αρχιτεκτονική του Σημασιολογικού Ιστού. Στην αρχιτεκτονική αυτή παρατηρούμε ότι ο Σημασιολογικός Ιστός αποτελείται από στρώματα (layers). Κάθε στρώμα υλοποιεί μια λειτουργικότητα (functionality), χρησιμοποιώντας και επεκτείνοντας τη λειτουργικότητα και τις τεχνολογίες που παρέχονται από τα χαμηλότερα στρώμα. Έτσι λοιπόν στα χαμηλά επίπεδα υλοποιούνται λειτουργίες οι οποίες είναι πολύ κοντά στον Παγκόσμιο Ιστό και στις μηχανές, όπως είναι οι τεχνολογίες που ασχολούνται με τον καθορισμό και την αναγνώριση των πόρων (URIs), ενώ καθώς ανεβαίνουμε στην ιεραρχία των επιπέδων διατρέχουμε επίπεδα τα οποία υλοποιούν λειτουργικότητες αναπαράστασης γνώσης, πολύπλοκης συλλογιστικής και εμπιστοσύνης πλησιάζοντας στην ανθρώπινη γνώση και σκέψη. Θα ασχοληθούμε με τα ακόλουθα επίπεδα:

1. **Επίπεδο μεταδεδομένων:** Στο επίπεδο αυτό εισάγεται μια πολύ βασική και απλή γλώσσα αναπαράστασης γνώσης για τον Παγκόσμιο Ιστό. Το μοντέλο της γλώσσας αυτής προσφέρει ουσιαστικά μόνο τη δυνατότητα δημιουργίας ισχυρισμών (assertions) για τα στοιχεία του διαδικτύου. Οι έννοιες που εισάγονται είναι αυτές του *πόρου (resource)* και τι *ιδιότητας (property),* οι οποίες και χρησιμοποιούνται για την περιγραφή μετα-πληροφορίας χωρίς να περιγράφουν κάποια περίπλοκη γνώση. Για παράδειγμα μπορούμε να περιγράψουμε κάποιον πόρο αποδίδοντάς του μια ή περισσότερες ιδιότητες με

τις τιμές που κατέχει για τις ιδιότητες αυτές. Η γλώσσα που υλοποιεί το επίπεδο αυτό είναι η RDF (Resource Description Framework)

2. **Επίπεδο σχήματος:** Στο επίπεδο αυτό εισάγονται κάποια επιπλέον βασικά στοιχεία για την περιγραφή γνώσης στο Σημασιολογικό Ιστό. Πιο συγκεκριμένα εισάγονται για πρώτη φορά οι έννοιες της *κλάσης (class)* και της ιεραρχίας κλάσεων και ιδιοτήτων. Για να οριστούν αυτές οι έννοιες χρησιμοποιείται η λειτουργικότητα του επιπέδου μεταδεδομένων. Η γλώσσα η οποία υλοποιεί το επίπεδο αυτό είναι η γλώσσα RDF-S (RDF-Schema)
3. **Λογικό επίπεδο:** Στο επίπεδο αυτό υλοποιούνται περισσότερο εκφραστικές γλώσσες αναπαράστασης γνώσης. Οι γλώσσες αυτές χρησιμοποιούν και επεκτείνουν τη λειτουργικότητα του επιπέδου σχήματος παρέχοντας περισσότερες εκφραστικές δυνατότητες. Η γλώσσα η οποία υλοποιεί τη λειτουργικότητα του επιπέδου αυτού είναι η OWL .
4. **Επίπεδο κανόνων:** Στο επίπεδο αυτό η λειτουργικότητα των γλωσσών του λογικού επιπέδου επεκτείνεται ακόμη περισσότερο παρέχοντας τη δυνατότητα καταγραφής κανόνων.

Στη συνέχεια θα μελετήσουμε τη σύνταξη και τη σημασιολογία των γλωσσών αναπαράστασης γνώσης RDF, RDF-S και OWL.

2. Οι γλώσσες RDF και RDF-S

2.1 Περιγραφή Μεταδεδομένων με την RDF

Η RDF είναι μια γλώσσα η οποία χρησιμοποιείται για την απλή περιγραφή *πόρων (resources)* του διαδικτύου. Η περιγραφή αυτή εστιάζεται κυρίως στην απόδοση μετα-πληροφορίας για τις οντότητες αυτές, όπως είναι η περιγραφή του τίτλου, του ονόματος, της ημερομηνίας δημιουργίας και άλλων χαρακτηριστικών κάποιου πόρου του διαδικτύου. Με την έννοια πόρος αναφερόμαστε σε οποιαδήποτε οντότητα του Παγκόσμιου ιστού, όπως είναι μια ιστοσελίδα, ένα τμήμα ή ένα σύνολο από ιστοσελίδες, ηλεκτρονικά αρχεία ή ακόμα και αντικείμενα τα οποία δεν είναι άμεσα διαθέσιμα στο διαδίκτυο, όπως είναι για παράδειγμα ένα βιβλίο.

Η RDF βασίζεται στην ιδέα ότι οι πόροι οι οποίοι πρέπει να περιγραφούν έχουν *ιδιότητες (properties)* οι οποίες έχουν συγκεκριμένη τιμή. Έτσι λοιπόν μια μετα-πληροφορία για ένα πόρο αποτελείται από μια ιδιότητα και την τιμή που έχει ο πόρος για την ιδιότητα αυτή. Μια έκφραση για έναν πόρο ονομάζεται RDF *πρόταση (sentence)*. Συνοψίζοντας λοιπόν, μια RDF πρόταση αποτελείται από μια τριάδ*α (triple)* ενός *υποκειμένου (subject)*, *ιδιότητας (property)* και *αντικειμένου (object)*. Τη θέση του υποκειμένου καταλαμβάνει ο πόρος, τη θέση της ιδιότητας η ιδιότητα που του αποδίδουμε, ενώ τη θέση του αντικειμένου η τιμή που έχει ο πόρος αυτός για την ιδιότητα. Η τιμή αυτή μπορεί να είναι κάποιος άλλος πόρος ή κάποια τιμή δεδομένων. Συντακτικά οι προτάσεις αυτές δηλώνονται από μια διατεταγμένη τριάδα της μορφής, s p o. όπου τα s, p και ο αντιπροσωπεύουν το υποκείμενο, την ιδιότητα και το αντικείμενο, αντίστοιχα ενώ η τριάδα τερματίζεται με το σύμβολο της τελείας. Ένα σύνολο από τριάδες RDF μπορούμε να το αντιληφθούμε και ως έναν γράφο. Σε αυτόν τον γράφο τα αντικείμενα και τα υποκείμενα παίζουν το ρόλο των κόμβων ενώ οι ιδιότητες παίζουν το ρόλο των ακμών τους συνδέουν. Όπως αναφέραμε και στην εισαγωγή η RDF είναι μια γλώσσα αναπαράστασης γνώσης για το Σημασιολογικό Ιστό. Έτσι λοιπόν το πρότυπο της RDF καθορίζει και μια σύνταξη η οποία έχει σαν σκοπό οι RDF τριάδες να δομούνται με έναν τρόπο επεξεργάσιμο από υπολογιστικά συστήματα και εφαρμογές. Η σύνταξη αυτή δε θα μπορούσε να

βασίζεται σε άλλο πρότυπο παρά μόνο στην XML. Η σύνταξη αυτή αναφέρεται ως RDF/XML σύνταξη. Για λόγους γενικότητας η RDF χρησιμοποιεί URI references για να προσδιορίσει τις οντότητες οι οποίες βρίσκονται στη θέση του υποκειμένου, της ιδιότητας και του αντικειμένου. Ένα URI reference (URIRef) αποτελείται από ένα URI και από ένα προαιρετικό fragment identifier. Για παράδειγμα το URIRef http://www.example.org/index.html#section2 αποτελείται από το URI http://www.example.org/index.html και από τον fragment identifier Section2 τον οποίο διακρίνουμε από το URI με τη χρήση του συμβόλου #.
Ας δούμε τώρα πως μπορούμε να ορίσουμε μετα-πληροφορία σε διάφορους πόρους χρησιμοποιώντας τη γλώσσα RDF. Έστω ότι θέλουμε να περιγράψουμε τη γνώση ότι η ιστοσελίδα http://www.example.org/index.html έχει δημιουργό (creator) τον John Smith. Στην πρόταση αυτή μπορούμε να διακρίνουμε ένα υποκείμενο, τη σελίδα, ένα αντικείμενο τον John Smith και μια ιδιότητα, αυτή του δημιουργού της ιστοσελίδας. Αυτή η μετα-πληροφορία μπορεί να περιγραφεί σε μια τριάδα RDF ως εξής:

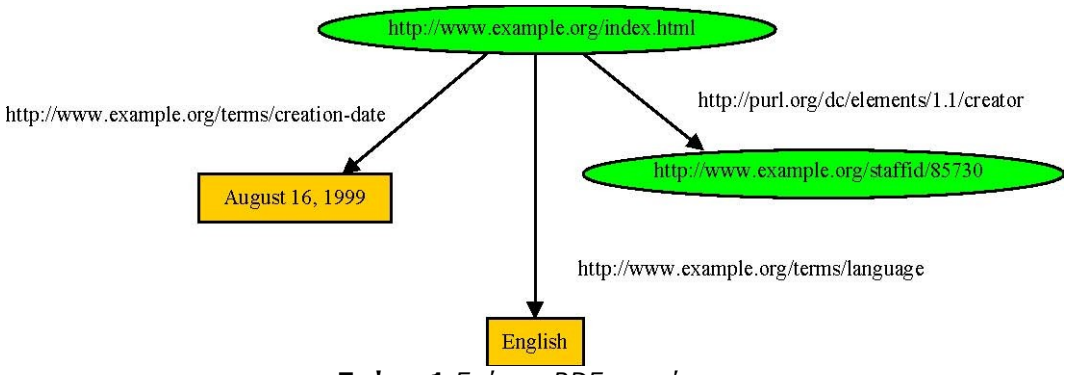
<http://.../index.html> <http://.../creator> <http://www.example.org/85730>.

όπου 85730 είναι ένα αντικείμενο που αντιπροσωπεύει τον John Smith. Παρατηρούμε ότι η RDF χρησιμοποιεί URIrefs όχι μόνο για να προσδιορίσει τον πότο τον οποίο και περιγράφουμε αλλά και την ιδιότητα και το αντικείμενο. Όπως γίνεται αντιληπτό ίσως είναι λίγο άβολο να γράφουμε συνέχεια ολόκληρα τα URI των οντοτήτων που χρησιμοποιούμε. Έτσι λοιπόν πολλές φορές χρησιμοποιούνται συντομεύσεις σε μορφή προθεμάτων (prefixes) τα οποία ονομάζονται QNames. Θα χρησιμοποιήσουμε τα QNames, rdf, rdfs, dc, ex, exterms και exdateid για να αναπαραστήσουμε τα URIs http://www.w3.org/1999/02/22-rdf-syntax-ns, http://www.w3.org/2000/01/ rdf-schema, http://purl.org/dc/elements/1.1/, http://www.example.org/, http://www.example.org/terms/, και http://www.example.org/date/, αντίστοιχα. Έτσι λοιπόν η παραπάνω τριάδα μπορεί να γραφτεί ως, ex:index.html dc:creator exstaff:85730.
Όπως είναι φυσικό αν επιθυμούμε μπορούμε να προσδώσουμε επιπλέον μετα-πληροφορίες για στον αρχικό μας πόρο. Για παράδειγμα μπορούμε να πούμε ότι η ημερομηνία δημιουργίας (creation-date) της ιστοσελίδας ήταν η 16η Αυγούστου του 1999 ή ότι η γλώσσα στην οποία είναι γραμμένη είναι τα Αγγλικά. Οι πληροφορίες αυτές μπορούν να περιγραφούν με τις ακόλουθες τριάδες:

ex:index.html exterms:creation-date "August 16, 1999".
ex:index.html exterms:language "English".
Όπως αναφέραμε προηγουμένως οι τριάδες RDF μπορούν να αναπαρασταθούν σε μια μορφή γράφου. Για το παραπάνω σύνολο τριάδων που περιγράψαμε ο RDF γράφος που δημιουργείται φαίνεται στο Σχήμα 1.
Από το γράφο αυτόν φαίνεται ότι το αντικείμενο σε μια RDF πρόταση μπορεί να είναι είτε ένα URIref είτε σταθερές τιμές οι οποίες και ονομάζονται *λεκτικά (literals)*. Είναι σημαντικό να τονίσουμε ότι ένα λεκτικό απαγορεύεται να εμφανίζεται στη θέση του υποκειμένου ή της ιδιότητας. Ας δούμε τώρα τι μορφή έχει η γνώση μας αυτή σε RDF/XML σύνταξη. Η RDF/XML μορφή των παραπάνω προτάσεων είναι η ακόλουθη:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc=http://purl.org/dc/elements/1.1/
      xmlns:exterms="http://www.example.org/terms/">
          <rdf:Description rdf:about="http://www.example.org/index.html">
          <exterms:creation-date>August 16, 1999</exterms:creation-date>
```

<exterms:language>English</exterms:language> <dc:creator rdf:resource="http://www.example.org/staffid/85730"/> </rdf:Description> </rdf:RDF>



Σχήμα 1 *Γράφος RDF προτάσεων.*

Ας εξηγήσουμε το παραπάνω RDF/XML κείμενο. Αρχικά παρατηρούμε ότι όπως σε όλα τα XML κείμενα έτσι και σε αυτό εισάγουμε τη δήλωση <?xml version="1.0"?> πριν από κάθε άλλο ορισμό. Εν συνεχεία το στοιχείο rdf:RDF δηλώνει ότι αυτό που ακολουθεί είναι ένα σύνολο από RDF προτάσεις. Μέσα στο στοιχείο αυτό δηλώνουμε επίσης και τις συντομεύσεις που θα χρησιμοποιήσουμε για τα URIrefs. Πιο συγκεκριμένα η συντόμευση rdf δηλώνει ότι οποιαδήποτε στοιχείο έχει ως πρόθεμα το rdf: αναφέρεται στο *χώρο ονομάτων (namespace)* που καθορίζεται από το URI http://www.w3.org/1999/02/22-rdf-syntax-ns#. Αντίστοιχα έχουμε ορίσει τα προθέματα dc και exterms. Στη συνέχεια ακολουθούν οι RDF/XML προτάσεις που δηλώνουν τις ματα-πληροφορίες που αναπαριστώνται από το γράφο του Σχήματος

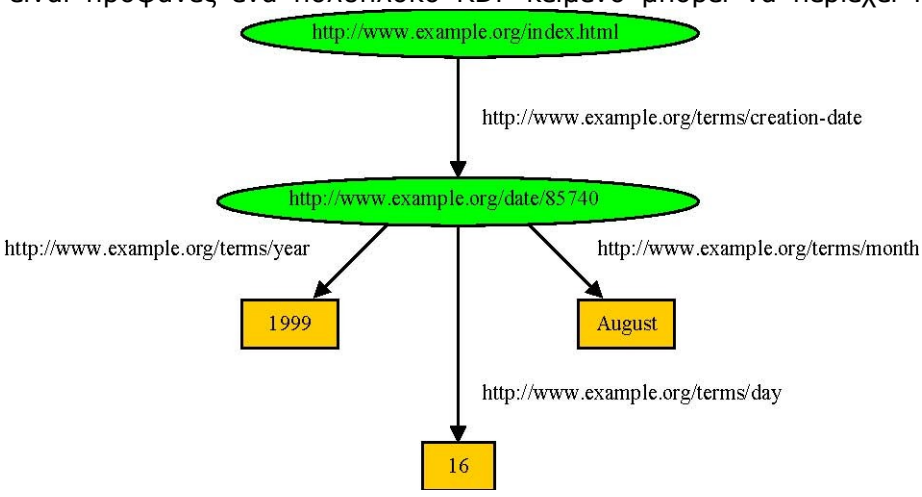
1. Πιο συγκεκριμένα το στοιχείο rdf:Description υποδηλώνει την αρχή της περιγραφής ενός πόρου. Η δήλωση αυτή συνεχίζει περιγράφοντας σε ποιόν πόρο αναφερόμαστε χρησιμοποιώντας το στοιχείο rdf:about και το URIref του πόρου. Ο πόρος αυτός αντιστοιχεί στο υποκείμενο μιας τριάδας.

Στη συνέχεια περιγράφουμε τις ιδιότητες και τις τιμές τις οποίες ο πόρος αυτός έχει για τις ιδιότητες αυτές. Πιο συγκεκριμένα στην έκτη γραμμή χρησιμοποιήσαμε ως *ετικέτα (tag)* το στοιχείο exterms:create-date για να δηλώσουμε την ιδιότητα που κατέχει ο πόρος, ενώ ως περιεχόμενο της ετικέτας, δηλαδή αυτό το οποίο βρίσκεται ανάμεσα στην ετικέτα εκκίνησης (start-tag) <exterms:creation-date> και την ετικέτα τερματισμού (end-tag) </exterms:creation-date>, αποτελεί και την τιμή του πόρου για την ιδιότητα αυτή, δηλαδή το αντικείμενο της τριάδας. Με τον ίδιο ακριβώς τρόπο εργαζόμαστε και για την ιδιότητα που περιγράφει τη γλώσσα της ιστοσελίδας, όμως όπως παρατηρούμε η σύνταξη της τρίτης ιδιότητας είναι διαφορετική. Αυτό συμβαίνει γιατί η τιμή που είχαν οι προηγούμενες δύο ιδιότητες ήταν λεκτικά ενώ στην τρίτη ιδιότητα η τιμή είναι ένας άλλος πόρος. Για να ξεχωρίσουμε τις δύο αυτές περιπτώσεις, όπως είναι φυσικό χρησιμοποιούμε διαφορετική σύνταξη. Έτσι λοιπόν, στην περίπτωση αυτή χρησιμοποιούμε το QName dc:creator για να δηλώσουμε το όνομα της ιδιότητας όμως επιπλέον χρησιμοποιούμε το στοιχείο rdf:resource που παρέχει η γλώσσα RDF για να μπορέσουμε να δηλώσουμε το URIref του πόρου με τον οποίο συνδέετε ο πόρος τον οποίο και περιγράφουμε. Είναι σημαντικό στο σημείο αυτό να σημειώσουμε ότι ενώ μπορούμε να χρησιμοποιήσουμε τα QNames για την καταγραφή του ονόματος μιας ετικέτας, η χρήση τους στις τιμές των στοιχείων απαγορεύεται από το πρότυπο της RDF/XML. Το σύνολο των στοιχείων το οποίο παρέχει η γλώσσα RDF για την περιγραφή γνώσης αποτελεί το *λεξιλόγιο (vocabulary)* της γλώσσας.

Από το προηγούμενο παράδειγμα παρατηρούμε ότι η χρήση πόρων στη θέση των αντικειμένων μας δίνει τη δυνατότητα να δημιουργήσουμε αλυσίδες από τριάδες RDF όπου το αντικείμενο της μιας τριάδας εμφανίζεται ως υποκείμενο της άλλης. Αυτό το χαρακτηριστικό είναι αρκετά χρήσιμο καθώς μας δίνεται η δυνατότητα οι τιμές που αποδίδουμε στην ιδιότητα ενός πόρου να έχουν και αυτές με τη σειρά τους κάποια περιγραφή και άρα να αναπαριστούν μια περίπλοκη δομή. Για παράδειγμα στις προηγούμενες τριάδες αποδώσαμε ως τιμή της ιδιότητας ημερομηνία δημιουργίας το λεκτικό August 16, 1999. Ίσως μια καλύτερη προσέγγιση να ήταν η ημερομηνία αυτή να αναπαρασταθεί με κάποιο πόρο, ο οποίος θα ονομάζεται ημερομηνία και ο οποίος θα έχει τις ιδιότητες, μήνας, ημέρα και έτος οι οποίες στη συγκεκριμένη περίπτωση θα έχουν τις τιμές August, 16 και 1999, αντίστοιχα. Έτσι λοιπόν μπορούμε να κάνουμε τις παρακάτω δηλώσεις,

ex:index.html	externs:creation-date	exdateid:85740.
exdateid:85740	externs:month	August.
exdateid:85740	externs:day	16.
exdateid:85740	externs:year	1999.

Η μορφή του RDF γράφου για τις παραπάνω δηλώσεις φαίνεται στο Σχήμα 2. Αυτός ο τρόπος γραφής όμως συνεπάγεται ότι οποτεδήποτε χρειαζόμαστε να αναφερθούμε σε μια ενδιάμεση οντότητα, όπως είναι η ημερομηνία, πρέπει να εισάγουμε έναν καινούργιο πόρο για την οντότητα αυτή και να του αποδώσουμε κάποιο URIref. Στην περίπτωση που ο πόρος αυτός χρησιμοποιείται μόνο μια φορά χωρίς να κατέχει κάποια παραπάνω χρησιμότητα στη γνώση μας ίσως να μην επιθυμούμε να εισάγουμε ρητά τον πόρο αυτόν αποδίδοντάς του κάποιο URIref. Σε αυτήν την περίπτωση η RDF μας παρέχει την έννοια του *κενού κόμβου (blank node).* Διαισθητικά η ύπαρξη ενός κενού κόμβου σε μια θέση υποδηλώνει την παρουσία κάποιας οντότητας στη θέση αυτή χωρίς όμως να δηλώνεται ρητά, μέσω κάποιου URIref, ποια είναι αυτή η οντότητα. Γραφικά ένας κενός κόμβος αναπαριστάται χρησιμοποιώντας το οβάλ σχήμα που χρησιμοποιούμε για τους πόρους, χωρίς όμως να αναγράφουμε κάποιο URIref.
Όπως είναι προφανές ένα πολύπλοκο RDF κείμενο μπορεί να περιέχει πολλούς



Σχήμα 2 *Δημιουργία της δομής της ημερομηνίας.*

κενούς κόμβους. Σε αυτήν την περίπτωση είναι απαραίτητο να διακρίνουμε του κενούς κόμβους μεταξύ τους. Γι αυτό το λόγο και η RDF χρησιμοποιεί τους λεγόμενους προσδιοριστές κενών *κόμβων (blank node identifiers)*. Οι προσδιοριστές

κενών κόμβων έχουν συντακτικά τη μορφή _:name, η οποία υποδηλώνει την ύπαρξη ενός κενού κόμβου και ταυτόχρονα δίνει και ένα όνομα στον κόμβο αυτό. Για παράδειγμα, αν χρησιμοποιήσουμε κενό κόμβο για την αναπαράσταση της ημερομηνίας οι τριάδες μας θα έχουν την ακόλουθη μορφή,

ex:index.html	externs:creation-date	_:pagedate.
_:pagedate	externs:month	August.
_:pagedate	externs:day	16.
_:pagedate	externs:year	1999.

ενώ η RDF/XML μορφή των τριάδων αυτών είναι η ακόλουθη,

```
<rdf:Description rdf:about="http://www.example.org/index.html">
<externs:creation-date rdf:parseType="Resource">
<externs:month>August</externs:month>
        <externs:day>16</externs:day>
        <externs:year>1999</externs:year>
        </externs:creation-date>
 </rdf:Description>
```

Η δήλωση rdf:parseType="Resource" χρησιμοποιείται για να υποδείξει ότι τα περιεχόμενα ενός στοιχείου πρέπει να ερμηνευτούν ως η περιγραφή ενός νέου κενού κόμβου. Η περιγραφή αυτού του κενού κόμβου γίνεται χωρίς να απαιτείται η δημιουργία μιας RDF περιγραφής με τη χρήση του στοιχείου rdf:Description, όπως είδαμε στο προηγούμενο παράδειγμα. Έτσι λοιπόν η δήλωση rdf:parseType="Resource" μας λέει ότι η τιμή της ιδιότητας externs:creation-date είναι ένας νέος κενός κόμβος και ότι οτιδήποτε ακολουθεί είναι οι τιμές του κενού κόμβου στις ιδιότητες που περιγράφονται. Αξίζει να αναφέρουμε στο σημείο αυτό ότι υπάρχουν και άλλοι τρόποι να αναπαραστήσουμε κενούς κόμβους στη σύνταξη RDF/XML. Πιο συγκεκριμένα μπορούμε να γράψουμε,

```
<rdf:Description rdf:about="http://www.example.org/index.html">
<externs:creation-date rdf:nodeID="abc"/>
</rdf:Description>
 <rdf:Description rdf:nodeID="abc">
            <externs:month>August</externs:month>
           <externs:day>16</externs:day>
             <externs:year>1999</externs:year>
</rdf:Description>
```

όπου το στοιχείο rdf:nodeID χρησιμοποιείται για να υποδείξει ότι αυτό που ακολουθεί είναι το όνομα ενός κενού κόμβου. Η μέθοδος αυτή έχει το πλεονέκτημα ότι αναθέτοντας ονόματα στους κενούς κόμβους μπορούμε να τους χρησιμοποιήσουμε και στις περιγραφές άλλων πόρων.

Στο προηγούμενο παράδειγμα είδαμε πως μπορούμε να δημιουργήσουμε τη δομή ημερομηνία από τα επιμέρους στοιχεία της ημέρας, του μήνα και του έτους. Όταν όμως δώσαμε τιμή στην ημέρα και στο μήνα χρησιμοποιήσαμε και για τις δύο αυτές ιδιότητες ένα λεκτικό το οποίο είναι ουσιαστικά μια συμβολοσειρά. Έτσι λοιπόν η τιμή "16" δεν αναπαριστά τον αριθμό δεκαέξι αλλά μια συμβολοσειρά της οποίας ο πρώτος χαρακτήρας είναι ο "1" και ο δεύτερος ο "6". Αυτού του είδους τα λεκτικά στην ορολογία της RDF ονομάζονται απλ*ά λεκτικά (plain literals)*. Ενώ στην περίπτωση του μήνα κάτι τέτοιο είναι αποδεκτό στην περίπτωση της ημέρας και του χρόνου θα ήταν φυσικό κάποιος να θέλει να αποδώσει ως τιμή των ιδιοτήτων αυτών

έναν αριθμό παρά μια συμβολοσειρά. Η κοινή πρακτική στις γλώσσες προγραμματισμού και στις βάσεις δεδομένων είναι η παροχή επιπρόσθετης πληροφορίας η οποία δηλώνει το πως πρέπει η τιμή που καθορίζουμε να ερμηνευτεί. Η πληροφορία αυτή δεν είναι άλλη παρά ο τύπος δεδομένων *(datatype)*, όπως για παράδειγμα οι τύποι δεδομένων δεκαδικός (decimal) και ακέραιος (integer). Τότε μια εφαρμογή η οποία καταλαβαίνει τους τύπους δεδομένων θα μπορεί να χειριστεί και την κάθε τιμή με τον κατάλληλο τρόπο. Για την παροχή της πληροφορίας αυτής η RDF παρέχει την έννοια των δακτυλογραφημένων *λεκτικών (typed literals)*.
Ένα *δακτυλογραφημένο λεκτικό* ορίζεται ως ένα ζεύγος μιας συμβολοσειράς και ενός URIref το οποίο προσδιορίζει τον τύπο δεδομένων της οντότητας που περιγράφεται από τη συμβολοσειρά. Για παράδειγμα χρησιμοποιώντας δακτυλογραφημένα λεκτικά μπορούμε να περιγράψουμε την ημερομηνία ως, _:pagedate exterms:year "1999"^^xsd:integer, όπου xsd είναι συντόμευση για το URI http://www.w3.org/2001/XMLSchema#. Βλέπουμε ότι χρησιμοποιούμε τη συμβολοσειρά «1999» για την περιγραφή του έτους, όμως επισυνάπτουμε επιπλέον την πληροφορία ότι αυτό το οποίο περιγράφεται από τη συμβολοσειρά είναι ένας ακέραιος. Από την άλλη χρησιμοποιώντας τύπους δεδομένων θα μπορούσαμε να μη δημιουργήσουμε έναν κενό κόμβο που να αναπαριστά την ημερομηνία αλλά θα μπορούσαμε να γράψουμε την τριάδα, ex:index.html exterms:creation-date "1999-08-16"^^xsd:date. Η τελευταία τριάδα σε RDF/XML σύνταξη γράφεται ως εξής,

```
<rdf:Description rdf:about="http://www.example.org/index.html">
<exterms:creation-date rdf:datatype=
"http://www.w3.org/2001/XMLSchema#date">1999-08-16
</exterms:creation-date>
 </rdf:Description>
```

Παρατηρούμε ότι χρησιμοποιούμε το στοιχείο rdf:datatype για να προσδιορίσουμε τον τύπο δεδομένων που πρέπει να χρησιμοποιηθεί για να ερμηνεύσουμε την τιμή που ακολουθεί. Όπως αναφέραμε προηγουμένως η χρήση συντομεύσεων στις τιμές ενός στοιχείου απαγορεύεται. Στην περίπτωση όμως των τύπων δεδομένων της XML μπορούμε να χρησιμοποιήσουμε τα στοιχεία *XML* (XML entities) για να βελτιώσουμε την αναγνωσιμότητα των κειμένων μας. Ο τρόπος για να το πετύχουμε αυτό είναι ο ακόλουθος. Αρχικά ορίζουμε τη συντόμευση xsd ως, <!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>. Στη συνέχεια μπορούμε να χρησιμοποιήσουμε τη συντόμευση αυτή γράφοντας &xsd;date αντί για ολόκληρο το URIref.
Μέχρι στιγμής είδαμε πως να χρησιμοποιούμε τις τριάδες RDF για να αποδίδουμε τιμές σε συγκεκριμένες ιδιότητες που κατέχει κάποιος πόρος του διαδικτύου. Αν θέλαμε να κάνουμε μια αντιστοίχηση με τις Περιγραφικές Λογικές, θα λέγαμε ότι μέχρι στιγμής έχουμε περιγράψει μόνο πως μπορούμε να δημιουργήσουμε ισχυρισμούς ρόλων. Πιο συγκεκριμένα μια τριάδα s p o. μπορεί να θεωρηθεί ως ένας ισχυρισμός της μορφής p(s,o), ο οποίος μας λέει ότι το άτομο s έχει την τιμή ο στην ιδιότητα (ρόλο) p. Το ερώτημα που τίθεται, λοιπόν, είναι αν η RDF προσφέρει κάποιο μηχανισμό για την περιγραφή σχέσεων στιγμιοτύπου ανάμεσα σε κάποιον πόρο και σε κάποια έννοια (κλάση). Η λειτουργικότητα αυτή παρέχεται στην RDF μέσω του στοιχείου rdf:type. Όταν χρησιμοποιούμε το στοιχείο rdf:type στη θέση της ιδιότητας μιας τριάδας RDF ουσιαστικά δηλώνουμε ότι αυτό το οποίο ακολουθεί στη θέση του αντικειμένου αποτελεί μια κλάση της οποίας μέλος αποτελεί ο πόρος που βρίσκεται στη θέση του υποκειμένου. Έτσι λοιπόν μπορούμε να πούμε ότι ο John Smith είναι κάποιος άνθρωπος γράφοντας την τριάδα, ex:85730 rdf:type exterms:Person. Η RDF/XML σύνταξη της τριάδα αυτής είναι ίδια με όσες έχουμε δει μέχρι στιγμής με τη μόνη διαφορά ότι στη θέση της ιδιότητας πρέπει να αναγραφεί το στοιχείο rdf:type. Το στοιχείο rdf:type όμως έχει και έναν εναλλακτικό

πιο συνοπτικό τρόπο σύνταξης στη μορφή RDF/XML. Αυτός δημιουργείται αφαιρώντας το στοιχείο rdf:type και αντικαθιστώντας το rdf:Description με το QName της κλάσης. Χρησιμοποιώντας αυτόν τον τρόπο γραφής στο προηγούμενο παράδειγμα έχουμε την ακόλουθη σύνταξη,
<exterms:Person rdf:about="ex:85730"/>

Όπως αναφέραμε και στο παράδειγμα της δημιουργίας της δομής της ημερομηνίας ένας πόρος που βρίσκεται στη θέση του αντικειμένου μπορεί σε μια άλλη τριάδα να εμφανιστεί στη θέση του αντικειμένου και έτσι μπορούμε να δημιουργήσουμε αλυσίδες από τριάδες RDF. Έτσι λοιπόν είναι καθ' όλα νόμιμο κάποιος να γράψει τις τριάδες ex:Harry rdf:type ex:Eagle και ex:Eagle rdf:type ex:EndangeredSpeciesList. Η πρώτη τριάδα μας δηλώνει ότι ο Harry είναι στιγμιότυπο της κλάσης των αετών, δηλαδή μέλος του συνόλου των αετών, ενώ η δεύτερη τριάδα μας δηλώνει ότι η κλάση των αετών είναι στιγμιότυπο της κλάσης που αναπαριστά τη λίστα των ειδών που απειλούνται προς εξαφάνιση. Αυτό σημαίνει ότι ο πόρος που αναπαριστά τους αετούς επενεργεί ταυτόχρονα και ως κλάση (έννοια) αλλά και ως άτομο. Η δυνατότητα δημιουργίας τέτοιων εκφράσεων ονομάζεται *μετα-μοντελοποίηση (metamodeling)* διότι η κλάση EndangeredSpeciesList είναι ουσιαστικά μια κλάση που επενεργεί πάνω σε μια άλλη κλάση, δηλαδή μια μετα-κλάση. Η RDF όμως προχωράει ένα βήμα παραπάνω επιτρέποντας σε έναν πόρο να χρησιμοποιηθεί σε οποιαδήποτε θέση μιας τριάδας. Προφανώς αυτή η λειτουργικότητα δεν προσφέρεται από τις ΠΛ, οι οποίες διαχωρίζουν αυστηρά το σύνολο των ατόμων, των κλάσεων (εννοιών) και το σύνολο των ιδιοτήτων (ρόλων).
Σε μερικές εφαρμογές χρειαζόμαστε τη δυνατότητα να μπορούμε να δημιουργούμε προτάσεις οι οποίες αναφέρονται σε άλλες προτάσεις RDF. Έτσι για παράδειγμα μπορούμε να περιγράψουμε το πότε δημιουργήθηκε κάποια τριάδα ή το ποιος τη δημιούργησε. Το λεξιλόγιο της RDF παρέχει τα στοιχεία για τη δημιουργία τέτοιου είδους προτάσεων. Τα στοιχεία αυτά είναι τα rdf:Statement, rdf:object, rdf:subject και rdf:predicate. Η διαδικασία δημιουργίας προτάσεων με τη χρήση του λεξιλογίου αυτού ονομάζεται *reification*. Χρησιμοποιώντας το λεξιλόγιο αυτό και εφαρμόζοντας τη διαδικασία του reification στην τριάδα ex:index.html exterms:creation-date exdateid:85740. προκύπτουν οι ακόλουθες προτάσεις RDF,

ex:triple100	rdf:type	rdf:Statement
ex:triple100	rdf:subject	ex:index.html
ex:triple100	rdf:predicate	exterms:creation-date
ex:triple100	rdf:object	exdateid:85740

Μπορούμε επίσης να περιγράψουμε πληροφορία όπως για παράδειγμα το ποιος δημιούργησε την πρόταση αυτή γράφοντας ex:triple100 dc:creator exstaff:85730. Η μορφή που έχουν οι προτάσεις αυτές σε RDF/XML σύνταξη είναι η ακόλουθη,

<rdf:Statement rdf:about="#triple100">
<rdf:subject rdf:resource=" http://www.example.org/index.html"/>
<rdf:predicate rdf:resource="http://www.example.com/terms/creation-date"/>
<rdf:object rdf:resource=" http://www.example.org/85740" />
</rdf:Statement>
Ολοκληρώνοντας την παρουσίασή μας στην RDF αξίζει να σημειώσουμε ότι η RDF παρέχει επιπλέον στοιχεία για την περιγραφή συνόλων (bags), ακολουθιών (sequences), εναλλακτικών επιλογών (alternatives) και συλλογών (collections)

πόρων. Το RDF λεξιλόγιο που παρέχεται για τον ορισμό αυτών των δομών είναι τα στοιχεία, rdf:bag, rdf:seq και rdf:alt για τη δημιουργία συνόλων, ακολουθιών και εναλλακτικών επιλογών, αντίστοιχα και τα στοιχεία, rdf:first, rdf:rest και rdf:nil για τη δημιουργία συλλογών. Για παράδειγμα μπορούμε να ορίσουμε ένα σύνολο από φοιτητές χρησιμοποιώντας τον παρακάτω τμήμα δήλωσης RDF/XML,

```
<rdf:Description rdf:about="http://example.org/courses/6.001">
    <s:students>
        <rdf:Bag>
         <rdf:li rdf:resource="http://example.org/students/Amy"/>
         <rdf:li rdf:resource="http://example.org/students/Mohamed"/>
         </rdf:Bag>
    </s:students>
 </rdf:Description>
```

Όπως παρατηρούμε από την παραπάνω σύνταξη εφόσον δε χρησιμοποιούμε κάποιο URIref στον ορισμό του συνόλου (Bag) τότε αυτό αναπαριστά έναν κενό κόμβο. Έτσι λοιπόν το στοιχείο s:students είναι μια ιδιότητα η οποία ως τιμή έχει έναν κενό κόμβο αυτόν που ορίζεται από το σύνολο.

4.2.2 Απλή Αναπαράσταση με τη Γλώσσα RDF-S

Η RDF μας παρέχει τη δυνατότητα να δημιουργήσουμε απλές προτάσεις για τους πόρους τους οποίους θέλουμε να περιγράψουμε χρησιμοποιώντας ιδιότητες, τιμές και URIref για τον προσδιορισμό των συστατικών που συμμετέχουν σε μια πρόταση. Η RDF όμως δεν παρέχει δυνατότητα να ορίσουμε και να περιγράψουμε ένα επιπλέον λεξιλόγιο το οποίο πιθανόν να επιθυμούμε να χρησιμοποιήσουμε στις εφαρμογές μας. Πιο συγκεκριμένα δεν έχουμε τη δυνατότητα να ορίσουμε τις κλάσεις (έννοιες) οι οποίες εμφανίζονται σε μια εφαρμογή. Επιπρόσθετα, είναι φυσικό να επιθυμούμε την περιγραφή των κλάσεων και των ιδιοτήτων μας δηλώνοντας σχέσεις υπαγωγής ανάμεσά τους. Η γλώσσα η οποία παρέχει τη λειτουργικότητα αυτή είναι η RDF-S. Ουσιαστικά η RDF-S παρέχει ένα επιπλέον λεξιλόγιο πάνω σε αυτό της RDF το οποίο περιλαμβάνει στοιχεία τα οποία προορίζονται στο να προσδώσουν την επιπρόσθετη αυτή λειτουργικότητα.

Η δυνατότητα ορισμού κλάσεων υλοποιείται μέσω του στοιχείου rdfs:Class του λεξιλογίου της RDF-S. Όπως είδαμε στην προηγούμενη ενότητα προκειμένου να δηλώσουμε ότι ένας πόρος είναι στιγμιότυπο μιας κλάσης χρησιμοποιήσαμε το στοιχείο rdf:type. Έτσι λοιπόν προκειμένου να δηλώσουμε μια νέα κλάση, έστω την ex:MotorVehicle, αυτό που έχουμε να κάνουμε είναι να δημιουργήσουμε την τριάδα ex:MotorVehicle rdf:type rdfs:Class, στην οποία η συντόμευση ex: αναπαριστά το URIref http://www.example.org/schemas/vehicles. Εν συνεχεία μπορούμε να δηλώσουμε ότι ο πόρος ex:Audi είναι μέλος της κλάσης ex:MotorVehicle γράφοντας, ex:Audi rdf:type ex:MotorVehicle.

Με αυτόν τον τρόπο μπορούμε να δημιουργήσουμε το δικό μας λεξιλόγιο, δηλαδή να ορίσουμε το σύνολο των κλάσεων που εμφανίζονται σε μια εφαρμογή μας. Για παράδειγμα μπορούμε επιπρόσθετα να κάνουμε τις δηλώσεις ex:Van rdf:type rdfs:Class. και ex:Truck rdf:type rdfs:Class. Παρατηρείστε ότι προκειμένου να δηλώσουμε μια κλάση πρέπει να κάνουμε χρήση της μετα-μοντελοποίησης. Αυτό γιατί η κλάση ex:Van ορίζεται ως στιγμιότυπο της μετα-κλάσης rdfs:Class. Αλλά ακόμα περισσότερο εφόσον το στοιχείο rdfs:Class είναι προφανώς και αυτό μια κλάση πρέπει να ισχύει, και όντως σε κάθε RDF-S αρχείο ισχύει, η τριάδα rdfs:Class rdf:type rdfs:Class!

Από τη στιγμή που έχουμε ορίσει τις κλάσεις μας μπορεί να μας ενδιαφέρει να ορίσουμε σχέσεις υπαγωγής ανάμεσα σε αυτές. Στην RDF-S αυτό επιτυγχάνεται με τη χρήση του στοιχείου rdfs:subClassOf. Έτσι λοιπόν μπορούμε να δημιουργήσουμε τις ακόλουθες προτάσεις,

ex:Van rdfs:subClassOf ex:MotorVehicle
ex:Truck rdfs:subClassOf ex:MotorVehicle

Η RDF/XML σύνταξη των παραπάνω ορισμών είναι η ακόλουθη,

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://example.org/schemas/vehicles">
<rdfs:Class rdf:ID="MotorVehicle"/>

<rdfs:Class rdf:ID="Truck">
<rdfs:subClassOf rdf:resource="#MotorVehicle"/>
 </rdfs:Class>

<rdfs:Class rdf:ID="Van">
<rdfs:subClassOf rdf:resource="#MotorVehicle"/>
 </rdfs:Class>
```

Παρατηρείστε ότι στο παραπάνω παράδειγμα χρησιμοποιήσαμε την περιεκτική γραφή του στοιχείου rdf:type. Αυτό μπορούμε να το κάνουμε γιατί όπως είπαμε το στοιχείο rdfs:Class είναι και αυτό μια κλάση. Έτσι λοιπόν μπορούμε να αντικαταστήσουμε το στοιχείο rdf:Description με αυτό. Τέλος παρατηρούμε ότι χρησιμοποιήσαμε ένα νέο στοιχείο το οποίο δεν έχουμε συναντήσει μέχρι στιγμής. Το στοιχείο αυτό είναι το rdf:ID. Αυτό μας επιτρέπει να μειώσουμε τα ονόματα γράφοντας μόνο τους fragment identifiers. Το URI το οποίο μπαίνει ως πρόθεμα στον fragment identifier για να δημιουργηθεί το URIref του πόρου είναι το URI του αρχείου στο οποίο βρίσκονται οι δηλώσεις ή το URI που ορίζεται από το xml:base.
Εκτός από τη δυνατότητα περιγραφής κλάσεων, μέσω σχέσεων υπαγωγής, η RDF-S παρέχει επιπλέον τη δυνατότητα δημιουργίας περιγραφών για τις ιδιότητες που επιθυμούμε να χρησιμοποιήσουμε σε μια εφαρμογή μας. Πιο συγκεκριμένα μπορούμε να περιγράψουμε σχέσεις υπαγωγής ανάμεσα σε δύο ιδιότητες, αλλά επιπρόσθετα μπορούμε να ορίσουμε το πεδίο ορισμού και το πεδίο τιμών μιας σχέσης. Το πεδίο ορισμού μας δηλώνει το τι τύπου επιτρέπεται να είναι οι πόροι οι οποίοι βρίσκονται στη θέση του υποκειμένου μιας τριάδας ενώ το πεδίο τιμών μας λέει τι τύπου είναι οι πόροι οι οποίοι βρίσκονται στη θέση του αντικειμένου. Τα στοιχεία που παρέχονται από την RDF-S για την υλοποίηση της λειτουργικότητας αυτής είναι τα στοιχεία rdfs:subPropertyOf, rdfs:domain και rdfs:range.
Έστω για παράδειγμα ότι θέλουμε να εμπλουτίσουμε την προηγούμενη γνώση για τα οχήματα εισάγοντας και περιγράφοντας τις ιδιότητες ex:driver και ex:primaryDriver. Καταρχάς μπορούμε να δηλώσουμε ότι τα στοιχεία αυτά αντιπροσωπεύουν ιδιότητες του κόσμους μας. Οι δηλώσεις αυτές μπορούν να γίνουν με τις τριάδες ex:driver rdf:type rdf:Property. και ex:primaryDriver rdf:type rdf:Property. Παρατηρήστε ότι στις προηγούμενες τριάδες δε χρησιμοποιήσαμε κανένα στοιχείο του λεξιλογίου της RDF-S. Είναι πολύ φυσικό να απαιτήσουμε η ιδιότητα ex:primaryDriver είναι υπο-ιδιότητα της ιδιότητας ex:driver. Η πληροφορία αυτή μπορεί να περιγραφεί σε μια τριάδα χρησιμοποιώντας το λεξιλόγιο της RDF-S. Η τριάδα είναι η, ex:primaryDriver rdfs:subPropertyOf ex:driver. Τέλος όπως είναι

αντιληπτό η ιδιότητα ex:driver συνδέει ένα όχημα με έναν άνθρωπο. Η γνώση αυτή μπορεί να περιγραφεί με τις τριάδες, ex:driver rdfs:domain ex:MotoveVehicle και ex:driver rdfs:domain ex:Person. Η RDF/XML μορφή των παραπάνω προτάσεων είναι η ακόλουθη,

```
<rdf:Property rdf:ID="driver">
<rdfs:domain rdf:resource="#MotorVehicle"/>
<rdfs:range rdf:resource="#Person">
</rdf:Property>

<rdf:Property rdf:ID="primaryDriver">
  <rdfs:subPropertyOf rdf:resource="#driver"/>
 </rdf:Property>
```

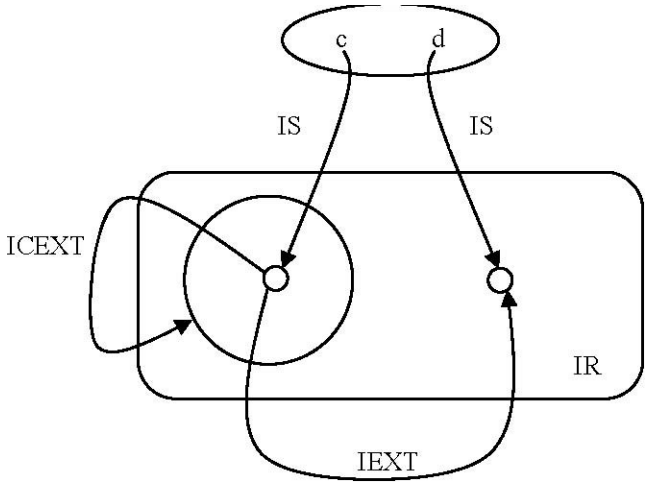
Τέλος η RDF-S μας παρέχει δύο στοιχεία τα οποία μπορούμε να χρησιμοποιήσουμε για να δημιουργήσουμε περιγραφές σε ελεύθερο κείμενο (free text) των πόρων των οποίων περιγράφουμε. Τα στοιχεία αυτά είναι το rdfs:comment και rdfs:label. Το στοιχείο rdfs:comment μπορεί να χρησιμοποιηθεί για να περιγραφθεί ένας πόρος χρησιμοποιώντας ελεύθερο κείμενο. Από την άλλη το στοιχείο rdfs:label μπορεί να χρησιμοποιηθεί για να δηλώσουμε ένα εναλλακτικό περισσότερο περιγραφικό όνομα για έναν πόρο μας. Τελειώνοντας να αναφέρουμε ότι για να αναφερθούμε στις γλώσσες RDF και RDF-S χρησιμοποιούμε πολλές φορές την ονομασία RDF(S).

4.2.3 Σημασιολογία των RDF και RDF-S

Όπως κάθε γλώσσα αναπαράστασης γνώσης έτσι και για τις γλώσσες RDF και RDF-S ορίζεται τυπική σημασιολογία (formal semantics) η οποία αποσκοπεί στο να αποδώσει νόημα (σημασία) στο λεξιλόγιο των γλωσσών RDF και RDF-S αλλά και σημασία στις προτάσεις (τριάδες) RDF. Στην περίπτωση της σημασιολογίας των Περιγραφικών Λογικών, η κατάσταση ήταν σχετικά απλή και ξεκάθαρη. Υπενθυμίζουμε ότι μία ερμηνεία σε μια ΠΛ αντιστοιχεί ένα άτομο σε ένα αντικείμενο, μια έννοια σε ένα σύνολο αντικειμένων και έναν ρόλο (ιδιότητα) σε ένα σύνολο από ζευγάρια αντικειμένων του χώρου ερμηνείας. Στην περίπτωση όμως των RDF και RDF-S η κατάσταση είναι πολύ πιο περίπλοκη. Αυτό συμβαίνει λόγω των δυνατοτήτων της μετα-μοντελοποίησης που παρέχει η RDF. Υπενθυμίζουμε ότι στη μετα-μοντελοποίηση ένας πόρος μπορεί να παίξει τόσο το ρόλο μιας κλάσης, μιας ιδιότητας αλλά και ενός ατόμου. Αυτός ο πολυμορφικός χαρακτήρας των πόρων καταστεί δύσκολη την ερμηνεία τους καθώς σε μια τέτοια περίπτωση αφενός κάποιος πόρος θα πρέπει να ερμηνευτεί ως αντικείμενο του χώρου ερμηνείας αλλά επιπλέον θα πρέπει να ερμηνευτεί και ως σύνολο αντικειμένων ή ακόμα και σύνολο ζευγαριών αντικειμένων. Η θεωρία μοντέλων (model theory) της RDF είναι αρκετά περίπλοκη και δύσκολη. Έτσι λοιπόν η παρουσίασή μας θα εστιαστεί στις βασικές έννοιες και τα συστατικά της θεωρίας αυτής. Ο αναγνώστης μπορεί στη συνέχεια να απευθυνθεί στο (Hayes 2003) για μια πλήρη παρουσίαση της θεωρίας μοντέλων της RDF.
Όπως αναφέραμε οι πόροι σε ένα RDF αρχείο μπορούν να κατέχουν διπλό ρόλο. Για να μπορέσουμε να ξεπεράσουμε τη δυσκολία αυτή οι ερμηνείες των προτάσεων και του λεξιλογίου των γλωσσών RDF και RDF-S βασίζονται σε μια απλή αλλά ταυτόχρονα περιεκτική ιδέα. Οι RDF ερμηνείες πραγματοποιούνται σε δύο στάδια. Στο πρώτο στάδιο όλοι οι πόροι ανεξάρτητα με το αν αναπαριστούν άτομα, κλάσεις ή ιδιότητες ερμηνεύονται (αντιστοιχούνται) σε αντικείμενα ενός χώρου ερμηνείας. Στο δεύτερο στάδιο για κάθε ένα από τα αντικείμενα αυτά προσδίδουμε μια *επέκταση (extension)* η οποία ουσιαστικά τους αποδίδει το τελικό τους νόημα.

Έστω για παράδειγμα η τριάδα c rdf:type c η οποία δηλώνει ότι το c είναι στιγμιότυπο του εαυτού του. Αυτό σημαίνει ότι το c συμπεριφέρεται τόσο ως άτομο

όσο και ως κλάση. Σε πρώτο βήμα χρησιμοποιώντας μια συνάρτηση ερμηνείας IS, αντιστοιχίζουμε τον πόρο c σε ένα άτομο του χώρου ερμηνείας IR, δηλαδή δημιουργούμε την αντιστοίχηση $c \rightarrow IS(c)$ όπου $IS(c) \in IR$. Αυτό συνεπάγεται ότι όσον αφορά την εμφάνιση του c ως άτομο του έχουμε αποδώσει νόημα (ερμηνεία). Σε δεύτερο βήμα χρησιμοποιούμε μια δεύτερη συνάρτηση ερμηνείας, την ICEXT, η οποία αποδίδει μια επέκταση στο αντικείμενο IS(c). Εφόσον το c συμπεριφέρεται και ως κλάση η επέκταση αυτή θα αντιστοιχίσει στο αντικείμενο IS(c) ένα σύνολο αντικειμένων του χώρου ερμηνείας. Αναθέτουμε δηλαδή στο αντικείμενο IS(c) το σύνολο ICEXT(IS(c)). Έτσι λοιπόν στο δεύτερο αυτό βήμα προσδίδουμε νόημα και στην έκφανση του πόρου c ως κλάση. Με τον ίδιο τρόπο το c θα μπορούσε σε κάποια άλλη πρόταση RDF να εμφανίζεται ως ιδιότητα. Σε αυτήν την περίπτωση θα χρησιμοποιούσαμε μια άλλη συνάρτηση ερμηνείας, την IEXT, η οποία και θα αντιστοιχίσει στο αντικείμενο IS(c) ένα σύνολο από ζευγάρια αντικειμένων του χώρου ερμηνείας. Στο Σχήμα 3 φαίνεται σχηματικά ένα παράδειγμα μιας τέτοιας ερμηνείας για τις τριάδες c rdf:type c. και c c d. οι οποίες δηλώνουν ότι το c είναι στιγμιότυπο του εαυτού του και ότι επίσης συμπεριφέρεται ως ιδιότητα που συνδέει τον εαυτό της με κάποιον άλλο πόρο d.



Σχήμα 3
Σύνθετη ερμηνεία της θεωρίας μοντέλων της RDF.

Όπως φαίνεται από το σχήμα ο πόρος c έχει ερμηνευτεί τόσο ως αντικείμενο (IS(c)), κλάση η οποία και περιέχει ως μέλος της την ερμηνεία του c ως άτομο, δηλαδή ισχύει $IS(c) \in ICEXT(IS(c))$, αλλά και ως ιδιότητα η οποία περιέχει μια σύνδεση από το αντικείμενο IS(c) στο αντικείμενο IS(d), δηλαδή ισχύει $(IS(c),IS(d)) \in IEXT(IS(c))$.

3. Εκφραστική Αναπαράσταση με την OWL

Στις προηγούμενες ενότητες είδαμε τις γλώσσες RDF και RDF-S. Είδαμε ότι το λεξιλόγιο των γλωσσών αυτών προσφέρει μια πολύ βασική εκφραστική δυνατότητα. Οι γλώσσες αυτές έχουν ουσιαστικά ως σκοπό να αποτελέσουν το θεμέλιο για πιο εκφραστικές γλώσσες ανώτερων επιπέδων, όπως είναι οι γλώσσες του λογικού επιπέδου. Μιλώντας με όρους Περιγραφικής Λογικής είδαμε ότι οι RDF και RDF-S παρέχουν τη δυνατότητα δημιουργίας σχέσεων υπαγωγής μεταξύ κλάσεων (δηλαδή εννοιών) και ιδιοτήτων (δηλαδή ρόλων) αλλά και τη δυνατότητα δημιουργίας ισχυρισμών. Επιπρόσθετα, η RDF-S παρέχει τη δυνατότητα καθορισμού του πεδίου ορισμού και του πεδίου τιμών μιας ιδιότητας με τη μορφή αξιωμάτων. Όπως θα δούμε στη συνέχεια τα δύο αυτά αξιώματα μπορούν να περιγραφούν ως δύο αξιώματα υπαγωγής στις ΠΛ.

Οι RDF και RDF-S, λοιπόν, δεν παρέχουν τη δυνατότητα να ορίσουμε κλάσεις από την ένωση, την τομή ή την άρνηση μιας ή και περισσοτέρων άλλων κλάσεων. Οι δυνατότητες αυτές και πολλές ακόμα παρέχονται από τις ΠΛ. Άρα οι ΠΛ είναι ένας καλός υποψήφιος για την δημιουργία μιας γλώσσας αναπαράστασης γνώσης για το λογικό επίπεδο. Όμως, όπως έχουμε αναφέρει ήδη μια τέτοια γλώσσα θα πρέπει να πληροί ορισμένες προϋποθέσεις για να μπορέσει να χρησιμοποιηθεί στο Σημασιολογικό Ιστό. Πιο συγκεκριμένα θα πρέπει να διαθέτει μια μορφή σύνταξης η οποία να συμβαδίζει με το πρότυπο XML. Επιπρόσθετα η γλώσσα αυτή θα πρέπει να είναι συμβατή με τις γλώσσες των χαμηλότερων επιπέδων, δηλαδή τις RDF και RDFS, αλλά ακόμα να είναι κατανοητή και εύκολη στη χρήση. Η νέα γλώσσα η οποία έρχεται να υλοποιήσει τη λειτουργικότητα του λογικού επιπέδου είναι η OWL (Web Ontology Language) Όπως θα δούμε στη συνέχεια, η OWL έχει πολύ μεγάλη σχέση με εκφραστικές ΠΛ όπως οι γλώσσες $\mathcal{SHOIN}(\mathbf{D})$ και $\mathcal{SHIF}(\mathbf{D})$.

Το πρότυπο της OWL καθορίζει ουσιαστικά τρεις υπογλώσσες αυξανόμενης εκφραστικής δυνατότητας. Οι γλώσσες αυτές είναι οι ακόλουθες:

•**OWL Lite:** Η γλώσσα αυτή απευθύνεται σε χρήστες οι οποίοι επιθυμούν να χρησιμοποιήσουν την OWL για την περιγραφή γνώσης σε εφαρμογές που δεν έχουν μεγάλες απαιτήσεις σε εκφραστικές δυνατότητες. Έτσι δίνεται η δυνατότητα ανάπτυξης εξειδικευμένων εργαλείων και μηχανισμών εξαγωγής συμπερασμάτων τα οποία αναμένεται να λειτουργούν ταχύτερα από εργαλεία τα οποία υλοποιούν περισσότερο εκφραστικές γλώσσες. Μιλώντας με όρους Περιγραφικών Λογικών θα λέγαμε ότι η γλώσσα παρέχει την ίδια εκφραστική δυνατότητα με τη γλώσσα *SHIF(***D***).*

•**OWL DL:** Η γλώσσα αυτή δίνει τη μέγιστη εκφραστική δυνατότητα που προσφέρεται από τη γλώσσα OWL χωρίς όμως να χάνονται οι καλές υπολογιστικές ιδιότητές της. Αυτό σημαίνει ότι η γλώσσα αυτή, σε αντίθεση με την τελευταία υπογλώσσα της OWL, είναι *αποφασίσιμη (decidable)*. Συγκριτικά με τις ΠΛ, η OWL DL παρέχει την ίδια εκφραστική δυνατότητα με τη γλώσσα *SHOIN(***D***).*

•**OWL Full:** Η γλώσσα αυτή προσφέρει το ίδιο λεξιλόγιο με τη γλώσσα OWL DL. Επιπρόσθετα όμως παρέχει τη συντακτική ελευθερία και τα χαρακτηριστικά της γλώσσας RDF και πιο συγκεκριμένα τη δυνατότητα μεταμοντελοποίησης. Η γλώσσα αυτή είναι εμφανώς *μη-αποφασίσιμη (undecidable)*.

Από τα παραπάνω γίνεται αντιληπτό ότι η μοναδική γλώσσα η οποία παρέχει συμβατότητα με το μοντέλο και τη σημασιολογία της RDF είναι η OWL Full. Από την άλλη όμως η μεγάλη εκφραστική δυνατότητά της την καταστεί μη-αποφασίσιμη και μέχρι σήμερα δεν είναι γνωστός κανένας αλγόριθμος εξαγωγής συμπερασμάτων γι αυτήν. Η μη-αποφασισιμότητα της OWL Full ανάγκασε τη ομάδα εργασίας τη OWL (OWL Working Group) να δημιουργήσει τις υπογλώσσες OWL Lite και OWL DL για τις οποίες βελτιστοποιημένοι αλγόριθμοι ήταν γνωστοί αλλά και υλοποιημένοι. Όπως είναι προφανές οι αλγόριθμοι αυτοί δεν είναι άλλοι παρά αλγόριθμοι εξαγωγής συμπερασμάτων για τις ΠΛ στις οποίες οι γλώσσες αυτές αντιστοιχούν.

Όπως είπαμε καθώς η OWL είναι μια γλώσσα αναπαράστασης γνώσης για το Σημασιολογικό Ιστό, πρέπει να διαθέτει μια μορφή σύνταξης που είναι συμβατή με την XML. Η σύνταξη αυτή δεν είναι άλλη από τη σύνταξη RDF/XML που είδαμε σε προηγούμενες ενότητες. Καθώς όμως η OWL παρέχει αρκετά εκφραστικούς κατασκευαστές και αξιώματα η σύνταξη αυτή γίνεται πολλές φορές αρκετά μεγάλη, περίπλοκη και με ελάχιστη διδακτική σημασία. Έτσι λοιπόν η OWL διαθέτει και μια άλλη μορφή σύνταξης η οποία αναφέρεται ως αφηρημέν*η σύνταξη (abstract*

syntax). Στις επόμενες ενότητες θα χρησιμοποιήσουμε την αφηρημένη σύνταξη για να παρουσιάσουμε τις εκφραστικές δυνατότητες που παρέχει η γλώσσα OWL. Αντίστοιχα με τις ΠΛ η OWL περιλαμβάνει ένα αλφάβητο το οποίο αποτελείται από *κλάσεις (classes)*, *ιδιότητες (properties)* και *άτομα (individuals)*. Οι κλάσεις της OWL αποτελούν ένα ανάλογο των εννοιών των Περιγραφικών Λογικών. Έτσι λοιπόν διασθητικά μια κλάση αναπαρηστά ένα σύνολο από αντικείμενα τα οποία έχουν κοινά χαρακτηριστικά, όπως για παράδειγμα η κλάση των ανθρώπων ή η κλάση των αυτοκινήτων. Όπως οι ΠΛ έτσι και η OWL περιλαμβάνει τόσο ατομικές κλάσεις όσο και *περιγραφές κλάσεων (class descriptions)*. Επιπρόσθετα, παρέχονται και *αξιώματα κλάσεων (class axioms)* όπως αξιώματα υπαγωγής, ισοδυναμίας αλλά και πολλά ακόμα που θα δούμε στην συνέχεια. Οι ιδιότητες είναι το αντίστοιχο των ρόλων στις ΠΛ, έτσι λοιπόν αναπαριστούν δυαδικές σχέσης, δηλαδή ζευγάρια αντικειμένων. Επιπρόσθετα η OWL προσφέρει τη δυνατότητα ορισμού αξιωμάτων ιδιοτήτων (property axioms), όπως αξιώματα μεταβατικών ρόλων αλλά και άλλα αξιώματα τα οποία δεν εμφανίζονται άμεσα παρά μόνο έμμεσα στις ΠΛ. Τέλος η OWL προσφέρει τη δυνατότητα ορισμού ισχυρισμών, οι οποίοι στην περίπτωση της OWL ονομάζονται *γεγονότα (facts)*.

4.3.1 Περιγραφές και Αξιώματα Κλάσεων

Η πρώτη δυνατότητα που μας παρέχει η OWL είναι η περιγραφή κλάσεων ως η *τομή (intersection)* μιας λίστας άλλων περιγραφών κλάσεων. Σε αφηρημένη σύνταξη μια τέτοια δήλωση έχει τη μορφή intersectionOf(C_1, C_2, …, C_n), όπου τα Ci, $\leq i \leq n$ είναι περιγραφές κλάσεων. Έτσι λοιπόν μια τέτοια έκφραση δηλώνει την κλάση η οποία είναι η τομή όλων των κλάσεων C_1, C_2, …, C_n. Η αντίστοιχη περιγραφή έννοιας στις ΠΛ θα ήταν η $C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$. Όπως και στην περίπτωση των ΠΛ μια τέτοια έκφραση απλώς δηλώνει την ύπαρξη μιας κλάσης χωρίς να της δίνει κάποιο όνομα. Αν θέλουμε να ονομάσουμε την κλάση αυτή θα πρέπει να χρησιμοποιήσουμε τα αξιώματα της OWL τα οποία θα δούμε αργότερα. Με παρόμοιο τρόπο η OWL μας προσφέρει τη δυνατότητα δημιουργίας περιγραφών κλάσεων από την *ένωση (union)* μιας λίστας περιγραφών κλάσεων ή το *συμπλήρωμα (complement)* μιας περιγραφής κλάσης. Στην πρώτη περίπτωση η αφηρημένη σύνταξη είναι η εξής, unionOf(C_1, C_2, …, C_n), ενώ στη δεύτερη περίπτωση η σύνταξη είναι η εξής, complementOf(C). Το συμπλήρωμα μιας κλάσης περιγραφής δηλώνει την κλάση η οποία περιλαμβάνει όλα τα αντικείμενα τα οποία δεν ανήκουν στην κλάση C.

Όπως οι ΠΛ έτσι και η OWL παρέχει περιγραφές κλάσεων μέσω περιορισμού του τύπου και του πλήθους των τιμών που μπορεί να πάρει μια ιδιότητα. Η μέθοδος αυτή ονομάζεται *περιορισμός ιδιότητας (property restriction)*. Σε αφηρημένη σύνταξη ένας περιορισμός ιδιότητας δηλώνεται με τη *λέξη κλειδί (keyword)* restriction. Στη συνέχεια δηλώνουμε την ιδιότητα στην οποία εφαρμόζουμε τον περιορισμό και τέλος δηλώνουμε τον περιορισμό που επιθυμούμε. Πιο συγκεκριμένα η OWL προσφέρει τον καθολικό και τον υπαρξιακό περιορισμό ιδιότητας. Οι περιορισμοί αυτοί δηλώνονται με τις λέξεις κλειδιά allValuesFrom και someValuesFrom ακολουθούμενες με από μια περιγραφή κλάσης. Για παράδειγμα αν θέλουμε να δηλώσουμε την κλάση των αντικειμένων τα οποία έχουν ένα παιδί το οποίο είναι Θηλυκό τότε σε αφηρημένη σύνταξη μπορούμε να γράψουμε, restriction(hasChild someValuesFrom(Female)). Παρόμοια, η κλάση των αντικειμένων που όλα τους τα παιδιά, αν έχουν, είναι είτε ψηλά είτε έξυπνα δηλώνεται ως, restriction(hasChild allValuesFrom(unionOf(Tall, Smart))). Η αντίστοιχη περιγραφή έννοιας στις ΠΛ είναι η $\forall$hasChild.(Tall$\sqcup$Smart). Επιπρόσθετα, η OWL παρέχει περιορισμούς πληθυκότητας. Όπως και οι ΠΛ η OWL προσφέρει τους τελεστές το-πολύ και το-λιγότερο όμως, επιπλέον, παρέχει και έναν άλλο τελεστή με τον οποίο μπορούμε να δηλώσουμε μια κλάση αντικειμένων περιγράφοντας την

ακριβή πληθυκότητα των μελών της σε μια ιδιότητα. Οι κατασκευαστές αυτοί δηλώνονται με τις λέξεις κλειδιά, minCardinality, maxCardinality και cardinality, αντίστοιχα, ακολουθούμενες από έναν φυσικό αριθμό ο οποίος δηλώνει την πληθυκότητα. Για παράδειγμα μπορούμε να δηλώσουμε την κλάση των αντικειμένων που περιέχουν το λιγότερο 4 παιδιά ή την κλάση των αντικειμένων που έχουν ακριβώς δύο αυτοκίνητα ως, restriction(hasChild minCardinality(4)) και restriction(hasCar cardinality(2)). Τέλος μπορούμε να περιγράψουμε κλάσεις ανάλογα με το αν συμμετέχουν σε μια σχέση με κάποιο συγκεκριμένο αντικείμενο. Σε αφηρημένη σύνταξη ο περιορισμός αυτός δηλώνεται με τη λέξη κλειδί value ακολουθούμενη από το όνομα του αντικειμένου. Για παράδειγμα, μπορούμε να περιγράψουμε την κλάση των αντικειμένων που συμμετέχουν με το αντικείμενο Αγγλία στη σχέση citizen-of. Διαισθητικά αυτό περιγράφει την κλάση που είναι κάτοικοι της Αγγλίας και η μορφή της σε αφηρημένη σύνταξη είναι η εξής, restriction(citizen-of value(England)). Παρατηρήστε ότι στην προκειμένη περίπτωση η λέξη England είναι ένα άτομο και όχι μια κλάση. Σε σύνταξη ΠΛ αυτό είναι ανάλογο με την ονοματική έννοια {England}, ενώ η παραπάνω κλάση αντιστοιχεί στην ΠΛ έννοια ∃citizen-of.{England}.

Τέλος μας παρέχεται η δυνατότητα ορισμού κλάσεων με *απαρίθμηση (enumeration)* των μελών τους. Ένα παράδειγμα είναι ο ορισμός της έννοιας (στην περίπτωση μας κλάσης) των ημερών της εβδομάδας, απαριθμώντας τα μέλη της, δηλαδή τη Δευτέρα, την Τρίτη κοκ. Στην OWL μια τέτοια κλάση μπορεί να οριστεί με τη εξής δήλωση σε αφηρημένη σύνταξη, oneof(o_1, o_2, …, o_n), όπου τα o_1, o_2, …, o_n είναι άτομα. Η δήλωση μιας τέτοιας έννοιας σε μια ΠΛ γλώσσα η οποία διαθέτει τον κατασκευαστή της ονοματικής έννοιας και τον κατασκευαστή της ένωσης έχει την εξής σύνταξη, {Δευτέρα}⊔{Τρίτη}⊔...⊔{Κυριακή}.

Ας δούμε τώρα τι μορφές αξιωμάτων κλάσεων μας προσφέρει η OWL. Όπως είναι αναμενόμενο η OWL μας δίνει τη δυνατότητα να ορίσουμε αξιώματα υπαγωγής και αξιώματα ισοδυναμίας. Η δυνατότητα περιγραφής σχέσεων υπαγωγής ήταν ήδη διαθέσιμη από την RDF-S μέσω του στοιχείου rdfs:subClassOf. Έτσι λοιπόν η OWL χρησιμοποιεί το στοιχείο αυτό για να ορίσει σχέσεις υπαγωγής ανάμεσα σε δύο κλάσεις. Η αφηρημένη σύνταξη των δηλώσεων αυτών είναι η subClassOf(C_1, C_2). Εκτός όμως από αυτήν την απλή μέθοδο δήλωσης μιας σχέσης υπαγωγής η OWL μας προσφέρει και μια επέκτασή της. Πιο συγκεκριμένα μπορούμε να δηλώσουμε μια σχέση υπαγωγής ανάμεσα σε μια κλάση και σε μια τομή από περιγραφές κλάσεων. Η δήλωση αυτή έχει τη μορφή Class(A partial C_1, C_2, …, C_n) ενώ σε ΠΛ η μορφή της
είναι η εξής, $A \sqsubseteq C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$. Έτσι λοιπόν συνδυάζοντας περιγραφές κλάσεων και το παραπάνω αξίωμα μπορούμε να δημιουργήσουμε μια δήλωση της μορφής, Class(A partial unionOf(C, D), restriction(R allValuesFrom(F))) η οποία έχει την ίδια σημασία με την δήλωση $A \sqsubseteq (C \sqcup D) \sqcap \forall R.F$, σε σύνταξη ΠΛ.

Αντίθετα από την RDF και RDF-S και παρόμοια με τις ΠΛ η OWL μας προσφέρει τη δυνατότητα να ορίσουμε σχέσεις ισοδυναμίας ανάμεσα σε κλάσεις. Πιο συγκεκριμένα μας δίνεται η δυνατότητα να ορίσουμε ισοδυναμίες ανά δύο σε μια λίστα από n κλάσεις. Οι δηλώσεις αυτές έχουν τη μορφή EquivalentClasses(C_1, C_2, …, C_n) ενώ η σημασιολογία τους είναι η ίδια με τη σημασιολογία που προσφέρεται από τις δηλώσεις $C_i \equiv C_j$, $1 \leq i < j \leq n$ σε μια ΠΛ. Όπως και στην περίπτωση των σχέσεων υπαγωγής έτσι και τώρα έχουμε τη δυνατότητα να ορίσουμε μια κλάση ως ισοδύναμη με την τομή μιας λίστας κλάσεων. Η σύνταξη των αξιωμάτων αυτών είναι η ακόλουθη, Class(A complete C_1, C_2, …, C_n), ενώ η σημασιολογία του αξιώματος αυτού είναι ίδια με αυτή του αξιώματος $A \equiv C_1 \sqcap C_2 \sqcap \ldots \sqcap C_n$ στις ΠΛ. Επιπρόσθετα χρησιμοποιώντας την περιγραφή κλάσεων με απαρίθμηση μπορούμε να δηλώσουμε μια κλάση ως την απαρίθμηση κάποιων ατόμων. Η σύνταξη είναι η, EnumeratedClass(A complete o_1, o_2, …, o_n).

Τέλος η OWL μας προσφέρει τη δυνατότητα να δηλώσουμε ότι μια λίστα από κλάσεις είναι ξένες μεταξύ τους ανά δύο. Για παράδειγμα μπορούμε να πούμε ότι οι κλάσεις Αρσενικό και Θηλυκό ή οι κλάσεις Ψηλός και Κοντός είναι ξένες μεταξύ τους. Οι δηλώσεις αυτές σχηματίζονται με τη λέξη κλειδί DisjointClasses η οποία ακολουθείται από μια λίστα από κλάσεις. Για παράδειγμα θα μπορούσαμε να γράψουμε, DisjointClasses(Αρσενικό, Θηλυκό). Τελειώνοντας, να αναφέρουμε ότι η OWL μας προσφέρει δύο κλάσεις οι οποίες αντιστοιχούν στις ΠΛ έννοιες, $\top$ και $\bot$. Οι κλάσεις αυτές στην OWL αναφέρονται με τα ονόματα owl:Thing και owl:Nothing, αντίστοιχα.

4.3.2 Περιγραφές και Αξιώματα Ιδιοτήτων

Η OWL ξεχωρίζει ανάμεσα σε δύο είδη ιδιοτήτων. Το πρώτο ονομάζεται *ιδιότητες αντικειμένων (object properties)* ενώ το δεύτερο *ιδιότητες τύπων δεδομένων (datatype properties).* Οι ιδιότητες αντικειμένων χρησιμοποιούνται για να συνδέσουν δύο άτομα μεταξύ τους, όπως για παράδειγμα η ιδιότητα εχωΠατερα συνδέει ένα άτομο με ένα άλλο που αντιπροσωπεύει τον πατέρα του, ενώ οι ιδιότητες τύπων δεδομένων συνδέουν ένα άτομο με μια τιμή τύπου δεδομένων, όπως για παράδειγμα η ιδιότητα εχωΗλικια η οποία συνδέει ένα άτομο με μια τιμή που αντιπροσωπεύει την ηλικία του. Παρόμοια με τις ΠΛ η OWL προσφέρει ένα σύνολο από αξιώματα ιδιοτήτων. Στη συνέχεια παρουσιάζουμε τα αξιώματα αυτά.

Στην περίπτωση των ιδιοτήτων τιμών δεδομένων μπορούμε μόνο να ορίσουμε μια ιδιότητα ως, DatatypeProperty(R). Παρατηρήστε, ότι στις ιδιότητες τύπων δεδομένων δε μας δίνεται η δυνατότητα να ορίσουμε μια ιδιότητα ως την αντίστροφη μιας άλλης. Αυτό συμβαίνει γιατί η αντίστροφη ιδιότητα μιας ιδιότητας τύπων δεδομένων θα συνέδεε μια τιμή με ένα αντικείμενο το οποίο δεν επιτρέπεται.

Όπως και στις ΠΛ, μας δίνεται η δυνατότητα να ορίσουμε ένα αξίωμα υπαγωγής ιδιοτήτων. Οι ορισμοί αυτοί δίνονται από την αφηρημένη σύνταξη, SubPropertyOf(R S), όπου R και S είναι ιδιότητες αντικειμένων ή τύπων δεδομένων. Επιπλέον η OWL μας προσφέρει τη δυνατότητα να ορίσουμε μια ιδιότητα ως υποιδιότητα μιας λίστας ιδιοτήτων. Η δήλωση αυτή γίνεται γράφοντας, ObjectProperty(R super(R_1) … super(R_n)), όπου οι σχέσεις R, R_1 … R_n είναι ιδιότητες αντικειμένων ή τύπων δεδομένων. Επιπρόσθετα μπορούμε να ορίσουμε μια ιδιότητα ως την αντίστροφη μιας άλλης γράφοντας, ObjectProperty(R inverseOf(S)), το οποίο δηλώνει ότι η ιδιότητα R είναι η αντίστροφη της S. Αυτό το αξίωμα σε ΠΛ αντιστοιχεί σε δύο αξιώματα της μορφής $S \sqsubseteq R^-$ και $R^- \sqsubseteq S$ τα οποία σε σημασιολογία Περιγραφικών Λογικών ερμηνεύονται ως, $\forall a,b \in \Delta^I$ αν $S^I(a,b)$ τότε $R^I(b,a)$ και αντίστροφα. Αντίθετα με τις ΠΛ στην OWL μας δίνεται η δυνατότητα να ορίσουμε ισοδυναμία μεταξύ δύο ιδιοτήτων χρησιμοποιώντας τη σύνταξη EquivalentProperties(R S), όπου R και S είναι όπως πριν. Το αξίωμα αυτό δε μας προσφέρει ουσιαστικά κάποια μεγαλύτερη εκφραστική δυνατότητα σε σχέση με τις ΠΛ αφού είναι ισοδύναμο με δύο αξιώματα υπαγωγής της μορφής, $S \sqsubseteq R$ και $R \sqsubseteq S$. Στη συνέχεια θα γράφουμε $S \equiv R$ ως συντόμευση των δύο αυτών υπαγωγών. Επιπλέον, η OWL κληρονομεί τα αξιώματα ορισμού πεδίου ορισμού και συνόλου τιμών μιας ιδιότητας αντικειμένων ή τύπων δεδομένων. Η αφηρημένη σύνταξη για αυτά τα αξιώματα είναι η ObjectProperty(R domain(C)) και ObjectProperty(R range(C)), αντίστοιχα, όπου R είναι μια ιδιότητα αντικειμένων και C είναι μια κλάση. Στην περίπτωση των ιδιοτήτων τύπων δεδομένων η λέξη κλειδί ObjectProperty αντικαθίσταται με τη λέξη κλειδί DatatypeProperty και η κλάση C στην περίπτωση του ορισμού του πεδίου τιμών αντικαθίσταται με έναν τύπο δεδομένων. Πιο συγκεκριμένα ο ορισμός γίνεται γράφοντας, DatatypeProperty(T range(d)).

Επίσης η OWL μας προσφέρει μια σειρά από αξιώματα ιδιοτήτων κάποια από τα οποία δεν εμφανίζονται άμεσα, παρά μόνο έμμεσα, στις Περιγραφικές Λογικές. Πιο συγκεκριμένα μπορούμε να δηλώσουμε ότι μια ιδιότητα αντικειμένων είναι

συμμετρική (symmetric). Μια ιδιότητα R ονομάζεται συμμετρική ανν ∀a,b, R(a,b) συνεπάγεται ότι R(b,a). Η δήλωση αυτή γίνεται γράφοντας σε αφηρημένη σύνταξη, ObjectProperty(R symmetric). Οι ΠΛ δεν παρέχουν άμεσα μια τέτοια δήλωση όμως
μπορούμε να δηλώσουμε ένα αξίωμα της μορφής R≡R⁻ το οποίο ουσιαστικά αναγκάζει την R να ερμηνευτεί ως συμμετρική σε όλες τις ερμηνείες που ικανοποιούν το αξίωμα. Μπορούμε επιπλέον να δηλώσουμε ότι μια ιδιότητα αντικειμένων R είναι μεταβατική, γράφοντας, ObjectProperty(R transitive). Επιπρόσθετα, μπορούμε να δηλώσουμε μια ιδιότητα αντικειμένων ή τύπων δεδομένων ως *συναρτησιακή (functional)*. Μια σχέση ονομάζεται συναρτησιακή αν, ∀a,b,c R(a,b) και R(a,c) συνεπάγεται ότι b=c. Για παράδειγμα η σχέση εχωΦυσικοΠατερα είναι συναρτησιακή καθώς είναι αδύνατο κάποιος να έχει παραπάνω από έναν φυσικό πατέρα. Η δήλωση αυτή γίνεται γράφοντας, ObjectProperty(R functional). Αντίθετα από τα προηγούμενα, το αξίωμα αυτό μπορεί να γίνει και σε ιδιότητες τύπων δεδομένων. Σε αυτήν την περίπτωση το αξίωμα έχει τη μορφή DatatypeProperty(T functional), όπου T είναι μια ιδιότητα τύπων δεδομένων. Για παράδειγμα η ιδιότητα τύπων δεδομένων εχωΗλικια είναι συναρτησιακή. Εκ πρώτης όψης το αξίωμα αυτό φαίνεται να μας δίνει μεγαλύτερη εκφραστική δυνατότητα από αυτή των ΠΛ. Στην πραγματικότητα όμως ένα τέτοιο
αξίωμα μπορεί σε μια ΠΛ να δηλωθεί από το αξίωμα υπαγωγής $\top \sqsubseteq \leq 1R$, το οποίο διαισθητικά δηλώνει ότι κάθε αντικείμενο του Δ^I συμμετέχει στη σχέση R το πολύ με ένα άλλο αντικείμενο. Τέλος μας δίνεται η δυνατότητα να ορίσουμε μια σχέση ως *αντίστροφη συναρτησιακή (inverse functional).* Διαισθητικά το αξίωμα αυτό δηλώνει ότι το πολύ ένα αντικείμενο μπορεί να συνδέεται με ένα άλλο αντικείμενο, δηλαδή αν ισχύουν τα R(a,c) και R(b,c) τότε πρέπει να ισχύει και a=b.

4.3.3 Άτομα

Όπως οι ΠΛ έτσι και η OWL μας παρέχει τη δυνατότητα δημιουργίας ισχυρισμών ανάμεσα σε ένα άτομο και μια κλάση, ανάμεσα σε ένα ζεύγος ατόμων και μια ιδιότητα αντικειμένων ή ανάμεσα σε ένα ζεύγος ατόμου και τιμής και μια ιδιότητα τύπων δεδομένων. Σε αφηρημένη σύνταξη οι δηλώσεις αυτές γίνονται γράφοντας Individual(o type(C_1) ... type(C_n) value(R_1, o_1) … value(R_n, o_n)), όπου τα o, o_1 έως o_n είναι άτομα τα C_1 ... C_n είναι κλάσεις και τα R, R_1 έως και R_n είναι ιδιότητες. Στη περίπτωση όπου μια ιδιότητα R_i είναι ιδιότητα τύπων δεδομένων το άτομο o_i πρέπει να αντικατασταθεί από μια τιμή, δηλαδή κάποιο plain ή typed literal. Η OWL όμως προσφέρει και επιπλέον δυνατότητες στη δημιουργία ισχυρισμών. Πιο συγκεκριμένα μας δίνεται η δυνατότητα να ορίσουμε μια λίστα από άτομα ως *ταυτοδύναμα* γράφοντας SameIndividual(o_1 … o_n) ή ότι μια λίστα από άτομα είναι διαφορετικά μεταξύ τους γράφοντας DifferentIndividuals(o_1 … o_n).

4.3.4 OWL Lite, OWL DL και OWL Full

Το παραπάνω σύνολο περιγραφής και αξιωμάτων κλάσεων και ιδιοτήτων είναι το μέγιστο δυνατό που μας παρέχει η γλώσσα OWL. Οι υπογλώσσες που το προσφέρουν είναι η OWL DL και η OWL Full. Όπως είπαμε και στην εισαγωγή η διαφορά μεταξύ OWL DL και OWL Full είναι ότι η OWL Full προσφέρει επιπλέον τη δυνατότητα μεταμοντελοποίησης. Μπορούμε δηλαδή να χρησιμοποιήσουμε ένα στοιχείο A και ως κλάση και ως ιδιότητα αλλά και ως άτομο ταυτόχρονα μέσα σε μια βάση γνώσης. Για παράδειγμα μπορεί κάποιος να γράψει τη δήλωση Individual(A type(A)). Ακόμα περισσότερο στην OWL Full μπορεί κάποιος να χρησιμοποιήσει τα ίδια τα δομικά στοιχεία της γλώσσας σε οποιαδήποτε θέση επιθυμεί. Μπορούμε δηλαδή να γράψουμε μια έκφραση της μορφής SameIndividual(A unionOf), το οποίο δηλώνει ότι το στοιχείο A είναι το ίδιο με το στοιχείο unionOf και άρα μπορεί να χρησιμοποιηθεί για να δηλώνει την ένωση μιας λίστας κλάσεων. Αντίθετα η γλώσσα OWL DL διαχωρίζει αυστηρά το σύνολο των κλάσεων, των ιδιοτήτων και των ατόμων.

Αντίθετα από τις OWL DL και OWL Full η OWL Lite προσφέρει ένα πιο περιορισμένο σύνολο περιγραφών και αξιωμάτων κλάσεων. Αυτό είχε ως σκοπό τη δημιουργία μιας λιγότερο εκφραστικής υπογλώσσας στην οποία οι αλγόριθμοι εξαγωγής συμπερασμάτων θα είναι πολύ γρήγοροι. Έτσι λοιπόν στην OWL Lite κάποια από τα δομικά στοιχεία της OWL που είδαμε σε προηγούμενες ενότητες απαγορεύεται να χρησιμοποιηθούν. Πιο συγκεκριμένα η OWL Lite απαγορεύει την χρήση των λέξεων κλειδιών, owl:oneOf, owl:unionOf, owl:complementOf, owl:hasValue και owl:dishointWith. Επιπρόσθετα, οι περιορισμοί πληθυκότητας, minCardinality και maxCardinality περιορίζονται σε συναρτησιακούς περιορισμούς. Αυτό σημαίνει ότι οι βαθμοί πληθυκότητας μπορούν να είναι μόνο το 0 και το 1. Αν θέλαμε να μιλήσουμε με όρους Περιγραφικών Λογικών θα λέγαμε ότι η OWL Lite απαγορεύει τις ονοματικές έννοιες (nominlas), οι οποίες συμβολίζονται με το γράμμα $\mathcal{O}$, ενώ αντί για τον πλήρη περιορισμό πληθυκότητας ($\mathcal{N}$) επιτρέπει μόνο συναρτησιακούς περιορισμούς ($\mathcal{F}$). Σε αυτό το σημείο ο αναγνώστης είναι απολύτως φυσικό να υποθέσει ότι η OWL Lite απαγορεύει επίσης την περιγραφή κλάσεων με τη
χρήση του συμπληρώματος (¬), το οποίο και συμβολίζεται με $\mathcal{C}$, ή τη χρήση του κατασκευαστή της ένωσης (⊔), ο οποίος υπενθυμίζουμε ότι συμβολίζεται με το γράμμα ($\mathcal{U}$). Στην πραγματικότητα όμως κάτι τέτοιο δεν είναι αληθές. Πιο συγκεκριμένα το σύνολο των στοιχείων (κατασκευαστών) που προσφέρει η OWL Lite είναι τέτοιο ώστε αυτοί οι κατασκευαστές να μπορούν να προσομοιωθούν. Για παράδειγμα έστω η έννοια A η οποία σε σύνταξη ΠΛ ορίζεται από το αξίωμα ισοδυναμίας A≡C⊓D. Αν ο χρήστης θέλει να ορίσει την άρνηση της έννοιας A μπορεί
να εργαστεί ως εξής. Ορίζει έναν νέο ρόλο, έστω τον forA. Στη συνέχεια ορίζει επιπλέον την έννοια A ως ισοδύναμη της έννοιας ≥1forA. Τέλος, ορίζει την έννοια negA ως, negA≡≤0forA. Η έννοια negA είναι το συμπλήρωμα της έννοιας, A. Αυτό συμβαίνει γιατί η έννοια ≥1forA είναι το συμπλήρωμα της έννοιας ≤0forA. Έτσι λοιπόν χρησιμοποιώντας κάθε φορά και έναν νέο ρόλο μπορούμε να ορίσουμε την άρνηση μιας έννοιας. Παρομοίως μπορούμε να εργαστούμε και για τον ορισμό της ένωσης δύο εννοιών. Συνοψίζοντας, η OWL Lite αντιστοιχεί στην ΠΛ $\mathcal{SHIF}(\mathbf{D})$ η οποία είναι μια αρκετά εκφραστική γλώσσα και άρα ο αρχικός σκοπός για τη δημιουργία μιας μη εκφραστικής γλώσσας απέτυχε.

4.3.5 RDF/XML Σύνταξη

Όπως αναφέραμε και στην εισαγωγή εφόσον η OWL είναι μια γλώσσα αναπαράστασης γνώσης για το Σημασιολογικό Ιστό εκτός από την αφηρημένη σύνταξη πρέπει να διαθέτει και RDF/XML σύνταξη. Η σύνταξη αυτή μας δίνει τη δυνατότητα να διαμορφώσουμε τη γνώση μας σε μια μορφή προσπελάσιμη από εφαρμογές του Παγκοσμίου Ιστού. Εφόσον η OWL διαθέτει αρκετά περίπλοκους κατασκευαστές η σύνταξη αυτή είναι πολλές φορές περίπλοκη. Στην ενότητα αυτή θα δούμε ορισμένα παραδείγματα της σύνταξης αυτής χωρίς να κάνουμε μια λεπτομερή παρουσίαση.

Στις περισσότερες περιπτώσεις η RDF/XML σύνταξη προκύπτει από την αντιστοίχηση των λέξεων κλειδιών της αφηρημένης σύνταξης της OWL σε στοιχεία της γλώσσας XML. Ας θεωρήσουμε για παράδειγμα τον ορισμό της κλάσης restriction(R allValuesFrom(C)), ο οποίος μας δηλώνει μια OWL κλάση εφαρμόζοντας έναν περιορισμό στην ιδιότητα R. Σε RDF/XML σύνταξη η λέξη κλειδί restriction αντιστοιχεί στο στοιχείο owl:Restriction, ενώ η λέξη κλειδί allValuesFrom αντιστοιχεί στο στοιχείο owl:allValuesFrom. Για να μπορέσουμε όμως να δηλώσουμε σε ποια ιδιότητα εφαρμόζουμε τον περιορισμό, με σκοπό τον ορισμό της κλάσης, χρειαζόμαστε ένα επιπλέον XML στοιχείο. Το στοιχείο αυτό είναι το στοιχείο owl:onProperty. Συνοψίζοντας η RDF/XML μορφή της παραπάνω δήλωσης είναι η ακόλουθη,

<owl:Restriction><owl:onProperty rdf:resource="#R" />
<owl:allValuesFrom rdf:resource="#C">
</owl:Restriction>

Παρομοίως, οι λέξεις κλειδιά Class και unionOf αντιστοιχούν στα στοιχεία owl:Class και owl:unionOf. Έτσι λοιπόν η δήλωση Class(A complete unionOf(C D)) αντιστοιχεί στη RDF/XML δήλωση,

<owl:Class rdf:ID="A">
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#C" />
<owl:Class rdf:about="#D" />
</owl:unionOf>
</owl:Class>

Έστω τώρα ότι σε αφηρημένη σύνταξη OWL κάνουμε τη δήλωση ObjectProperty(R domain(C) range(D) transitive symmetric) η οποία μας δηλώνει ότι το πεδίο ορισμού της ιδιότητας R είναι η έννοια C το πεδίο τιμών είναι η έννοια D και τέλος ότι η ιδιότητα R είναι μεταβατική και συμμετρική. Χρησιμοποιώντας τα στοιχεία XML που αντιστοιχούν στις παραπάνω λέξεις κλειδιά μπορούμε να κάνουμε τη δήλωση αυτή χρησιμοποιώντας τη σύνταξη RDF/XML. Η δήλωση αυτή είναι η ακόλουθη,

<owl:SymmetricProperty rdf:ID="R">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
<rdfs:domain rdf:resource="#C"/>
<rdfs:range rdf:resource="#D"/>
<rdf:type df:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:SymmetricProperty>

Τέλος ας δούμε ένα αξίωμα ατόμων. Έστω η δήλωση Individual(a type(C) type(D) value(R, b)) η οποία μας δηλώνει ότι το άτομο a είναι στιγμιότυπο των κλάσεων C και D και συνδέεται με το άτομο b μέσω της σχέσης R. Η αντίστοιχη δήλωση σε RDF/XML σύνταξη είναι η ακόλουθη,

<C rdf:ID="o">
<rdf:type rdf:resource="#D"/>
<R rdf:resource="#b" />
</C>

Εφόσον θεωρητικά μια OWL οντολογία πρέπει να επεκτείνει μια RDF οντολογία, ένα OWL αρχείο εκκινεί με τις ίδιες ετικέτες με τις οποίες εκκινεί και ένα RDF αρχείο. Πιο συγκεκριμένα ένα OWL αρχείο ξεκινά με την ετικέτα <rdf:RDF>. Στη συνέχεια όμως προκειμένου να δηλώσουμε ότι αυτό που ακολουθεί είναι μια OWL οντολογία χρησιμοποιούμε την ετικέτα owl:Ontology. Η ετικέτα αυτή περιέχει το στοιχείο rdf:about το οποίο αποδίδει ένα όνομα για την οντολογία. Εκτός από τα στοιχεία τα οποία είδαμε και έχουν σχέση με περιγραφή και αξιώματα κλάσεων και ιδιοτήτων η OWL προσφέρει και μια σειρά από στοιχεία τα οποία χρησιμοποιούνται για την περιγραφή μεταπληροφορίας για την οντολογία. Πιο συγκεκριμένα το στοιχείο owl:priorVersion δηλώνει ένα URL στο οποίο βρίσκεται μια OWL οντολογία η οποία είναι προηγούμενη έκδοση της οντολογίας που δηλώνεται από το τρέχον αρχείο OWL. Επίσης το στοιχείο owl:versionInfo χρησιμοποιείται για να δηλώσει πληροφορίες για το τρέχον αρχείο όπως για παράδειγμα ημερομηνία και ώρα δημιουργίας και άλλα. Τέλος ένα στοιχείο της OWL το οποίο χρησιμοποιείται αρκετά αλλά επίσης έχει αποτελέσει αμφιλεγόμενο σημείο για τους ερευνητές όσον αφορά τη σημασιολογία του είναι το στοιχείο owl:imports. Με το στοιχείο αυτό μπορούμε να

εισάγουμε τους ορισμούς, κλάσεις, αξιώματα και ιδιότητες μιας άλλης οντολογίας στο τρέχον αρχείο.

4.3.6 Σημασιολογία

Η σημασιολογία της OWL περιγράφεται στο κείμενο (Patel-Schneider et. at., 2004) του προτύπου της OWL. Η σημασιολογία αυτή αποδίδεται απευθείας στα RDF/XML στοιχεία της OWL, όπως είναι το allValuesFrom και άλλα. Αυτό συνεπάγεται ότι προκειμένου να ερμηνευτούν κάποιες περίπλοκες και περιττές δομές που χρησιμοποιούνται στην RDF/XML σύνταξη της OWL η σημασιολογία αυτή περιπλέκεται χωρίς να υπάρχει κάποιος ιδιαίτερος λόγος. Για το λόγο αυτό συνήθως για να αποδώσουμε ερμηνεία στα δομικά στοιχεία, τα αξιώματα και τους κατασκευαστές της OWL, η οποία να έχει διδακτική και ουσιαστική σημασία εργαζόμαστε διαφορετικά. Ανάγουμε τους κατασκευαστές και τα δομικά στοιχεία της OWL σε εκφραστικές Περιγραφικές Λογικές. Η αναγωγή αυτή οφείλεται στους Horrocks και Patel-Schneider (2003). Εκτός από την παροχή σημασιολογίας στους κατασκευαστές και τα αξιώματα της OWL η αναγωγή αυτή μας προσφέρει ακόμα ένα πλεονέκτημα. Μπορούμε να ανάγουμε τα προβλήματα εξαγωγής συμπερασμάτων των OWL οντολογιών στα προβλήματα εξαγωγής συμπερασμάτων των Περιγραφικών Λογικών για τα οποία υπάρχουν υλοποιημένοι και βελτιστοποιημένοι αλγόριθμοι. Στο σχήμα 4 παρουσιάζουμε την αντιστοιχία ανάμεσα στην αφηρημένη σύνταξη των περιγραφών OWL κλάσεων και των περιγραφών εννοιών των Περιγραφικών Λογικών. Όπως παρατηρούμε, από την αντιστοιχία αυτή, προκύπτει και η σημασιολογία των δομικών στοιχείων της γλώσσας OWL.

Πίνακας 4 *Περιγραφές OWL κλάσεων*

Αφηρημένη Σύνταξη	**Σύνταξη ΠΛ**	**Σημασιολογία**
owl:Thing	$\top$	$\top^{\mathcal{I}}=\Delta^{\mathcal{I}}$
owl:Nothing	$\bot$	$\bot^{\mathcal{I}}=\emptyset$
intersectionOf(C_1, …, C_n)	$C_1 \sqcap \ldots \sqcap C_n$	$(C_1 \sqcap \ldots \sqcap C_n)^{\mathcal{I}}=C_1^{\mathcal{I}} \cap \ldots \cap C_n^{\mathcal{I}}$
unionOf(C_1, …, C_n)	$C_1 \sqcup \ldots \sqcup C_n$	$(C_1 \sqcup \ldots \sqcup C_n)^{\mathcal{I}}=C_1^{\mathcal{I}} \cup \ldots \cup C_n^{\mathcal{I}}$
complementOf(C)	$\neg C$	$(\neg C)^{\mathcal{I}}=\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
oneOf(o_1, …, o_n)	$\{o_1\} \sqcup \ldots \sqcup \{o_n\}$	$(\{o_1\} \sqcup \ldots \sqcup \{o_n\})^{\mathcal{I}}=\{o_1^{\mathcal{I}},\ldots,o_n^{\mathcal{I}}\}$
restriction(R someValuesFrom(C))	$\exists R.C$	$(\exists R.C)^{\mathcal{I}}=\{a^{\mathcal{I}} \mid \exists b^{\mathcal{I}}.\ (a^{\mathcal{I}},b^{\mathcal{I}}) \in R^{\mathcal{I}} \ \&\ b^{\mathcal{I}} \in C^{\mathcal{I}}\}$
restriction(R allValuesFrom(C))	$\forall R.C$	$(\forall R.C)^{\mathcal{I}}=\{a^{\mathcal{I}} \mid \forall b^{\mathcal{I}}.\ (a^{\mathcal{I}},b^{\mathcal{I}}) \in R^{\mathcal{I}} \rightarrow b^{\mathcal{I}} \in C^{\mathcal{I}}\}$
restriction(R hasValue(o))	$\exists R.\{o\}$	$(\exists R.\{o\})^{\mathcal{I}}=\{a^{\mathcal{I}} \mid (a^{\mathcal{I}},o^{\mathcal{I}}) \in R^{\mathcal{I}}\}$
restriction(R minCardinality(n))	$\geq n R$	$(\geq n R)^{\mathcal{I}}=\{a^{\mathcal{I}} \mid \#\{b^{\mathcal{I}} \mid (a^{\mathcal{I}},b^{\mathcal{I}}) \in R^{\mathcal{I}}\} \geq n\}$
restriction(R maxCardinality(n))	$\leq n R$	$(\leq n R)^{\mathcal{I}}=\{a^{\mathcal{I}} \mid \#\{b^{\mathcal{I}} \mid (a^{\mathcal{I}},b^{\mathcal{I}}) \in R^{\mathcal{I}}\} \leq n\}$

restriction(R cardinality(n))	$\geq nR \sqcap \leq nR$	$(\geq nR \sqcap \leq nR)^I = \{a^I \mid \#\{b^I (a^I, b^I) \in R^I\} = n\}$

Παρόμοια αντιστοιχία μπορεί να οριστεί και στην περίπτωση των αξιωμάτων κλάσεων της γλώσσας OWL. Τα αξιώματα αυτά μπορούν να αντιστοιχηθούν σε αξιώματα υπαγωγής και ισοδυναμίας μεταξύ εννοιών μιας Περιγραφικής Λογικής. Οι αντιστοιχίες φαίνονται στον πίνακα 5 και στις περισσότερες των περιπτώσεων είναι αρκετά προφανής. Η μόνη περίπτωση που αξίζει να σχολιάσουμε είναι αυτή των ξένων κλάσεων. Από τη θεωρία συνόλων γνωρίζουμε ότι δύο σύνολα είναι ξένα μεταξύ τους αν η τομή τους είναι το κενό σύνολο. Αν μεταφερθούμε από το χώρο των συνόλων στον αφηρημένο χώρο των εννοιών τότε μπορούμε να ορίσουμε δύο έννοιες ως ξένες αν η τομή τους υπάγεται στην κενή έννοια. Αξίζει στο σημείο αυτό να επισημάνουμε ότι υπάρχει ένας εναλλακτικός τρόπος δήλωσης ότι δύο έννοιες είναι ξένες μεταξύ τους. Πιο συγκεκριμένα η έννοια C είναι ξένη με την έννοια D αν και μόνο αν $C \sqsubseteq \neg D$.

Πίνακας 5 *Αξιώματα OWL κλάσεων*

Αφηρημένη Σύνταξη	**Σύνταξη ΠΛ**	**Σημασιολογία**
Class(A partial C_1, …, C_n)	$A \sqsubseteq C_1 \sqcap \ldots \sqcap C_n$	$A^I \subseteq C_1^I \cap \ldots \cap C_n^I$
Class(A complete C_1, …, C_n)	$A \equiv C_1 \sqcap \ldots \sqcap C_n$	$A^I \subseteq C_1^I \cap \ldots \cap C_n^I$
EnumeratedClass(A o_1, …, o_n)	$A \equiv \{o_1\} \sqcup \ldots \sqcup \{o_n\}$	$(\{o_1\} \sqcup \ldots \sqcup \{o_n\})^I = \{o_1^I, \ldots, o_n^I\}$
SubClassOf(C_1, C_2)	$C_1 \sqsubseteq C_2$	$C_1^I \subseteq C_2^I$
EquivalentClasses(C_1, …, C_n)	$C_1 \equiv \ldots \equiv C_n$	$C_1^I = \ldots = C_n^I$
DisjointClasses(C_1, …, C_n)	$C_i \sqcap C_j \sqsubseteq \bot$, $1 \leq i < j \leq n$	$C_i^I \cap C_j^I \subseteq \varnothing$

Στον πίνακα 6 παρουσιάζουμε την αντιστοιχία ανάμεσα στα αξιώματα ιδιοτήτων της OWL και αξιώματα ρόλων των Περιγραφικών Λογικών. Για ακόμα μια φορά παρατηρούμε ότι η OWL δεν παρέχει τίποτα περισσότερο από την εκφραστική δυνατότητα που μας παρέχει μια ΠΛ ακόμα και στις περιπτώσεις που κάτι τέτοιο δεν είναι άμεσα προφανές, όπως στην περίπτωση των συναρτησιακών ρόλων που εξηγήσαμε παραπάνω. Από τον πίνακα αυτό όμως παρατηρούμε και κάτι άλλο πολύ ενδιαφέρον. Τα αξιώματα ορισμού του πεδίου τιμών και του πεδίου ορισμού μιας ιδιότητας, τα οποία συναντήσαμε στη γλώσσα RDF-S, μπορούν και αυτά να κωδικοποιηθούν με τη χρήση αξιωμάτων Περιγραφικών Λογικών. Πιο συγκεκριμένα ένα αξίωμα ορισμού του πεδίου τιμών της μορφής ObjectProperty(R domain(C))

αντιστοιχεί στο αξίωμα υπαγωγής $\exists R.\top \sqsubseteq C$, ενώ ένα αξίωμα ορισμού του πεδίου τιμών της μορφής ObjectProperty(R range(C)) αντιστοιχεί στο αξίωμα υπαγωγής, $\top \sqsubseteq \forall R.C$. Διαισθητικά το πρώτο αξίωμα μας λέει ότι: «αν ένα αντικείμενο συνδέεται μέσω της σχέσης R με κάτι, τότε το αντικείμενο αυτό είναι τύπου C». Άρα αυτό το αξίωμα μας λέει ότι όλα τα άτομα τα οποία βρίσκονται στην πρώτη θέση ενός ζευγαριού (a,b) το οποίο ανήκει στη σχέση R θα είναι τύπου C. Το δεύτερο αξίωμα μας δηλώνει ότι «για όλα τα αντικείμενα του κόσμου, όποτε αυτά συνδέονται με κάποιο άλλο αντικείμενο μέσω της σχέσης R τότε το αντικείμενο αυτό είναι τύπου C». Άρα όλα τα άτομα τα οποία βρίσκονται στη δεύτερη θέση ενός ζευγαριού (a,b) το οποίο ανήκει στη σχέση R θα είναι τύπου C το οποίο αντιστοιχεί με τον ορισμό πεδίου τιμών.

Πίνακας 6 *Αξιώματα ιδιοτήτων αντικειμένων*

Αφηρημένη Σύνταξη	**Σύνταξη ΠΛ**	**Σημασιολογία**
SubPropertyOf(R_1, R_2)	$R \sqsubseteq S$	$R^I \subseteq S^I$
EquivalentProperties(R_1, …, R_n)	$R_1 \equiv \ldots \equiv R_n$	$R_1^I = \ldots = R_n^I$
ObjectProperty(R super(R_1) … super(R_n) domain(C_1) … domain(C_n) range(C_1) … range(C_n) inverseOf(S) Symmetric Functional InverseFunctional Transitive)	$R \sqsubseteq R_i$ $\exists R.\top \sqsubseteq C_i$ $\top \sqsubseteq \forall R.C_i$ $R \equiv S^-$ $R \equiv R^-$ $\top \sqsubseteq \leq 1 R$ $\top \sqsubseteq \leq 1 R$ $Tr(R)$	$R^I \subseteq R_i^I$ $R^I \subseteq C_i^I \times \Delta^I$ $R^I \subseteq \Delta^I \times C_i^I$ $R^I = (S^-)^I$ $R^I = (R^-)^I$ $\forall a^I.\#\{b \mid (a^I,b^I) \in R^I\} \leq 1$ $\forall a^I.\#\{b \mid (a^I,b^I) \in (R^-)^I\} \leq 1$ $\{(a^I,b^I), (b^I,c^I)\} \subseteq R^I \rightarrow (a^I,c^I) \in R^I$

Εν συνεχεία τα αξιώματα ατόμων παρουσιάζονται στον πίνακα 7

Πίνακας 7 *Αξιώματα ατόμων*

Αφηρημένη Σύνταξη	**Σύνταξη ΠΛ**	**Σημασιολογία**
Individual(o type(C_1) … type(C_n)	$o:C_i$, $1 \leq i \leq n$	$o^I \in C_i^I$, $1 \leq i \leq n$
Value(R_1, o_1) … value(R_n, o_n)	$(o,o_i):C_i$, $1 \leq i \leq n$	$(o^I,o_i^I) \in C_i^I$, $1 \leq i \leq n$
SameIndidivual(o_1, …, o_n)	$o_1 = \ldots = o_n$	$o_1^I = \ldots = o_n^I$
DifferentIndividuals(o_1, …, o_n)	$o_i \neq o_j$, $1 \leq i < j \leq n$	$o_i^I \neq o_j^I$, $1 \leq i < j \leq n$

Τέλος στον πίνακα 8 παρουσιάζουμε την αντιστοίχηση ανάμεσα στην αφηρημένη σύνταξη περιγραφών κλάσεων και αξιωμάτων ιδιοτήτων τα οποία σχετίζονται με τύπους δεδομένων. Έτσι λοιπόν η ιδιότητα T είναι μια ιδιότητα τύπων δεδομένων ενώ το σύμβολο d παριστάνει έναν τύπο δεδομένων.

Πίνακας 8 *Αξιώματα ιδιοτήτων τύπων δεδομένων*

Αφηρημένη Σύνταξη	**Σύνταξη ΠΛ**	**Σημασιολογία**
restriction(T someValuesFrom(d))	$\exists T.d$	$(\exists T.d)^I = \{a^I \mid \exists y.(a^I,y) \in T^I \& y \in d_D\}$
restriction(T allValuesFrom(d))	$\forall T.d$	$(\forall T.d)^I = \{a^I \mid \forall y.(a^I,y) \in T^I \rightarrow y \in d_D\}$

restriction(T minCardinality(n))	$\geq nT$	$(\geq nT)^I=\{a^I \mid \#\{y \mid (a^I,y)\in T^I\}\geq n\}$
restriction(T maxCardinality(n))	$\leq nT$	$(\leq nT)^I=\{a^I \mid \#\{y \mid (a^I,y)\in T^I\}\leq n\}$
DatatypeProperty(T super(T_1)…super(T_n) domain(C_1) … domain(C_n) range(d_1) … range(d_n) Functional)	$T\sqsubseteq T_i$ $\exists T.\top\sqsubseteq C_i$ $\top\sqsubseteq\forall T.d_i$ $\top\sqsubseteq\leq 1T$	$T^I\subseteq T_{il}$ $T^I\subseteq C_{il}\times\Delta_D$ $T^I\subseteq\Delta_D\times C_{il}$ $\forall a^I.\#\{y\mid(a^I,y)\in T^I\}\leq 1$

Εισαγωγή στις Περιγραφικές Λογικές

Σύνταξη, Σημασιολογία

1. Εισαγωγή

Ένα από τα προβλήματα με το οποίο ασχολείται η Επιστήμη των Υπολογιστών και ιδιαίτερα ο τομέας της Τεχνητή Νοημοσύνης είναι το πώς μπορεί να καταγραφεί η ανθρώπινη γνώση σε ένα Υπολογιστικό Σύστημα ή αλγόριθμο. Η γνώση αυτή, αφού εισαχθεί στο σύστημα ή τον αλγόριθμο από τον άνθρωπο, θα μπορεί να αξιοποιηθεί για την εξαγωγή πιο ευφυών αποτελεσμάτων τα οποία θα πλησιάζουν σε ποιότητα αυτά της ανθρώπινης συλλογιστικής και σκέψης. Στην πραγματικότητα το πρόβλημα αυτό απασχόλησε πρώτους του μαθηματικούς, καθώς από την εποχή του Αριστοτέλη προσπαθούν να βρουν έναν *τυπικό (formal)* (μαθηματικό) τρόπο για να καταγράψουν την ανθρώπινη γνώση. Τι σημαίνει όμως τυπική καταγραφή της γνώσης και γιατί αυτή δεν μπορεί να γίνει με την περιγραφή της σε απλή φυσική γλώσσα, όπως άλλωστε ένας μεγάλος όγκος γνώσης καταγράφεται στο παρών βιβλίο; Η απάντηση έρχεται από τους μαθηματικούς οι οποίοι τονίζουν ότι η φυσική γλώσσα δεν έχει τέτοιο μαθηματικό υπόβαθρο που να επιτρέπει τη χρήση της για αυτούς τους σκοπούς. Απεναντίας μάλιστα, η χρήση της αντιβαίνει στην αυστηρότητα και την τυπικότητα των μαθηματικών. Ας θεωρήσουμε για παράδειγμα την πρόταση «Χτύπησε το παιδί με το ξύλο». Η πρόταση αυτή παρόλο που είναι καθ' όλα νόμιμη επιδέχεται διττής ερμηνείας. Μπορεί να σημαίνει ότι το παιδί που κρατούσε ένα ξύλο στα χέρια του χτύπησε ή ότι κάποιος άλλος χτύπησε το παιδί χρησιμοποιώντας ένα ξύλο. Βλέπουμε λοιπόν ότι παρ' όλο που οι επιμέρους έννοιες της πρότασης, όπως είναι το παιδί, το ξύλο κλπ επιδέχονται μοναδικής ερμηνείας η σύνδεση των εννοιών επιφέρει αβεβαιότητα και αμφισημία. Για το λόγο αυτό οι μαθηματικοί έχουν εισάγει ειδικές γλώσσες οι οποίες και αναφέρονται ως *γλώσσες αναπαράστασης γνώσης (knowledge representation languages)*.

Οι γλώσσες αναπαράστασης γνώσης δε διαφέρουν σε μεγάλο βαθμό από τις φυσικές γλώσσες που εμείς γνωρίζουμε όπως τα Ελληνικά ή τα Αγγλικά. Αποτελούνται από ένα *αλφάβητο (alphabet)*, ένα *συντακτικό (syntax)* και μια *σημασιολογία (semantics).* Έκτος όμως από την τυπικότητά τους οι γλώσσες αναπαράστασης γνώσης έχουν και ένα επιπλέον χαρακτηριστικό. Το επιπλέον αυτό στοιχείο ονομάζεται *θεωρία αποδείξεων* (*proof theory*) ή *μηχανισμός εξαγωγής συμπερασμάτων (reasoning algorithm)*. Ο ρόλος του είναι να περιγράφει κανόνες με βάση τους οποίους στηριζόμενοι σε μια αρχική γνώση, η οποία έχει περιγραφεί με μια γλώσσα αναπαράστασης γνώσης, να μπορούμε να εξάγουμε νέα γνώση και νέα συμπεράσματα από τα αρχικά γεγονότα. Όπως γίνεται αντιληπτό το χαρακτηριστικό αυτό είναι ιδιαίτερα σημαντικό στους επιστήμονες της κλάδου των υπολογιστών, καθώς η καταγραφή και μόνο γνώσης σε ένα υπολογιστικό σύστημα δε συνεπάγεται και τη δημιουργία ευφυών εφαρμογών, αλλά χρειάζεται και ένας τρόπος για την εκμαίευση γνώσης από την ήδη υπάρχουσα.

Κατά τη διάρκεια των χρόνων έχει αναπτυχθεί μια πληθώρα από γλώσσες αναπαράστασης γνώσης. Κάποιες, όπως είναι η Προτασιακή Λογική (Propositional Logic) (Mendelson, 1987) ή η Κατηγορηματική Λογική Πρώτης Τάξης (First-Order Predicate Logic) (Mendelson, 1987), έχουν αναπτυχθεί από μαθηματικούς, άλλες όμως όπως τα Σημασιολογικά Δίκτυα (Semantic Networks) (Quillian, 1967) και οι Περιγραφικές Λογικές (ΠΛ) (Description Logics – DLs) (Baader et. al., 2001) έχουν

αναπτυχθεί από επιστήμονες πληροφορικούς για να επιτελέσουν και να καλύψουν συγκεκριμένες ανάγκες του κλάδου τους. Για το λόγο αυτό σε πολλές περιπτώσεις οι γλώσσες αυτές ξέφευγαν από την αυστηρότητα και την τυπικότητα που πρέπει να διέπει τις γλώσσες αναπαράστασης γνώσης. Για παράδειγμα τα Σημασιολογικά Δίκτυα αναπαριστούν γνώση χρησιμοποιώντας ένα οπτικό (γραφικό) μοντέλο. Αυτό έχει το πλεονέκτημα οι γλώσσες αυτές να είναι απλούστερες στην κατανόηση και τη χρήση από επιστήμονες πληροφορικούς ή και απλούς χρήστες, οι οποίοι στις περισσότερες των περιπτώσεων έχουν περιορισμένες γνώσεις μαθηματικής λογικής, από την άλλη όμως η έλλειψη αυστηρής τυπικότητας δημιουργούσε προβλήματα.

Από την προσπάθεια καθορισμού τυπικής σημασιολογίας στη γλώσσα των Σημασιολογικών Δικτύων προέκυψε η γλώσσα των Περιγραφικών Λογικών (ΠΛ). Όπως θα δούμε και στη συνέχεια, τα δομικά στοιχεία των ΠΛ είναι οι έννοιες, οι ρόλοι και τα άτομα. Επιπλέον κάθε ΠΛ διαθέτει και ένα σύνολο *κατασκευαστών εννοιών* (concept constructors) οι οποίοι επενεργούν πάνω σε έννοιες με σκοπό τη δημιουργία περισσότερο περίπλοκων εννοιών. Για παράδειγμα μπορούμε να περιγράψουμε την έννοια Πατέρας χρησιμοποιώντας τις επιμέρους έννοιες Αρσενικό,

Άνθρωπος, το ρόλο εχειΠαιδι αλλά και τους κατασκευαστές, ⊓ και ∃ ως εξής:

$$\text{Πατέρας} \equiv \text{Αρσενικό} \sqcap \exists \text{εχειΠαιδι}.\text{Άνθρωπος}$$

Το σύμβολο ≡ δηλώνει τον ορισμό της έννοιας Πατέρας από τις επιμέρους έννοιες και τους κατασκευαστές. Επιπρόσθετα, χρησιμοποιώντας το άτομο Πέτρος, μπορούμε να δηλώσουμε ότι ο Πέτρος είναι Πατέρας γράφοντας, Πατέρας(Πέτρος). Όπως γίνεται αντιληπτό ανάλογα με το σύνολο των κατασκευαστών που χρησιμοποιούμε κάθε φορά ορίζουμε και μια διαφορετική ΠΛ. Εν κατακλείδι οι ΠΛ συνδυάζουν τόσο τυπική σημασιολογία όσο και απλότητα στη χρήση και την κατανόησή τους, πράγμα που τις έχει καταστήσει μέχρι αυτή τη στιγμή τις de facto γλώσσες για την περιγραφή γνώσης στο Σημασιολογικό Ιστό.

2. Βασικές Περιγραφικές Λογικές

Στο κεφάλαιο αυτό θα μελετήσουμε κάποιους από τους βασικούς κατασκευαστές εννοιών των περιγραφικών λογικών. Η μελέτη αυτή θα επικεντρωθεί στη σύνταξη και τη σημασιολογία των κατασκευαστών αυτών καθώς επίσης και στη γενική θεωρία σημασιολογίας των Περιγραφικών Λογικών. Όπως αναφέραμε και στην εισαγωγή, όπως κάθε γλώσσα έτσι και οι ΠΛ έχουν ένα αλφάβητο. Αντίθετα όμως με τις γλώσσες που γνωρίζουμε το αλφάβητο αυτό δεν είναι σταθερό αλλά μπορεί να οριστεί από το χρήστη. Το αλφάβητο ορίζεται από ένα σύνολο *ατομικών εννοιών (atomic concepts)* **C**, ένα σύνολο *ατομικών ρόλων (atomic roles)* ή αλλιώς *σχέσεων (relations)* **R**, και από ένα σύνολο *ατόμων (individuals)* **I**. Συνήθως χρησιμοποιούμε τα γράμματα A, B για να αναπαραστήσουμε ατομικές έννοιες, τα γράμματα R, S για να αναπαραστήσουμε ρόλους και τα γράμματα a, b για να αναπαραστήσουμε άτομα. Π*εριγραφές εννοιών (concept descriptions)* ή αλλιώς *περίπλοκες έννοιες (complex concepts)* μπορούμε να δημιουργήσουμε από τις πρωτογενείς έννοιες σε συνδυασμό με τους κατασκευαστές εννοιών των ΠΛ και συνήθως χρησιμοποιούμε τα γράμματα C, D για να αναφερθούμε σε αυτές. Επιπρόσθετα, προσέξτε ότι χρησιμοποιούμε λέξεις που ξεκινούν με κεφαλαία για την αναπαράσταση εννοιών, πρωτογενών ή μη, π.χ. Άνθρωπος, ενώ λέξεις που ξεκινούν με μικρό για την αναπαράσταση ρόλων, π.χ. εχειΠαιδι.

Μια από τις πιο βασικές ΠΛ είναι η γλώσσα $\mathcal{AL}$ (attributive language) (Schmidt-Schauss M., & Smolka G., 1991). Αυτή δημιουργείται από ένα αλφάβητο πρωτογενών εννοιών και ρόλων, από το σύνολο κατασκευαστών {¬, ⊓, ∀, ∃} και από δυο περιγραφές εννοιών οι οποίες έχουν ιδιαίτερη σημασία για τις ΠΛ γλώσσες

και συμβολίζονται με ⊤ και ⊥. Ας δούμε όμως πώς ορίζονται τυπικά οι περιγραφές εννοιών στην ΠΛ $\mathcal{AL}$. Έστω A μια ατομική έννοια, C, D δυο περιγραφές εννοιών και R ένας ατομικός ρόλος. Οι περιγραφές εννοιών στη γλώσσα $\mathcal{AL}$ ορίζονται επαγωγικά από την ακόλουθη *αφηρημένη σύνταξη (abstract syntax):*

$$C, D \rightarrow A \mid \top \mid \bot \mid \neg A \mid C \sqcap D \mid \forall R.C \mid \exists R.\top$$

Οι έννοιες ⊤ και ⊥ ονομάζονται *καθολική έννοια (universal concept)* και *κενή έννοια (bottom concept)*, αντίστοιχα. Από την άλλη οι έννοιες ∀R.C και ∃R.⊤ ονομάζονται *περιορισμός τιμής (value restriction)* ή αλλιώς *καθολικός περιορισμός (universal restriction)* και *περιορισμένος υπαρξιακός περιορισμός (limited existential restriction)*, αντίστοιχα. Τέλος παρατηρήστε ότι στη γλώσσα $\mathcal{AL}$ η άρνηση μπορεί να εμφανιστεί μόνο μπροστά από ατομικές έννοιες.

Ας δώσουμε λοιπόν κάποια παραδείγματα εννοιών που μπορούμε να περιγράψουμε χρησιμοποιώντας την εκφραστικότητα που παρέχει η ΠΛ $\mathcal{AL}$. Έστω το αλφάβητο που περιέχει τις πρωτογενείς έννοιες Άνθρωπος, Θηλυκό και το ρόλο εχωΠαιδι. Τότε η περιγραφική έννοια Άνθρωπος⊓Θηλυκό περιγράφει όλες τις γυναίκες ενώ η έννοια Άνθρωπος⊓¬Θηλυκό περιγράφει όλους τους άντρες. Επιπρόσθετα η περιγραφική έννοια Άνθρωπος⊓∀εχειΠαιδι.¬Θηλυκό περιγράφει τους ανθρώπους που όλα τους τα παιδιά (αν υπάρχουν) είναι αρσενικά, ενώ η έννοια Άνθρωπος⊓∃εχειΠαιδι.⊤ περιγράφει τους ανθρώπους που έχουν τουλάχιστον ένα παιδί, δηλαδή του γονιούς. Τέλος μπορούμε να περιγράψουμε αυτούς που δεν έχουν κανένα παιδί ως Άνθρωπος⊓∀εχειΠαιδι.⊥.

Είναι πολύ σημαντικό στο σημείο αυτό να παρατηρήσουμε ότι μέχρι στιγμής έχουμε περιγράψει μόνο τη σύνταξη της Περιγραφικής Λογικής $\mathcal{AL}$ ενώ δεν έχουμε αναφέρει τίποτα για τη σημασιολογία της. Σε μια φυσική γλώσσα, όπως είναι τα Ελληνικά, οι ερμηνείες των εννοιών είναι συνήθως συγκεκριμένες. Για παράδειγμα η ερμηνεία της έννοιας «άνθρωπος» αποτελεί την οντότητα αυτή που όλοι γνωρίζουμε μαζί με τα βιομετρικά χαρακτηριστικά που τη διακρίνουν. Μέλη της έννοιας αυτή είναι όλοι οι άνθρωποι του κόσμου. Αντίθετα όμως σε μια ΠΛ, όπως επίσης και σε οποιαδήποτε γλώσσα αναπαράστασης γνώσης όταν γράφουμε την έννοια «Άνθρωπος» αυτό δε σημαίνει αυτόματα ότι η έννοια αυτή έχει και ερμηνεία η οποία και είναι το σύνολο όλων των ανθρώπων. Πιο συγκεκριμένα πρέπει εμείς να αποδώσουμε μια συγκεκριμένη ερμηνεία στην έννοια αυτή για να αποκτήσει κάποιο νόημα και σημασία. Ας δούμε λοιπόν πώς ορίζονται μαθηματικά οι ερμηνείες σε μια ΠΛ. Μια ΠΛ *ερμηνεία (interpretation)* I ορίζεται από ένα ζεύγος $(\Delta^I, \cdot^I)$, όπου Δ^I

είναι ένα μη-κενό σύνολο που ονομάζεται *χώρος ερμηνείας (domain of interpretation)* και περιέχει στοιχεία που ονομάζονται *αντικείμενα (objects),* και $\cdot^I$

είναι μια *συνάρτηση ερμηνείας (interpretation function)* που ερμηνεύει κάθε ατομική έννοια A ως ένα υποσύνολο A^I του Δ^I ($A^I \subseteq \Delta^I$) και κάθε ρόλο R ως ένα υποσύνολο R^I του $\Delta^I \times \Delta^I$ ($R^I \subseteq \Delta^I \times \Delta^I$). Τέλος η συνάρτηση ερμηνείας μπορεί να επεκταθεί για να δώσει ερμηνεία και σε περιγραφές εννοιών. Η σημασιολογία τους είναι η ακόλουθη:

$\top^I = \Delta^I$

$\bot^I = \varnothing$

$(\neg A)^I = \Delta^I \setminus A^I$

$(C \sqcap D)^I = C^I \cap D^I$

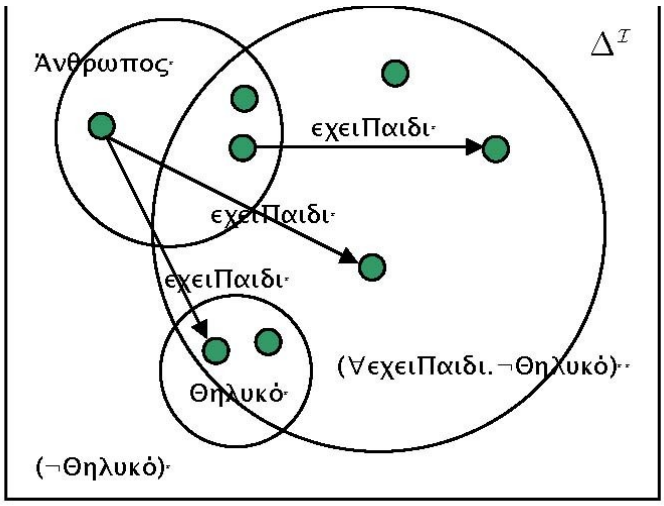
$$(\forall R.C)^{I}\{a\in\Delta^{I} \mid \forall b\in\Delta^{I}.\ (a,b)\in R^{I}\rightarrow b\in C^{I}\}$$

$$(\exists R.\top)^{I}\{a\in\Delta^{I} \mid \exists b\in\Delta^{I}.\ (a,b)\in R^{I}\}$$

Όπως γίνεται αντιληπτό κάθε $\mathcal{AL}$ έννοια ερμηνεύεται ως ένα υποσύνολο του Δ^{I}. Για παράδειγμα η έννοια $\top$ ερμηνεύεται ως το σύνολο το οποίο περιέχει όλα τα αντικείμενα του χώρου ερμηνείας, ενώ η έννοια $\bot$ ερμηνεύεται ως το κενό σύνολο, το οποίο και δικαιολογεί την ονομασία που τους έχουμε προσδώσει. Εν συνεχεία η έννοια $C\sqcap D$ ερμηνεύεται ως το σύνολο το οποίο προκύπτει από την τομή των ερμηνειών των εννοιών C και D. Επιπρόσθετα, η ερμηνεία της έννοιας $\forall R.C$ περιέχει το σύνολο των αντικειμένων του Δ^{I} τα οποία αν συμμετέχουν στο ρόλο R^{I} με κάποιο άλλο αντικείμενο, τότε το αντικείμενο αυτό ανήκει στην ερμηνεία της έννοιας C δηλαδή στο σύνολο C^{I}. Είναι πολύ σημαντικό στο σημείο αυτό να τονίσουμε ότι ένα αντικείμενο ανήκει στην ερμηνεία της έννοιας $\forall R.C$ ακόμα και αν δε σχετίζεται μέσω της σχέσης R^{I} με κανένα άλλο αντικείμενο. Έτσι λοιπόν, επιστρέφοντας στα παραδείγματά μας για τις έννοιες μιας οικογένειας έχουμε ότι και η ερμηνεία της έννοιας Άνθρωπος⊓∀εχειΠαιδι.¬Θηλυκό αποτελεί ένα σύνολο του Δ^{I}. Το σύνολο αυτό περιέχει τα αντικείμενα του Δ^{I} τα οποία ανήκουν ταυτόχρονα στην ερμηνεία της έννοιας Άνθρωπος, δηλαδή στο σύνολο $Άνθρωπος^{I}$, και αν συμμετέχουν στο ρόλο $εχειΠαιδι^{I}$ με κάποιο άλλο αντικείμενο, τότε το αντικείμενο αυτό δεν ανήκει στο σύνολο αυτό που εμείς έχουμε αποδώσει ως ερμηνεία της έννοιας Θηλυκό.

Άνθρωπος⊓∀εχειΠαιδι.¬Θηλυκό

Στο σχήμα 4.1 φαίνεται διαισθητικά η ερμηνεία της



Σχήμα 1.1 *Γραφική αναπαράσταση της ερμηνείας της έννοιας*

προαναφερθείσας έννοιας. Στο σχήμα αυτό παρατηρούμε τις ερμηνείες των εννοιών Άνθρωπος, Θηλυκό, ∀εχειΠαιδι.¬Θηλυκό και ¬Θηλυκό αλλά και του ρόλου εχειΠαιδι. Βλέπουμε ότι όλες αυτές οι έννοιες, ακόμα και η ∀εχειΠαιδι.¬Θηλυκό, ερμηνεύονται ως υποσύνολα του χώρου ερμηνείας Δ^{I}. Η ερμηνεία της έννοιας Άνθρωπος⊓∀εχειΠαιδι.¬Θηλυκό είναι η τομή των συνόλων $Άνθρωπος^{I}$ και $(\forall εχειΠαιδι.\neg Θηλυκό)^{I}$. Παρατηρούμε επίσης ότι η ερμηνεία της έννοιας ∀εχειΠαιδι.¬Θηλυκό περιέχει και αντικείμενα τα οποία δε συνδέονται με κανένα άλλο αντικείμενο μέσω του ρόλου $εχειΠαιδι^{I}$. Επιπρόσθετα παρατηρούμε ότι τα αντικείμενα που ανήκουν στο σύνολο $Άνθρωπος^{I}$ όχι όμως στην τομή με το σύνολο $(\forall εχειΠαιδι.\neg Θηλυκό)^{I}$ είναι αυτά τα οποία έχουν τουλάχιστον μια σύνδεση με κάποιο αντικείμενο του συνόλου $Θηλυκό^{I}$ μέσω της σχέσης $εχειΠαιδι^{I}$.

Από τα προηγούμενα μπορούμε να παρατηρήσουμε ότι οι έννοιες σε μια ΠΛ δεν επιδέχονται μοναδικής ερμηνείας. Αντίθετα μάλιστα υπάρχουν περιπτώσεις που μια έννοια μπορεί να έχει άπειρες ερμηνείες, αλλά και περιπτώσεις εννοιών όπου δεν υπάρχει καμία ερμηνεία που να τις ερμηνεύει ως μη κενές, όπως για παράδειγμα η έννοια Άνθρωπος⊓¬Άνθρωπος. Οι έννοιες αυτές ονομάζονται *μη-ικανοποιήσιμες (unsatisfiable)*.

Προσθέτοντας επιπλέον κατασκευαστές εννοιών στην απλή περιγραφική λογική $\mathcal{AL}$ μπορούμε να δημιουργήσουμε περισσότερο εκφραστικές γλώσσες οι οποίες θα μας δίνουν τη δυνατότητα να περιγράφουμε πιο πολύπλοκες έννοιες. Πρώτα απ' όλα ας θεωρήσουμε τον κατασκευαστή ένωσης (union) (ο οποίος συμβολίζεται με το γράμμα $\mathcal{U}$). Η σύνταξη της ένωση δυο εννοιών είναι η εξής C⊔D και η ερμηνεία της από τη συνάρτηση ερμηνείας $\cdot^{\mathcal{I}}$ η ακόλουθη:
$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$

Για παράδειγμα με τη γλώσσα $\mathcal{ALU}$ μπορούμε να περιγράψουμε την έννοια Πατέρας⊔Μητέρα που διαισθητικά περιγράφει την έννοια του γονιού.

Επιπρόσθετα είναι απολύτως φυσικό να επεκτείνουμε τον περιορισμένο υπαρξιακό περιορισμό στον *πλήρη υπαρξιακό περιορισμό* (ο οποίος συμβολίζεται με το γράμμα $\mathcal{E}$). Η σύνταξη του είναι η ∃R.C ενώ η σημασιολογία του είναι η ακόλουθη:

$$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b \in \Delta^{\mathcal{I}}.\ (a,b) \in R^{\mathcal{I}} \text{ και } b \in C^{\mathcal{I}}\}.$$

Έτσι για παράδειγμα με τη γλώσσα $\mathcal{ALE}$ μπορούμε να περιγράψουμε την έννοια των ανθρώπων που έχουν τουλάχιστον ένα παιδί που είναι κορίτσι ως,
Άνθρωπος⊓∃ΕχειΠαιδι.Θηλυκό. Είναι σημαντικό να παρατηρήσουμε τη διαφορά μεταξύ των εννοιών ∃ΕχειΠαιδι.Θηλυκό και ∀ΕχειΠαιδι.Θηλυκό. Στην πρώτη περίπτωση ζητάμε την ύπαρξη τουλάχιστον ενός παιδιού που να είναι θηλυκό χωρίς να αποκλείουμε να υπάρχουν άλλα παιδιά που δεν είναι θηλυκά, ενώ στη δεύτερη περίπτωση απαιτούμε όλα τα παιδιά, αν αυτά υπάρχουν, να είναι θηλυκά. Ένας άλλος κατασκευαστής εννοιών που χρησιμοποιείται αρκετά στις ΠΛ και αυξάνει σημαντικά την εκφραστικότητά τους είναι ο κατασκευαστής *περιορισμού πληθυκότητας (number restriction)*. Ο κατασκευαστής αυτός συμβολίζεται με το γράμμα $\mathcal{N}$ και αποτελείται από δυο επιμέρους κατασκευαστές. Τον κατασκευαστή έννοιας *το-πολύ (at-most)*, ο οποίος έχει σύνταξη ≤nR και τον κατασκευαστή *το-λιγότερο (at-least)*, ο οποίος έχει σύνταξη ≥nR, όπου n είναι ένας φυσικός αριθμός και R ένας ρόλος. Διαισθητικά οι κατασκευαστές αυτοί ορίζουν έννοιες λαμβάνοντας υπόψη το πλήθος των συνδέσεων ενός αντικειμένου a με άλλα σε μια σχέση R. Πιο τυπικά η σημασιολογία των κατασκευαστών αυτών είναι η ακόλουθη.

$$(\leq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \#\{b \mid (a,b) \in R^{\mathcal{I}}\} \leq n\},$$
$$(\geq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \#\{b \mid (a,b) \in R^{\mathcal{I}}\} \geq n\},$$

όπου με # συμβολίζεται η *πληθυκότητα (cardinality)* ενός συνόλου. Έτσι λοιπόν η ερμηνεία της έννοιας ≥nR περιλαμβάνει το σύνολο αυτών των αντικειμένων που συμμετέχουν στην ερμηνεία της σχέσης R το λιγότερο με n άλλα αντικείμενα. Χρησιμοποιώντας τους κατασκευαστές αυτούς μπορούμε να περιγράψουμε έννοιες όπως είναι η έννοια των ανθρώπων που έχουν ακριβώς ένα παιδί, Άνθρωπος⊓≥1εχειΠαιδι⊓≤1εχειΠαιδι. Αν τώρα συνδυάσουμε τον κατασκευαστή πληθυκότητας με τον κατασκευαστή ένωσης, και άρα έχουμε τη γλώσσα $\mathcal{ALUN}$ μπορούμε να περιγράψουμε την έννοια των ανθρώπων που ή έχουν πολλά παιδιά όλα από τα οποία είναι θηλυκά ή έχουν λίγα όλα από τα οποία είναι αρσενικά ως, Άνθρωπος⊓((≥3εχειΠαιδι⊓∀εχειΠαιδι.Θηλυκό)⊔
(≤2εχειΠαιδι⊓∀εχειΠαιδι.Αρσενικό).

Στην περίπτωση που επιτρέπουμε μόνο την τιμή πληθυκότητας 1 ο κατασκευαστής ονομάζεται *συναρτησιακός περιορισμός πληθυκότητας (functional number restriction)* και η ύπαρξή του συμβολίζεται με το γράμμα $\mathcal{F}$.

Όπως είναι φανερό ο κατασκευαστής αυτός εμφανίζει κάποιες αδυναμίες όσον αφορά το μέγεθος της εκφραστικότητας που μας προσφέρει. Για παράδειγμα δεν μπορούμε να περιγράψουμε την έννοια των ανθρώπων που έχουν πολλά θηλυκά παιδιά αφήνοντας ανοικτό το ενδεχόμενο να υπάρχουν και άλλα παιδιά που είναι είτε θηλυκά είτε αρσενικά. Η αδυναμία αυτή ξεπερνάτε από μια απλή αλλά ταυτόχρονα ισχυρή επέκταση του κατασκευαστή αυτού. Ο νέος κατασκευαστής, ο οποίος συμβολίζεται με $\mathcal{Q}$ ονομάζεται *προσοντούχος περιοριστής πληθυκότητας (qualified number restriction ή qualified cardinality restriction)*. Η σύνταξη της μορφής το
πολύ είναι ≤nR.C, όπου n και R είναι όπως πριν ενώ C είναι μια οποιαδήποτε ΠΛ έννοια. Παρόμοια για τη μορφή το-λιγότερο. Η σημασιολογία του δίνεται από μια απλή επέκταση της σημασιολογίας του απλού περιοριστή πληθυκότητας και για την περίπτωση του το-πολύ είναι η ακόλουθη: $(\leq nR.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \#\{b \mid (a,b) \in R^{\mathcal{I}} \text{ και } b \in C^{\mathcal{I}}\} \leq n\}$. Έτσι λοιπόν η προαναφερθείσα έννοια μπορεί να περιγραφεί ως εξής:
Άνθρωπος⊓(≥3εχειΠαιδι.Θηλυκό⊔≤2εχειΠαιδι.Αρσενικό).

Μια άλλη προφανής επέκταση που μπορούμε να κάνουμε είναι η επέκταση της άρνησης σε περίπλοκες έννοιες. Η ΠΛ που επιτρέπει αυτόν τον κατασκευαστή συμβολίζεται με το γράμμα *C*, η σύνταξή του είναι ¬C ενώ η σημασιολογία του δίνεται από την σχέση $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$. Χρησιμοποιώντας τη γλώσσα $\mathcal{ALC}$ μπορούμε για παράδειγμα να περιγράψουμε την έννοια των ανθρώπων που δεν έχουν παιδιά ως Άνθρωπος⊓¬(∃εχειΠαιδί.⊤), ή ως ¬(Πατέρας⊔Μητέρα) αν χρησιμοποιήσουμε ως
πρωτογενείς έννοιες τις Πατέρας και Μητέρα αλλά επίσης και τον κατασκευαστή ένωσης.

Σε αυτό το σημείο ίσως έχει αρχίσει να γίνεται αντιληπτό ότι η εισαγωγή άρνησης σε μια ΠΛ κάνει κάποιες από τις προαναφερθείσες γλώσσες να μην είναι διακριτές μεταξύ τους. Για παράδειγμα για να περιγράψουμε την έννοια των ανθρώπων που δεν έχουν παιδιά χρησιμοποιήσαμε τη γλώσσα $\mathcal{AL}$ για να γράψουμε,
Άνθρωπος⊓∀εχειΠαιδι.⊥, αντί για Άνθρωπος⊓¬(∃εχειΠαιδί.⊤). Επιπρόσθετα, κάποιος θα μπορούσε να περιγράψει την έννοια αυτή ως, ¬Πατέρας⊓¬Μητέρα αντί να χρησιμοποιήσει τον κατασκευαστή ένωσης για να γράψει ¬(Πατέρας⊔Μητέρα). Πιο τυπικά χρησιμοποιώντας τους γνωστούς κανόνες DeMorgan, από τη θεωρία συνόλων, προκύπτουν οι ισοδυναμίες, C⊔D≡¬(¬C⊓¬D) και ∃R.C≡¬∀R.¬C. Έτσι λοιπόν οποτεδήποτε αναφερόμαστε στη γλώσσα $\mathcal{ALC}$ θα θεωρούμε και την ύπαρξη των κατασκευαστών ένωσης και πλήρη υπαρξιακού περιορισμού χωρίς να χρειάζεται να ονομάζουμε τη γλώσσα $\mathcal{ALUEC}$. Παρομοίως ο κατασκευαστής $\mathcal{Q}$ έχει τέτοια εκφραστική δυνατότητα που μπορεί να περιγράψει τον πλήρη υπαρξιακό περιορισμό,

καθώς η ακόλουθη ισοδυναμία ισχύει: ∃R.C≡≥1R.C. Έτσι λοιπόν μπορούμε απλώς να γράψουμε $\mathcal{ALQ}$ υπονοώντας ουσιαστικά τους κατασκευαστές που ορίζονται από τη γλώσσα $\mathcal{ALEQ}$. Τέλος αξίζει να επισημάνουμε ότι ο κατασκευαστής $\mathcal{Q}$ έχει τη δυνατότητα να εκφράσει και τον κατασκευαστή του περιορισμού τιμής καθώς ισχύει
∀R.C≡≤0R.¬C.

2.1 Ορολογίες

Μέχρι στιγμής είδαμε πως μπορούμε να χρησιμοποιήσουμε τους κατασκευαστές που μας προσφέρει μια ΠΛ σε συνδυασμό με τις πρωτογενείς έννοιες και ρόλους με σκοπό τη δημιουργία πολύπλοκων εννοιών. Οι ΠΛ όμως μας προσφέρουν μια επιπλέον δυνατότητα αυτή του να μπορούμε να αποδίδουμε ονόματα στις περίπλοκες έννοιες που θέλουμε να περιγράψουμε, αλλά ακόμα και να περιγράφουμε σχέσεις ανάμεσα σε αυτές. Οι σχέσεις αυτές παρουσιάζονται με τη μορφή αξιωμάτων που ονομάζονται *αξιώματα ορολογίας (terminological axioms)*. Πιο συγκεκριμένα αν C και D είναι ΠΛ έννοιες τότε τα αξιώματα ορολογίας έχουν τη μορφή,

C⊑D ή C≡D

Αξιώματα του πρώτου τύπου ονομάζονται *αξιώματα υπαγωγής (subsumption axioms*
ή inclusion axioms) ενώ του δεύτερου τύπου ονομάζονται *αξιώματα ισοδυναμίας (equivalence axioms)*. Διαισθητικά ένα αξίωμα υπαγωγής της μορφής C⊑D δηλώνει ότι η έννοια D είναι πιο γενική από την έννοια C ή αλλιώς ότι η έννοια C είναι υπο έννοια της D. Αντίστοιχα το αξίωμα C≡D σημαίνει ότι οι δυο έννοιες είναι ταυτόσημες. Ένα σύνολο από αξιώματα υπαγωγής ή ισοδυναμίας αποτελούνε το *σώμα ορολογίας (TBox – Terminological Box)* ή απλώς μια *ορολογία (terminology)* η οποία συμβολίζεται με το γράμμα $\mathcal{T}$.

Ας δούμε τώρα πως ερμηνεύονται τα αξιώματα ορολογίας. Σύμφωνα με τα παραπάνω εφόσον το αξίωμα C⊑D σημαίνει ότι η έννοια D είναι πιο γενική από την έννοια C και εφόσον τα C και D ερμηνεύονται σαν σύνολα είναι φυσικό να πούμε ότι μια ερμηνεία I *ικανοποιεί (satisfies)* ένα αξίωμα υπαγωγής C⊑D αν $C^I \subseteq D^I$, δηλαδή αν η ερμηνεία I ερμηνεύει την έννοια D ως υπερσύνολο της έννοιας C. Αντίστοιχα μια ερμηνεία I ικανοποιεί ένα αξίωμα ισοδυναμίας C≡D αν $C^I = D^I$. Τέλος μια ερμηνεία I ικανοποιεί ένα σώμα ισχυρισμών $\mathcal{T}$ αν ικανοποιεί όλα τα αξιώματα υπαγωγής και ισοδυναμίας που υπάρχουν στο $\mathcal{T}$. Τότε λέμε ότι η I είναι *μοντέλο (model)* του $\mathcal{T}$. Διαισθητικά μια ορολογία $\mathcal{T}$ αποτελεί έναν περιορισμό στη δομή των μοντέλων που μπορούν να οριστούν. Για παράδειγμα ένα αξίωμα της μορφής C⊑D⊓E σημαίνει ότι

σε όλες τις ερμηνείες οι οποίες ικανοποιούν το αξίωμα, το σύνολο C^I πρέπει να είναι υποσύνολο της τομής των συνόλων D^I και E^I ανεξάρτητα με την ερμηνεία που θα αποδώσουμε ατομικά στις έννοιες C, D και E.

Το TBox λοιπόν μας δίνει τη δυνατότητα να ονομάσουμε τις πολύπλοκες έννοιές μας. Για παράδειγμα μπορούμε να πούμε ότι η έννοια Πατέρας⊔Μητέρα περιγράφει ακριβώς την έννοια του γονιού γράφοντας, Γονιός≡Πατέρας⊔Μητέρα. Σε πολλές περιπτώσεις μπορεί να μην μπορούμε να δώσουμε ακριβή ορισμό για μια έννοιά. Σε αυτήν την περίπτωση μπορούμε να ορίσουμε μια σχέση υπαγωγής ανάμεσα στις έννοιες. Για παράδειγμα μπορούμε να πούμε ότι η τίγρης είναι ένα
είδος ζώου γράφοντας, Τίγρης⊑Ζώο. Οι σχέσεις υπαγωγής είναι αρκετά σημαντικές καθώς μας επιτρέπουν τη δημιουργία ιεραρχιών. Αξίζει να σημειώσουμε ότι τόσο οι έννοιες στο αριστερό αλλά και στο δεξί μέρος των αξιωμάτων ορολογίας μπορούν

να είναι οποιαδήποτε περίπλοκη έννοια. Έτσι λοιπόν μπορούμε να γράψουμε το αξίωμα
∃ΕχειΠαιδί.⊤⊓ΥγιήςΕυτυχισμένος, που δηλώνει ότι αν έχεις παιδί και έχεις και την υγεία σου τότε είσαι και ευτυχισμένος, αλλά ακόμα περισσότερο μπορούμε να ορίσουμε αξιώματα που περιέχουν *κύκλους (cycles)*, όπως για παράδειγμα το αξίωμα ΔυαδικόΔέντρο≡Δέντρο⊓≤2ΕχειΚλαδι⊓∃ΕχειΚλαδι.ΔυαδικόΔέντρο που ορίζει τη δομή δεδομένων δυαδικό δέντρο. Τα αξιώματα της πρώτης μορφής ονομάζονται *υπαγωγές γενικών εννοιών (general concept inclusions – GCIs)* ή *γενικευμένα αξιώματα (general axioms)*. Στον πίνακα 4.2 φαίνεται ένα παράδειγμα TBox με αξιώματα για την περιγραφή μιας οικογένειας.

Πίνακας 4.2 *TBox οικογένειας*

Γυναίκα ≡ Άνθρωπος⊓Θηλυκό
Άντρας ≡ Άνθρωπος⊓¬Γυναίκα
Μητέρα ≡ Γυναίκα⊓∃ΕχειΠΑιδι.Άνθρωπος
Πατέρας ≡ Άντρας⊓∃ΕχειΠΑιδι.Άνθρωπος
Γονιός ≡ Πατέρας⊔Μητέρα
Πολύτεκνος ≡ Γονιός⊓≥3ΕχειΠαιδι

2.2 Ισχυρισμοί

Έκτος από τη δυνατότητα ορισμού σχέσεων μεταξύ εννοιών οι ΠΛ μας επιτρέπουν να κάνουμε και υποθέσεις όσον αφορά τα άτομα του κόσμου τον οποίο μοντελοποιούμε. Μας δίνει δηλαδή τη δυνατότητα καθορισμού *σχέσεων στιγμιοτύπου (instance relations)* ανάμεσα σε ένα άτομο (ζευγάρι ατόμων) και μια έννοια (ρόλο), τα οποία ονομάζονται *ισχυρισμοί (assertions)*. Υπάρχουν λοιπόν δυο είδη ισχυρισμών, οι *ισχυρισμοί εννοιών (concept assertions)*, που έχουν τη σύνταξη a:C ή C(a), και οι *ισχυρισμοί ρόλων (role assertion)* που έχουν τη σύνταξη (a,b):R ή R(a,b). Αν λοιπόν θεωρήσουμε ως άτομα το Μιχάλης, Ντόρα και Γιώργος, μπορούμε να κάνουμε ισχυρισμούς πάνω στις έννοιες και τους ρόλους που εισάγαμε στο TBox της οικογένειας, όπως Γονιός(Ντόρα), εχειΠαιδι(Ντόρα, Μιχάλης) και Άντρας(Γιώργος). Το σύνολο των ισχυρισμών αποτελεί το *σώμα ισχυρισμών (ABox – Assertional Box)* το οποίο συμβολίζεται με $\mathcal{A}$. Έτσι λοιπόν μια ΠΛ *βάση γνώσης (knowledge base)* ορίζεται ως ένα ζευγάρι ενός TBox $\mathcal{T}$ και ενός ABox $\mathcal{A}$, $\mathcal{K}=(\mathcal{T},\mathcal{A})$.

Όπως με κάθε στοιχείο μιας ΠΛ γλώσσας έτσι και οι ισχυρισμοί χρειάζονται να ερμηνευτούν. Για το σκοπό αυτό η συνάρτηση ερμηνείας επεκτείνεται αντιστοιχίζοντας ένα άτομο a σε ένα αντικείμενο a^I του Δ^I. Είναι σημαντικό σε αυτό το σημείο να επισημάνουμε τη διαφορά μεταξύ ατόμου και αντικειμένου. Τα άτομα ως στοιχεία του αλφάβητου μιας ΠΛ γλώσσας χρειάζεται να ερμηνευτούν για να αποκτήσουν συγκεκριμένο νόημα. Έτσι λοιπόν το άτομο Γιώργος δεν αντιπροσωπεύει κάποιο συγκεκριμένο αντικείμενο σε κάποιο κόσμο. Αν θέλουμε να ερμηνεύσουμε το άτομο Γιώργος πρέπει να ορίσουμε τον κόσμο μας (το Δ^I) και να αναθέσουμε ένα αντικείμενο του κόσμου ως ερμηνεία του ατόμου Γιώργος.

Μια ερμηνεία I ικανοποιεί έναν ισχυρισμό a:C ((a,b):R) αν $a^I \in C^I$ ($(a^I,b^I)\in R^I$), ενώ ικανοποιεί ένα σώμα ισχυρισμών *A* αν ικανοποιεί όλους τους ισχυρισμούς που βρίσκονται στο *A*. Τότε λέμε ότι η I είναι μοντέλο του *A*. Στα σώματα ισχυρισμών έχουμε έναν επιπλέον ορισμό ο οποίος είναι αρκετά σημαντικός. Λέμε ότι μια

ερμηνεία I ικανοποιεί ένα σώμα ισχυρισμών $\mathcal{A}$ *με βάση το* (μβτ) σώμα ορολογίας $\mathcal{T}$ αν η I είναι μοντέλο του $\mathcal{A}$ αλλά και ταυτόχρονα μοντέλο του $\mathcal{T}$.

1. Υπηρεσίες Εξαγωγής Συμπερασμάτων

Όπως αναφέραμε στην εισαγωγή μας ένα σύστημα το οποίο μπορεί μόνο να αποθηκεύει γνώση σε κάποια μορφή δε συνεπάγεται απαραίτητα και τη δημιουργία ενός ευφυούς συστήματος. Για να μπορέσουμε να δημιουργήσουμε τέτοιες εφαρμογές θα πρέπει να δίνεται η δυνατότητα να εξάγουμε συμπεράσματα βασισμένοι στη γνώση την οποία έχουμε περιγράψει. Μέχρι στιγμής έχουμε δει πως μπορούμε να χρησιμοποιήσουμε τα σώματα ορολογίας και ισχυρισμών των ΠΛ για τη δημιουργία βάσεων γνώσης. Στο παρών κεφάλαιο θα παρουσιάσουμε τις *υπηρεσίες εξαγωγής συμπερασμάτων (inference services)* που μας παρέχουν οι ΠΛ. Επιπρόσθετα θα αναλύσουμε τον τρόπο με τον οποίο όλες οι υπηρεσίες εξαγωγής συμπερασμάτων μπορούν να αναχθούν σε μια μόνο υπηρεσία, καθώς επίσης πώς υπηρεσίες που ορίζονται με βάση ένα TBox μπορούν να αναχθούν σε υπηρεσίες μβτ κενό TBox. Όπως γίνεται αντιληπτό και οι δυο αυτοί μηχανισμοί είναι πολύ σημαντικοί καθώς αφενός αν υλοποιήσουμε έναν αλγόριθμο που λύνει την υπηρεσία στην οποία οι άλλες ανάγονται τότε έχουμε υλοποιήσει έναν αλγόριθμο για όλες και αφετέρου εξαλείφοντας το TBox ο αλγόριθμος απλοποιείται σε μεγάλο βαθμό.

3.1 Υπηρεσίες Σωμάτων Ορολογίας

Οι ΠΛ παρέχουν υπηρεσίες τόσο πάνω στα σώματα ορολογίας όσο και στα σώματα ισχυρισμών. Ας ξεκινήσουμε από το σώμα ορολογίας. Έστω $\mathcal{T}$ένα TBox.

•Ικανοποιησιμότητα (satisfiability): Η έννοια C είναι *ικανοποιήσιμη (satisfiable)* μβτ *T* αν υπάρχει μοντέλο I του *T* τέτοιο ώστε $C^I \neq \varnothing$.

- **Υπαγωγή (subsumption):** Η έννοια C υπάγεται στην έννοια D (subsumed by) μβτ *T* αν $C^I \subseteq D^I$ για κάθε μοντέλο I του *T*. Σε αυτήν την περίπτωση γράφουμε $T \models C \sqsubseteq D$.
- **Ισοδυναμία (equivalence):** Η έννοια C είναι ισοδύναμη με την έννοια D μβτ *T* αν $C^I = D^I$ για κάθε μοντέλο I του *T*. Σε αυτήν την περίπτωση γράφουμε $T \models C \equiv D$.

•Ξένες Έννοιες (disjointness): Η έννοια C είναι *ξένη* με την έννοια D *(disjoint with)* μβτ *T* αν $C^I \cap D^I = \varnothing$ για κάθε μοντέλο I του *T*.

Παρατηρήστε πως διαφέρει το πρόβλημα την ικανοποιησιμότητας από τα υπόλοιπα προβλήματα. Διαισθητικά στο πρόβλημα της ικανοποιησιμότητας αναζητούμε αν μια έννοια C έχει φυσική υπόσταση με βάση τα αξιώματα που περιγράφονται σε ένα TBox, άρα μας ενδιαφέρει ακόμα και μια ερμηνεία στην οποία το «νόημα» της να μην είναι κενό. Αντίθετα για να δούμε αν μια έννοια υπάγεται σε μια άλλη πρέπει να δούμε όλα τα μοντέλα του $\mathcal{T}$. Προφανώς στα προηγούμενα προβλήματα το TBox μπορεί να είναι κενό.

Ας δούμε μερικά παραδείγματα. Οι παρακάτω προτάσεις ισχύουν:

1. Άντρας⊓¬Άντρας δεν είναι ικανοποιήσιμη μβτ κενό TBox.

2. Άντρας⊓Γυναίκα είναι ικανοποιήσιμη μβτ κενό TBox.

3. Άντρας⊓Γυναίκα δεν είναι ικανοποιήσιμη μβτ TBox του πίνακα 4.2.
4. ∃εχειΠαιδι.Αρσενικό⊓∃εχειΠαιδι.¬Αρσενικό είναι ικανοποιήσιμη μβτ κενό TBox.
5. ∀εχειΠαιδι.Αρσενικό⊓∀εχειΠαιδι.¬Αρσενικό είναι ικανοποιήσιμη μβτ κενό TBox.
6. Η έννοια Μητέρα δεν υπάγεται στην έννοια Γονιός μβτ κενό TBox.
7. Μητέρα⊑Γονιός μβτ TBox $\mathcal{T}$ του πίνακα 4.2.

Η πρώτη περίπτωση είναι προφανής καθώς δεν μπορεί να βρεθεί ερμηνεία τέτοια ώστε ένα αντικείμενο να ανήκει ταυτόχρονα σε ένα σύνολο και στο συμπλήρωμά του. Στη δεύτερη περίπτωση, καθώς δεν έχουμε κάποια γνώση ότι ο άντρας και η γυναίκα είναι ξένες έννοιες (αφού το ερώτημα είναι μβτ κενό TBox) η έννοια αυτή είναι ικανοποιήσιμη. Για την τρίτη περίπτωση, από τα αξιώματα του πίνακα 4.2 βλέπουμε ότι ο Άντρας έχει οριστεί ως η τομή του Ανθρώπου και του συμπληρώματος της έννοιας Γυναίκα. Έτσι λοιπόν σε όλα τα μοντέλα του TBox του πίνακα 4.2 τα σύνολα Άντρας^I και Γυναίκα^I είναι ξένα μεταξύ τους. Στην περίπτωση 4 μπορούμε να φτιάξουμε το μοντέλο $(a^I,b^I)\in\text{εχειΠαιδι}^I$, $(a^I,c^I)\in\text{εχειΠαιδι}^I$, $b^I\in\text{Αρσενικό}^I$, $c^I\notin\text{Αρσενικό}^I$. Για την έννοια της περίπτωσης 5, όπως αναφέραμε και στην ενότητα που είδαμε τη σημασιολογία της έννοιας ∀R.C ένα αντικείμενο a^I ανήκει στο σύνολο $(\forall R.C)^I$ ακόμα και αν το a^I δε συνδέεται μέσω της R^I με κανένα άλλο αντικείμενο. Έτσι λοιπόν μπορούμε να θεωρήσουμε ένα αντικείμενο που ανήκει στην έννοια και δεν έχει κανένα παιδί. Η περίπτωση 6 είναι παρόμοια με την περίπτωση 2. Εφόσον δεν έχουμε κάποια γνώση που να μας επιβάλει το σύνολο Μητέρα^I να είναι υποσύνολο του συνόλου Γονιός^I σε όλα τα μοντέλα, τότε μπορούμε να φτιάξουμε μια ερμηνεία στην οποία το σύνολο Μητέρα^I να είναι υπερσύνολο του συνόλου Γονιός^I. Στην περίπτωση 7 από τα αξιώματα του πίνακα 4.2 σε κάθε μοντέλο του TBox η ερμηνεία της έννοιας Γονιός ορίζεται ως η ένωση των ερμηνειών Μητέρα και Πατέρας.
Όπως αναφέραμε στην εισαγωγή της ενότητας αυτής όλες οι υπηρεσίες εξαγωγής συμπερασμάτων μπορούν να αναχθούν σε ακριβώς μια και πιο συγκεκριμένα μπορούν να αναχθούν στο πρόβλημα της μη-ικανοποιησιμότητας (unsatisfiability) (Donini et. al., 1994). Πιο συγκεκριμένα αν η ΠΛ περιέχει άρνηση τότε έχουμε τις εξής ισοδυναμίες:

- Η έννοια C υπάγεται στην έννοια D αν και μόνο αν (ανν) η έννοια C⊓¬D είναι μη-ικανοποιήσιμη.
- Η έννοιες C και D είναι ισοδύναμες ανν και οι δυο έννοιες C⊓¬D και ¬C⊓D είναι μη-ικανοποιήσιμες.
- Οι έννοιες C και D είναι ξένες μεταξύ τους ανν η έννοια C⊓D είναι μη-ικανοποιήσιμη.

Για παράδειγμα έστω ότι θέλουμε να εξετάσουμε αν ισχύει ∃R.C⊓∃R.D⊓≤1R⊑∃R.(C⊓D) μβτ κενό TBox. Τότε μετασχηματίζουμε το ερώτημα υπαγωγής στην ικανοποιησιμότητα της έννοιας ∃R.C⊓∃R.D⊓≤1R⊓¬(∃R.(C⊓D)) μβτ κενό TBox. Χρησιμοποιώντας τώρα την ισοδυναμία μεταξύ του ∃ και του ∀ που

είδαμε στην προηγούμενη ενότητα η παραπάνω έννοια μπορεί να γραφτεί ως, $\exists R.C \sqcap \exists R.D \sqcap \leq 1R \sqcap (\forall R.\neg(C \sqcap D))$, η οποία εν συνεχεία χρησιμοποιώντας την ισοδυναμία μεταξύ του κατασκευαστή $\sqcap$ και $\sqcup$ μπορεί να γραφτεί ως $\exists R.C \sqcap \exists R.D \sqcap \leq 1R \sqcap (\forall R.(\neg C \sqcup \neg D))$.

3.2 Υπηρεσίες Σωμάτων Ισχυρισμών

Ας στρέψουμε τώρα την προσοχή μας στα σώματα ισχυρισμών. Έστω *A* ένα ABox και *T* ένα TBox.

•Συνέπεια (consistency): Το *A* είναι *συνεπές (consistent)* μβτ *T* αν υπάρχει μοντέλο του *T* το οποίο είναι και μοντέλο του *A*.

•Συνεπαγωγή (entailment): Το *A συνεπάγεται (entails)* έναν ισχυρισμό φ μβτ *T*, αν κάθε μοντέλο του *A* και του *T* ικανοποιεί τον ισχυρισμό. Σε αυτήν την περίπτωση γράφουμε $T,A \models \varphi$

Ας δούμε μερικά παραδείγματα. Οι παρακάτω ισχυρισμοί ισχύουν:

0Το ABox *A*={Άντρας(Γιώργος), Γυναίκα(Γιώργος)} είναι συνεπές μβτ κενό TBox.
1Το παραπάνω ABox δεν είναι συνεπές μβτ TBox του πίνακα 4.2.
2Έστω *A*={εχειΠαιδι(Πέτρος,Γιώργος), Άντρας(Γιώργος)}. Τότε *A*⊨Πέτρος:∃εχειΠαιδι.Άντρας μβτ κενό TBox.
3Έστω το προηγούμενο ABox. Τότε το *A* δε συνεπάγεται τον ισχυρισμό Πέτρος:∀εχειΠαιδι.Άντρας μβτ κενό TBox.

Όπως είδαμε στα παραδείγματα ικανοποιησιμότητας εννοιών η έλλειψη πληροφορίας για τις έννοιες Άντρας και Γυναίκα συνεπάγεται ότι μπορούμε να κατασκευάσουμε τουλάχιστον μια (στην πραγματικότητα μπορούμε άπειρες) ερμηνεία όπου οι ερμηνείες των εννοιών αυτών να έχουν κοινά αντικείμενα. Αρκεί να αντιστοιχίσουμε το άτομο Γιώργος σε ένα από αυτά τα αντικείμενα. Με την εισαγωγή τώρα της γνώσης του TBox του πίνακα 4.2 το παραπάνω ABox δεν είναι πια συνεπές καθώς οι έννοιες είναι πλέον ξένες μεταξύ τους. Ας δούμε την περίπτωση 3. Σε κάθε μοντέλο του ABox θα ισχύει ότι $(\text{Πέτρος}^{\mathcal{I}}, \text{Γιώργος}^{\mathcal{I}}) \in \text{εχειΠαιδι}^{\mathcal{I}}$ και $\text{Γιώργος}^{\mathcal{I}} \in \text{Άντρας}^{\mathcal{I}}$, έτσι ώστε να ικανοποιούνται οι ισχυρισμοί του ABox. Αυτό σημαίνει ότι σε κάθε μοντέλο του ABox υπάρχει ένα αντικείμενο του $\Delta^{\mathcal{I}}$ που να συμμετέχει στη σχέση $\text{εχειΠαιδι}^{\mathcal{I}}$ με το $\text{Πέτρος}^{\mathcal{I}}$ και ταυτόχρονα το αντικείμενο αυτό να ανήκει στο σύνολο $\text{Άντρας}^{\mathcal{I}}$ δηλαδή στην ερμηνεία της έννοιας Άντρας. Ας εξετάσουμε τώρα την περίπτωση 4. Όπως αναλύσαμε στην περίπτωση 3 το ABox μας εγγυάται την ύπαρξη ενός αντικειμένου για το οποίο ισχύουν οι παραπάνω ισχυρισμοί. Αυτό όμως δε αποκλείει την περίπτωση σε μια ερμηνεία εκτός από το αντικείμενο αυτό να υπάρχουν και άλλα που ενώ συμμετέχουν στη σχέση $\text{εχειΠαιδι}^{\mathcal{I}}$ με το $\text{Πέτρος}^{\mathcal{I}}$ να μη συμμετέχουν στο σύνολο $\text{Άντρας}^{\mathcal{I}}$. Αυτό ουσιαστικά σημαίνει ότι τα ABoxes αποτελούνε μια αφαίρεση ενός μόνο κομματιού (τμήματος) του κόσμου μας και ποτέ δεν αντιπροσωπεύουν ολόκληρο τον κόσμο. Άρα αν κάποιος ισχυρισμός δε βρίσκεται μέσα στο ABox αυτό δε σημαίνει ότι ο ισχυρισμός δεν ισχύει αλλά ότι δεν ξέρουμε αν ισχύει και ότι μπορούμε να δημιουργήσουμε ερμηνείες στις οποίες ο ισχυρισμός αυτό να ικανοποιείται και άλλες στις οποίες μπορεί να μην ικανοποιείται. Η υπόθεση αυτή ονομάζεται *υπόθεση ανοικτού κόσμου (open-world assumption)*.

Όπως και στην περίπτωση των προβλημάτων των σωμάτων ορολογιών έτσι και στην περίπτωση των προβλημάτων των σωμάτων ισχυρισμών όλα τα προβλήματα και πιο συγκεκριμένα το πρόβλημα της συνεπαγωγής μπορεί να αναχθεί
στο πρόβλημα της συνέπειας ενός ABox. Έτσι λοιπόν έχουμε $\mathcal{A} \models \varphi$ μβτ TBox $\mathcal{T}$ αν το $\mathcal{A} \cup \{\neg\varphi\}$ είναι ασυνεπές (inconsistent) μβτ $\mathcal{T}$. Αυτή η αναγωγή μπορεί να γίνει αντιληπτή ως εξής. Αν το $\mathcal{A} \cup \{\neg\varphi\}$ είναι συνεπές μβτ $\mathcal{T}$, δηλαδή αν μπορέσουμε να βρούμε μια ερμηνεία που να ικανοποιεί τους ισχυρισμούς του $\mathcal{A}$ και του $\mathcal{T}$ αλλά ταυτόχρονα και τον ισχυρισμό $\neg\varphi$ τότε θα έχουμε βρει μια ερμηνεία που να είναι μοντέλο του $\mathcal{A}$ και του $\mathcal{T}$ και να μην ικανοποιεί τον ισχυρισμό φ. Τέλος το πρόβλημα της ικανοποιησιμότητας μιας έννοιας C μβτ $\mathcal{T}$ μπορεί και αυτό να αναχθεί στο πρόβλημα της συνέπειας ενός ABox $\mathcal{A}$ μβτ $\mathcal{T}$. Πιο συγκεκριμένα το C είναι ικανοποιήσιμο μβτ $\mathcal{T}$ αν το $\mathcal{A}=\{C(a)\}$ είναι συνεπές μβτ $\mathcal{T}$, όπου a είναι ένα καινούργιο άτομο. Είναι σημαντικό να προσέξουμε ότι το πρόβλημα της υπαγωγής μιας έννοιας σε μια άλλη δεν επηρεάζεται από την ύπαρξη και τους ισχυρισμούς ενός
ABox (Nebel, 1990a).

3.3 Εξάλειψη του Σώματος Ορολογίας

Όπως έχουμε δει μέχρι στιγμής όλα τα ενδιαφέροντα ερωτήματα γίνονται πάντα με βάση κάποιο TBox $\mathcal{T}$. Αυτό γιατί το TBox αυτό αποτελεί ένα σημαντικό κομμάτι της γνώσης μας το οποίο μας θέτει περιορισμούς στα πιθανά μοντέλα, όπως για παράδειγμα στην περίπτωση των εννοιών του Άντρα και της Γυναίκας. Όπως είδαμε ένα TBox μπορεί να περιλαμβάνει περίπλοκα αξιώματα. Για παράδειγμα οι έννοιες που ορίζουν μια άλλη έννοια μπορούν και αυτές να έχουν ορισμούς σε κάποιο άλλο σημείο του TBox. Μπορεί επίσης να έχουμε αξιώματα με κύκλους αλλά και υπαγωγές γενικών εννοιών. Το ερώτημα του τίθεται λοιπόν είναι με ποιον τρόπο μπορούμε να λάβουμε υπόψη μας τη γνώση που μας περιγράφει ένα TBox; Στη συνέχεια θα δούμε ότι εφαρμόζοντας μια προ-επεξεργασία πάνω σε ένα TBox μπορούμε να το εξαλείψουμε πλήρως και να ανάγουμε τα προβλήματα μας μβτ κενό TBox.
Οι διαδικασίες που εφαρμόζονται για την απαλοιφή ενός TBox χωρίζονται σε δυο κατηγορίες ανάλογα με το είδος των αξιωμάτων που περιέχονται μέσα σε αυτό. Έτσι λοιπόν διακρίνουμε δυο είδη TBox. Τα *απλά (simple)* TBox και τα *γενικευμένα (general)* και/ή *κυκλικά (cyclic)*. Τα απλά TBox περιέχουν μόνο αξιώματα της μορφής $A \sqsubseteq C$ και $A \equiv C$, όπου A είναι μια πρωτογενής έννοια και C είναι μια οποιαδήποτε ΠΛ έννοια. Τα γενικευμένα TBox είναι αυτά που περιέχουν αξιώματα
GCI, δηλαδή $C \sqsubseteq D$, ενώ κυκλικά είναι αυτά που τα αξιώματά τους περιέχουν κύκλους. Για παράδειγμα τα TBox $\mathcal{T}=\{C \sqsubseteq D, D \sqsubseteq E, E \sqsubseteq C\}$ και $\mathcal{T}=\{C \sqsubseteq D \sqcap \exists R.C\}$ περιέχουν και τα δυο κύκλους.
Η διαδικασία που εφαρμόζουμε σε ένα απλό TBox ονομάζεται *ξεδίπλωμα (unfolding)* ή *επέκταση (expansion)*. Η διαδικασία αυτή λειτουργεί ως εξής: Κάθε αξίωμα υπαγωγής της μορφής $A \sqsubseteq C$ αντικαθίσταται από ένα αξιώματα ισοδυναμίας της μορφής $A \equiv A' \sqcap C$, όπου A' είναι μια νέα πρωτογενής έννοια που δεν εμφανίζεται πουθενά στο $\mathcal{T}$. Διαισθητικά η έννοια αυτή αναπαριστά τα χαρακτηριστικά εκείνα που ξεχωρίζουν την έννοια A από τα υπόλοιπα μέλη της έννοιας C. Στη συνέχεια αν η έννοια C είναι μια πολύπλοκη έννοια, που σχηματίζεται από επιμέρους πρωτογενής έννοιες, ανατρέχουμε το TBox ψάχνοντας για τον ορισμό των εννοιών αυτών αντικαθιστώντας τους στη θέση στην οποία βρίσκονται στο C. Η επέκταση ενός TBox $\mathcal{T}$ συμβολίζεται με $\mathcal{T}'$.
Ας θεωρήσουμε για παράδειγμα το TBox $\mathcal{T}=\{\text{Γυναίκα} \sqsubseteq \text{Άνθρωπος}, \text{Άντρας} \equiv \text{Άνθρωπος} \sqcap \neg\text{Γυναίκα}\}$ και ας εφαρμόσουμε τη διαδικασία επέκτασης. Το

πρώτο αξίωμα δε μας καθορίζει πλήρως την έννοια Γυναίκα. Έτσι λοιπόν το αντικαθιστούμε με το αξίωμα ισοδυναμίας Γυναίκα≡Γυναίκα'⊓Άνθρωπος. Αν κοιτάξουμε την αναλογία αυτού του αξιώματος με αυτά του πίνακα 4.2 θα δούμε διαισθητικά ότι η έννοια Γυναίκα' θα μπορούσε να αντιστοιχηθεί στην έννοια Θηλυκό και άρα είναι αυτό το οποίο έλλειπε για να καθοριστεί η έννοια Γυναίκα πλήρως. Προχωρώντας βλέπουμε ότι η έννοια Άντρας ορίζεται από την πρωτογενή έννοια Άνθρωπος και την έννοια Γυναίκα η οποία και ορίζεται κάπου αλλού στο TBox. Έτσι λοιπόν αντικαθιστούμε τον ορισμό της και λαμβάνουμε το αξίωμα Άντρας≡Άνθρωπος⊓¬(Γυναίκα'⊓Άνθρωπος). Αξίζει να σημειώσουμε ότι η διαδικασία επέκτασης ενός TBox μπορεί να οδηγήσει σε ένα νέο TBox που είναι εκθετικά μεγαλύτερο από το αρχικό TBox. Μια τέτοια ορολογία είναι η $\mathcal{T}=\{C_i=\forall R.C_{i+1}\sqcap\forall S.C_{i+1}\}$ για $1\leq i\leq n-1$. Παρόλα αυτά υπάρχουν τεχνικές στις οποίες το ξεδίπλωμα του TBox γίνεται μόνον όταν αυτό χρειάζεται πράγμα το οποίο μας οδηγεί σε αλγορίθμους που χρειάζονται πολυωνυμικό χώρο σε σχέση με το αρχικό TBox. Η τεχνική αυτή ονομάζεται *αργό ξεδίπλωμα (lazy unfolding)* και οφείλεται στον Lutz (1999). Συνοψίζοντας, αν θέλουμε να ελέγξουμε τη ικανοποιησιμότητα
της έννοιας Άντρας⊓Γυναίκα μβτ TBox $\mathcal{T}$ του πίνακα 4.2, επεκτείνουμε το TBox και τέλος αντικαθιστούμε τους ορισμούς των εννοιών Άντρας και Γυναίκα στην προηγούμενη έννοια. Εφαρμόζοντας τη διαδικασία αυτή λαμβάνουμε την έννοια έννοιας, Άνθρωπος⊓¬(Άνθρωπος⊓Θηλυκό)⊓(Άνθρωπος⊓Θηλυκό). Αντί τώρα να ελέγξουμε την ικανοποιησιμότητα της αρχικής έννοιας μβτ TBox του πίνακα 4.2 ελέγχουμε την ικανοποιησιμότητα της έννοιας αυτής μβτ κενό TBox.

Ας ασχοληθούμε τώρα με τα γενικευμένα και τα κυκλικά TBox. Όπως είναι προφανές στα TBox αυτά δεν μπορούμε να εφαρμόσουμε τη μέθοδο της επέκτασης καθώς στην περίπτωση κυκλικών αξιωμάτων αυτή δε θα τελείωνε ποτέ, ενώ στην περίπτωση των υπαγωγών γενικών εννοιών για πολλές έννοιες μπορεί να μην έχουμε τους ορισμούς τους. Η περίπτωση αυτών των TBox αντιμετωπίζεται με μια διαδικασία που ονομάζεται *εσωτερίκευση (internalization)*. Εν συντομία η μέθοδος αυτή κωδικοποιεί τους σημασιολογικούς περιορισμούς που περιγράφονται από τα αξιώματα υπαγωγής και ισοδυναμίας σε μια και μοναδική έννοια. Στη συνέχεια παίρνει το συνδυασμό των εννοιών και τέλος σχηματίζει ένα και μοναδικό γενικευμένο αξίωμα το οποίο ο αλγόριθμος μπορεί να χειριστεί αποδοτικά.

Έστω το αξίωμα $C\sqsubseteq D$. Γενικά μπορούμε πολύ εύκολα να αντιληφθούμε ότι για οποιοδήποτε αντικείμενο του κόσμου μας a^I είτε θα συμβαίνει το a^I να ανήκει στο σύνολο C^I είτε το a^I να μην ανήκει στο σύνολο αυτό άρα να ανήκει στο συμπλήρωμα του $(\neg C)^I$. Στην πρώτη περίπτωση για να ικανοποιεί η ερμηνεία I το αξίωμα υπαγωγής θα πρέπει το C^I να είναι υποσύνολο του D^I άρα λοιπόν θα πρέπει το a^I να ανήκει και στο σύνολο D^I. Εφόσον αυτό πρέπει να ισχύει για όλες τις ερμηνείες που ικανοποιούν το αξίωμα, μπορούμε να αφαιρεθούμε από αυτές και να γράψουμε γενικά ότι $C\sqsubseteq D$ ανν κάθε άτομο a ανήκει ή στην έννοια $\neg C$ ή στην έννοια D, άρα δηλαδή στην έννοια $\neg C\sqcup D$. Έτσι λοιπόν ένα γενικευμένο και/ή κυκλικό TBox της μορφής $\mathcal{T}=\{C_i\sqsubseteq D_i\}$ για $1\leq i\leq n$ μετασχηματίζεται στην έννοια $C_{\mathcal{T}}\equiv(\neg C_1\sqcup D_1)\sqcap(\neg C_2\sqcup D_2)\sqcap\ldots\sqcap(\neg C_n\sqcup D_n)$. Τα αξιώματα της μορφής $C\equiv D$ αντιμετωπίζονται ως δυο αξιώματα της μορφής $C\sqsubseteq D$ και $D\sqsubseteq C$. Τέλος δημιουργούμε το γενικευμένο αξίωμα $\top\sqsubseteq C_{\mathcal{T}}$ για να υποδηλώσουμε ότι οι παραπάνω περιορισμοί πρέπει να εφαρμοστούν σε όλα τα άτομα του κόσμου μας. Αξίζει να αναφέρουμε ότι η παραπάνω διαδικασία μπορεί επίσης να εφαρμοστεί και σε ένα απλό TBox.

Έστω για παράδειγμα το TBox $\mathcal{T}$={C⊔D⊑E}. Η έννοια C υπάγεται στην έννοια E μβτ $\mathcal{T}$; Προφανώς η απάντηση είναι καταφατική καθώς σε κάθε μοντέλο του $\mathcal{T}$ η ερμηνεία της E θα περιλαμβάνει την ένωση των ερμηνειών της C και D. Ας δούμε όμως πως λειτουργεί η μέθοδος. Η εσωτερικευμένη μορφή του TBox είναι η εξής: ⊤⊑¬(C⊔D)⊔E. Εφαρμόζοντας τους κανόνες DeMorgan η έννοια του δεξιού μέλους μπορεί να μετασχηματιστεί σε (¬C⊓¬D)⊔E. Στη συνέχεια ανάγουμε την υπαγωγή στο πρόβλημα της ικανοποιησιμότητας της έννοιας C⊓¬E και τέλος ανάγουμε αυτό στο πρόβλημα της συνέπειας του ABox {a:C⊓¬E}. Λόγω του γενικευμένου αξιώματος που προέκυψε από τη διαδικασία εσωτερίκευσης σε επόμενο βήμα το ABox μετασχηματίζεται στο {a:C⊓¬E, a:(¬C⊓¬D)⊔E}. Αν εξετάσουμε προσεκτικά το ABox αυτό θα δούμε ότι δεν μπορούμε να βρούμε ερμηνεία που να ικανοποιεί ταυτόχρονα τους ισχυρισμούς αυτούς. Πιο συγκεκριμένα για να ικανοποιείται ο ισχυρισμός a:C⊓¬E θα πρέπει να ισχύει $a^I \in (C \sqcap \neg E)^I$. Αν βασιστούμε στη σημασιολογία του κατασκευαστή ⊓ τότε καταλήγουμε ότι πρέπει $a^I \in C^I$ και $a^I \in (\neg E)^I$. Επίσης όμως λόγω του δεύτερου ισχυρισμού πρέπει είτε $a^I \in (\neg E)^I$, το οποίο έρχεται σε σύγκρουση με τους προηγούμενους περιορισμούς, ή $a^I \in (\neg C \sqcap \neg D)^I$ το οποίο συνεπάγεται ότι $a^I \in (\neg C)^I$ και $a^I \in (\neg D)^I$ το οποίο επίσης έρχεται σε σύγκρουση με τους αρχικούς περιορισμούς. Άρα λοιπόν το ABox δεν είναι συνεπές και η έννοια C υπάγεται στην έννοια D. Παρατηρήστε ότι στην περίπτωση της εσωτερίκευσης που δείξαμε δεν έχουμε εξαλείψει πλήρως το TBox αλλά έχουμε ένα αξίωμα. Ουσιαστικά αυτό το αξίωμα μας λέει ποιους περιορισμούς πρέπει να εφαρμόσουμε στα άτομα του κόσμου μας. Όπως θα δούμε σε επόμενη ενότητα σε πιο εκφραστικές ΠΛ ακόμα και αυτό το μοναδικό αξίωμα μπορεί να εξαλειφθεί τελείως.

Υπηρεσίες Διαδικτύου

Εισαγωγή

Τα τελευταία χρόνια πολλές ερευνητικές ομάδες στο χώρο του διαδικτύου έχουν επικεντρώσει τις προσπάθειές τους στην ανάπτυξη υπηρεσιών, οι οποίες θα δώσουν επιπρόσθετες δυνατότητες στους χρήστες για την εξυπηρέτηση των ολοένα αυξανόμενων αναγκών τους. Οι υπηρεσίες αυτές έχουν υιοθετηθεί από πολλούς κατασκευαστές λογισμικού, σαν μέσο για τον εμπλουτισμό των ιστοσελίδων – και όχι μόνο, με δυναμικά στοιχεία, τα οποία βασίζονται σε συγκεκριμένες διεπαφές.

Διάφοροι οργανισμοί εμπλέκονται στον καθορισμό προδιαγραφών για τις «Υπηρεσίες Διαδικτύου» με αποτέλεσμα αρκετοί ορισμοί να έχουν διατυπωθεί, με κυρίαρχο αυτόν της ερευνητικής ομάδας W3C Web Services Architecture Working Group

«Μια υπηρεσία ιστού (web service), είναι μια εφαρμογή λογισμικού, η οποία προσδιορίζεται από ένα URI , της οποίας οι διεπαφές και οι συνδέσεις είναι δυνατόν να καθοριστούν, να περιγραφούν και να εντοπισθούν από τεχνουργήματα XML, και υποστηρίζουν απευθείας αλληλεπίδραση με άλλες εφαρμογές λογισμικού, χρησιμοποιώντας μηνύματα βασισμένα σε XML, διαμέσου πρωτοκόλλων βασισμένων στο διαδίκτυο.»

Σύμφωνα με τον παραπάνω ορισμό, η «Υπηρεσία Διαδικτύου» καθορίζεται σαν ένας πόρος, όχι μόνο μέσα στον παγκόσμιο ιστό, αλλά και σε επίπεδο ενός τοπικού εσωτερικού δικτύου (intranet), ο οποίος παρέχεται από κάποιον οργανισμό. Σημαντικό είναι το γεγονός της ανεξαρτησίας ανάμεσα στις υπολογιστικές πλατφόρμες από τη μεριά του παροχέα της υπηρεσίας και του τελικού χρήστη αυτής (cross platform applications).

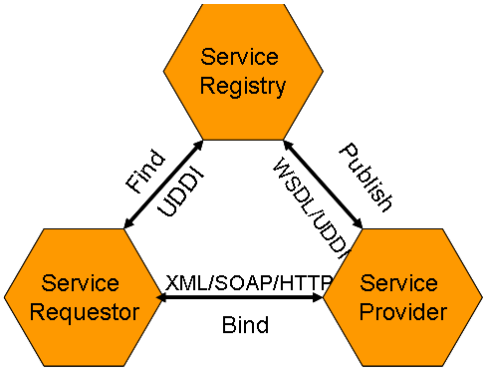
Η «Υπηρεσία Διαδικτύου» είναι διαθέσιμη μέσω των δημοσιευθέντων διεπαφών της, στις οποίες είναι δυνατή η κλήση. Η σχέση αυτή είναι ανάλογη με αυτή του διαφυλλιστή διαδικτύου (web browser) και του παροχέα του δικτυακού τόπου (web application server). Ο διαφυλλιστής δεν επηρεάζεται από τον παροχέα της ιστοσελίδας ή το μηχανισμό που τον εξυπηρετεί, αρκεί το αποτέλεσμα να είναι εκφρασμένο σε γλώσσα HTML.

Από τεχνικής πλευράς, μια «Υπηρεσία Διαδικτύου» είναι μια συλλογή από συσχετιζόμενες υπηρεσίες, που είναι προσβάσιμες μέσα από κάποιο δίκτυο και έχουν μια συγκεκριμένη περιγραφή. Σε αυτό το επίπεδο, η έννοια της «Υπηρεσίας Διαδικτύου» δεν είναι κάτι καινούργιο στην επιστήμη των υπολογιστών. Μέσω των «Υπηρεσιών Διαδικτύου», η επιστήμη των υπολογιστών προσπαθεί να επιλύσει την πρωταρχική πρόκληση του κατανεμημένου υπολογισμού, την εύρεση και την πρόσβαση σε απομακρυσμένους πόρους. Η διαφορά αυτής της προσέγγισης είναι η χρήση ανοιχτών τεχνολογιών και προδιαγραφών (open standards), ελεγχόμενων από ευρείας συναίνεσης οργανισμούς, όπως ο OASIS και ο W3C.

Βασική αρχή των «Υπηρεσιών Διαδικτύου» είναι η μη υλοποίησή τους σε αυστηρά συνεκτικά συνδεδεμένα εξαρτήματα και υπο-ρουτίνες, που η γνώση τους θα είναι αναγκαία κατά τη φάση της ανάπτυξης της υπηρεσίας. Αντίθετα, επιλέχθηκε μια αρκετά πιο δυναμική προσέγγιση. Δόθηκε έμφαση στις διεπαφές, το σημείο επικοινωνίας μεταξύ της ίδιας της υπηρεσίας και του λογισμικού που την καλεί. Βέβαια, αυτό που είναι πρωτοποριακό, δεν είναι η χρήση διεπαφών, αλλά το γεγονός ότι μια «Υπηρεσία Διαδικτύου» είναι άθροισμα πολλών ανεξάρτητων οντοτήτων, μη αυστηρά συνδεδεμένων μεταξύ τους, που όμως επικοινωνούν με απλές ανοιχτές προδιαγραφές.

Υπηρεσιοκεντρική Αρχιτεκτονική (Service Oriented Architecture – SOA)

Το μοντέλο που χρησιμοποιείται στις μέρες μας για την περιγραφή των «Υπηρεσιών Διαδικτύου» βασίζεται κυρίως στο κλασικό επιχειρησιακό μοντέλο, το οποίο έχουν υιοθετήσει οι διάφοροι ιδιωτικού οργανισμοί και απεικονίζεται στο Σχήμα 1.



Σχήμα 1. Αρχιτεκτονική γύρω από την υπηρεσία

Από το παραπάνω σχήμα μπορούμε να διακρίνουμε τις οντότητες που απαρτίζουν την τρέχουσα αρχιτεκτονική των υπηρεσιών διαδικτύου. Αυτές είναι η οντότητα που ζητάει την υπηρεσία (Service Requestor), η οντότητα που παρέχει την υπηρεσία (Service Provider) και τέλος η οντότητα του καταλόγου υπηρεσιών (Service Registry).

Με περισσότερη λεπτομέρεια, κάθε μία οντότητα έχει τους ακόλουθους ρόλους:

- Η οντότητα, η οποία ζητάει την υπηρεσία (Service Requestor), είναι στην ουσία ο «αιτούμενος» (Client) της υπηρεσίας και πυροδοτεί την έναρξη της όλης διαδικασίας. Αρχικά επιφορτίζεται με το έργο της αναζήτησης της κατάλληλης περιγραφής μιας υπηρεσίας τις καταχωρήσεις της υπηρεσίας καταλόγου. Στη συνέχεια, και ενώ έχει εντοπίσει τον επιθυμητό παροχέα, εγκαθιδρύει σύνδεση με αυτόν, καλεί την υπηρεσία και λαμβάνει τα αποτελέσματα.
- Η οντότητα που παρέχει την υπηρεσία (Service Provider), δημοσιοποιεί την περιγραφή της υπηρεσίας ιστού στην οντότητα του καταλόγου υπηρεσιών (Service Registry) και ουσιαστικά παρέχει την υπηρεσία στο διαδίκτυο. Επιπλέον, της ανατίθεται ο ρόλος της λήψης των μηνυμάτων κλήσης για την υπηρεσία από έναν ή περισσότερους αιτούμενους και της παροχής των απαραίτητων αποτελεσμάτων.
- Τέλος, η οντότητα του καταλόγου υπηρεσιών (Service Registry), περιέχει καταχωρήσεις με τις περιγραφές των ήδη δημοσιοποιηθέντων υπηρεσιών διαδικτύου. Για τις υπηρεσίες αυτές παρέχονται τρόποι αναζήτησης ανάμεσα στις διάφορες περιγραφές.

Οι λειτουργίες που παρέχονται μέσω αυτής της αρχιτεκτονικής είναι οι ακόλουθες:

- Η καταχώρηση των απαραίτητων πληροφοριών για μια υπηρεσία από τη μεριά του παροχού (publishing), η οποία επιτυγχάνεται μέσω της χρήσης της γλώσσας περιγραφής WSDL.
- Η αναζήτηση και η εύρεση στους καταλόγους της κατάλληλης περιγραφής μιας «Υπηρεσίας Διαδικτύου» (finding). Η επικρατούσα τεχνολογία καταλόγου ονομάζεται UDDI.
- Τέλος, έχουμε την εγκαθίδρυση σύνδεσης μεταξύ αιτούμενου και παροχέα (binding), κλήση της κατάλληλης υπηρεσίας και αποστολή των αποτελεσμάτων.

Όλα τα παραπάνω επιτυγχάνονται με την ανταλλαγή μηνυμάτων SOAP μεταξύ των τριών εμπλεκόμενων οντοτήτων, η οποία βασίζεται στη γλώσσα μεταδεδομένων XML.

Το κλειδί της παραπάνω αρχιτεκτονικής είναι η περιγραφή της «Υπηρεσίας Διαδικτύου». Η περιγραφή είναι εκείνη που δημοσιοποιείται από τον παροχέα της υπηρεσίας στον κατάλογο υπηρεσιών. Η περιγραφή είναι το αποτέλεσμα της αναζήτησης του αιτούμενου στον κατάλογο υπηρεσιών. Η περιγραφή είναι εκείνη που λέει στον αιτούμενο όσα ακριβώς πρέπει να γνωρίζει προκειμένου να κάνει κλήση της υπηρεσίας που τον ενδιαφέρει. Τέλος, η περιγραφή της υπηρεσίας μπορεί να περιέχει πληροφορία για το τι είδους είναι το αποτέλεσμα που αναμένεται να επιστραφεί ύστερα από την κλήση της υπηρεσίας.

Με την παραπάνω αρχιτεκτονική ήδη υπάρχουσες εφαρμογές μπορούν εύκολα να μετασχηματισθούν σε υπηρεσίες που να εξυπηρετούν άλλες υπάρχουσες – ή και νέες – εφαρμογές. Πολλές υπηρεσίες μπορούν να συνδυαστούν και να συνεργασθούν για την παραγωγή ενός αποτελέσματος με τρόπο διαφανή προς τον τελικό χρήστη, στον οποίο δίνεται η αίσθηση ότι κλήθηκε μία και μόνο υπηρεσία. Επιπρόσθετα, οργανισμοί και επιχειρήσεις μπορούν ευκολότερα να κατασκευάσουν λογισμικό, που να αλληλεπιδρά με επιχειρησιακές διαδικασίες και να ανταποκρίνεται γρήγορα στις αλλαγές του επιχειρησιακού περιβάλλοντος.

Service Oriented Architecture και «Υπηρεσίες Διαδικτύου»

Παρόλο που η τεχνολογία των «Υπηρεσιών Διαδικτύου» και η προαναφερθείσα αρχιτεκτονική SOA συχνά συναντώνται μαζί ή σε κάποιο συνδυασμό, πρέπει να αναφερθεί ότι οι δύο αυτοί όροι είναι διαφορετικοί και πρέπει να ξεχωρίζουν. Η αρχιτεκτονική SOA είναι ένα σενάριο, ή μια προσέγγιση, για την οικοδόμηση συστημάτων, τα οποία επικεντρώνονται σε ένα σύνολο από συστατικά που μπορούν να συνεργασθούν δυναμικά. Οι «Υπηρεσίες Διαδικτύου» από την άλλη μεριά, είναι μια προσέγγιση για τον τρόπο που μπορεί να αναπτυχθεί μια αρχιτεκτονική SOA. Το μοντέλο των «Υπηρεσιών Διαδικτύου» παρέχει προδιαγραφές για ένα σύνολο από γλώσσες και τεχνολογίες βασισμένες σε XML, που μπορούν να χρησιμοποιηθούν για την οικοδόμηση συστημάτων με αρχιτεκτονική SOA.

Πρωτόκολλο SOAP

Στα τέλη της δεκαετίας του 90, και συγκεκριμένα το 1997, μεγάλες εταιρείες, όπως η Microsoft, άρχισαν να διερευνούν κατά πόσο ο κατανεμημένος υπολογισμός μπορεί να βασιστεί στη γλώσσα XML. Ο σκοπός της έρευνας αυτής ήταν να γίνει εφικτή η επικοινωνία μεταξύ των εφαρμογών μέσω απομακρυσμένων κλήσεων διαδικασιών (Remote Procedure Calls – RPCs), χρησιμοποιώντας απλά πρωτόκολλα δικτύου, όπως το HTTP. Το 1999 έκανε την εμφάνισή του το SOAP, ένας RPC μηχανισμός βασισμένος σε XML. Το 2000 ο οργανισμός W3C ασχολείται με την ιδέα αυτή και ύστερα από αρκετές αλλαγές, βελτιώσεις και τροποποιήσεις 2 ολόκληρων χρόνων, το 2003 δηλαδή, το SOAP με την έκδοση 1.2 γίνεται η προτεινόμενη προδιαγραφή – πρωτόκολλο για τις υπηρεσίες διαδικτύου.

Στον πυρήνα του, το SOAP, είναι μια προδιαγραφή για ένα απλό αλλά ταυτόχρονα ευέλικτο XML πρωτόκολλο δεύτερης γενιάς. Το σημαντικό είναι ότι καθώς η έρευνα ξεκίνησε από τον κατανεμημένο υπολογισμό, το SOAP είναι το πλέον κατάλληλο πρωτόκολλο αφού ειδικεύεται σε τέτοια περιβάλλοντα. Το SOAP παρέχει τα κάτωθι χαρακτηριστικά και μηχανισμούς:

- Μηχανισμός για τον ορισμό της πληροφορίας στην επικοινωνία – Στο SOAP, όλη η πληροφορία είναι καταχωρημένη σε ένα ξεκάθαρο και ταυτοποιήσιμο SOAP μήνυμα (SOAP message). Αυτό γίνεται μέσω ενός SOAP φακέλου

(SOAP envelope), που περικλείει όλες τις απαραίτητες πληροφορίες. Ένα μήνυμα SOAP, μπορεί να έχει σώμα (body), το οποίο περιέχει πληροφορία σε XML δομή. Επίσης, μπορεί να διαθέτει έναν αριθμό από επικεφαλίδες (headers), οι οποίες ενσωματώνουν επιπλέον πληροφορίες έξω από το σώμα του μηνύματος.

- Διεργασιακό μοντέλο – Αυτό το μοντέλο ορίζει ένα σύνολο κανόνων βάσει των οποίων γίνονται οι διαπραγματεύσεις μεταξύ των SOAP μηνυμάτων και του λογισμικού. Διακρίνεται από την απλότητά του.
- Μηχανισμός για αντιμετώπιση σφαλμάτων – Το SOAP παρέχει το μηχανισμό αυτό με τη μορφή των SOAP faults, τα οποία όταν χρησιμοποιούνται μπορεί να προσδιοριστεί η πηγή που προκάλεσε το σφάλμα. Επιπλέον, παρέχουν τη δυνατότητα ανταλλαγής διαγνωστικών πληροφοριών μεταξύ των μελών της επικοινωνίας.
- Μοντέλο επεκτασιμότητας – Το μοντέλο αυτό χρησιμοποιεί SOAP επικεφαλίδες για την υλοποίηση προεκτάσεων πάνω στο SOAP. Οι επικεφαλίδες περιέχουν κομμάτια από δεδομένα με δυνατότητα επέκτασης, τα οποία ταξιδεύουν μαζί με το μήνυμα και μπορούν να γίνουν στόχος για επέκταση σε συγκεκριμένους κόμβους του δικτύου.
- Ευέλικτος μηχανισμός για αναπαράσταση δεδομένων – Ο μηχανισμός αυτός επιτρέπει στα δεδομένα σε οποιαδήποτε σειριακή μορφή κι αν βρίσκονται να αναπαρασταθούν σε XML μορφή.
- Σύμβαση για αναπαράσταση των RPCs και των απαντήσεών τους σαν SOAP μηνύματα – Οι απομακρυσμένες κλήσεις διαδικασιών είναι αρκετά διαδεδομένες στον κατανεμημένο προγραμματισμό/ υπολογισμό και μπορούν να αναπαρασταθούν καλά μέσω του μηχανισμού αυτού σαν SOAP μηνύματα.
- Πρωτόκολλο εγκαθίδρυσης σύνδεσης – Το πρωτόκολλο αυτό ορίζει μια αρχιτεκτονική για την οικοδόμηση συνδέσεων επικοινωνίας ώστε να είναι εφικτή η ανταλλαγή SOAP μηνυμάτων πάνω σε μέσα μεταφοράς και επικοινωνιακά κανάλια. Χρησιμοποιείται το HTTP πρωτόκολλο, καθώς είναι το πιο διαδεδομένο και ευρέως χρησιμοποιούμενο στο διαδίκτυο.

Όσον αφορά στα μηνύματα SOAP, εφόσον το πρωτόκολλο είναι βασισμένο στην XML, είναι αναμενόμενο τα μηνύματα να έχουν μια τέτοια μορφή και ουσιαστικά να μην είναι τίποτε άλλο, παρά XML έγγραφα. Το στοιχείο-ρίζα του μηνύματος είναι το soapenv:Envelope, το οποίο περικλείει το στοιχείο soapenv:Body, που περιέχει πληροφορία σχετιζόμενη με το σκοπό του μηνύματος. Το στοιχείο soapenv:Body με τη σειρά του μπορεί να περιέχει άλλα στοιχεία, τα οποία να σχετίζονται ή να αναπαριστούν την απομακρυσμένη κλήση διαδικασίας που πρόκειται να εκτελεστεί. Το όνομα της μεθόδου που θα κληθεί και το όνομα του στοιχείου αυτού περιέχονται μέσα στο σώμα του μηνύματος.

Τα μηνύματα αυτά στέλνονται μέσω πρωτοκόλλου HTTP .με τη μέθοδο POST. Η απάντηση σε ένα τέτοιο μήνυμα, που και πάλι μεταφέρεται μέσω HTTP είναι και αυτή ένα SOAP μήνυμα. Το μήνυμα της απάντησης περικλείεται κι αυτό από το συστατικό soapenv:Envelope, που περιέχει το soapenv:Body, το οποίο ενσωματώνει και τη βασική πληροφορία του μηνύματος, στην οποία συγκαταλέγεται και μια κωδικοποιημένη αναπαράσταση του αποτελέσματος της απομακρυσμένης κλήσης που πραγματοποιήθηκε.

Το SOAP παρέχει, όπως ήδη αναφέρθηκε, τη δυνατότητα σε ένα μήνυμα να συμπεριλαμβάνονται στοιχεία – επικεφαλίδες. Ο ρόλος τους είναι και ο σημαντικότερος λόγος που χρησιμοποιούμε τα μηνύματα SOAP και όχι απλά XML έγγραφα. Τα στοιχεία αυτά (headers) έχουν την ιδιότητα να αναπαριστούν ένα επιπρόσθετο και επεκτάσιμο είδος πληροφορίας το οποίο μεταφέρεται μαζί με το υπόλοιπο μήνυμα, χωρίς να τροποποιείται ο κυρίως πυρήνας του μηνύματος. Για να γίνει κατανοητό αυτό ας δούμε ένα παράδειγμα της αληθινής ζωής ακριβώς

αντίστοιχο με την ιδέα των SOAP μηνυμάτων. Ας υποθέσουμε ότι θέλουμε να στείλουμε ένα έγγραφο αλλά και κάποιες επιπρόσθετες πληροφορίες χωρίς όμως να θέλουμε να μαρκάρουμε το έγγραφο με αυτές. Τοποθετούμε, λοιπόν, το έγγραφο στο φάκελο και στη συνέχεια προσθέτουμε μία ή δύο σελίδες χαρτί, οι οποίες περιγράφουν την επιπλέον πληροφορία που θέλαμε να στείλουμε. Η αντιστοιχία του παραδείγματος αυτού με το πρωτόκολλο που περιγράφουμε είναι ότι ο φάκελος είναι το στοιχείο soapenv:Envelope, το έγγραφο που θέλουμε να στείλουμε είναι το στοιχείο soapenv:Body και οι σελίδες με την επιπλέον πληροφορία είναι τα στοιχεία επικεφαλίδες (headers).

Η χρήση των επικεφαλίδων για την προσθήκη λειτουργικότητας σε ένα μήνυμα SOAP είναι γνωστή και σαν κατακόρυφη επέκταση, γιατί τα στοιχεία αυτά τοποθετούνται στην κορυφή του μηνύματος πριν το σώμα του.

Για λόγους ασφάλειας σε περιπτώσεις επεκτασιμότητας χρησιμοποιώντας στοιχεία επικεφαλίδες, η προδιαγραφή του SOAP ορίζει τη μεταβλητή mustUnderstand. Όταν η τιμή της είναι αληθής, τότε ο παραλήπτης του μηνύματος πρέπει να συμφωνήσει ότι αποδέχεται όλους τους όρους της επικεφαλίδας του μηνύματος προκειμένου να συνεχιστεί η επεξεργασία του. Στην περίπτωση που η τιμή της μεταβλητής αυτή είναι ψευδής, τότε η επικεφαλίδα έχει προαιρετικό ρόλο και η επεξεργασία του μηνύματος μπορεί να γίνει και από παραλήπτες που αγνοούν τις προδιαγραφές που θέτει η επικεφαλίδα.
Αντίστοιχα με την κατακόρυφη επεκτασιμότητα, υπάρχει και η οριζόντια, η οποία έχει να κάνει με την αντιστοίχηση διαφορετικών κομματιών του μηνύματος με διαφορετικούς παραλήπτες. Αυτό επιτυγχάνεται μέσω των μεσολαβητών SOAP (SOAP intermediaries). Αυτές οι εφαρμογές μπορούν να επεξεργάζονται μέρη του μηνύματος SOAP καθώς αυτό ταξιδεύει προς τον τελικό του προορισμό. Οι μεσολαβητές μπορούν να αποδέχονται και να προωθούν μηνύματα καθώς κάνουν και κάποιο είδος επεξεργασίας. Σε γενικές γραμμές, στον αληθινό κόσμο σπάνια ξεκινά ένα μήνυμα από κάποια τοποθεσία και φθάνει απευθείας στον προορισμό του. Για παράδειγμα ας πάρουμε το ηλεκτρονικό ταχυδρομείο. Όταν στέλνουμε ένα μήνυμα αυτό θα περάσει από διάφορους εξυπηρέτες και δρομολογητές προτού «εντοπίσει» τον προορισμό του. Αυτός είναι ένας από τους κυριότερους λόγους που χρειάζονται οι μεσολαβητές στα μηνύματα SOAP, καθώς επίσης χάρη σε αυτούς μπορούμε να έχουμε υπηρεσίες προστιθέμενης αξίας σε κατανεμημένα συστήματα.

Έχοντας περιγράψει αρκετά το SOAP, φτάνουμε στα εξής βήματα που τελικά πρέπει να πραγματοποιηθούν από έναν παραλήπτη ή ένα μεσολαβητή κάποιου μηνύματος SOAP:

- Πρέπει να κατανοηθεί το σύνολο των κανόνων που περιέχει το μήνυμα που παραλείφθηκε. Τα στοιχεία επικεφαλίδες κυρίως μπορεί να περιέχουν τέτοιο πληροφορία.
- Πρέπει να προσδιοριστούν όλα τα σύνολα που ορίζουν τα στοιχεία επικεφαλίδες με προσοχή.
- Αν κάποιο σημείο δεν είναι κατανοητό – και αυτό έχει να κάνει με τη μεταβλητή mustUnderstand – τότε πρέπει να συνταχθεί ένα μήνυμα λάθους (SOAP fault). Σε τέτοιο περίπτωση η περαιτέρω επεξεργασία του μηνύματος πρέπει να σταματήσει. Λάθη που έχουν να κάνουν με το σώμα του μηνύματος δεν εντοπίζονται σε αυτό το βήμα.
- Ακολουθεί επεξεργασία των επικεφαλίδων. Σε περίπτωση που το μήνυμα βρίσκεται στον τελικό παραλήπτη και όχι σε κάποιον ενδιάμεσο, τότε γίνεται και η επεξεργασία του σώματος του μηνύματος.
- Στην περίπτωση που το μήνυμα βρίσκεται σε κάποιον ενδιάμεσο μεσολαβητή και δεν υπάρχει κάποιο πρόβλημα – με την έννοια ότι δεν υπήρξε ανάγκη να συνταχθεί κάποιο μήνυμα λάθους – τότε η πορεία του

μηνύματος συνεχίζεται με την προώθησή του στον επόμενο κόμβο, όπου ακολουθείται η ίδια διαδικασία.

Ουσιαστικά οι μεταβλητή mustUnderstand έχει σα ρόλο να μας δίνει την ευκαιρία να ορίζουμε τι ακριβώς διαδικασίες θα γίνονται σε κάθε κόμβο από τον οποίο περνάει το μήνυμα μέχρι να φτάσει στον τελικό του προορισμό.

Όσον αφορά στα μηνύματα λάθους, αυτά είναι απλά μηνύματα SOAP, τα οποία περιέχουν ένα απλό στοιχείο μέσα στο soapenv:Body, το soapenv:Fault. Η παρουσία αυτού του στοιχείου σηματοδοτεί ότι κάπου υπήρξε κάποιο λάθος. Το μήνυμα λάθους έχει κάποια συγκεκριμένη και καλά ορισμένη δομή ώστε να παρέχει πληροφορία σχετική με το λάθος που συνέβη. Το στοιχείο env:Code περιέχει όλη την απαραίτητη κωδικοποιημένη πληροφορία για το σκοπό αυτό. Από τις πληροφορίες αυτές μπορούμε να διαπιστώσουμε εάν το λάθος προέκυψε λόγω λανθασμένης ή ελλιπούς πληροφορίας από τον αποστολέα, ή λόγω κάποιου λάθους που συνέβη κατά την επεξεργασία του από κάποιον παραλήπτη. Επίσης το λάθος μπορεί να σχετίζεται με την σχέση κάποιου μεσολαβητή με την τιμή της μεταβλητής mustUnderstand. Τέλος, ο λόγος προελεύσεώς του μπορεί να είναι οι μη συμβατές εκδόσεις του πρωτοκόλλου που χρησιμοποιούνταν από κόμβο σε κόμβο. Τα μηνύματα λάθους, ως κανονικά SOAP μηνύματα που είναι, μπορούν να επεκτείνονται με την παρουσία στοιχείων επικεφαλίδων, τα οποία μπορεί να περιέχουν επιπλέον πληροφορία για την ανίχνευση του λάθους και να συμβάλλουν στη συντήρηση των υπηρεσιών πάνω στο δίκτυο.

Ένα εργαλείο για την πραγματοποίηση SOAP συναλλαγών είναι ο Apache Axis , μια μηχανή SOAP ανοιχτού λογισμικού.

Γλώσσα Περιγραφής Υπηρεσιών Διαδικτύου – WSDL

Μέχρι εδώ έχουμε μιλήσει για το SOAP. Έχουμε εξηγήσει ότι είναι βασισμένο στην XML και ότι μέσω αυτού του πρωτοκόλλου μπορούν να γίνουν αιτήσεις για απομακρυσμένες κλήσεις διαδικασιών από κάποιον πελάτη σε κάποιον εξυπηρέτη. Από τη μεριά του πελάτη όμως υπάρχουν δυσκολίες στη σύνταξη του μηνύματος SOAP, καθώς ο πελάτης δε μπορεί να ξέρει τι ακριβώς μήνυμα να στείλει. Το SOAP καθορίζει κάποιους κανόνες και προσφέρει μια συγκεκριμένη φόρμα για τα μηνύματα, αλλά πρέπει να βρεθεί κάποιος άλλος τρόπος ώστε ο πελάτης να είναι σε θέση να γνωρίζει τι μήνυμα να στείλει στον εξυπηρέτη του παροχέα της υπηρεσίας. Επίσης, ο πελάτης πρέπει να γνωρίζει αρκετές λεπτομέρειες προτού στείλει το μήνυμα. Πρέπει να γνωρίζει καταρχάς που ακριβώς να το στείλει, ή ποιες μεθόδους θέλει να καλέσει από την υπηρεσία, ή ποια πρωτόκολλα επικοινωνίας υποστηρίζονται από τον παροχέα της υπηρεσίας.

Προκειμένου να γνωρίζει ο πελάτης όλες τις απαραίτητες πληροφορίες για την κλήση μιας υπηρεσίας διαδικτύου, πρέπει να έχει στην κατοχή του μια περιγραφή αυτής. Για να γίνει η όλη διαδικασία ακόμη πιο εύκολη και πιο τυποποιημένη χρειάζεται ένας καθορισμένος μηχανισμός για την περιγραφή των υπηρεσιών διαδικτύου. Αυτός ο μηχανισμός θα παρέχει πληροφορίες για ακριβώς αυτά που χρειάζεται να ξέρει ο χρήστης που θέλει να καλέσει την υπηρεσία. Επιπρόσθετα, χάρη σε αυτόν το μηχανισμό είναι δυνατή η ανάπτυξη εφαρμογών που θα βοηθούν τους προγραμματιστές στην ανάπτυξη των υπηρεσιών μέσω των περιγραφών τους.

Όπως έχει αναφερθεί και σε προηγούμενη ενότητα, ο ρόλος της υπηρεσίας καταλόγου (Service Registry) σε μια SOA αρχιτεκτονική είναι πολύ σημαντικός. Ο πελάτης, προκειμένου να ανακαλύψει την υπηρεσία που τον ενδιαφέρει, ψάχνει πρώτα στον κατάλογο αυτό. Το αποτέλεσμα της αναζήτησης δεν είναι τίποτε άλλο από την περιγραφή της υπηρεσίας που ζήτησε. Ο παροχέας της υπηρεσίας

διαδικτύου, λοιπόν, δημοσιεύει την περιγραφή της υπηρεσίας του γι' αυτό το λόγο. Για να ξέρει ο κάθε πιθανός πελάτης πώς να εγκαθιδρύσει επικοινωνιακή σύνδεση μαζί του και πως ακριβώς να καλέσει την υπηρεσία που του παρέχει.

Μια καλή περιγραφή μιας υπηρεσίας διαδικτύου πρέπει να αποτελείται από δύο μέρη:

- Τη λειτουργική περιγραφή – Η περιγραφή αυτή περιγράφει ποιες ακριβώς λειτουργίες είναι διαθέσιμες από την υπηρεσία διαδικτύου και ποια είναι η απαιτούμενη σύνταξη που θα πρέπει να έχει το SOAP μήνυμα προκειμένου να γίνει με επιτυχία η κλήση. Επίσης, πληροφορίες σχετικές με την τοποθεσία της υπηρεσίας είναι διαθέσιμες ώστε ο χρήστης να έχει το ακριβές URL στο οποίο θα γίνει η κλήση. Είναι χρέος της περιγραφής αυτής να δώσει όλες τις απαραίτητες πληροφορίες στον πελάτη ώστε να καλέσει την υπηρεσία.
- Τη μη-λειτουργική περιγραφή – Η λειτουργική περιγραφή είναι πολύ σημαντική, αφού χωρίς αυτή δεν είναι δυνατόν να επιτευχθεί η κλήση της υπηρεσίας. Η μη-λειτουργική περιγραφή όμως είναι εξίσου σημαντική. Όπως υποδηλώνει και το όνομά της, προσφέρει πληροφορίες που σχετίζονται με μη λειτουργικές απαιτήσεις της υπηρεσίας. Τέτοιες πληροφορίες μπορεί να περιγράφουν το λόγο που κάποιος πελάτης να επιθυμεί να καλέσει την υπηρεσία. Επίσης, μπορεί να περιγράφουν τον παροχέα της υπηρεσίας με αρκετή λεπτομέρεια. Μπορεί να προσφέρουν προσωπικά του στοιχεία και τηλέφωνα επικοινωνίας. Τέλος, πληροφορίες για την ασφάλεια γύρω από την υπηρεσία και την πολιτική του παροχέα μπορεί να είναι διαθέσιμες μέσω της μη λειτουργικής περιγραφής της υπηρεσίας.

Η ευρέως χρησιμοποιούμενη και καθιερωμένη γλώσσα περιγραφής υπηρεσιών διαδικτύου είναι η WSDL (Web Services Description Language). Είναι μια γλώσσα βασισμένη στην XML και περιγράφει τρεις σημαντικές ιδιότητες της υπηρεσίας:

- Τι κάνει η υπηρεσία – Εδώ περιγράφονται οι μέθοδοι της υπηρεσίας, οι παράμετροί της και τα αποτελέσματα που επιστρέφει.
- Πώς γίνεται η πρόσβαση στην υπηρεσία – Εδώ παρέχονται όλες οι απαραίτητες πληροφορίες για τη σύνταξη των SOAP μηνυμάτων που θα ανταλλαγούν, καθώς και τα πρωτόκολλα που υποστηρίζονται από τον παροχέα.
- Που βρίσκεται η υπηρεσία – Εδώ παρέχονται πληροφορίες για τη δικτυακή διεύθυνση της υπηρεσίας. Συνήθως είναι ένα URL.

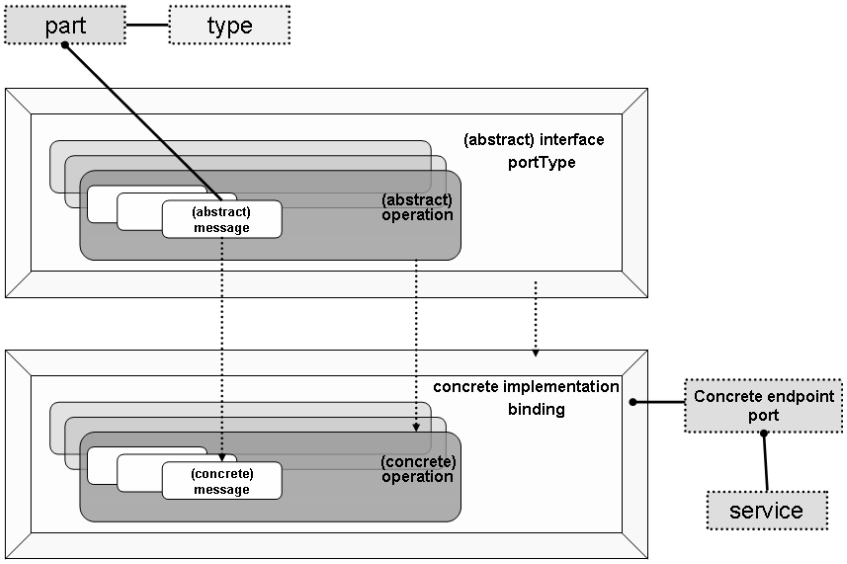
Όπως κάθε καλά ορισμένη γλώσσα βασισμένη σε XML, έτσι και η WSDL ορίζει κάποια βασικά στοιχεία που συναντώνται σε ένα WSDL έγγραφο. Τα κυριότερα από αυτά είναι:

- portType – Είναι ένας αφαιρετικός ορισμός της διασύνδεσης της υπηρεσίας διαδικτύου. Ουσιαστικά, σκοπός του είναι η περιγραφή αυτής της διασύνδεσης. Όπως γίνεται σε έναν κώδικα Java, που η διασύνδεση αποτελείται μόνο από τις υπογραφές των μεθόδων, έτσι κι εδώ ο ορισμός του portType είναι η συλλογή των λειτουργιών που παρέχει η υπηρεσία διαδικτύου. Ένα τέτοιο στοιχείο μπορεί να έχει ένα απλό όνομα, το οποίο πιθανόν να σχετίζεται με τη λειτουργία που προσφέρει η υπηρεσία. Εάν ένα έγγραφο WSDL περιέχει πολλά τέτοια στοιχεία, τότε το κάθε ένα οφείλει να έχει διαφορετικό όνομα.
- Message – Καθορίζει τη μορφή του μηνύματος, καθώς και το σύνολο των παραμέτρων που σχετίζονται με τις μεθόδους της υπηρεσίας. Το στοιχείο αυτό μπορεί να αποσυντεθεί σε πολλά μέρη, που ονομάζονται parts. Κάθε στοιχείο message μέσα σε ένα WSDL έγγραφο πρέπει να έχει ένα μοναδικό όνομα ώστε να ξεχωρίζει από τα υπόλοιπα. Ουσιαστικά, τα στοιχεία αυτά

δεν παρουσιάζουν ιδιαίτερο ενδιαφέρον αφού δεν είναι τίποτε άλλο από μια συλλογή από parts και κάθε part έχει ένα όνομα, που συνήθως αντικατοπτρίζει την πληροφορία που περιέχεται στο part.

- Types – Ορίζει τη συλλογή όλων των τύπων δεδομένων που χρησιμοποιούνται από την υπηρεσία διαδικτύου. Το εξ ορισμού σύστημα των τύπων δεδομένων στην WSDL είναι το XML σχήμα (XML Schema – XSD). Το σχήμα αυτό μπορεί να χρησιμοποιηθεί για να περιγράψει όλους τους τύπους δεδομένων που μπορεί να χρησιμοποιηθούν στα διάφορα μηνύματα που ανταλλάσσονται για την κλήση της υπηρεσίας. Το στοιχείο types είναι ένα μέρος για το WSDL έγγραφο στο οποίο ο χρήστης ορίζει τύπους δεδομένων που θα χρησιμοποιηθούν σε άλλα στοιχεία του εγγράφου.
- Binding – Περιέχει λεπτομέρειες για το πώς τα στοιχεία σε μια αφαιρετική δομή (όπως το portType) μπορούν να μετασχηματισθούν σε μια συμπαγή αναπαράσταση των δεδομένων και πρωτοκόλλων που θα χρησιμοποιηθούν για την επικοινωνία μεταξύ του πελάτη και της υπηρεσίας. Ουσιαστικά είναι το στοιχείο που θα πει στον πελάτη πώς ακριβώς να μορφοποιήσει το μήνυμα που θα στείλει για την κλήση της υπηρεσίας. Κάθε στοιχείο portType μπορεί να έχει ένα ή περισσότερα στοιχεία binding συσχετισμένα με αυτό. Για ένα συγκεκριμένο στοιχείο portType, ένα στοιχείο binding περιγράφει τον τρόπο που πρέπει να γίνει η κλήση των μεθόδων της υπηρεσίας χρησιμοποιώντας κάποιο καθορισμένο πρωτόκολλο, όπως το SOAP, πάνω σε κάποιο επίσης καθορισμένο πρωτόκολλο μεταφοράς, όπως το HTTP. Το όνομα του binding πρέπει να είναι μοναδικό καθώς πολλά τέτοια στοιχεία μπορεί να υπάρχουν σε ένα WSDL έγγραφο.
- Port – Περιγράφει πως ακριβώς εκφράζεται ένα binding σε μια φυσική θύρα του δικτύου. Ο ρόλος της είναι πολύ απλός αφού δεν κάνει τίποτε άλλο από μια απλή αντιστοιχία. Τα στοιχεία αυτά, όπως και τα προηγούμενα, οφείλουν να φέρουν ένα όνομα, μοναδικό μέσα στο WSDL έγγραφο. Συνήθως, τα στοιχεία αυτά υποδεικνύουν το URL στο οποίο πρέπει να σταλούν τα μηνύματα SOAP ώστε να γίνει η κλήση της υπηρεσίας. Τα στοιχεία αυτά δε συναντούνται μόνα τους μέσα στο έγγραφο της περιγραφής της υπηρεσίας. Είναι στοιχεία – παιδιά του συστατικού service.
- Service – Ένα τέτοιο στοιχείο είναι μια συλλογή από ports. Το στοιχείο αυτό μπορεί να έχει ένα μοναδικό όνομα μέσα στο έγγραφο. Παρόλο που ο ρόλος του δεν έχει ιδιαίτερη σημασία, είναι καλό τα στοιχεία port να ομαδοποιούνται κάτω από ένα στοιχείο service. Συνήθως όμως ένα μόνο στοιχείο port συναντάται και έτσι καταλήγουμε στο να έχουμε ένα στοιχείο service, που να περιέχει ένα στοιχείο port.

Το σχήμα που ακολουθεί απεικονίζει τις συσχετίσεις μεταξύ των συστατικών της WSDL.



Σχήμα 11. Το πληροφοριακό μοντέλο της WSDL

Η δομή ενός WSDL εγγράφου ξεκινά με ένα στοιχείο ορισμών, το οποίο καλείται definitions. Το στοιχείο αυτό περιέχει όλα τα υπόλοιπα στοιχεία που μπορεί να συναντηθούν μέσα στο έγγραφο, δηλαδή μπορεί να περιέχει:

- Κανένα ή οσαδήποτε στοιχεία τεκμηρίωσης. Τα στοιχεία αυτά καλούνται documentation elements και χρησιμοποιούνται για να παρέχουν χρήσιμες πληροφορίες κατανοητές από τον άνθρωπο γύρω από την υπηρεσία διαδικτύου.
- Κανένα ή οσαδήποτε στοιχεία εισαγωγών. Τα στοιχεία αυτά καλούνται import elements. Τα στοιχεία δίνουν τη δυνατότητα επαναχρησιμοποίησης ήδη υπαρχόντων WSDL εγγράφων επιτρέποντας σε διάφορα στοιχεία να έχουν αναφορές σε άλλα ήδη υπάρχοντα που βρίσκονται σε διαφορετικό έγγραφο σε διαφορετικό αρχείο.
- Προαιρετικά ένα στοιχείο types.
- Κανένα ή οσαδήποτε στοιχεία message.
- Κανένα ή οσαδήποτε στοιχεία portType. Συνήθως μόνο ένα συναντάται.
- Κανένα ή οσαδήποτε στοιχεία binding. Συνήθως μόνο ένα συναντάται, το οποίο έχει να κάνει με το στοιχείο portType.
- Κανένα ή οσαδήποτε στοιχεία service. Και πάλι συνήθως μόνο ένα είναι αρκετό.

Τέλος, η έκδοση που χρησιμοποιείται ευρέως είναι η 1.1 της WSDL. Σημαντικές βελτιώσεις αναμένονται στην έκδοση 2.0, στην οποία υπάρχει ο ισχυρισμός ότι θα υποστηρίζει κάποιου είδους σημασιολογικής πληροφορίας σε μικρό βαθμό.

Universal Description, Discovery and Integration (UDDI)

Στις αρχές του 2000, και καθώς η ιδέα των υπηρεσιών ιστού άρχισε να κερδίζει έδαφος μέσα στην κοινότητα, έγινε σαφές ότι η καταχώρηση σε καταλόγους των υπηρεσιών ιστού ήταν κάτι απαραίτητο για να μπορέσει να εφαρμοστεί πρακτικά η ιδέα. Επιπλέον, λόγο της γενικότερης στροφής προς τα ανοικτά πρότυπα, οτιδήποτε εκτός από έναν πρότυπο μηχανισμό αλληλεπίδρασης και αναζήτησης δε θα μπορούσε να γίνει αποδεκτό. Ένα τέτοιο πρότυπο καταλόγου, θα έπρεπε να υποστηριχθεί και από τις περισσότερες, αν όχι από όλες τις μεγάλες εταιρίες παροχής λογισμικού, και να υιοθετηθεί από τις διάφορες βιομηχανίες. Έτσι η πρωτοβουλία του UDDI, που ήταν το αποτέλεσμα πολλών μηνών συνεργασίας μεταξύ αντιπροσώπων από τις Ariba, IBM, και Microsoft, που ξεκίνησε την άνοιξη

του 2000, γεννήθηκε και ανακοινώθηκε επίσημα στις 6 Σεπτεμβρίου 2000. Η υποστήριξη για το UDDI επεκτάθηκε και πέρα από τις τρεις εταιρίες που συμμετείχαν αρχικά και αυτή τη στιγμή, το έργο UDDI περιλαμβάνει μια κοινότητα που αριθμεί πάνω από 310 εταιρίες.

Ο σκοπός του UDDI είναι να διευκολύνει την ανακάλυψη υπηρεσιών και κατά το χρόνο σχεδίασης, και δυναμικά κατά το χρόνο εκτέλεσης. Συνεπώς το UDDI λειτουργεί ως ένα κοινό άμεσα συνδεδεμένο κατάλογο (και των αντιστοίχων υπηρεσιών), το οποίο λειτούργησε για πρώτη φορά στις 2 Μαΐου 2001. Ο κατάλογος αυτός, συνήθως αναφέρεται ως κατάλογος επιχειρήσεων UDDI (UDDI Business Registry). Ο κατάλογος αυτός, στην πραγματικότητα αποτελείται από 2 πανομοιότυπους καταλόγους, που διατηρούνται αυτή τη στιγμή από δύο εταιρίες (IBM και Microsoft), οι οποίοι και αποκαλούνται οι διαχειριστές του UDDI.

Για να γίνει κάποιος διαχειριστής ενός τέτοιου καταλόγου, πρέπει να ακολουθήσει αυστηρές συμφωνίες για την αντιγραφή των δεδομένων, το απόρρητο των δεδομένων και τις διάφορες πολιτικές. Από την οπτική μιας καταχωρημένης επιχείρησης αλλά και από την οπτική του απλού χρήστη, η απαίτηση είναι να μην υπάρχει διαφορά ανάμεσα στα διαφορετικά αντίγραφα του καταλόγου. Οι επιχειρήσεις μπορούν να κάνουν εγγραφή σε οποιονδήποτε διαχειριστή UDDI, και τα δεδομένα τους θα αντιγραφούν σε όλα τα άλλα αντίγραφα του καταλόγου στους άλλους διαχειριστές. Επομένως, οι χρήστες μπορούν να αναζητήσουν στον κατάλογο οποιουδήποτε διαχειριστή για να βρουν τις επιχειρήσεις που αναζητούν, άσχετα με το διαχειριστή που έχει επιλέξει η συγκεκριμένη επιχείρηση για να εγγραφεί. Υπάρχουν όμως κάποια λεπτά θέματα που έχουν σχέση με την απόφαση σχετικά με το ποιον διαχειριστή να χρησιμοποιήσει μια επιχείρηση για να εγγραφεί. Για παράδειγμα, κάποιοι διαχειριστές μπορεί να ζητούν επιπρόσθετες προαιρετικές πληροφορίες που δεν απαιτούνται από το UDDI. Αυτή η πληροφορία δεν αντιγράφεται στα αντίγραφα του καταλόγου στους άλλους διαχειριστές. Επιπλέον, αυτό που πρέπει να σημειωθεί είναι ότι από τη στιγμή που μια επιχείρηση επιλέξει κάποιο διαχειριστή για να εγγραφεί, η οποιαδήποτε ενημέρωση και τροποποίηση των στοιχείων θα πρέπει να γίνει από τον ίδιο διαχειριστή. Αυτό πρέπει να γίνεται έτσι δεδομένου ότι ο κάθε διαχειριστής ακολουθεί διαφορετικές πολιτικές πιστοποίησης και ασφάλειας, και επομένως η πληροφορία πιστοποίησης δεν είναι εύκολο να αντιγραφεί στους άλλους διαχειριστές.

Το UDDI είναι κάτι παραπάνω από ένα κατάλογο επιχειρήσεων και υπηρεσιών. Ορίζει ένα σύνολο από δομές δεδομένων και προδιαγραφές API, για την προγραμματική αναζήτηση και καταχώρηση εταιριών, υπηρεσιών, δεσμών, και ειδών υπηρεσιών. Στα τυπικά σενάρια υπηρεσιών διαδικτύου , οι παροχείς υπηρεσιών διαδικτύου θα θελήσουν να δημοσιεύσουν τις περιγραφές των εταιριών και των υπηρεσιών τους σε ένα κατάλογο, και όσοι αναζητούν υπηρεσίες, είτε κατά τη σχεδίαση, είτε κατά την εκτέλεση, θα θελήσουν να αναζητήσουν στον κατάλογο περιγραφές υπηρεσιών. Οι προδιαγραφές API του UDDI παρέχουν γι’ αυτό το λόγο ένα σύνολο από προγραμματιστικές διεπαφές (API) δημοσίευσης για την καταχώρηση υπηρεσιών, και αντίστοιχες διεπαφές ερωτήσεων API για την εύρεση υπηρεσιών.

Επιπλέον από την παροχή μιας προγραμματιστικής διεπαφής, οι διαχειριστές του καταλόγου UDDI, παρέχουν ένα σύστημα αλληλεπίδρασης με το χρήστη, βασισμένο στο διαδίκτυο, για την καταχώρηση, διαχείριση, ανεύρεση εταιριών και υπηρεσιών μέσα στον κατάλογο.

Το μοντέλο χρήσης του UDDI

Σε αυτήν την ενότητα θα εξεταστούν αρχικά οι τυπικές απαιτήσεις για ένα κατάλογο μεταδεδομένων, ορίζοντας κάποια τυπικά στοιχεία δεδομένων και

κάποιες λειτουργίες. Τέλος, θα γίνει μια γενική περιγραφή των δομών δεδομένων και των API του UDDI, συσχετίζοντάς τα με τις αρχικές απαιτήσεις.

3.6.1.1. Οι απαιτήσεις του καταλόγου UDDI

Οι κατάλογοι γενικά, είτε είναι φτιαγμένα για στοιχεία λογισμικού, για υπηρεσίες, είτε για κάποιο άλλο σκοπό, έχουν κάποιες βασικές απαιτήσεις:

- Ένα σύνολο από προδιαγραφές δομών δεδομένων για τα μεταδεδομένα που θα αποθηκευθούν στον κατάλογο.
- Ένα σύνολο από προδιαγραφές λειτουργιών Δημιουργίας, Ανάγνωσης, Ενημέρωσης, Διαγραφής (CRUD – Create, Read, Update, Delete) για την αποθήκευση, διαγραφή, και ερώτηση των δεδομένων του καταλόγου.

Οι συνήθεις απαιτήσεις για τα μεταδεδομένα είναι:

- Ιδιοκτησία και περιεχόμενο, δηλαδή ότι κάποιος πρέπει να είναι ο ιδιοκτήτης των δεδομένων που δημοσιεύει, καθώς και όλων των δεδομένων που περιέχονται σε αυτά, εκτός εάν δηλωθεί διαφορετικά.
- Κατηγοριοποίηση, δηλαδή τα δεδομένα μπορούν να είναι ταξινομημένα σε μια ή περισσότερες κατηγορίες, για να διευκολύνουν την οργάνωση και την αναζήτηση.

Οι συνήθεις απαιτήσεις για τις λειτουργίες του καταλόγου είναι:

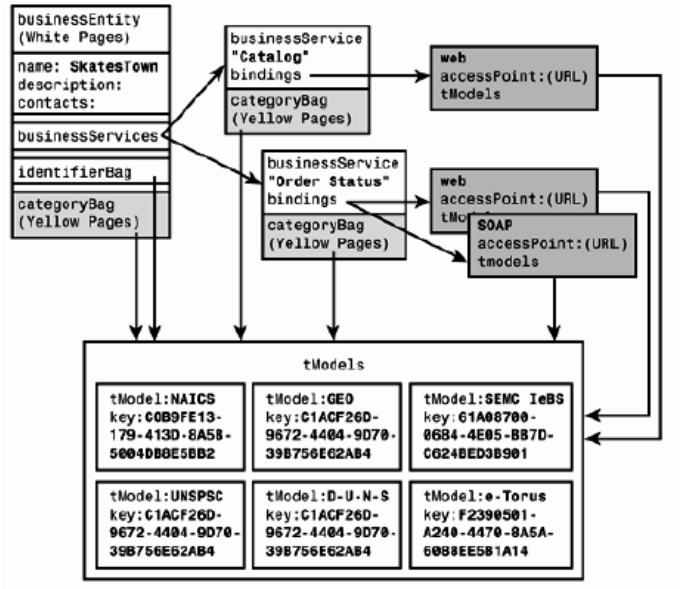
- Η πιστοποίηση για διαδικασίες που αλλάζουν δεδομένα και για δημόσιους καταλόγους.
- Η ελεύθερη πρόσβαση για λειτουργίες ερώτησης και ανάγνωσης.

Το τμήμα του καταλόγου UDDI, διαθέτει ένα μοντέλο πληροφορίας που αντιπροσωπεύει τις προδιαγραφές των δομών δεδομένων, και μια προγραμματιστική διεπαφή για τις προδιαγραφές των λειτουργιών. Το μοντέλο πληροφοριών του UDDI σχεδιάστηκε με τέτοιο τρόπο ώστε να είναι ευέλικτο και επεκτάσιμο, επιτρέποντας έτσι και τη χρήση οποιουδήποτε μοντέλου περιγραφής υπηρεσιών που θα επιθυμούσε να ορίσει κάποιος χρήστης.

Επειδή οι προδιαγραφές του UDDI είναι μια διαδικασία συνεχούς βελτίωσης και επέκτασης, στη συνέχεια θα περιγραφούν οι προδιαγραφές της έκδοσης 1.0 του UDDI. Οι μελλοντικές εκδόσεις το μόνο που θα κάνουν είναι να προσθέσουν περαιτέρω λειτουργικότητες πάνω στην τρέχουσα έκδοση, και να αντιμετωπίσουν προβλήματα που ανακύπτουν σε θέματα αντιγραφής, εκδόσεων και ασφάλειας.

3.6.1.2 Οι δομές δεδομένων του UDDI – Γενική περιγραφή

Οι προδιαγραφές της έκδοσης 1.0 του UDDI, επιτρέπουν σε οντότητες όπως εταιρίες και οργανισμοί να καταχωρούν δημόσια πληροφορία για αυτούς και πληροφορίες για τις υπηρεσίες που παρέχουν. Επιπλέον επιτρέπει σε οντότητες όπως επιχειρήσεις, ομάδες προτύπων, και ομάδες βιομηχανιών, να καταχωρούν πληροφορίες για τα είδη υπηρεσιών που έχουν ορίσει, περιγραφές προτύπων και εννοιών, και να αναφέρονται σε αυτά με αναγνωριστικά ονόματα που τους έχουν ορίσει. Κατά συνέπεια, το UDDI φαίνεται σαν να προσφέρει δυο διαφορετικούς καταλόγους: έναν επαγγελματικό, και ένα κατάλογο τύπου αναφοράς. Σε απόλυτη αντιστοιχία, το UDDI έχει στον πυρήνα του δυο βασικές ριζικές δομές δεδομένων, τις "*businessEntity"* και "*tModel*".



Σχήμα 2. Μοντέλο πληροφορίας του UDDI

3.6.1.3 Η δομή δεδομένων businessEntity

Η πληροφορία της οντότητας της επιχείρησης διαχωρίζεται θεμελιωδώς σε μέρη που είναι γνωστά με τους όρους Λευκές, Κίτρινες, και Πράσινες σελίδες:

- Οι **Λευκές σελίδες** περιέχουν γενικές πληροφορίες επικοινωνίας με την οντότητα.
- Οι **Κίτρινες σελίδες** περιέχουν πληροφορίες ταξινόμησης για τους τύπους και την τοποθεσία των υπηρεσιών που προσφέρει η οντότητα.
- ΟΙ **Πράσινες σελίδες** περιέχουν πληροφορίες για τις τον τρόπο με τον οποίο μπορεί κάποιος να έχει πρόσβαση στις υπηρεσίες που προσφέρονται.

Είναι απαραίτητο να σημειωθεί ότι αυτό είναι απλά ένα θεμελιώδες μοντέλο, και πως το UDDI δεν καθορίζει κάποιο συγκεκριμένο μοντέλο αποθήκευσης. Μια καταχώρηση μιας οντότητας στις λευκές σελίδες θα περιέχει και πληροφορίες ταξινόμησης, που θα δείχνουν τη θέση της οντότητας αυτής στις κίτρινες σελίδες, και επίσης, μια καταχώρηση μιας υπηρεσίας κάποιας οντότητας θα περιέχει πληροφορίες υλοποίησης που θα δείχνουν τη θέση της οντότητας αυτής μέσα στις πράσινες σελίδες. Αυτό αντανακλάται στις λεπτομέρειες της δομής *"businessEntity"*. Το στοιχείο "*businessEntity*", πέρα από κάποιες πληροφορίες επικοινωνίας, αναγνώρισης και ταξινόμησης, περιέχει ένα *"businessServices"* στοιχείο, που είναι ένα πεδίο για στοιχεία τύπου "*businessService*". Ένα στοιχείο "*businessService"* με τη σειρά του, περιέχει ένα στοιχείο "*bindingTemplates*", το οποίο είναι ένα πεδίο για στοιχεία τύπου "*bindingTemplate"*. Τέλος, οι καταχωρήσεις "*bindingTemplate"* δείχνουν σε καταχωρήσεις "*tModel*". Οι συσχετίσεις που περιγράφονται παραπάνω, απεικονίζονται στο .

3.6.1.4 Η δομή δεδομένων tModel

Η άλλη σημαντική δομή του μοντέλου πληροφορίας του UDDI, είναι το στοιχείο "*tModel*". Οι καταχωρήσεις στις πράσινες σελίδες, είναι στην ουσία εξειδικευμένες πληροφορίες κάθε επιχείρησης που συνδέουν πληροφορίες για τύπους υπηρεσιών και άλλες έννοιες που ορίζονται αλλού. Αυτές οι επαναχρησιμοποιήσιμες έννοιες αποκαλούνται μοντέλα τεχνολογίας, και η δομή δεδομένων του UDDI που αντιστοιχεί είναι το στοιχείο "*tModel*". Αυτό το στοιχείο δίνει τη δυνατότητα σε ομάδες βιομηχανιών, σε σώματα προτύπων αλλά και σε μεμονωμένες επιχειρήσεις να μπορούν να καθορίζουν επαναχρησιμοποιήσιμους εννοιολογικούς ορισμούς

τύπων υπηρεσιών, οι οποίοι μπορούν να χρησιμοποιηθούν και να συνδυαστούν, σχηματίζοντας μια υπογραφή για μια υπηρεσία.

Στο σχήμα αναγνώρισης του UDDI, σε κάθε στιγμιότυπο των τεσσάρων τύπων εγγραφών εκχωρείται ένα μοναδικό αναγνωριστικό από τον κόμβο του UDDI κατά την αρχική εγγραφή. Επιπρόσθετα, και ανάλογα με το σχήμα περιεχομένου του UDDI, κάθε στιγμιότυπο μιας περιεχόμενης δομής κρατάει μια αναφορά κλειδί προς την περιεχόμενη οντότητα, σε μια αυστηρή σχέση γονέα – τέκνου. Αυτά τα αναγνωριστικά είναι "UUIDs" (*Universal Unique IDentifiers*), που ακολουθούν τις συμβάσεις του OSF για κατανεμημένο υπολογιστικό περιβάλλον (*Distributed Computing Environment*), και ακολουθούν το σχήμα 8-4-4-4-12 (για παράδειγμα F775A3A6-FF3E-4B85-B1DE-F0F99D0E2C6D).

Θεωρήθηκε πολύ σημαντικό για την επεκτασιμότητα του μοντέλου, οι ιδιωτικοί κατάλογοι UDDI να έχουν την ικανότητα να επεκτείνουν τη δομή "*businessEntity*", έτσι ώστε να μπορεί να διατηρεί δεδομένα που χρειάζονται για τους δικούς τους σκοπούς. Γι' αυτήν την περίπτωση δημιουργήθηκε η οντότητα "*businessEntityExt*". Οι διαχειριστές των καταλόγων UDDI πάντως δεν θα παρέχουν τέτοιες επεκτάσεις στα δεδομένα των επιχειρήσεων που διατηρούν στις δημόσια προσβάσιμες βάσεις τους.

Προγραμματιστική διεπαφή του UDDI – Γενική περιγραφή

Εκτός από τον ορισμό των δομών δεδομένων που θα αποθηκεύονται στον κατάλογο, οι προδιαγραφές του UDDI παρέχουν και δύο βασικούς τύπους προγραμματιστικών διεπαφών: την προγραμματιστική διεπαφή δημοσίευσης και την προγραμματιστική διεπαφή ερώτησης. Αυτές οι δύο διεπαφές ορίζουν ένα σύνολο από 20 μηνύματα SOAP για το πρωτόκολλο HTTP ή για το HTTPS, τα οποία πρέπει να είναι κατανοητά από οποιοδήποτε κατάλογο συμβατό με το UDDI πρότυπο. Το SOAP τμήμα του μηνύματος καθώς και οι απαντήσεις είναι δομές της XML που καθορίζονται στις προδιαγραφές του UDDI.

Η προγραμματιστική διεπαφή δημοσίευσης ορίζει είναι ένα πιστοποιημένο σύνολο λειτουργιών που επιτρέπει στους οργανισμούς να δημοσιεύουν πληροφορία για εταιρίες, υπηρεσίες, υλοποιήσεις ή είδη υπηρεσιών στον κατάλογο UDDI. Το UDDI δεν καθορίζει κάποια μέθοδο πιστοποίησης, εκτός από την απαίτηση για ύπαρξη κουπονιού πιστοποίησης (*authorization token*), και τη χρήση του πρωτοκόλλου HTTPS κατά τις διαδικασίες πιστοποίησης. Έτσι ο κάθε διαχειριστής θα πρέπει να έχει το δικό του μηχανισμό πιστοποίησης.

Η αντίστοιχη προγραμματιστική διεπαφή ερωτήσεων, αποτελείται από ένα σύνολο από λειτουργίες οι οποίες δεν απαιτούν πιστοποίηση και επιτρέπουν στους χρήστες να λαμβάνουν τις πληροφορίες που αναζητούν από τον κατάλογο UDDI. Τα ερωτήματα μπορούν να γίνονται με βάση δύο γενικά σχήματα: ένα σχήμα με τη χρήση επόπτη διαδικτύου (*browser*) για την εύρεση γενικών πληροφοριών, το οποίο και συνήθως ακολουθείται από ένα σχήμα αναζήτησης σε βάθος, με βάση το οποίο αναζητούνται συγκεκριμένες λεπτομερείς πληροφορίες.

Το UDDI παρέχει 4 λειτουργίες αποθήκευσης (*save_business*, *save_service*, *save_binding* και *save_tModel*), 4 λειτουργίες διαγραφής (*delete_business*, *delete_service*, *delete_binding* και *delete_tModel*), 4 λειτουργίες αναζήτησης (*find_business*, *find_service*, *find_binding* και *find_tModel*), και 4 λειτουργίες αναζήτησης λεπτομερειών (*get_businessDetail*, *get_serviceDetail*, *get_bindingDetail* και *get_tModelDetail*), για κάθε ένα από τους 4 τύπους δομών (businessEntity, businessService, bindingTemplate και tModel) αντίστοιχα (Πίνακας 1).

	Business	Service	Binding	tModel
Save/Update	save_business	save_service	save_binding	save_tModel
Delete	delete_business	delete_service	delete_binding	delete_tModel
Find	find_business	find_service	find_binding	find_tModel
GetDetail	get_businessDetail	get_serviceDetail	get_bindingDetail	get_tModelDetail

Πίνακας 1: Μηνύματα των προγραμματιστικών διεπαφών που ορίζονται από τις προδιαγραφές του UDDI.

Οι λειτουργίες αποθήκευσης μπορούν να χρησιμοποιηθούν για τη δημιουργία μιας καινούριας εγγραφής αλλά και για τη διόρθωση μιας υπάρχουσας. Κατά τη δημιουργία μιας καινούριας εγγραφής, η αναγνωριστική παράμετρος (κλειδί της εγγραφής) δεν καταχωρείται, αλλά εκχωρείται από το διαχειριστή και επιστρέφεται κατά την απάντηση του διαχειριστή. Για την ενημέρωση μιας εγγραφής όμως χρειάζεται και η εισαγωγή του κλειδιού αυτού. Οι λειτουργίες διαγραφής δέχονται τα κλειδιά που εκχώρησαν οι διαχειριστές σαν παραμέτρους, και διαγράφουν τις αντίστοιχες εγγραφές από τον κατάλογο. Ειδική περίπτωση διαγραφής αποτελεί η λειτουργία "*delete_tModel*". Όταν ένας αυθεντικοποιημένος χρήστης, χρησιμοποιήσει την παραπάνω λειτουργία για την διαγραφή ενός "*tModel",* αυτό δεν διαγράφεται φυσικά από τον κατάλογο, αλλά σημειώνεται ως κρυφό. Με τον τρόπο αυτό, είναι ακόμα διαθέσιμο στον αυθεντικοποιημένο χρήστη, αλλά δεν επιστρέφεται ως αποτέλεσμα από τις λειτουργίες αναζήτησης. Ο χρήστης, μπορεί να επανεισάγει το "*tModel",* χρησιμοποιώντας τη λειτουργία "*save_tModel"* περνώντας ως παράμετρο το κλειδί του αρχικού "*tModel".*

Οι λειτουργίες αναζήτησης δέχονται ένα σύνολο από κριτήρια σαν είσοδο, και το αποτέλεσμά τους συνήθως είναι μια δομή λίστας η οποία περιέχει τις εγγραφές που ικανοποιούν τα κριτήρια που είχαν τεθεί. Οι λειτουργίες αναζήτησης λεπτομερειών δέχονται μια λίστα από κωδικούς αναγνώρισης και επιστρέφουν λεπτομερείς πληροφορίες σχετικά με τις καταχωρήσεις που αντιστοιχούν στους κωδικούς της εισόδου. Η προγραμματιστική διεπαφή, παρέχει και μια επιπρόσθετη λειτουργία αναζήτησης λεπτομερειών, την "*get_businessDetailExt"*, η οποία χρησιμοποιείται για την ανάκτηση εκτεταμένων πληροφοριών για εταιρίες από κόμβους μη διαχειριστών, που διατηρούν αυτές τις πρόσθετες πληροφορίες.

Επιπλέον, επειδή η προγραμματιστική διεπαφή δημοσίευσης είναι ένα σύνολο από πιστοποιημένες λειτουργίες, όπου η αναγνώριση ενός χρήστη βασίζεται στη χρήση κουπονιών, η διεπαφή αυτη παρέχει και δυο λειτουργίες πιστοποίησης, τις "*get_authToken"* και "*discard_authToken"*.

Τέλος, η λειτουργία "*getRegisteredInfo"*, είναι μια λειτουργία αυθεντικοποίησης, η οποία και επιστρέφει όλα τα κλειδιά "*businessEntity"* και "*tModel"*, που έχουν καταχωρηθεί από έναν εξουσιοδοτημένο χρήστη.

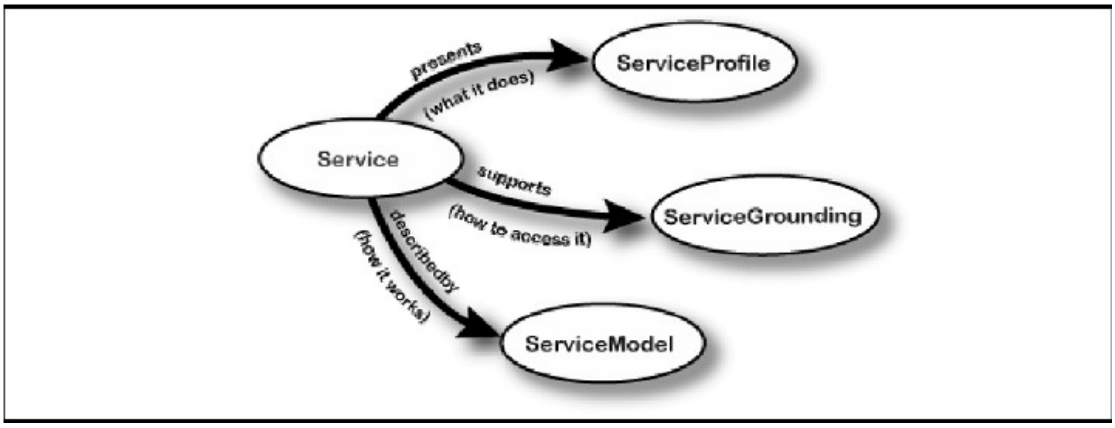
Κλείνοντας την αναφορά στην προγραμματιστική διεπαφή του UDDI, θα πρέπει να τονιστεί, ότι όλες σχεδόν οι λειτουργίες των προγραμματιστικών διεπαφών δημοσίευσης και ερώτησης είναι δυνατόν να πραγματοποιηθούν και από τις ιστοσελίδες των μεμονωμένων διαχειριστών. Η προγραμματιστική διεπαφή όμως, εκτός από του ότι είναι και πολύ πιο πλήρης από το σύστημα αλληλεπίδρασης ενός επόπτη διαδικτύου, παρέχει επίσης τη δυνατότητα στις επιχειρήσεις να εκτελούν όλες τις λειτουργίες προγραμματιστικά, βοηθώντας στην δυναμική (κατά το χρόνο εκτέλεσης) ανεύρεση υπηρεσιών.

Υπηρεσίες Οντολογίας OWL-S

Περιγραφή

Η OWL-S είναι μια owl υπηρεσία άνω οντολογίας που προσφέρει ένα λεξικό το οποίο μπορεί να χρησιμοποιηθεί σε συνδυασμό με την owl για να περιγράφει υπηρεσίες με μια αμφιλεγόμενη, υπολογιστικώς διερμηνευόμενη μορφή. Η OWL-S αναπτύχθηκε με σκοπό να επιτρέπει την ανακάλυψη,κλήση, σύνθεση και αυτόματο έλεγχο των υπηρεσιών διαδικτύου(web services).Η OWL-S θεωρεί την σύνθεση των υπηρεσιών σαν διεργασίες.Υπάρχει μια πολύ καθαρή διαφοροποίηση μεταξυ των ιδιοτήτων των διεργασιών , της δομής τους και της υλοποίησής τους σε OWL-S η οποία περέχει έναν τρόπο να μοντελοποιήσει μια διεργασία ανεξάρτητα από την υλοποίησή της.
Η κεντρική κλάση της υπηρεσίας οντολογίας ειναι η κλάση Service όπως φαίνεται στο σχήμα 1. Το τρέχον URI για αυτήν την οντολογία είναι

http:// www.daml.org/services/owl-s/1.2/Service.owl



Σχήμα 1. OWL-S Υπηρεσία οντολογίας

Το οποίο το συντομογραφούμε σαν &owls στα παραδείγματα που ακολουθούν.
Η κλάση ServiceProfile περιγράφει τι χρειάζεται η υπηρεσία και τι παρέχει στον πελάτη.Η Service και η ServiceProfile κλάσεις συνδέονται από την ιδιότητα presents και την αντίστροφή της ιδιότητα presentedBy.

Η κλάση ServiceModel περογράφει πώς δουλεύει η υπηρεσία και τις συνέπειες της εκτέλεσής της.Η Service και η ServiceModel κλάσεις συνδέονται με την ιδιότητα describedBy και την αντίστροφη ιδιότητά της describes.Μια υπηρεσία μπορεί να περιγραφεί απο το πολύ ένα στιγμιότυπο της κλάσης ServiceModel.

Η κλάση ServiceGrounding περιγράφει πως μια υπηρεσία μπορεί να προσπελαστεί.Η Service και η ServiceGrounding κλάσεις συνδέονται μεταξύ τους με την ιδιότητα supports και την αντίστροφη ιδιότητά τους supportedBy.Ενα στιγμιότυπο της κλάσης ServiceGrounding πρέπει να συνδέεται με ακριβώς μια υπηρεσία.

Service Profile (Τι κάνει)

Η κεντρική κλάση της profile οντολογίας ειναι η ServiceProfile κλάση, που ειναι υπεύθυνη για την περιγραφή των υπηρεσιών των ίδιων. Το τρέχον URI της profile οντολογίας ειναι

http:// www.daml.org/services/owl-s/1.2/Profile.owl

Η OWL-S ορίζει μια υποκλάση της κλάσης ServiceProfile που ονομάζεται Profile,σαν μια πιθανή αναπαράσταση για ενα προφίλ υπηρεσίας.Αυτή η αναπαράσταση κατέχει τρείς ιδιότητες που χρησιμοποιουνται για να περιγράψουν την υπηρεσία στους ανθρώπους παρά στις μηχανές.Αυτές οι ιδιότητες ειναι: η ιδιότητα contactInformation που περιγράφει ποιός παρέχει την υπηρεσία που έχουμε,η ιδιότητα serviceName που χρησιμοποιείται για ην αναγνώριση της διεργασίας και η ιδότητα textDescription που παρέχει μια ελεύθερη περιγραφή κειμένου της υπηρεσίας.
Πέρα απο τις προαναφερθείσες ιδιότητες, η κλάση Profile παρέχει μια περιγραφή των λειτουργιών της υπηρεσίας κάτω απο δύο προοπτικές:μετασχηματισμό πληροφορίας (που αναπαριστάται απο τις εισόδους και τις εξόδους της υπηρεσίας) και αλλαγές της κατάστασης(που αναπαριστώνται απο τα preconditions και τα effects της υπηρεσίας).
Η OWL-S επίσης εισάγει την έννοια της παραμέτρου.Η παράμετρος ειναι ενας τύπος μεταβλητής που χρησιμοποιείται επίσης στο πλαίσιο των γλωσσών κανόνων.Η κλάση Parameter ειναι υπερκλάση της κλάσης Input, που αντιπροσωπεύει τις παραμέτρους εισόδου, όπως επίσης ειναι η κλάση Output που αντιπροσωπεύει τις παραμέτρους εξόδου της ίδιας διεργασίας.Οι κλάσεις Parameter,Input και Output ειναι μέρος της OWL-S οντολογίας διεργασίας(process ontology).Τα preconditions και τα effects μιας διεργασίας απ' την άλλη εκφράζονται με λογικές εκφράσεις.
Για να περιγράψουμε τις παραμέτρους εισόδου και εξόδου και τα preconditions και τα effects μιας υπηρεσίας, η κλάση Profile έχει τις ακόλουθες ιδιότητες που φαίνονται και στο σχήμα 2.

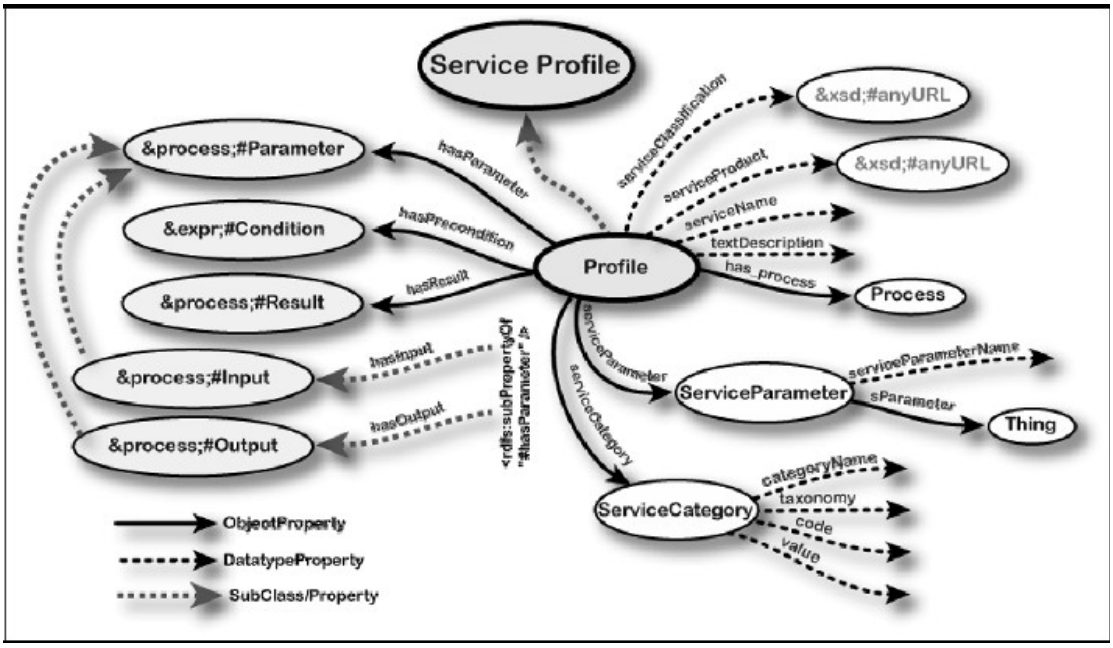
- hasParameter: μια ιδιότητα που το πεδίο ορισμού του ειναι η κλάση Parameter από την OWL-S οντολογία διεργασίας.Αυτή η ιδιότητα δεν χρησιμοποιείται απευθείας, χρησιμεύει σαν υπερκλάση για τις υπόλοιπες τις ιδιότητες.

- hasPrecondition:Μια ιδιότητα της οποίας το πεδίο τιμών ειναι η κλάση Precondition από την OWL-S οντολογία διεργασίας.

- hasResult: Μια ιδιότητα της οποίας το πεδίο τιμών ειναι η κλάση Result από την OWL-S οντολογία διεργασίας.Το αποτέλεσμα της διεργασίας υποδηλώνει τις συνθηκες κάτω απο τις οποίες γεννιούνται οι έξοδοί της(outputs).

- hasInput: μια ιδιότητα που το πεδίο ορισμού του ειναι η κλάση Input,υποκλάση της κλάσης Parameter από την OWL-S οντολογία διεργασίας.Η κλάση Input αντιπροσωπεύει την πληροφορία που απαιτείται απο την διεργασία για την εκτέλεσή της.

- hasOutput: μια ιδιότητα που το πεδίο ορισμού του ειναι η κλάση Output από την OWL-S οντολογία διεργασίας.



Σχήμα 2. OWL-S προφίλ οντολογίας.

Σε μια ιδεατή κατάσταση όλα τα στιγμιότυπα των κλάσεων Input, Output, Precondition και Result δημιουργούνται απο την οντολογία διεργασίας και αναφέρονται απο την κλάση Profile.Γι'αυτό η τιμή για κάθε νέα ιδιότητα ειναι μια παραπομπή σε ενα στιγμιότυπο μιας κλάσης απο την οντολογία διεργασίας.Η κλάση Profile εχει ενα ειδικό σύνολο απο ιδιότητες:

- serviceParameter:μια εκτεταμένη λίστα απο ιδιότητες των οποίων τα στοιχεία ειναι στιγμιότυπα της κλάσης ServiceParameter.Αυτή η κλάση κατέχει τις ιδιότητες serviceParameterName και sParameter(η τελευταία ειναι ενας δείκτης προς την τιμή της παραμέτρου σε κάποια άλλη OWL οντολογία).

- serviceCategory:Η τιμή αυτής της ιδιότητας ειναι μια παραπομπή σε μια υπάρχουσα ταξονομία υπηρεσίας.Αυτή η ιδιότητα χρησιμοποιείται για να περιγράψει κατηγορίες υπηρεσιών.Εχει ενα όνομα(categoryName), εναν δείκτη σε κάποια ταξονομία(taxonomy),μια ή περισσότερες τιμές σε αυτήν την ταξονομία και για κάθε τύπο σχετιζόμενης υπηρεσίας εναν ειδικό κωδικό(code).

Ειναι σημαντικό να σημειώσουμε οτι ένα προφίλ υπηρεσίας μπορεί να δημοσιοποιήσει μόνο ενα μέρος των υπηρεσιών που παρέχει.Παρόλα αυτά εαν δεν δηλωθούν οι λειτουργικότητες δεν θα είναι ανακαλύψιμες.

Service Model(Πώς το κάνει)

Στην OWL-S μια διεργασία ειναι μια λεπτομερής διεργασία που περιγράφεται απο ενα στιγμιότυπο της κλάσης Process, που ειναι υποκλάση του ServiceModel.Μια διεργασία εχει εναν αριθμό από συνθήκες που καθορίζουν τις καταστάσεις κάτω απο τις οποίες μπορεί να εκτελεστεί.Οι διεργασίες μπορεί να προέλθουν απο τον μετασχηματισμό των δεδομένων εισόδου.Η εκτέλεση της διεργασίας με το ίδιο μέτρο μπορεί να μετασχηματίσει τον κόσμο.

Με άλλα λόγια η διεργασία μετατοπίζει τον κόσμο σε μια νέα κατάσταση.Για να περιγράψουμε μια διεργασία σε OWL-S ειναι αναγκαίο να δηλώσουμε:

- παραμέτρους εισόδου
- παραμέρους εξόδου
- preconditions
- effects

που συχνά αναφέρονται ως IOPEs(Input, Output, Preconditions, και Effects).Ειναι πολύ σημαντικό να εγγυηθούμε οτι για οποιαδήποτε δοσμένη υπηρεσία, τα IOPEs που αντιπροσωπεύονται στο μοντέλο υπηρεσίας αντικατοπτρίζονται στο προφίλ υπηρεσίας.
Για να περιγράψουμε την σχέση μεταξύ μιας διεργασίας και κάθε μιας απο τις παραμέτρους της,κάνοντας ετσι δυνατή την δήλωση μετασχηματισμών δεδομένων, η OWL-S παρέχει τις ιδότητες hasInput, hasOutput και hasLocal.Αυτές όλες ειναι υποιδιότητες της ιδιότητας hasParameter.
Μαζί με τις παραμέτρους εισόδου και εξόδου μια διεργασία έχει επίσης preconditions που δηλώνονται με την ιδιότητα hasPrecondition.
Οταν μια διεργασία εκτελείται ,εκτός απο τις εξόδους της, η διεργασία μπορεί να έχει επίσης σαν αποτέλεσμα την αλλαγή της κατάστασης του κόσμου.Αυτές οι αλλαγές λέγονται effects με το λεξιλόγιο της OWL-S.Αυτά τα effects ειναι διαφορετικά από τα outputs γιατί αντι να παράγουν δεδομένα αλλάζουν την κατάσταση του κόσμου(π.χ. αφού λάβουμε μια πληρωμή το υπόλοιπό μας διαφοροποιείται).
Για να εκφράσουμε τα preconditions και effects μιας διεργασίας, η OWL-S παρέχει μια ειδική γλώσσα έκφρασης, που παριστάνεται απο την κλάση Expression και όλες τις υποκλάσεις της των οποίων το URI ειναι

http://www.daml.org/services/owl-s/1.2/generic/Expression.owl.

Η κλάση Expression κατέχει δύο ιδιότητες, expressionLanguage και expressionBody, και οι δυό με πληθικότητα(cardinality) ιση με ενα,δηλαδή δέχονται μόνο μιά μονή τιμή για την γλώσσα και το σώμα. Η πιό κοινή γλώσσα που χρησιμοποιείται σαν τιμή για την ιδιότητα expressionLanguage ειναι η SWRL.
Στην OWL-S τα outputs και τα effects δεν συνδέονται απευθείας με τις υποψήφιες διεργασίες γιατί μπορεί να διαφοροποιούνται ανάλογα με το περιβάλλον της υπηρεσίας που εφαρμόζεται.Για να διατηρηθεί η συνέπεια, μια κλάση με όνομα Result δημιουργήθηκε.Αυτή η κλάση διατηρεί μια απευθείας σχέση με τα outputs και τα effects μέσω μιας ιδιότητας με ονομα hasResult.
Η κλάση Result εχει πέντε ιδιότητες.Η ιδιότητα hasResultVar επιτρέπει για την δήλωση μιας μεταβλητής του τύπου ResultVar,που ειναι υποκλάση της Parameter.Αυτή η ιδότητα μπορεί να χρησιμοποιηθεί σε συνδυασμό με την ιδιότητα inCondition που ορίζει τις συνθήκες κάτω από τις οποίες το επιθυμητό αποτέλεσμα θα έρθει.Η ιδιότητα withOutput δηλώνει ις εξόδους της διεργασίας όταν το επιθυμητό αποτέλεσμα εχει επιτευχθεί.Παρομοίως η ιδιότητα hasEffect δηλώνει τις τιμές εκτέλεσης της διεργασίας πάνω απο το επιθυμητό αποτέλεσμα.Τέλος η ιδιότητα resultFrom παρέχει ενα ιδεατό XML πρότυπο(template) για τις εξόδους που πρέπει να αποσταλλούν στον πελάτη της υπηρεσίας (Service client).
Πάνω από ενα αποτέλεσμα μπορεί να δηλωθεί για μια OWL-S διεργασία.Κάποιος πρέπει να ειναι προσεχτικός να κάνει τις συνθήκες κάτω από τις οποίες κάθε αποτέλεσμα(result) μπορεί να επιτευχθεί με αμοιβαία αποκλειόμενο τρόπο για την αποφυγή διφορούμενης και αβέβαιας εκτέλεσης.

Μια OWL-S διεργασία περιλαμβάνει πάντοτε δυό πράκτορες(agents), συγκεκριμένα ενα στιγμιότυπο της κλάσης TheClient και ενα στιγμιότυπο της κλάσης TheServer.
Η κλάση Process αντιστοιχεί στην ένωση τριών OWL-S κλάσεων όπως ορίζεται από το επόμενο OWL κομμάτι(fragment),

```
<owl:Class rdf:ID="Process">
<rdfs:comment> The most general class of processes
</rdfs:comment>
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#AtomicProcess"/>
<owl:Class rdf:about="#SimpleProcess"/>
<owl:Class rdf:about="#CompositeProcess"/>
</owl:unionOf>
</owl:Class>
```

Ακολουθεί σύντομη επεξήγηση των τριλων OWL-S κλάσεων.

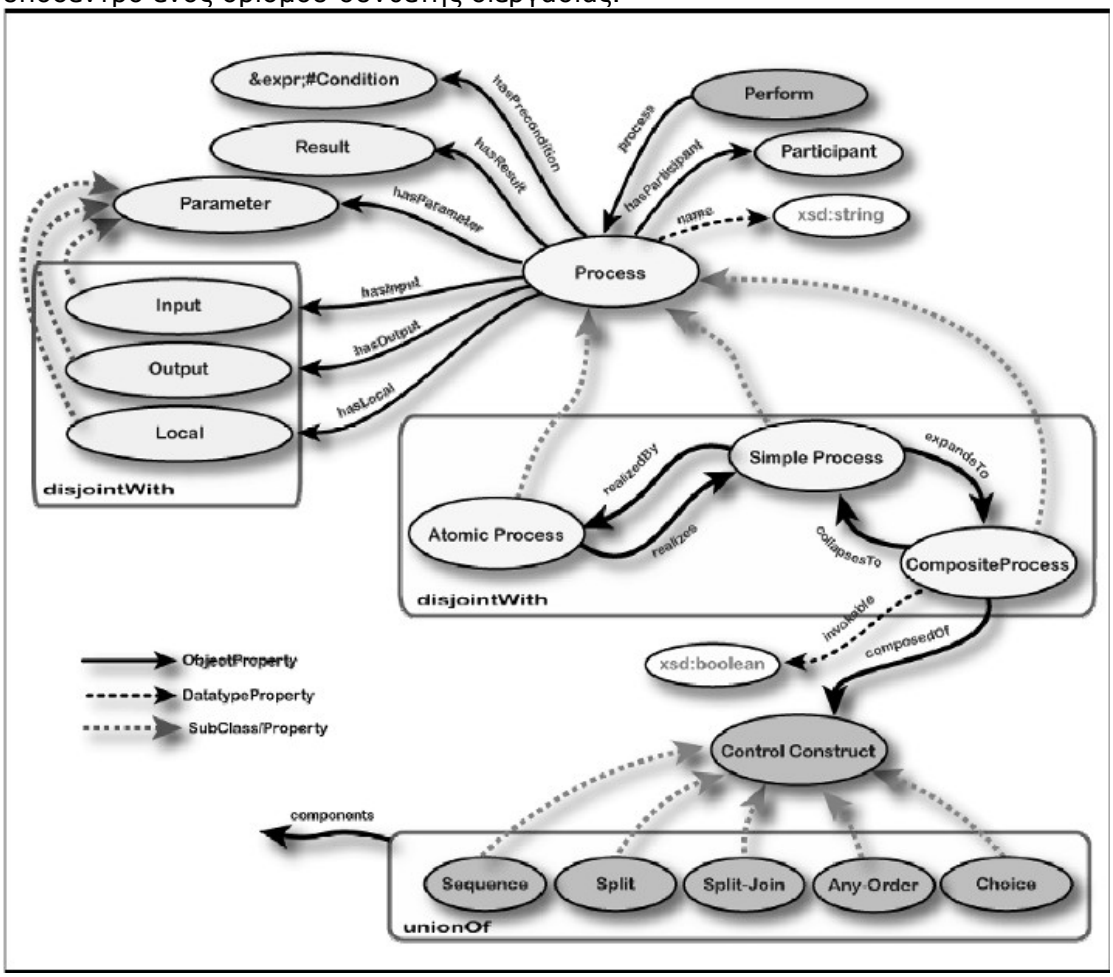
- AtomicProcess: Διεργασίες που μπορούν να κληθούν απευθείας,δεν κατέχουν υποδιεργασίες, και εκτελούν την υπηρεσία σε ενα βήμα.Για κάθε ατομική διεργασία πρέπει να υπάρχει ενα συνδεόμενο grounding(πως να την προσπελάσει).

- SimpleProcess: Διεργασίες που δεν μπορούν να κληθούν απευθείας αλλα όπως και οι ατομικές διεργασίες θεωρούνται διεργασίες ενός βήματος.Οι απλές διεργασίες χρησιμοποιούνται σαν αφαιρετικά στοιχεία.Μπορεί να χρησιμοποιηθούν για την παροχή μιας βλέψης πάνω από ατομική διεργασία ή να παρέχουν μια απλουστευμένη αναπαράσταση μιας σύνθετης διεργασίας.Εαν μια απλή διεργασία συνδεθεί με μια σύνθετη διεργασία αυτή η σχέση αναπαριστάται με την ιδιότητα realizedBy.Οι απλές διεργασίες μπορεί να έχουν τιμές εισόδου και εξόδου συνδεόμενες με αυτές όπως συμβαίνει και μετις υπόλοιπες τις διεργασίες.

- CompositeProcess:Διεργασίες που μπορούν να αναλυθούν σε άλλες διεργασίες.Κατασκευαστές ελέγχου (ControlConstruct) χρησιμοποιούνται για να περιγράψουν σύνθετες διεργασίες διαμέσου της composedOf ιδιότητας.Κάθε ControlConstruct με τη σειρά του εχει μια ιδιότητα components που δηλώνει την σύνθεση της διεργασίας.

Μια σύνθετη διεργασία μπορεί να ιδωθεί σαν ενα δέντρο του οποίου οι μή τερματικοί κόμβοι αναφέρονται(are labeled) με δομές ελέγχου, κάθε κόμβος έχοντας τον απόγονό του με ετικέττα χρησιμοποιώντας την ιδιότητα component.Τα φύλλα αυτού του δέντρου αναπαριστάνουν άλλες κλήσεις διεργασίας, που υποδηλώνονται σαν στιγμιότυπα της κλάσης Perform.Η ιδιότητα process της κλάσης Perform υποδηλώνει το όνομα της διεργασίας που πρόκειται να κληθεί. Η οντολογία διεργασίας φαίνεται στο σχήμα 3.
Η OWL-S παρέχει τις ακόλουθες δομές ελέγχου. Sequence, Split , Split-Join, Any-Order, Choice, If-Then-Else, Iterate, Repeat-While, Repeat-Until,και AsProcess. Ενα αντικείμενο AsProcess ειναι και μια δομή ελέγχου και μια διεργασία.Αν θεωρηθεί σαν ενας τύπος δομής ελέγχου το construct AsProcess δεν εχει συστατικά(components).Παρόλα αυτα εαν είναι μια CompositeProcess, μπορεί να έχει ενα σώμα (η τιμή της composedOf ιδιότητας).Επιτρέπει σχετικά outputs,

τοπικές μεταβλητές(local variables),effects και ακόμα preconditions με ενα τυχαίο υποδέντρο ενος ορισμού σύνθετης διεργασίας.



Σχήμα 3 :Οντολογία διεργασίας

Η σημασιολογία των δομών ελέγχου ειναι ίδια με τις αντίστοιχες δομές της Business Process Execution Language (BPEL) ή της Business Process Modeling Languages(BPML).

Service Grounding (Πως γίνεται η προσπέλαση)

Πως να προσπελάσει κάποιος μια υπηρεσία OWL-S περιγράφεται απο την κλάση ServiceGrounding.Αυτή η κλάση περιγράφει την μορφή των μηνυμάτων που πρέπει να ανταλλαχθούν,τα serialization πρωτόκολλα, τα πρωτόκολλα μεταφοράς και διευθυνσιοδότησης.Ενα OWL-S grounding μπορει να ιδωθεί σαν μια αντιστοίχηση απο ενα αφαιρετικό specification προς ενα specification συγκεκριμένου στοιχείου υπηρεσίας(concrete service element specification).Ανάμεσα στα έγγραφα που σχηματίζουν μια OWL-S specification το specification της service grounding ειναι το μόνο που εχει σχέση με συγκεκριμένες λεπτομέρειες υλοποίησης.Οι αναπαραστάσεις και του προφίλ υπηρεσίας και του μοντέλου υπηρεσίας αποτυπώνονται σε ενα πιό αφαιρετικό επίπεδο.
Η OWL-S δεν περιλαμβάνει μια αφαιρετική αναπαράσταση για μηνύματα.Στην πράξη, τα περιεχόμενα των μηνυμάτων έμμεσα υποδηλώνονται από τις ιδιότητες Input και Output της ατομικής διεργασίας.Συγκεκριμένα όμως μηνύματα απο την άλλη υποδηλώνονται απο το grounding.Τα μηνύματα υποδηλώνονται χρησιμοποιώντας την WSDL,όπως έχει περιγραφεί παραπάνω.Ο σκοπός του grounding ειναι στο να αποφασίσει πως οι είσοδοι και οι έξοδοι μιας ατομικής

διεργασίας μπορούν να περιγραφούν με συγκεκριμένα μηνύματα που ανταλλάσονται μεταξύ των διεργασιών.

Υλοποίηση περίπτωσης χρήσης(use case) σεναρίου έκδοσης άδειας οδήγησης.

Περίπτωση χρήσης(use case)

Η περίπτωση χρήσης παρακάτω περιγράφει την δημόσια υπηρεσία της έκδοσης μιας άδειας οδήγησης στην Ελλάδα.Η δημόσια υπηρεσία(Service Provider) που παρέχει την υπηρεσία ειναι το υπουργείο Μεταφορών και Επικοινωνιών.Η υπηρεσία παρέχεται σε επίπεδο νομαρχίας.
Προϋποθέσεις(preconditions) ή υποθέσεις(assumptions)
Ο υποψήφιος για άδεια οδήγησης πρέπει να τηρεί τις παρακάτω προϋποθέσεις:

1. Πρέπει να έχει εντοπιότητα στην Ελλάδα
2. Δεν έχει άδεια οδήγησης της ίδιας κατηγορίας που να έχει εκδοθεί στην Ελλάδα ή σε άλλη χώρα της ευρωπαικής ένωσης(συμπεριλαμβανομένων των Νορβηγίας, Ισλανδίας και Λιχτενσταιν)
3. Πρέπει να ειναι 18 ετών και πάνω
4. Πρέπει να τηρεί τις ελάχιστες απαιτήσεις για διανοητική και φυσική ικανότητα που ταιριάζουν για αυτήν την κατηγορία άδειας.
5. Πρέπει να έχει περάσει με επιτυχία μια θεωρητική εξέταση και μια εξέταση οδήγησης μετά από υποχρεωτική εκπαίδευση σε θεωρία και πρακτική.

Τα αναγκαία δικαιολογητικά είναι:

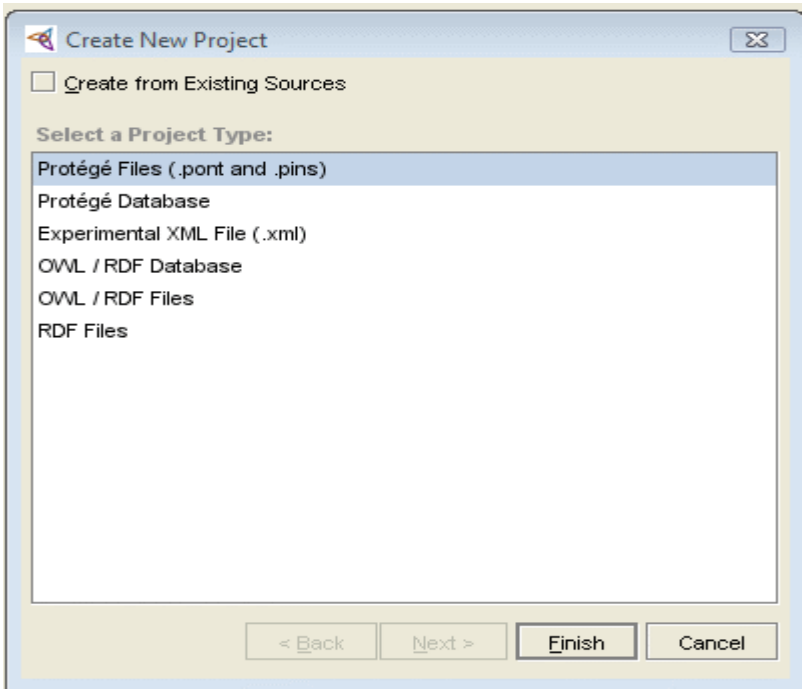
1. Μια νόμιμα υπογεγραμμένη αίτηση
2. Φωτοαντίγραφο της αστυνομικής ταυτότητας ή του διαβατηρίου ή οποιουδήποτε άλλου εγγράφου απο το οποίο μπορεί να ελεγχθεί η ηλικία του υποψηφίου
3. Φωτοαντίγραφο της άδειας παραμονής ή άλλου ισοδύναμου εγγράφου εαν ο ενδιαφερόμενος δεν είναι Ελληνας πολίτης.Αυτη η άδεια πρέπει να έχει εκδοθεί 185 ημέρες πριν την αίτηση ή 85 ημέρες εαν ο ενδιαφερόμενος ειναι απο χώρα της ευρωπαικής ενωσης(συμπεριλαμβανομένων των Νορβηγίας, Ισλανδίας και Λιχτενσταιν).Η άδεια παραμονής πρέπει να είναι νόμιμη (να ισχύει) εως την ημέρα παραλαβής της άδειας οδήγησης.
4. Δύο πρόσφατες Φωτογραφίες
5. Δύο πιστοποιητικά υγείας.Ενα απο εναν γιατρό γενικό παθολόγο και ενα απο εναν οφθαλμίατρο.Για να πάρει κάποιος τα πιστοποιητικά υγείας δύο παράβολα των 32 ευρώ χρειάζονται και τέσσερις φωτογραφίες.Οι γιατροί πρέπει να δουλεύουν για την τοπική υπηρεσία μεταφορών και επικοινωνιών.
6. Ενα παράβολο των έξι ευρώ
7. Ενα παράβολο των 18 ευρώ ή 27.02 ευρώ εαν πρόκειται για επέκταση άδειας οδήγησης σε άλλη κατηγορία.
8. Φωτοαντίγραφο της τρέχουσας άδειας οδήγησης σε περίπτωση επέκτασης της άδειας οδήγησης σε άλλη κατηγορία.

Διαδικασία

1. Η αίτηση υποβάλλεται στην τοπική υπηρεσία μεταφορών και επικοινωνιών(του υποψηφίου ο τόπος διαμονής)
2. Σε περίπτωση επέκτασης της τρέχουσας άδειας οδήγησης σε άλλη κατηγορία ένα νέο έγγραφο εκδίδεται ενω η προηγούμενη άδεια οδήγησης ακυρώνεται.
3. Εαν ο υποψήφιος εχει ήδη μια άδεια οδήγησης που εκδόθηκε στην Ελλάδα η σε άλλη χώρα της ευρωπαικής ένωσης(συμπεριλαμβανομένων των Νορβηγίας, Ισλανδίας και Λιχτενσταιν) αυτή πρέπει να κατατεθεί.

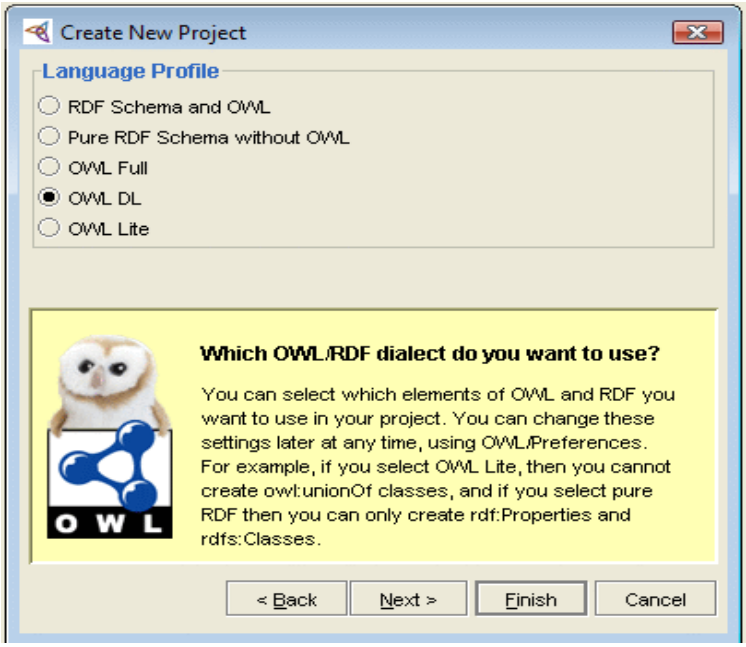
Ακολουθεί η υλοποίηση χρησιμοποιώντας το εργαλείο Protégé για την μοντελοποίηση των κλάσεων και των ιδιοτήτων του παραπάνω σεναρίου ετσι ώστε να ειναι δυνατή η αποτύπωση της περίπτωσης χρήσης «έκδοση άδειας οδήγησης». Το protégé περιέχει το owl plugin το οποίο μας δίνει την δυνατότητα να ορίζουμε κλάσεις και ιδιότητές τους και να δημιουργεί ένα αρχείο σε owl όπου δημιουργείται η οντολογία την οποία δημιουργήσαμε.

Στην αρχή δημιουργούμε ένα project με την επιλογή File->New Project και το παράθυρο που φαίνεται στο σχήμα 1.



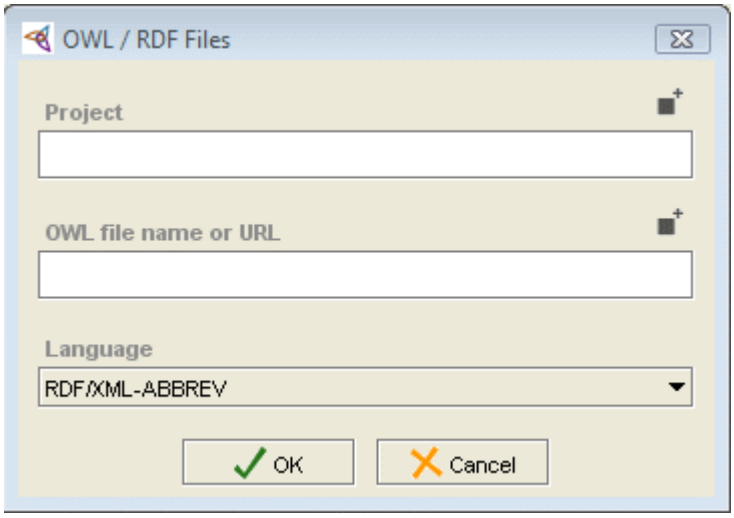
Σχήμα 1

Τα project στο protégé έχουν προέκταση .pprj. Επιλέγουμε OWL/RDF Files, Next και στο σχήμα 2 επιλέγουμε OWL DL και finish.



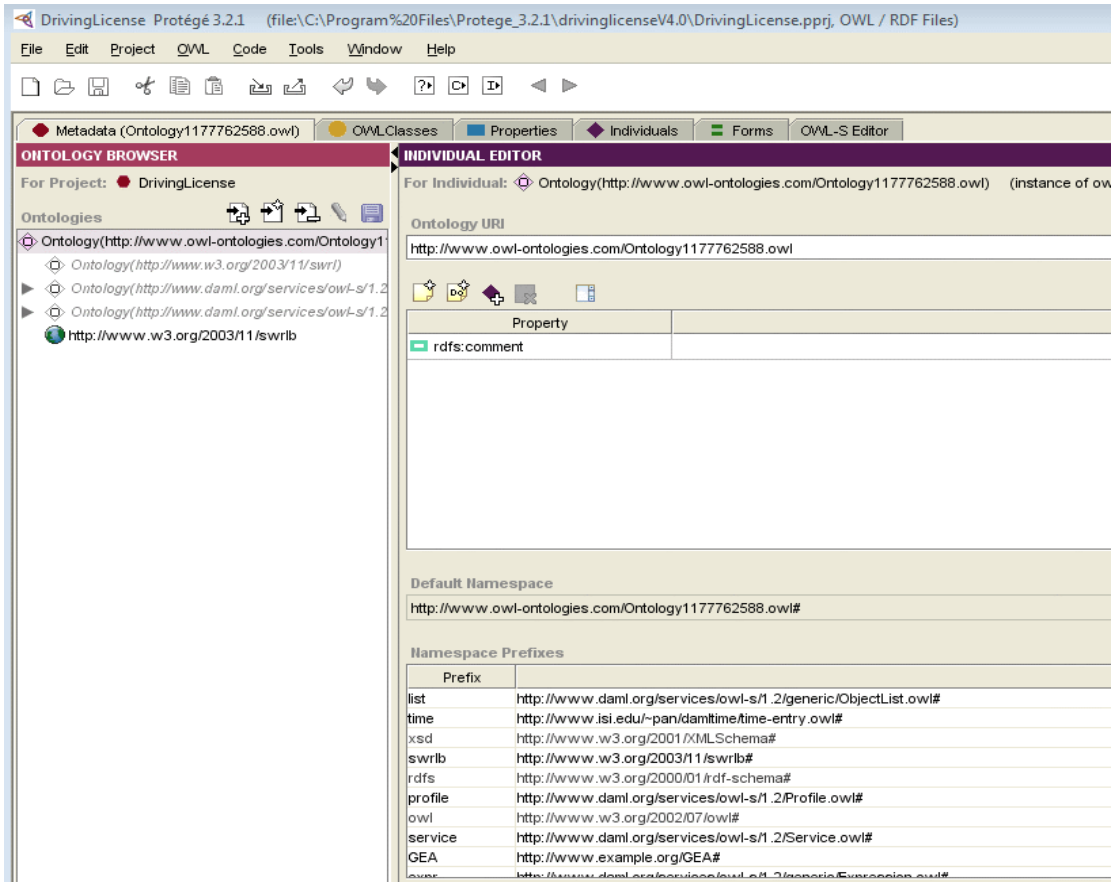
Σχήμα 2

Για να ονομάσουμε το project σαν DrivingLicense επιλέγουμε File->Save Project και στο παράθυρο του Σχήματος 3 γράφουμε το όνομα του project



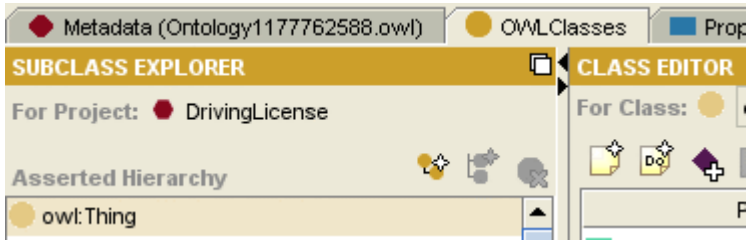
Σχήμα 3.

Η οθόνη που εμφανίζεται (σχήμα 4)ειναι η κεντρική του protégé που περιέχει κάποιες καρτέλλες όπως Metadata, OwlClasses όπου κατασκευάζουμε τις κλάσεις της οντολογίας μας, Properties όπου φιάχνουμε τις ιδιότητες των κλάσεων ειτε Object properties είτε Datatype properties, και την καρτέλλα OWL-SEditor ο οποίος είναι ενα plugin με το οποίο φιάχνουμε μια διεργασία που θέλουμε σε OWL-S. Στην περίπτωσή μας θα το χρησιμοποιήσουμε για να φιάξουμε την διεργασία για την έκδοση της DrivingLicense.

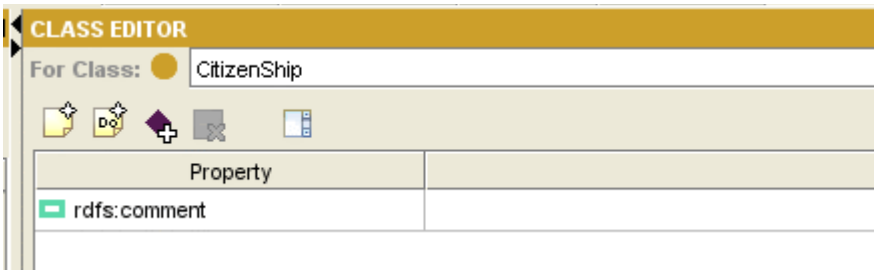


Σχήμα 4

Στη συνέχεια πρέπει να γράψουμε τις κλάσεις απο τις οποίες αποτελείται η οντολογία μας και αυτό το κάνουμε στην καρτέλλα OWLClasses του protégé.Στην περίπτωση της DrivingLicense οντολογίας δημιουργούμε τις κλάσεις που φαίνονται στα σχήματα 7 και 8 αντίστοιχα.Πιό κάτω θα περιγράψουμε τις κλάσεις αυτές.Δημιουργούμε μια κλάση σύμφωνα με το σχήμα 5 και γράφουμε το όνομα της κλάσης σύμφωνα με το σχήμα 6.Δεξιά από το Asserted Hierarchy με το πάτημα των δύο ενωμένων κύκλων δημιουργούμε μια κλάση και με το πάτημα του κύκλου με το X διαγράφουμε μια ήδη δημιουργημένη κλάση(Σχήμα 5).Στο σχήμα 6 βλέπουμε που γράφουμε το όνομα της κλάσης που έχουμε δημιουργήσει.



Σχήμα 5

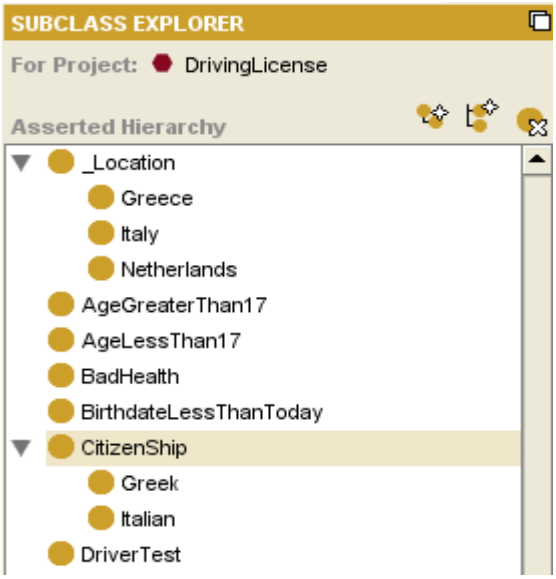


Σχήμα 6

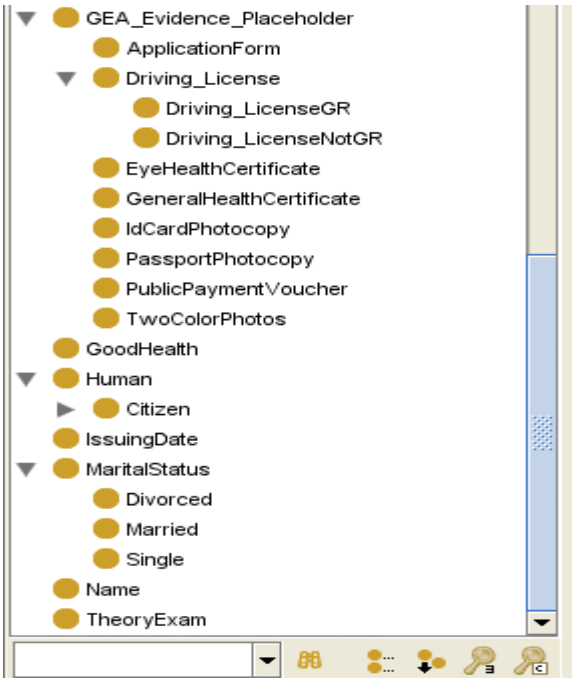
Παρακάτω περθγράφουμε τις κλάσεις απο τις οποίες αποτελείται η οντολογία μας DrivingLicense.Αυτές ειναι οι ακόλουθες:

- _Location με ενδεικτικές υποκλάσεις Greece, Italy Netherlands που δηλώνει τον τόπο διαμονής του υποψηφίου
- AgeGreaterThan17 και AgeLessThan17 ειναι κλάσεις που δηλώνουν το σύνολο των υποψηφίων πάνω και κάτω από 17 ετών
- GoodHealth και BadHealth ειναι κλάσεις που δηλώνουν το σύνολο των υποψηφίων που έχουν καλή και κακή υγεία αντίστοιχα
- BirthDateLessThanToday ειναι κλάση που δηλώνει το σύνολο των υποψηφίων με ημερομηνία γέννησης μικρότερη της σημερινής.
- CitizenShip με ενδεικτικές υποκλάσεις Greek και Italian ειναι κλάση που δηλώνει το σύνολο των υποψηφίων με κάποια εθνικότητα.
- DriverTest ειναι κλάση που δηλώνει εξέταση οδήγησης
- GEA_Evidence_Placeholder ειναι κλάση που δηλώνει κάποιο αντικείμενο ένδειξης και το οποίο αποτελείται από αρκετές υποκλάσεις που είναι:ApplicationForm η κλάση της αίτησης, DrivingLicense η κλάση της άδειας οδήγησης με υποκλάσεις τις Driving_LicenseGR και Driving_LicenseNotGR που δηλώνουν τις άδειες οδήγησης που έχουν εκδοθεί στην Ελλάδα και αυτές εκτός Ελλάδας αντίστοιχα, EyeHealthCertificate που είναι η κλάση του πιστοποιητικού υγείας ματιών, GeneralHealthCertificate που είναι η κλάση του γενικού πιστοποιητικού υγείας, IDCardPhotocopy η κλάση του φωτοαντίγραφου ταυτότητας, PassportPhotocopy η κλάση του αντίγραφου διαβατηρίου, PublicPaymentVoucher η κλάση του παράβολου και TwoColorPhotos η κλάση των δύο εγχρωμων φωτογραφιών.
- Human με υποκλάση Citizen και με υποκλάση Driver που δηλώνουν τις κλάσεις άνθρωπος, πολίτης και οδηγός αντίστοιχα
- IssuingDate η κλάση της ημερομηνίας έκδοσης του διπλώματος
- MaritalStatus με υποκλάσεις Divorced, Married και Single που ειναι η κλάση της οικογενειακής κατάστασης με υποκλάσεις χωρισμένος, παντρεμένος και εργένης
- Name η κλάση των ονομάτων
- TheoryExam η κλάση της θεωρητικής εξέτασης.

Αυτές οι κλάσεις φαίνονται στα σχήματα 7 και 8.



Σχήμα 7

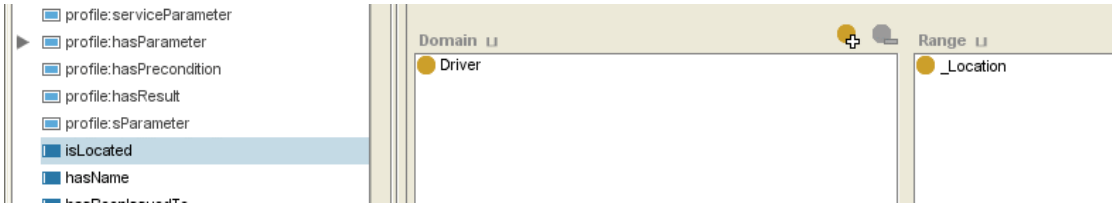


Σχήμα 8

Αφού καταγράψουμε τις κλάσεις το επόμενο βήμα ειναι να βρούμε και να γράψουμε τις ιδιότητες της οντολογίας μας τις οποίες θα καταγράψουμε σαν Object Properties στην καρτέλλα Properties του protégé.

Σχήμα 9

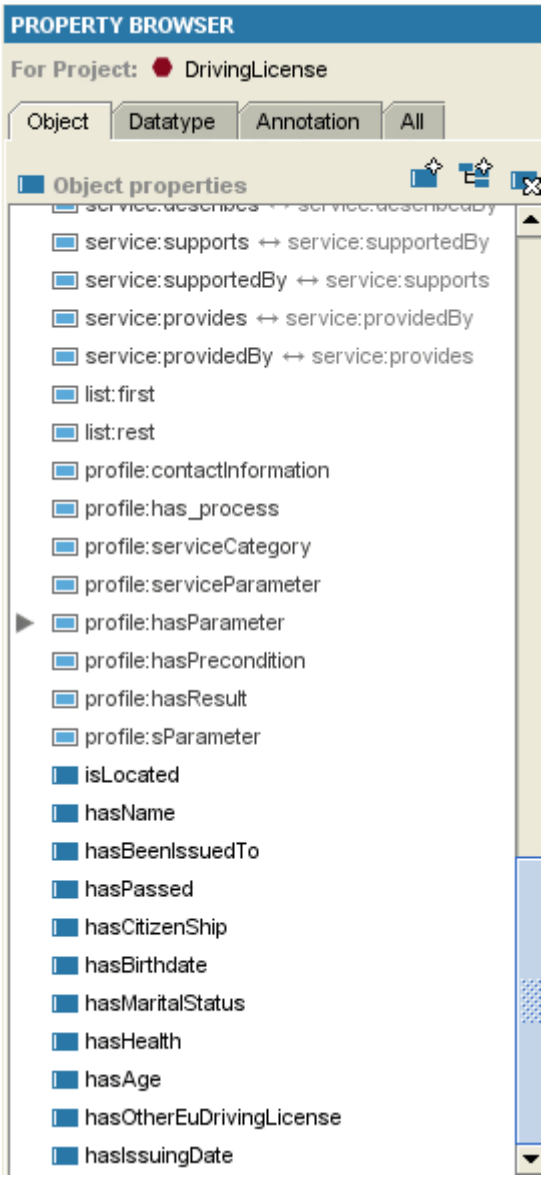
Στο σχήμα 9 φαίνεται που γράφουμε το όνομα της ιδιότητας στον Property Editor ενω μια νέα ιδιότητα αντικειμένου την δημιουργούμε πατώντας το μπλέ τετραγωνάκι δεξιά απ τις λέξεις Object properties.Πατώντας το εικονίδιο με το μπλέ τετραγωνάκι με το Χ διαγράφουμε μια ιδιότητα.
Παρακάτω αναλύουμε τις ιδιότητες της οντολογίας μας.Κάθε ιδιότητα έχει ενα πεδιο ορισμού (Domain) και ένα πεδίο τιμών (range) τα οποία τα δηλώνουμε στα πεδία που φαίνονται στο σχήμα 10 παρακάτω.Π.χ. η ιδιότητα isLocated εχει πεδίο ορισμού την κλάση Οδηγος Driver και πεδίο τιμών την κλάση _Location.Με τον κύκλο που εχει το + δημιουργούμε νέο πεδίο ορισμού.



Σχήμα 10

Οι ιδιότητες που καταγράψαμε ειναι οι ακόλουθες:

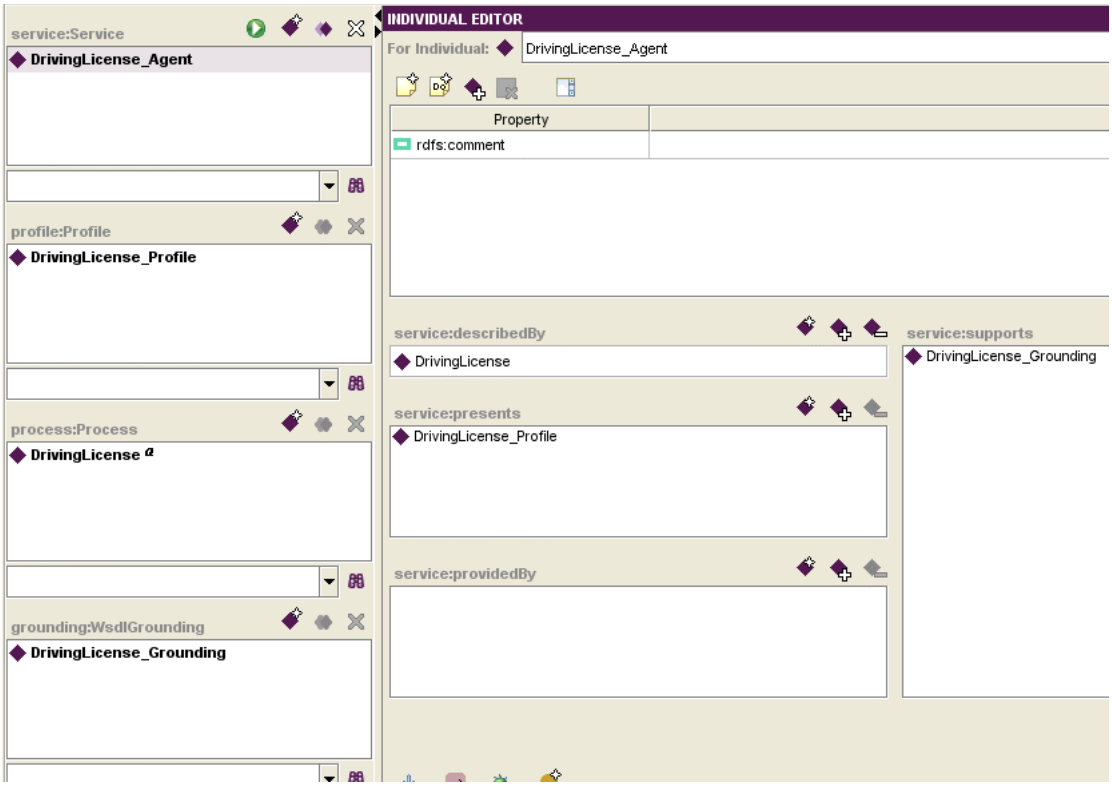
- isLocated – ο οδηγός βρίσκεται στον τόπο που ζεί
- hasName - ο οδηγός εχει όνομα Name
- hasBeenIssuedTo – ο οδηγός του έχει εκδοθεί μια άδεια οδήγησης
- hasPassed - ο οδηγός έχει περάσει την ένωση των κλάσεων DriverTest και TheoryExam
- hasCitizenShip – ο πολίτης εχει εντοπιότητα(CitizenShip)
- hasBirthDate – ο οδηγός έχει ημερομηνία γέννησης την κλάση BirthDateLessThanToday
- hasMaritalStatus – ο πολίτης έχει οικογενειακή κατάσταση MaritalStatus
- hasHealth - ο οδηγός έχει GoodHealth
- has Age - ο οδηγός έχει ηλικία μεγαλύτερη απο 17 ετών
- hasOtherEUDrivingLicense – ο οδηγός έχει άδεια οδήγησης εκτός Ελλάδας Driving_LicenseNotGR
- hasIssuingDate - η άδεια οδήγησης έχει ημερομηνία έκδοσης IssuingDate



Σχήμα 11

Στο σχήμα 11 φαίνεται ο property browser με τις ιδιότητες αντικειμένων της οντολογίας μας.Αν έπρεπε να δημιουργήσουμε datatype properties θα έπρεπε να δουλέψουμε στην καρτέλλα Datatype.

Στη συνέχεια πρέπει να μοντελοποιήσουμε με την βοήθεια του OWL-S Editor την διεργασία της έκδοσης της άδειας οδήγησης.Η υπηρεσία η οποία θα πειγραφεί ειναι η DriverLicense_Agent η οποία σύμφωνα με την ορολογία της OWL-S ειναι describedBy της DrivingLicense διεργασίας, παρουσιάζει (presents) DrivingLicense_Profile και supports ενα DrivingLicense_Grounding.Αυτό φαίνεται στον OWL-S Editor στο σχήμα 12



Σχήμα 12

Το DrivingLicense_Profile εχει hasInput, hasOutput, και hasParameter παραμέτρους.Αυτές τις παραθέτουμε τώρα

hasInput

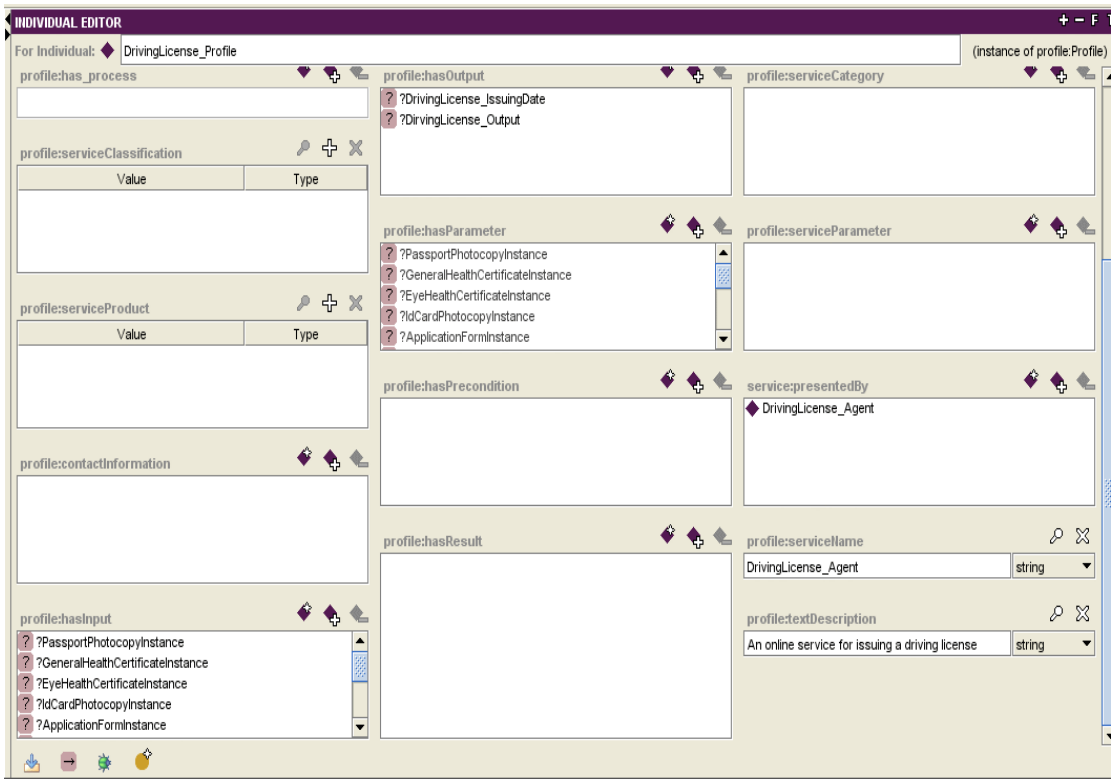
?PassportPhotocopyInstance
?GeneralHealthCertificateInstance
?EyeHealthCertificateInstance
?IDCardPhotocopyInstance
?ApplicationFormInstance
?PublicPaymentVoucherInstance
?BirthDateInput
?TwoColorPhotosInstance
?AgeInput
?LocationInput
?DriverInstance

hasOutput

?DrivingLicense_IssuingDate
?DrivingLicense_Output

hasParameter

Ειναι οι παράμετροι hasInput και hasOutput.το ονομα του Service είναι DrivingLicense_Agent.Τα παραπάνω φαίνονται στο σχήμα 13



Σχήμα 13

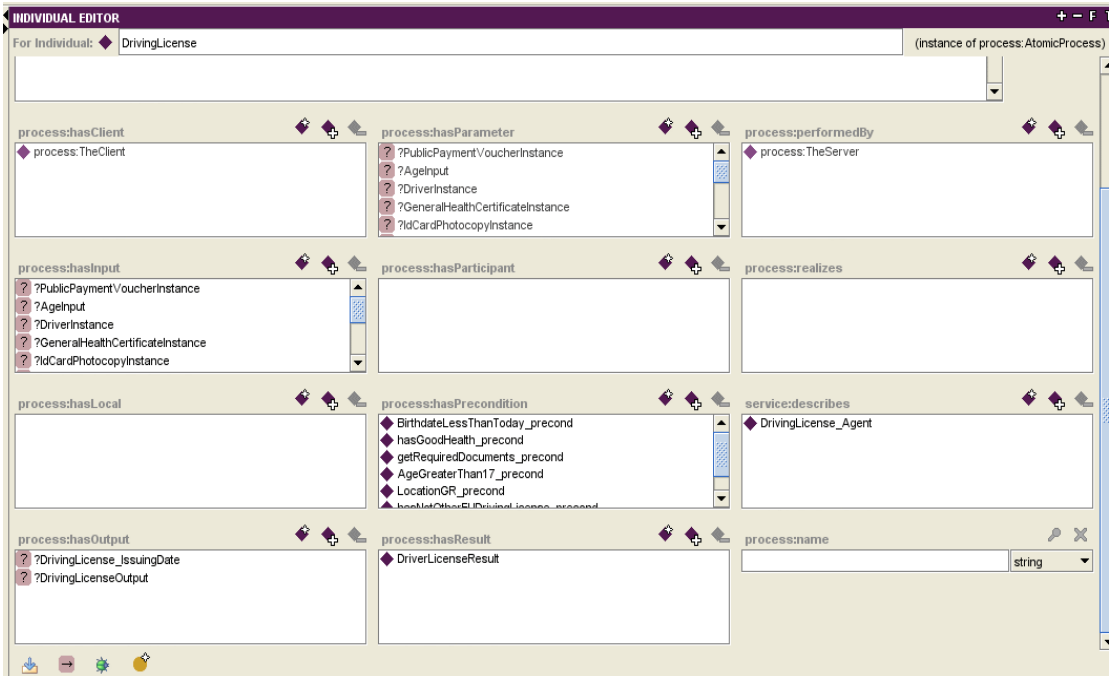
Το επόμενο βήμα ειναι να καθορίσουμε στον OWL-S Editor την ίδια την διεργασία την οποία δίνουμε το όνομα **DrivingLicense.** Τα πεδία hasInput, hasOutput και hasParameter ειναι τα ίδια με αυτά του Service Profile. Η διεργασία μας εχει process:hasClient τον TheClient και process:performedBy τον TheServer.Τώρα η διεργασία μας εχει 7 preconditions τα οποία τα υλοποιούμε με την γλώσσα SWRL στον SWRL editor που θα δούμε στη συνέχεια.Αυτά τα preconditions ειναι τα ακόλουθα:

hasPreconditions

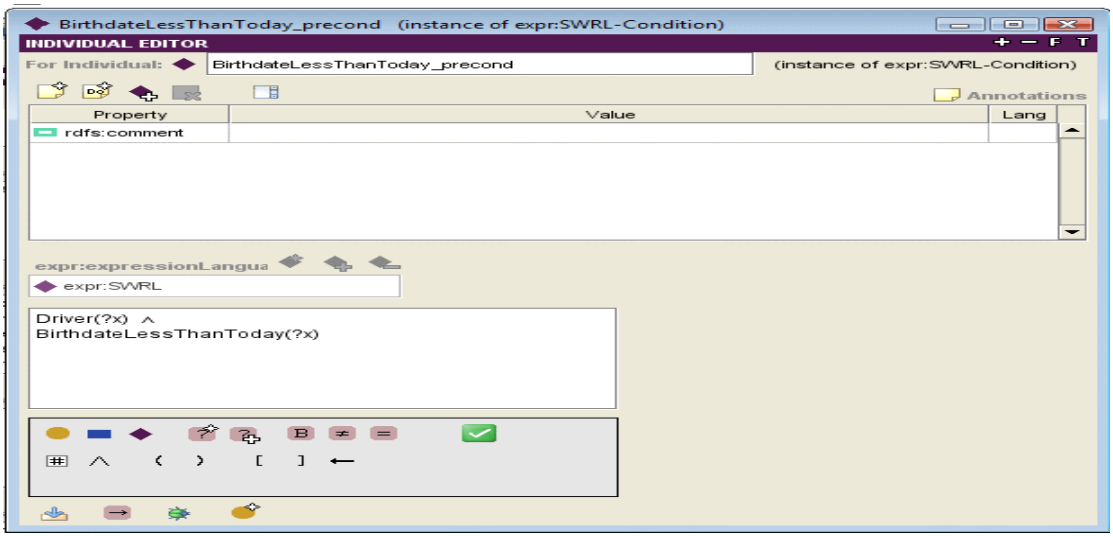
- BirthDateLessThanToday_precond με swrl συνθήκη Driver(?x) ^ BirthdateLessThanToday
- hasGoodHealth_precond με swrl συνθήκη Driver(?x) ^GoodHealth(?x)
- getRequiredDocuments_precond με swrl συνθήκη Driver(?x) ^GeneralHealthCertificate(?x)^EyeHealthCertificate(?x)^ TwoColorPhotos(?x)^ApplicationForm(?x)^PassportPhotocopy(?x) ^PublicPaymentVoucher(?x)^IdCardPhotocopy(?x)
- AgeGreaterThan17_precond με swrl συνθήκη Driver(?x) ^AgeGreaterThan17(?x)
- LocationGR_precond με swrl συνθήκη Driver(?x) ^Greece(?x)
- hasNotOtherEuDrivingLicense_precond με swrl συνθήκη Driver(?x) ^Driving_LicenseGR(?x)

- Tests_precond με swrl συνθήκη Driver(?x) ^ TheoryExam(?x)^DriverTest(?x)

Στα σχήματα 14 και 15 φαίνονται η οθόνη του OWL-S Editor για την DrivingLicense process και ο SWRL Editor για το precondition BirthDateLessThanToday_precond



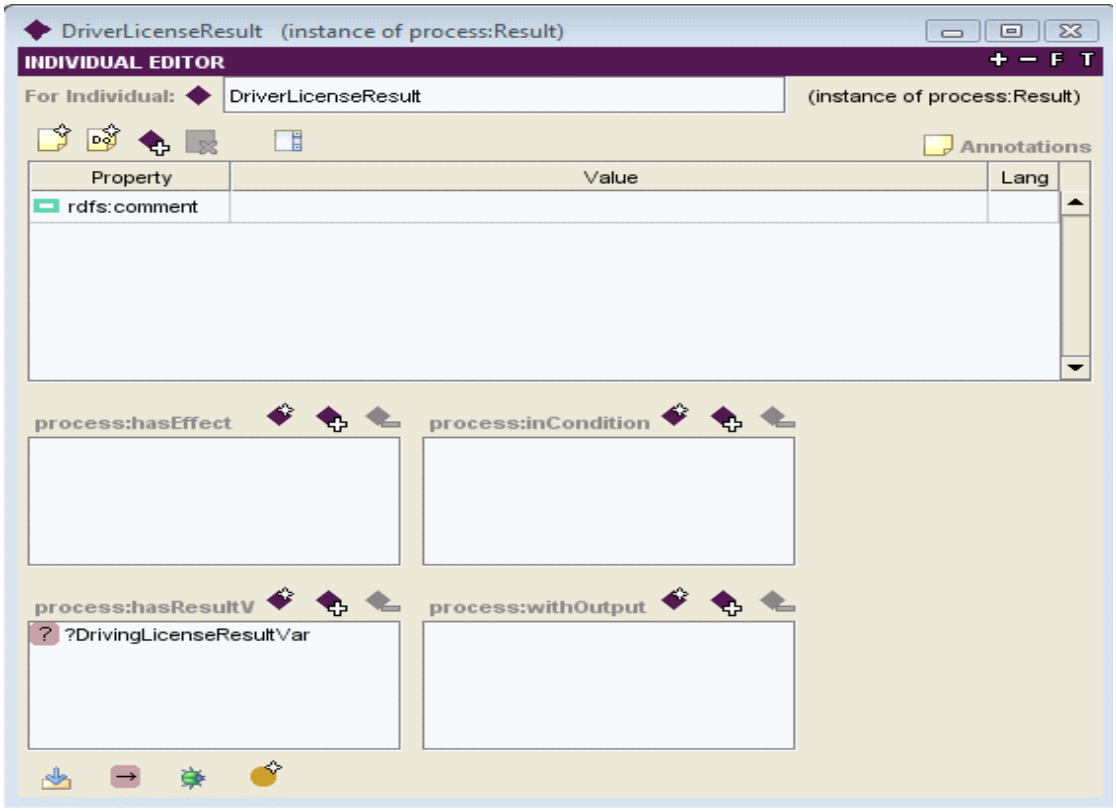
Σχήμα 14



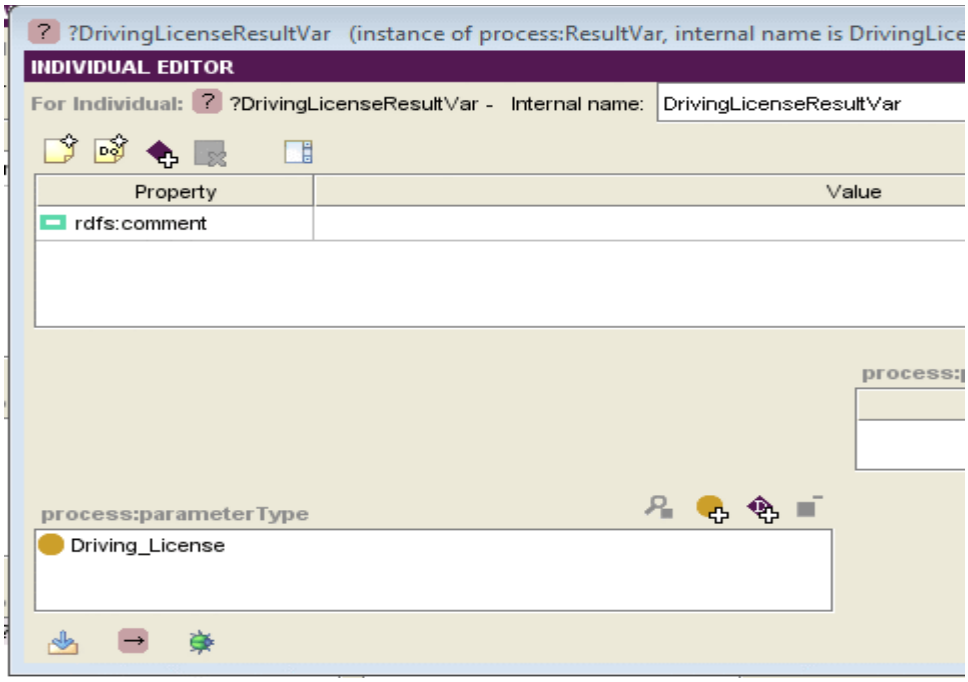
Σχήμα 15

Τέλος στο process:hasResult καθορίζουμε το DriverLicenseResult σύμφωνα με την θεωρία της OWL-S.Αυτά καθορίζονται με το DrivingLicenseResultVar και το οποίο εχει parameterType την κλάση Driving_License.Αυτά φαίνονται στα σχήματα

16 και 17 αντίστοιχα.Για να τα δηλώσουμε πρέπει να κάνουμε διπλό κλικ πάνω στο DriverLicenseResult.

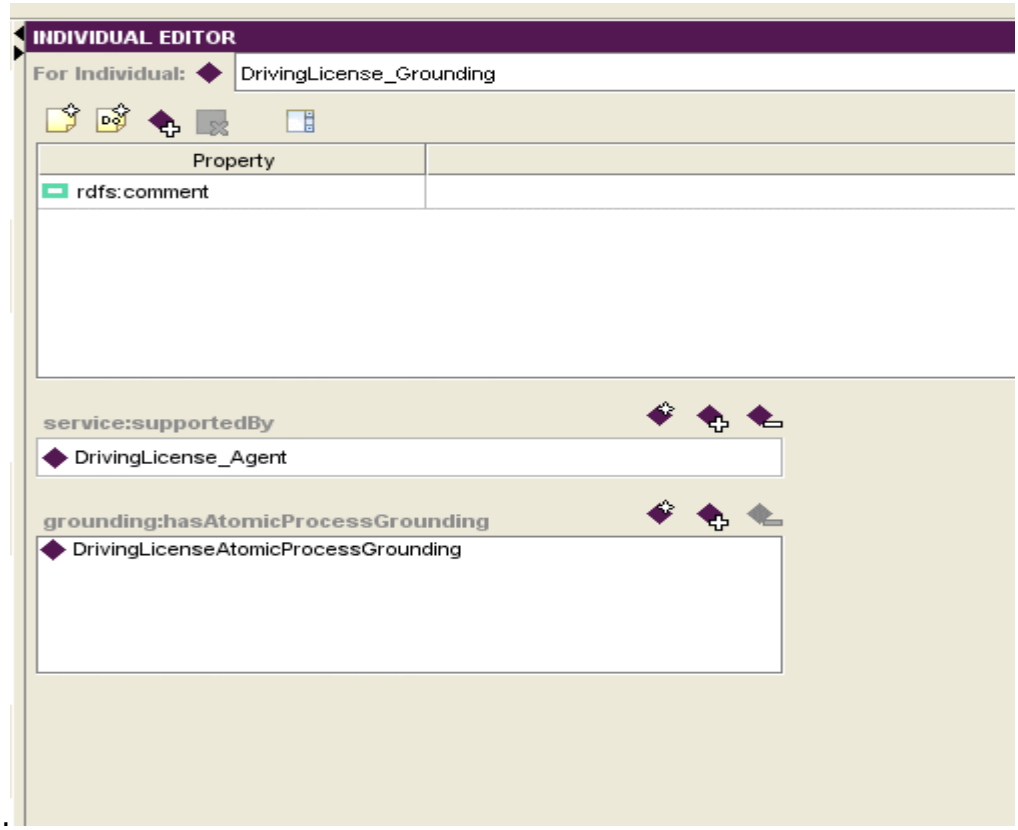


Σχήμα 16



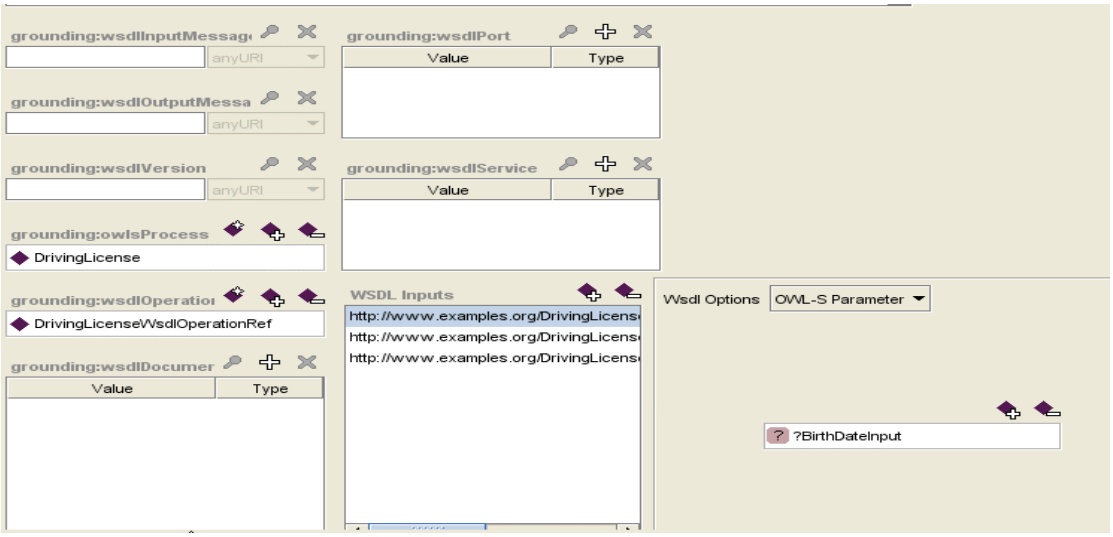
Σχήμα 17

Αυτό που απομένει ειναι σύμφωνα με την θεωρία της OWL-S να καθορίσουμε το Grounding της διεργασίας μας.Αυτή καθορίζεται στον OWL-S Editor όπως στο σχήμα 18

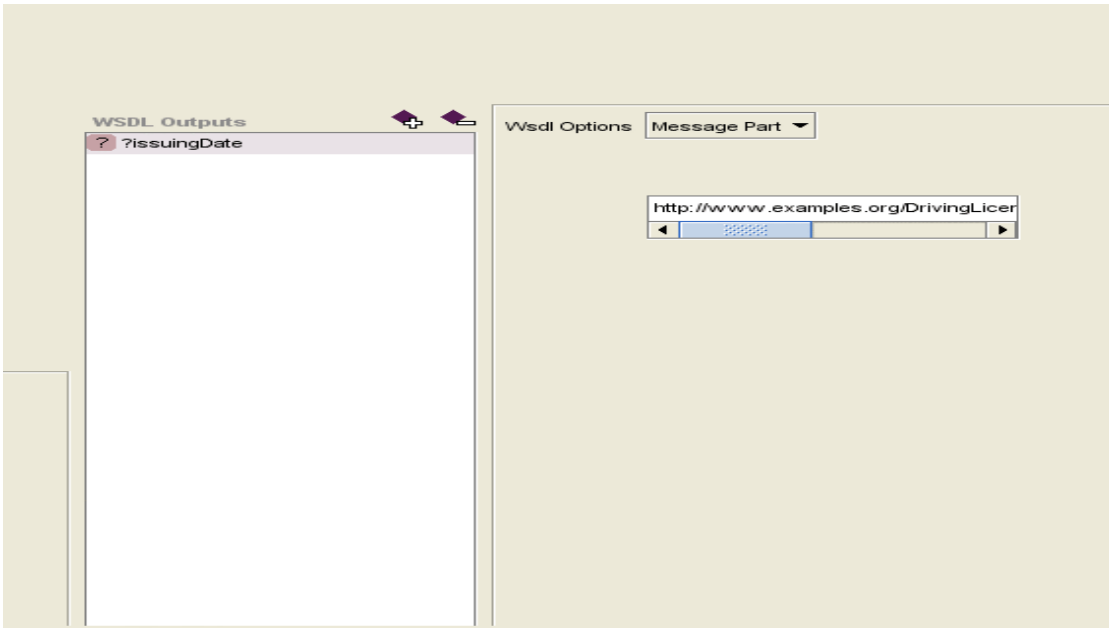


Σχήμα 18

Με διπλό κλικ στο DrivingLicenseAtomicProcessGrounding μπαίνουμε στην οθόνη DrivingLicenseAtomicProcessGrounding όπου αριστερά ειναι τα WSDLInputs και δεξιά τα WSDLOutputs. Αυτά φαίνονται στα σχήματα 19 και 20.

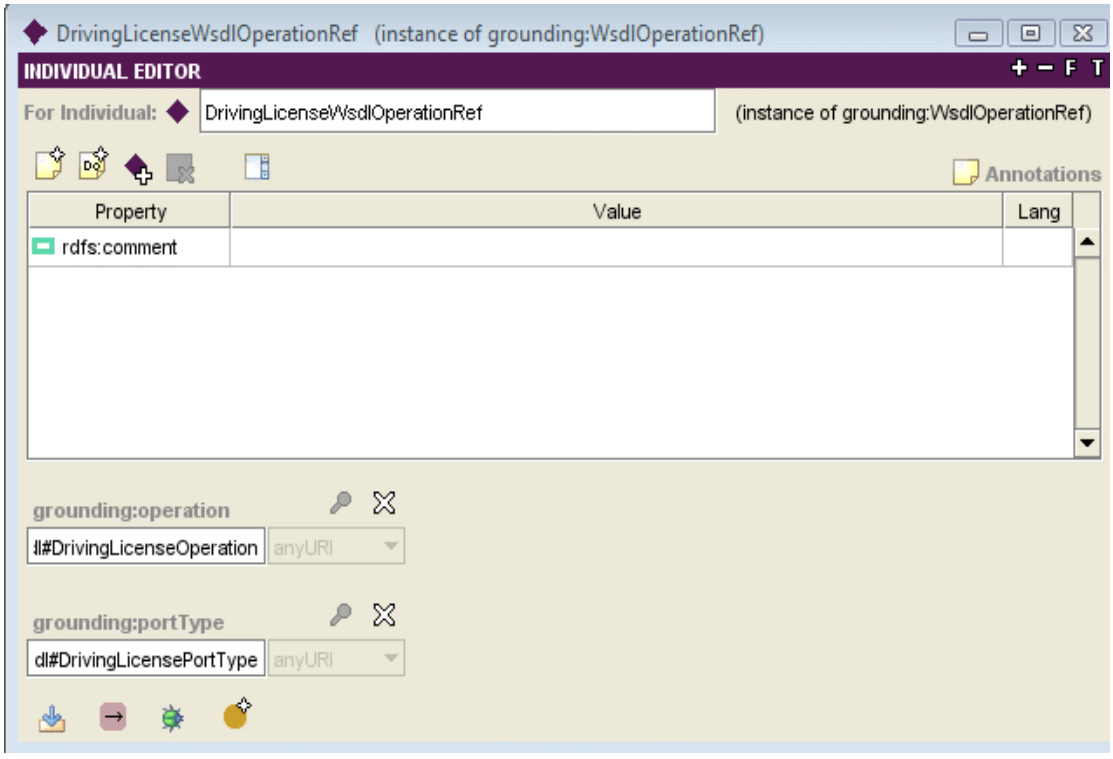


Σχήμα 19



Σχήμα 20

Με διπλό κλικ στο DrivingLicenseWsdloperationRef εμφανίζεται η οθόνη του σχήματος 21 η οποία καθορίζει το DrivingLicenseOperation και το DrivingLicensePortType τα οποία ειναι στοιχεία του WSDL αρχείου το οποίο θα παραθέσουμε το listing πιό κάτω.



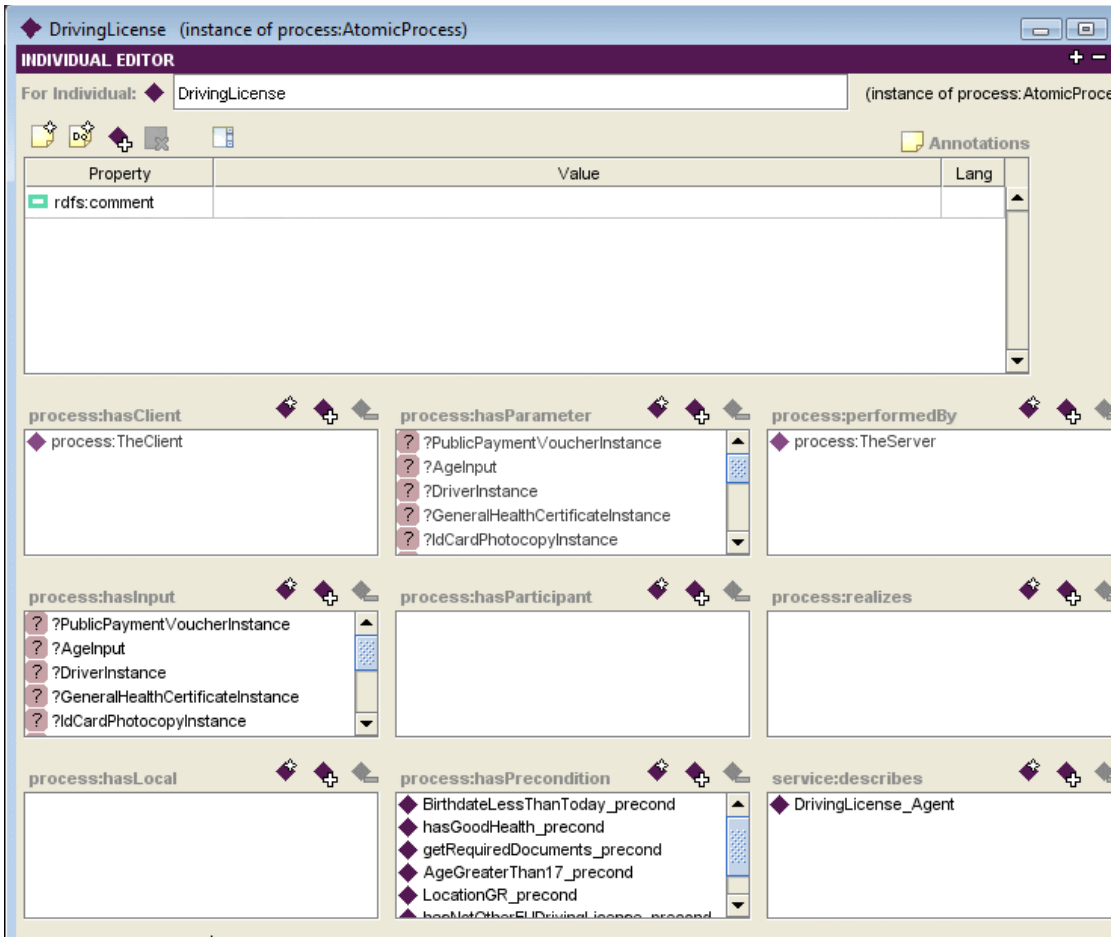
Σχήμα 21

Στο σχήμα 19 φαίνονται οι είσοδοι του WSDL αρχείου της διεργασίας μας.Αυτές οι είσοδοι ειναι: ?BirthDateInput, ?AgeInput, και ?LocationInput.Απο την αναδιπλούμενη λίστα wsdl options καθορίζουμε οτι οι είσοδοι ειναι OWL-S

παράμετροι.Αυτοί οι είσοδοι του ServiceGrounding αντιστοιχούν στα πεδία του wsdl αρχείου αντίστοιχα
http://www.examples.org/DrivingLicenseWebService/DrivingLicense.wsdl#BirthDate
http://www.examples.org/DrivingLicenseWebService/DrivingLicense.wsdl#Age
και
http://www.examples.org/DrivingLicenseWebService/DrivingLicense.wsdl#Location

Στο σχήμα 20 φαίνονται οι εξοδοι του WSDL αρχείου της διεργασίας μας.Αυτή η έξοδος είναι η ?IssuingDate. Απο την αναδιπλούμενη λίστα wsdl options καθορίζουμε οτι οι έξοδοι ειναι Message parts.Αυτή η έξοδος αντιστοιχεί στο πεδίο του wsdl αρχείου
http://www.examples.org/DrivingLicenseWebService/DrivingLicense.wsdl#issuingDate

Στο σχήμα 19 αν πατήσουμε με διπλό κλικ πάνω στο grounding:owlsprocess ->DrivingLicense εμφανίζεται η οθόνη της DrivingLicense process όπως στο σχήμα 22



Σχήμα 22

Παρακάτω παραθέτουμε το wsdl αρχείο που έχουμε δημιουργήσει για την διεργασία DrivingLicense.

WSDL LISTING

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="DrivingLicense"
targetNamespace="http://www.examples.org/DrivingLicenseWebService"
    xmlns:tns="http://www.examples.org/DrivingLicenseWebService"
    xmlns:dlxsd="http://www.examples.org/DrivingLicenseWebService/typ
es.xsd"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <!-- Type definitions -->
  <types>
    <xsd:schema
targetNamespace="http://www.examples.org/DrivingLicenseWebService/typ
es.xsd"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">

      <xsd:complexType name="DrivingLicenseInputs">
        <xsd:sequence>
          <xsd:element name="Age" type="xsd:integer"/>
          <xsd:element name="Location" type="xsd:string"/>
          <xsd:element name="Birthdate" type="xsd:date"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="issuingDate" type="xsd:date"/>
      <xsd:element name="error" type="xsd:string"/>

    </xsd:schema>
  </types>

  <!-- Message definitions -->

  <message name ="DrivingLicenseWSDLRequest">
    <part name="body" element="dlxsd:DrivingLicenseInputs"/>
  </message>

  <message name ="DrivingLicenseWSDLResponse">
    <part name="body" element="dlxsd:issuingDate"/>
  </message>

  <message name ="DrivingLicenseWSDLError">
    <part name="error" element="dlxsd:error"/>
  </message>
  -->

  <!-- Port type definitions -->

  <portType name="DrivingLicensePortType">
    <operation name="DrivingLicenseWSDLOperation">
      <input message="tns:DrivingLicenseWSDLRequest"/>
      <output message="tns:DrivingLicenseWSDLResponse"/>
      <fault message="tns:DrivingLicenseWSDLError" name="error"/>
    </operation>
  </portType>
```

```
  <!-- Binding definitions -->

  <binding name="DrivingLicenseBinding"
type="tns:DrivingLicensePortType">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document"/>
    <operation name="DrivingLicenseWSDLOperation">
      <soap:operation
soapAction="http://www.examples.org/DrivingLicenseWebServices"/>
      <input>
        <soap:body use="literal"
namespace="http://www.examples.org/DrivingLicenseWebService/types.xsd
"/>
      </input>
      <output>
        <soap:body use="literal"
namespace="http://www.examples.org/DrivingLicenseWebService/types.xsd
"/>
      </output>
      <fault name="error">
        <soap:fault name="error" use="literal"
namespace="http://www.examples.org/DrivingLicenseWebService/types.xsd
"/>
      </fault>
    </operation>
  </binding>


  <!-- Service and Port (network address) definitions -->

  <service name="DrivingLicenseService">
    <port name="DrivingLicenseWSDLPort"
binding="tns:DrivingLicenseBinding">
      <soap:address
location="http://www.examples.org/DrivingLicenseWebService"/>
    </port>
  </service>


</definitions>
```

Παραθέτουμε ένα μέρος του OWL αρχείου που δημιουργεί το πρόγραμμα protégé αφού δημιουργήσουμε τις κλάσεις, τις ιδιότητες και τα στοιχεία που δημιουργήσαμε στον OWL-S Editor.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:service="http://www.daml.org/services/owl-
s/1.2/Service.owl#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:time="http://www.isi.edu/~pan/damltime/time-entry.owl#"
    xmlns:list="http://www.daml.org/services/owl-
s/1.2/generic/ObjectList.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:expr="http://www.daml.org/services/owl-
s/1.2/generic/Expression.owl#"
```

```
    xmlns:GEA="http://www.example.org/GEA#"
    xmlns="http://www.owl-ontologies.com/Ontology1177762588.owl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:profile="http://www.daml.org/services/owl-
s/1.2/Profile.owl#"
    xmlns:process="http://www.daml.org/services/owl-
s/1.2/Process.owl#"
    xmlns:grounding="http://www.daml.org/services/owl-
s/1.2/Grounding.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xml:base="http://www.owl-ontologies.com/Ontology1177762588.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.w3.org/2003/11/swrl"/>
    <owl:imports rdf:resource="http://www.daml.org/services/owl-
s/1.2/Grounding.owl"/>
    <owl:imports rdf:resource="http://www.daml.org/services/owl-
s/1.2/Profile.owl"/>
    <owl:imports rdf:resource="http://www.w3.org/2003/11/swrlb"/>
  </owl:Ontology>
  <owl:Class rdf:ID="EyeHealthCertificate">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="GEA_Evidence_Placeholder"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Driving_LicenseNotGR">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Driving_License"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="TheoryExam"/>
  <owl:Class rdf:ID="Driving_LicenseGR">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Driving_License"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Greek">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="CitizenShip"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="MaritalStatus"/>
  <owl:Class rdf:ID="Driver">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Citizen"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Name"/>
  <owl:Class rdf:about="#Driving_License">
    <rdfs:subClassOf rdf:resource="#GEA_Evidence_Placeholder"/>
  </owl:Class>
  <owl:Class rdf:ID="TwoColorPhotos">
    <rdfs:subClassOf rdf:resource="#GEA_Evidence_Placeholder"/>
  </owl:Class>
  <owl:Class rdf:ID="IdCardPhotocopy">
    <rdfs:subClassOf rdf:resource="#GEA_Evidence_Placeholder"/>
  </owl:Class>
  <owl:Class rdf:ID="AgeGreaterThan17"/>
  <owl:Class rdf:ID="Human"/>
  <owl:Class rdf:ID="_Location"/>
  <owl:Class rdf:ID="IssuingDate"/>
```

```
<owl:Class rdf:ID="PublicPaymentVoucher">
  <rdfs:subClassOf rdf:resource="#GEA_Evidence_Placeholder"/>
</owl:Class>
<owl:Class rdf:ID="Italy">
  <rdfs:subClassOf rdf:resource="#_Location"/>
</owl:Class>
<owl:Class rdf:ID="BirthdateLessThanToday"/>
<owl:Class rdf:ID="Single">
  <rdfs:subClassOf rdf:resource="#MaritalStatus"/>
</owl:Class>
<owl:Class rdf:ID="Greece">
  <rdfs:subClassOf rdf:resource="#_Location"/>
</owl:Class>
<owl:Class rdf:ID="BadHealth"/>
<owl:Class rdf:ID="GoodHealth"/>
<owl:Class rdf:ID="PassportPhotocopy">
  <rdfs:subClassOf rdf:resource="#GEA_Evidence_Placeholder"/>
</owl:Class>
<owl:Class rdf:ID="Italian">
  <rdfs:subClassOf rdf:resource="#CitizenShip"/>
</owl:Class>
<owl:Class rdf:ID="ApplicationForm">
  <rdfs:subClassOf rdf:resource="#GEA_Evidence_Placeholder"/>
</owl:Class>
<owl:Class rdf:ID="Netherlands">
  <rdfs:subClassOf rdf:resource="#_Location"/>
</owl:Class>
<owl:Class rdf:ID="Divorced">
  <rdfs:subClassOf rdf:resource="#MaritalStatus"/>
</owl:Class>
<owl:Class rdf:ID="DriverTest"/>
<owl:Class rdf:about="#Citizen">
  <rdfs:subClassOf rdf:resource="#Human"/>
</owl:Class>
<owl:Class rdf:ID="Married">
  <rdfs:subClassOf rdf:resource="#MaritalStatus"/>
</owl:Class>
<owl:Class rdf:ID="AgeLessThan17"/>
<owl:Class rdf:ID="GeneralHealthCertificate">
  <rdfs:subClassOf rdf:resource="#GEA_Evidence_Placeholder"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="isLocated">
  <rdfs:domain rdf:resource="#Driver"/>
  <rdfs:range rdf:resource="#_Location"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasName">
  <rdfs:range rdf:resource="#Name"/>
  <rdfs:domain rdf:resource="#Driver"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasBeenIssuedTo">
  <rdfs:range rdf:resource="#Driving_License"/>
  <rdfs:domain rdf:resource="#Driver"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPassed">
  <rdfs:domain rdf:resource="#Driver"/>
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#DriverTest"/>
        <owl:Class rdf:about="#TheoryExam"/>
      </owl:unionOf>
```

```
      </owl:Class>
    </rdfs:range>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasCitizenShip">
    <rdfs:range rdf:resource="#CitizenShip"/>
    <rdfs:domain rdf:resource="#Citizen"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasBirthdate">
    <rdfs:domain rdf:resource="#Driver"/>
    <rdfs:range rdf:resource="#BirthdateLessThanToday"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasMaritalStatus">
    <rdfs:range rdf:resource="#MaritalStatus"/>
    <rdfs:domain rdf:resource="#Citizen"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasHealth">
    <rdfs:range rdf:resource="#GoodHealth"/>
    <rdfs:domain rdf:resource="#Driver"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasAge">
    <rdfs:domain rdf:resource="#Driver"/>
    <rdfs:range rdf:resource="#AgeGreaterThan17"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasOtherEuDrivingLicense">
    <rdfs:domain rdf:resource="#Driver"/>
    <rdfs:range rdf:resource="#Driving_LicenseNotGR"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasIssuingDate">
    <rdfs:domain rdf:resource="#Driving_License"/>
    <rdfs:range rdf:resource="#IssuingDate"/>
  </owl:ObjectProperty>
  <swrl:AtomList>
    <rdf:rest>
      <swrl:AtomList>
        <rdf:rest>
          <swrl:AtomList>
            <rdf:rest>
              <swrl:AtomList>
                <rdf:first>
                  <swrl:ClassAtom>
                    <swrl:argument1>
                      <swrl:Variable rdf:ID="x"/>
                    </swrl:argument1>
                    <swrl:classPredicate
rdf:resource="#TwoColorPhotos"/>
                  </swrl:ClassAtom>
                </rdf:first>
                <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#nil"/>
              </swrl:AtomList>
            </rdf:rest>
            <rdf:first>
              <swrl:ClassAtom>
                <swrl:argument1 rdf:resource="#x"/>
                <swrl:classPredicate
rdf:resource="#PassportPhotocopy"/>
              </swrl:ClassAtom>
            </rdf:first>
          </swrl:AtomList>
        </rdf:rest>
        <rdf:first>
```

```
          <swrl:ClassAtom>
            <swrl:argument1 rdf:resource="#x"/>
            <swrl:classPredicate
rdf:resource="#GeneralHealthCertificate"/>
          </swrl:ClassAtom>
        </rdf:first>
      </swrl:AtomList>
    </rdf:rest>
    <rdf:first>
      <swrl:ClassAtom>
        <swrl:classPredicate rdf:resource="#EyeHealthCertificate"/>
        <swrl:argument1 rdf:resource="#x"/>
      </swrl:ClassAtom>
    </rdf:first>
  </swrl:AtomList>
  <process:Input rdf:ID="Input_9"/>
  <swrl:AtomList>
    <rdf:rest>
      <swrl:AtomList>
        <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-
syntax-ns#nil"/>
        <rdf:first>
          <swrl:ClassAtom>
            <swrl:classPredicate
rdf:resource="#EyeHealthCertificate"/>
            <swrl:argument1 rdf:resource="#x"/>
          </swrl:ClassAtom>
        </rdf:first>
      </swrl:AtomList>
    </rdf:rest>
    <rdf:first>
      <swrl:ClassAtom>
        <swrl:classPredicate
rdf:resource="#GeneralHealthCertificate"/>
        <swrl:argument1 rdf:resource="#x"/>
      </swrl:ClassAtom>
    </rdf:first>
  </swrl:AtomList>
  <process:Output rdf:ID="Output_14"/>
  <process:Input rdf:ID="PublicPaymentVoucherInstance">
    <process:parameterType
rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.owl-
ontologies.com/Ontology1177762588.owl#PublicPaymentVoucher</process:p
arameterType>
  </process:Input>
  <swrl:AtomList>
    <rdf:first>
      <swrl:BuiltinAtom>
        <swrl:builtin
rdf:resource="http://www.w3.org/2003/11/swrlb#equal"/>
        <swrl:arguments>
          <rdf:List>
            <rdf:first>
              <process:Input rdf:ID="Location">
                <process:parameterType
rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
                >
```

Reasoning

Εκτός από την παροχή ενός API που κάνει δυνατή την προγραμματική εξερεύνηση και το editing των owl οντολογιών, το Protégé-OWL παρέχει ενα reasoning API που μπορεί να χρησιμοποιηθεί για να προσπελάσει εναν εξωτερικό DIG συμβατό reasoner και ετσι δίνει την δυνατότητα να γίνονται inferences σχετικά με κλάσεις και individuals σε οντολογίες.Παρακάτω θα περιγράψουμε πως το protégé-OWL επικοινωνεί με εναν reasoner και θα δωσουμε ενα παράδειγμα για την χρήση του protégé-OWL reasoning API.

Descriptions Logics Reasoners

Η OWL-DL έχει τις αρχές της στις περιγραφικές λογικές που είναι αποφασίσιμα κομάτια της first order logic(λογική πρώτης τάξης).Για ενα συγκεκριμένο θέμα μια λογική είναι αποφασίσιμη εαν ειναι δυνατόν να σχεδιάσουμε εναν αλγόριθμο που θα τερματίσει σε εναν πεπερασμένο αριθμό βημάτων(δηλ. Ο αλγόριθμος εγγυάται οτι δεν θα τρέχει για πάντα).Για παράδειγμα στην περιγραφική λογική ειναι δυνατόν να γράψουμε εναν αλγόριθμο που υπολογίζει εαν μια έννοια ειναι υποσύνολο μιας άλλης έννοιας και εγγυάται οτι θα τερματίσει υστερα απο πεπερασμένο αριθμό βημάτων.
Επειδή μια OWL-DL οντολογία μπορεί να μεταφραστεί σε μια αναπαράσταση περιγραφικής λογικής ειναι δυνατόν να κάνουμε αυτοματοποιημένο reasoning στην οντολογία χρησιμοποιώντας εναν description logic reasoner.Ενας description logic reasoner παρέχει διάφορες inferencing υπηρεσίες όπως υπολογισμό των inferred superclasses μιας κλάσης, αποφασίζοντας εαν μια κλάση ειναι consistent ή οχι(μια κλάση ειναι inconsistent εαν δεν εχει κανένα στιγμιότυπο), αποφασίζοντας εαν μια κλάση is subsumed απο μια άλλη.Μερικοί απο τους πιο δημοφιλείς description logics reasoners ειναι:

- RACER
- FaCT
- FaCT++
- Pellet

DIG συμβατοί Description Logics Reasoners

Επειδή α)για να γράψουμε ενα description logic reasoner δεν ειναι ενα απλό πράγμα και β) υψηλά βελτιστοποιημένοι description logics reasoners τρίτων εχουν ήδη αναπτυχθεί δεν υπάρχει λόγος να γράψει κάποιος εναν τέτοιον reasoner για μια συγκεκριμένη εφαρμογή.Ευτυχώς υπάρχει ένα πρότυπο που παρέχει ενα specification για εναν κοινό τρόπο σύνδεσης σε DL reasoners τρίτων.Αυτό το standard καλείται το «DIG Interface»(DIG σημαίνει DL Implementations Group).

Ενας DIG συμβατός description logic reasoner ειναι αυτός που παρέχει ενα standard interface για προσπέλαση που δίνει την δυνατότητα στον reasoner να προσπελαστεί πάνω απο HTTP χρησιμοποιώντας την DIG γλώσσα.Η DIG γλώσσα ειναι μια αναπαράσταση βασισμένη σε XML των οντολογικών οντοτήτων όπως κλάσεις ιδιότητες και individuals και επίσης αξιωμάτων όπως subclass axioms, disjoint axioms, και equivalent class axioms. Η DIG γλώσσα περιέχει δομικά στοιχεία που επιτρέπει στους clients να «πούν» σε εναν reasoner σχετικά με μια οντολογία (να περιγράψουν μια οντολογία στον reasoner) και επίσης να «ρωτήσουν» εναν reasoner σχετικά με το τι εχει γίνει inferred όπως subclass relationships, type relationships.Ενα παράδειγμα DIG γλώσσας ακολουθεί

```
<?xml version="1.0" encoding="UTF-8"?>

<tells uri="" xmlns="http://dl.kr.org/dig/2003/02/lang"

    xmlns:xs="http://www.w3.org/2001/XMLSchema"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://potato.cs.man.ac.uk/dig/level0/dig.x
sd">

    <equalc>

        <catom name="owl:Thing"/>

        <top/>

    </equalc>

    <defconcept name="ParmezanTopping"/>

    <equalc>

        <catom name="ParmezanTopping"/>

        <and>

            <catom name="CheeseTopping"/>

            <some>

                <ratom name="hasSpicyness"/>

                <catom name="MildSpiciness"/>

            </some>

        </and>

    </equalc>

    <disjoint>

        <catom name="ParmezanTopping"/>

        <catom name="MozzarellaTopping"/>

    </disjoint>

    <disjoint>       .
```

Το πλεονέκτημα του DIG είναι οτι οι εφαρμογές μπορούν να επικοινωνούν με εναν οποιονδήποτε DIG συμβατό reasoner χωρίς να χρειάζεται να ξέρει εξειδικευμένες λεπτομέρειες του reasoner ή τα reasoner interactions protocols.Αυτό σημαίνει οτι ειναι δυνατόν να κάνουμε plug-in οποιονδήποτε DIG συμβατό reasoner σε μια DIG-aware εφαρμογή. Για παράδειγμα, RACER, FaCT++ η οποιοσήποτε άλλος DIG συμβατός reasoner μπορεί να χρησιμοποιηθεί με την Protégé-OWL.

Η πλήρης εγκατάσταση του Protégé έρχεται με τρείς reasoner υλοποιήσεις:

- **`DefaultProtegeDIGReasoner`** : παρέχει μια σύνδεση σε εναν εξωτερικό DIG compliant reasoner
- **`ProtegePelletJenaReasoner`**: παρέχει μια απευθείας σύνδεση στον Pellet reasoner που προσπελάζεται μέσω του Jena API
- **`ProtegePelletOWLAPIReasoner`**: παρέχει μια απευθείας σύνδεση στον Pellet reasoner που προσπελάζεται μέσω του OWL-API

Το Reasoning API

Τα επόμενα μέρη παρέχουν ενα περίγραμμα για την χρήση του Protégé-OWL reasoning API.Τα παραδείγματα χρησιμοποιούν μια οντολογία που περιγράφει πίτσες.Αυτή η οντολογία στηρίζεται στην οντολογία πίτσας που εχει κατασκευαστεί στο διάσημο Protégé-OWL tutorial, που γράφτηκε απο τον Matthew Horridge απο το University of Manchester.Φυσικά αντι για αυτήν την οντολογία μπορούμε να χρησιμοποιήσουμε την δικιά μας της έκδοσης άδειας οδήγησης δηλαδή.

Το πακέτο inference

Το reasoning API υπάρχει στο πακέτο edu.stanford.smi.protegex.owl.inference. Οι κεντρικές κλάσεις που θα χρησιμοποιηθούν ειναι ο ReasonerManager(που χρησιμοποιείται για να αποκτήσει εναν reasoner) και ο ProtegeOWLReasoner (ενα interface στο external DIG reasoner).

Ο Reasoner Manager

Γενικά το πρώτο βήμα οταν χρησιμοποιούμε το reasoning API ειναι ειναι να αποκτήσουμε ενα στιγμιότυπο του ProtegeOWLReasoner για ενα OWL μοντέλο.Αυτο το στιγμιότυπο του reasoner μπορεί μετά να χρησιμοποιηθεί για να αποκτήσουμε inferred πληροφορία σχετικά με το μοντέλο όπως inferred superclasses, inferred equivalent classes και inferred types για individuals.Το ProtegeOWLReasoner διαχειρίζεται την επικοινωνία με τον εξωτερικό DIG reasoner σιγουρεύοντας οτι πάντοτε συγχρονίζεται κανονικά με το εσωτερικό Protégé-OWL μοντέλο.

Για να πάρουμε ενα στιγμιότυπο του ProtegeOWLReasoner για ενα OWL μοντέλο, ο ReasonerManager, που βρίσκεται στο πακέτο edu.stanford.smi.protegex.owl.inference.protegeowl πρέπει να χρησιμοποιηθεί. Ο ReasonerManager ειναι μια κλάση singleton που το στιγμιότυπό της μπορεί να αποκτηθεί χρησιμοποιώντας την στατική μέθοδο getInstance().

Η getReasoner`(OWLModel kb)` μέθοδος του ReasonerManager μπορεί να χρησιμοποιηθεί για να αποκτήσουμε εναν reasoner για το συγκεκριμένο

OWLModel.Αυτή η μέθοδος θα επιστρέψει το ιδιο στιγμιότυπο του ProtegeOWLReasoner για ενα δοσμένο μοντέλο κάθε φορά που καλείται.

Μια εναλλακτική μεθοδος `getReasoner(OWLModel kb, boolean createNew)` μπορεί να χρησιμοποιηθεί για να αποκτήσουμε εναν reasoner για μια βάση γνώσης. Ο παρακάτω κώδικας δείχνει πως μπορούμε να αποκτήσουμε εναν ProtegeOWLReasoner για ενα OWLModel.

```
// Get the reasoner manager and obtain a reasoner for the OWL
model.

ReasonerManager reasonerManager = ReasonerManager.getInstance();

ProtegeOWLReasoner reasoner = reasonerManager.getReasoner(model);
```

Setting και Checking την σύνδεση του Reasoner

Η επικοινωνία με εναν εξωτερικό DIG συμβατό reasoner ειναι εφικτή πάνω απο το HTTP έτσι το URL του εξωτερικού DIG συμβατού reasoner πρέπει να δοθεί.Αυτό γίνεται με την setReasonerURL(String url) μέθοδο της πάνω στο ProtegeOWLReasoner.Εχοντας θέσει το URL η σύνδεση μπορεί να ελεγχθεί με την isConnected() μέθοδο.Παράδειγμα ειναι ο ακόλουθος κώδικας:

```
// Set the reasoner URL (using the default URL for Racer here) and
test the connection.

reasoner.setURL(http://localhost:8080);

if (reasoner.isConnected()) {

    // Get the reasoner identity - this contains information

    // about the reasoner, such as it's name and version,

    // and the tell and ask operations that it supports.

    DIGReasonerIdentity reasonerIdentity = reasoner.getIdentity();

    System.out.println("Connected to " +
reasonerIdentity.getName());

}
```

Ρωτώντας τον Reasoner για inferred πληροφορία

Αφού η σύνδεση με τον reasoner εχει εγκαθιδρυθεί, ο reasoner μπορεί να ερωτηθεί για πληροφορία σχετικά με την οντολογία.Το `ProtegeOWLReasoner` interface περιέχει αρκετές μεθόδους για να αποκτήσουμε inferred πληροφορία

σχετικά με κλάσεις και individuals.Για παράδειγμα υπάρχουν μέθοδοι να πάρουμε τις inferred superclasses και subclasses μιας δοσμένης κλάσης.

Το παράδειγμα παρακάτω δείχνει πώς μπορούμε να πάρουμε τα inferred subclasses για μια ειδική κλάση στην περίπτωσή μας την VegetarianPizza, που ειναι μια ονομαστική κλάση της Pizza οντολογίας.Πρώτα οι asserted named subclasses της VegetarianPizza λαμβάνονται μέσω του OWLModel χρησιμοποιώντας την μέθοδο `vegetarianPizza.getNamedSubclasses()`.Μετά οι **inferred** subclasses της `VegetarianPizza` λαμβάνονται ρωτώντας τον reasoner, και μετά τυπώνονται.

```
// Get the VegetarianPizza OWLNamedClass from the OWLModel

OWLNamedClass vegetarianPizza =
model.getOWLNamedClass("VegetarianPizza");

if (vegetarianPizza != null) {

    // Get the number of asserted subclasses of VegetarianPizza

    Collection assertedSubclasses =
vegetarianPizza.getNamedSubclasses();

    System.out.println("Number of asserted VegetarianPizzas: " +
assertedSubclasses.size());



    // Now get the inferred subclasses of VegetarianPizza

    Collection inferredSubclasses =
reasoner.getSubclasses(vegetarianPizza, null);

    System.out.println("Number of inferred VegetarianPizzas: " +
inferredSubclasses.size());



    // Print out the inferred subclasses

    System.out.println("VegetarianPizzas:");

    for (Iterator it = inferredSubclasses.iterator();
it.hasNext();) {

        OWLNamedClass curClass = (OWLNamedClass) it.next();

        System.out.println(curClass.getName());

    }
```

```
}
```

Ενημερώνοντας το Protégé-OWL με inferred πληροφορία

Μερικές από τις μεθόδους στον ProtegeOWLReasoner ρωτάνε τον external DIG reasoner για να αποκτήσουν inferred πληροφορία και μετά εισάγουν αυτήν την πληροφορία στο OWLModel.Αυτό ειναι χρήσιμο για μεγάλα μοντέλα γιατί αντί να ρωτάμε τον reasoner για πληροφορία για κάθε κλάση στην οντολογία manually (κάθε φορά έχοντας το κόστος μιας http request/response) ενα bulk ερώτημα πραγματοποιείται ενα για consistency checking ενα για inferred superclass checking, και ενα για equivalent class checking.

Αφού εχουν εκτελεστεί αυτά τα bulk ερωτήματα το OWLModel ενημερώνεται που σημαίνει οτι το μοντέλο μπορεί να ελεγχθεί «offline» απο τον reasoner.Ενα παράδειγμα απο αυτό δείχνεται παρακάτω.Η `classifyTaxonomy` μέθοδος bulk queries τον reasoner γαι consistency, inferred superclasses, και equivalent classes για κάθε κλαση στην οντολογία (χρησιμοποιώντας μόνο 3 HTTP request/responses) και μετά βάζει την πληροφορία στο Protégé-OWL μοντέλο.

```
// We can classify the whole ontology, which will put the

// inferred class hierarchy information directly into the Protégé-
OWL model.

System.out.println("Classifying taxonomy...");

reasoner.classifyTaxonomy(null);

System.out.println("...Classified taxonomy!");



// We can then use the methods on OWLNamedClass for getting
inferred

// information, without having to make separate queries to the
external

// DIG reasoner.

System.out.println("Inferred subclasses of VegetarianPizza:");

inferredSubclasses = vegetarianPizza.getInferredSubclasses();

for (Iterator it = inferredSubclasses.iterator(); it.hasNext();) {

    OWLNamedClass curClass = (OWLNamedClass) it.next();

    System.out.println(curClass.getName());
```

```
}
```

Θα δούμε πώς μπορούμε να πάρουμε ενα Pellet reasoner στιγμιότυπο(προσπελάσιμο μέσω Jena).

Η `ProtegePelletJenaReasoner` υλοποίηση μετατρέπει ενα Protege-OWL μοντέλο σε ενα Jena μοντέλο και μετα χρησιμοποιεί την υπάρχουσα Pellet reasoner σύνδεση διαθέσιμη στην Jena για το inference.

Το επόμενο κομμάτι κώδικα δείχνει πώς να πάρουμε ενα Pellet reasoner στιγμιότυπο που προσπελάζεται μέσω Jena:

```
public ProtegeReasoner createPelletJenaReasoner(OWLModel owlModel) {

        // Get the reasoner manager instance
        ReasonerManager reasonerManager =
ReasonerManager.getInstance();

        //Get an instance of the Protege Pellet reasoner
        ProtegeReasoner reasoner =
reasonerManager.createProtegeReasoner(owlModel,
ProtegePelletJenaReasoner.class);

        return reasoner;
}
```

Θα δούμε πώς μπορούμε να πάρουμε ενα Pellet reasoner στιγμιότυπο(προσπελάσιμο μέσω OWL-API).

Η `ProtegePelletOWLAPIReasoner` υλοποίηση μετατρέπει ενα Protege-OWL μοντέλο σε ενα OWL- μοντέλο και μετα χρησιμοποιεί την υπάρχουσα Pellet reasoner σύνδεση διαθέσιμη στο OWL-API για το inference.

Το επόμενο κομμάτι κώδικα δείχνει πώς να πάρουμε ενα Pellet reasoner στιγμιότυπο που προσπελάζεται μέσω OWL-API:

```
public ProtegeReasoner createPelletOWLAPIReasoner(OWLModel owlModel)
{

        // Get the reasoner manager instance
        ReasonerManager reasonerManager =
ReasonerManager.getInstance();

        //Get an instance of the Protege Pellet reasoner
        ProtegeReasoner reasoner =
reasonerManager.createProtegeReasoner(owlModel,
ProtegePelletOWLAPIReasoner.class);

        return reasoner;
}
```

Γλώσες κανόνων

Εισαγωγή

Οι γλώσσες αναπαράστασης γνώσης, όπως η RDF Schema και OWL, είναι σχεδιασμένες για να καθορίζουν περιγραφές των πεδίων εφαρμογών.Συνήθως προσφέρουν κατασκευαστές(δομές) για να περιγράφουν κλάσεις, ιδιότητες, και σχέσεις όπως επίσης και δομές για να αποτυπώνουν περιορισμούς για κλάσεις και ιδιότητες και να ορίζουν σύνθετες κλάσεις.Σε αντίθεση οι γλώσσες κανόνων ειναι σχεδιασμένες να καθορίζουν κανόνες μετασχηματισμών δεδομένων που ορίζουν πώς να συμθέτουν νέα γεγονότα από αυτά που είναι αποθηκευμένα στη βάση γνώσης.Θα εισάγουμε τρείς γλώσσες κανόνων , Datalog, την Rule Markup Language(RuleML) και την Semantic Web Rule Language(SWRL).
Αυτές οι γλώσσες κανόνων βασίζονται σε αρχές που προέρχονται από τον λογικό προγραμματισμό από τον οποίο η Prolog ειναι η αρχέτυπη υλοποίηση.
Η Datalog ειναι ενα συντακτικά περιορισμένο υποσύνολο του λογικού προγραμματισμού όπου τα σύμβολα συναρτήσεων δεν επιτρέπονται.Η datalog αρχικά ορίστηκε σαν μια γλώσσα ερωτήσεων και κανόνων για βάσεις γνώσης.Ο υπολογισμός ερωτημάτων στην Datalog συνήθως ακολουθεί bottom-up στρατηγικές οι οποίες λειτουργούν αποδοτικά ακόμα και για μεγάλες βάσεις δεδομένων.
Η RuleML ειναι μια markup γλώσσα για δημοσιοποίηση και διαμοιρασμό βάσεων γνώσεων στον παγκόσμιο ιστό, χτισμένη πάνω απο την Datalog.Η γλώσσα αναπτύσσεται κάτω απο την Rule markup Initiative.
Η SWRL συνδιάζει την RuleML με την OWL και περιέχει ενα αφηρημένο συντακτικό υψηλού επιπέδου,μια σημασιολογία model-theoretic, και ενα XML συντακτικό.Η SWRL δίνει την δυνατότητα σε Datalog-like κανόνες να συνδιαστούν με μια OWL βάση γνώσης.

DATALOG

Η datalog ειναι ενα συντακτικά περιορισμένο υποσύνολο του λογικού προγραμματισμού, που αρχικά οριστηκε σαν μια γλώσσα ερωτημάτων και κανόνων για deductive βάσεις δεδομένων.Σε αυτό το κομμάτι εισάγουμε τις βασικές έννοιες της datalog σαν προετοιμασία για τα επόμενα κομμάτια.
Ενα αλφάβητο Datalog A ειναι ενα σύνολο απο κατηγορηματικά σύμβολα, σταθερές και μεταβλητές.Η πληθικότητα του κατηγορηματικού συμβόλου (arity) ειναι ενας μη αρνητικός ακέραιος που δηλώνει τον αριθμό των ορισμάτων του κατηγορήματος.Σε αντίθεση με την Prolog τα σύμβολα συναρτήσεων δεν επιτρέπονται στην βασική Datalog.
Για να απλουστεύσουμε τον συμβολισμό δεν παραθέτουμε επακριβώς τις μεταβλητές και τις σταθερές ενός αλφαβήτου,υιοθετώντας την παραδοχή οτι οι μεταβλητές δηλώνονται σαν συμβολοσειρές μικρών γραμμάτων και οι σταθερές εγκλείονται μεσα σε διπλούς απόστροφους.Με αυτήν την παραδοχή ενα αλφάβητο ελλατώνεται σε ενα σύνολο κατηγορηματικών συμβόλων.Ο πίνακας 1 δείχνει ένα αλφάβητο της BOOKS βάσης γνώσης την οποία χρησιμοποιούμε στα παραδείγματά μας.Ολοι οι ορισμοί που ακολουθούν ειναι σχετικοί με το δοσμένο αλφάβητο.
Ενας όρος(term) ειναι μια μεταβλητή ή μια σταθερή.Σημειώστε οτι επειδή η Datalog δεν περιέχει συναρτησιακά σύμβολα δεν επιτρέπει πιο σύνθετους όρους.

Predicate	Arity	Intended Interpretation
Book	3	Book(t,c,y) indicates a book with title t, country of publication c, and year of publication y
hasAuthor	2	
references	2	references(t,r) indicates that a book with title t references another book with title r
trReferences	2	trReferences(t,r) indicates that a book with title t transitively references another book with title r
UKBook	1	UKBook(t) indicates a book with title t that has as country of publication «United Kingdom"

Πίνακας 1. Το αλφάβητο της BOOKS βάση γνώσης

Ενα άτομο (atom) ειναι μια έκφραση της μορφής $P(t_1,...,t_n)$;όπου P ειναι ενα ν-οστό κατηγορηματικό σύμβολο και $t1,...,tn$ ειναι όροι(terms).Παράδειγμα ατόμων στο αλφάβητο BOOKS ειναι(“Semantic web”, “Brazil” και «2005» ειναι σταθερές και title και country ειναι μεταβλητές):

```
1. Book(“Semantic Web",”Brazil”,”2005")
2. Book(title,country,”2005")
```

Ενα clause ειναι ειτε ενα γεγονός(fact) ή ενας κανόνας.Ενα γεγονός ειναι μια έκφραση της μορφής

B.

Όπου B ειναι ένα άτομο δίχως μεταβλητές.Ενας κανόνας ειναι μια έκφραση της μορφής

C ←A1,...,Am.

Όπου $A_1,...,A_m$ και C ειναι άτομα. Τα $A_1,...,A_m$ καλούνται σαν η λίστα andecedents και c καλείται το συμπέρασμα ή κεφαλή (consequent or head) του κανόνα. Οι μεταβλητές θεωρούνται σαν καθολικώς ποσοδεικτούμενες,με την εμβέλειά τους περιορισμένες στον κανόνα.Οι μεταβλητές ειναι περιορισμένες ως προς το πεδίο(range restricted) με την έννοια οτι μόνο μεταβλητές που υπάρχουν στο antecedent ενός κανόνα μπορεί να υπάρξει στο consequent του κανόνα.

Ενα γεγονός πρέπει πάντοτε να ισχύει, ενω ενας κανόνας δηλώνει οτι εάν το antecedent ισχύει τοτε το consequent πρέπει να ισχύει.

Μια datalog βάση γνώσης ειναι ενα σύνολο από clauses. Σε αντίθεση με την Prolog η διάταξη των clauses δεν εχει σημασία στην Datalog.Ενας κανόνας ειναι αναδρομικός εαν το κατηγορηματικό σύμβολο που υπάρχει στο consequent επίσης υπάρχει στη λίστα των antecedents. Ενα σύνολο $S_0,...,S_{n-1}$ *κανόνων* ειναι αμοιβαίως αναδρομικοί εαν και μόνον εάν υπάρχουν κατηγορηματικά σύμβολα $P_0,...,P_{n-1}$ τέτοια ώστε τα P_i *να υπάρχουν στο consequent* του S_i *και* στη λίστα των antecedents των *Si+1*, για *i ανήκει* [0,*n-1*].

Αμοιβαίως αναδρομικοί κανόνες αυξάνουν σημαντικά την εκφραστική δύναμη της Datalog, αλλά δημιουργούν δυσκολίες στον ορισμό της σημασιολογίας των datalog βάσεων γνώσεων, και στην κατασκευή datalog επεξεργαστών.

Σε μια συγκεκριμένη στιγμή του χρόνου η BOOKS βάση δεδομένων μπορεί να αποτελείται απο το ακόλουθο σύνολο clauses.

1. Book("The Description Logic Handbook",
"United Kingdom", "2003").
2. Book("Principia Mathematica", "United Kingdom", "1910").
3. Book("Semantic Web","Brazil","2005").
4. hasAuthor("The Description Logic Handbook",
"Franz Baader").
5. hasAuthor("The Description Logic Handbook",
"Diego Calvanese").
6. hasAuthor("Principia Mathematica", "Bertrand Russell").
7. hasAuthor(«Principia Mathematica", «Alfred Whitehead").
8. hasAuthor(«Semantic Web", «Karin Breitman").
9. references(«Semantic Web",
«The Description Logic Handbook").
10. references(«The Description Logic Handbook",
«Principia Mathematica").
11. trReferences(x,y) ← references(x,y).
12. trReferences(x,y) ← references(x,z), trReferences(z,y).
13. UKBook(x) ← Book(x, «United Kingdom", y).
Note that Clauses 1 to 10 are facts. Clauses 11 and 12 capture what it

Σημειώστε οτι τα clauses 1 εως 10 ειναι γεγονότα(facts).Τα clauses 11 και 12 δίνουν την έννοια της μεταβατικής αναφοράς(transitive reference).Το clause 12 ειναι ενας αναδρομικός κανόνας. Τέλος το clause 13 ειναι ενας μη αναδρομικός κανόνας που δίνει την έννοια του τι σημαίνει ενα βιβλιο να έχει εκδοθεί στην μ.Βρεττανία(UK).

Ενας επεξεργαστής Datalog θα πρέπει να ειναι ικανός να συμπεράνει τα ακόλουθα γεγονότα απο την παραπάνω βάση γνώσης.

```
14. trReferences(«Semantic Web",
«The Description Logic Handbook").
15. trReferences(«The Description Logic Handbook",
«Principia Mathematica").
16. trReferences(«Semantic Web", «Principia Mathematica").
17. UKBook(«The Description Logic Handbook").
18. UKBook(«Principia Mathematica").
```

Προσέξτε οτι για να συμπεράνει κάποιος το clause 16, ο datalog επεξεργαστής θα έπρεπε να χρησιμοποιήσει το clause 12 όπως και τα clauses 9,10 και 11.

RuleML

Η Rule Markup Language (RuleML) βασκά παρέχει ένα XML συντακτικό για Datalog clauses.Σε αυτό το κομμάτι εισάγουμε το XML συντακτικό της RuleML με τη βοήθεια παραδειγμάτων βασισμένα στο BOOKS αλφάβητο του πίνακα 1.

Κάθε παράδειγμα εκφράζεται με τη RuleML XML συντακτικό και την Datalog.
Ενα άτομο (atom) στην RuleML εκφράζεται με τη βοήθεια των tags <atom>, <rel>, <var>, και <ind> όπως στα επόμενα παραδείγματα.

```
1. <Atom>
2. <Rel>Book</Rel>
3. <Ind>Semantic Web</Ind>
4. <Ind>Brazil</Ind>
5. <Ind>2005</Ind>
6. </Atom>
```

Datalog συμβολισμός: Book(«Semantic Web",»Brazil»,»2005")

```
7. <Atom>
8. <Rel>Book</Rel>
9. <Var>title</Var>
10. <Var>publ</Var>
11. <Ind>2005</Ind>
12. </Atom>
```

Datalog συμβολισμός: Book(title,publ,»2005")

Οι τιμές της <rel> tag στις γραμμές 2 και 8 δηλώνουν ένα κατηγορηματικό σύμβολο.Οι τιμές του <ind> tag στις γραμμές 3,4,5 και 11 δηλώνουν σταθερές.Και οι τιμές του tag <Var> στις γραμμές 9 και 10 δηλώνουν οτι title και publ πρέπει να καταληφθούν ώς μεταβλητές.Απο δώ γίνεται κατανοητό οτι οι μεταβλητές και οι σταθερές δεν χρειάζεται να ακολουθούν καμιά συντακτική σύμβαση γιατί ακριβώς υποδηλώνονται με τη βοήθεια των tags <Ind> και <Var>.

Ενας κανόνας στην RuleML εκφράζεται με τη βοήθεια των tags <Implies>,<head> και <body> όπως στο:

```
1. <Implies>
2. <head>
3.      <Atom>
4.      <Rel>UKBook</Rel>
5.      <Var>x</Var>
6.      </Atom>
7. </head>
8. <body>
9.      <Atom>
10.     <Rel>Book</Rel>
11.     <Var>x</Var>
12.     <Ind>United Kingdom</Ind>
13.     <Var>y</Var>
14.     </Atom>
15. </body>
16. </Implies>
```

Datalog συμβολισμός: UKBook(x) ← Book(x, «United Kingdom", y).

Οι κανόνες μπορεί να είναι πιό πολύπλοκοι και να περιλαμβάνουν conjunctions που δηλώνονται με το tag <And> που είναι φωλιασμένο μέσα στο <body> όπως φαίνεται στο επόμενο παράδειγμα:

```
1. <Implies>
2. <head>
3.      <Atom>
4.      <Rel>trReferences</Rel>
5.      <Var>x</Var>
6.      <Var>y</Var>
7.      </Atom>
8. </head>
9. <body>
10. <And>
11.     <Atom>
12.     <Rel>references</Rel>
13.     <Var>x</Var>
14.     <Var>z</Var>
15.     </Atom>
16. <Atom>
17.     <Rel>trReferences</Rel>
18.     <Var>z</Var>
19.     <Var>y</Var>
20. </Atom>
21. </And>
22. </body>
23. </Implies>
```

Datalog συμβολισμός: trReferences(x,y) ← references(x,z),trReferences(z,y).

Το επόμενο RuleML έγγραφο περιέχει την πλήρη BOOKS βάση γνώσης και το οποίο ελέγχθηκε με τον W3C validator για XML schema.

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <RuleML
3. xmlns="http://www.ruleml.org/0.91/xsd"
4. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xsi:schemaLocation="http://www.ruleml.org/0.91/xsd
6. http://www.ruleml.org/0.91/xsd/datalog.xsd">
7. <Assert mapClosure="universal">
8. <Atom>
9. <Rel>Book</Rel>
10. <Ind>The Description Logic Handbook</Ind>
11. <Ind>United Kingdom</Ind>
12. <Ind>2003</Ind>
13. </Atom>
14. <Atom>
15. <Rel>Book</Rel>
16. <Ind>Principia Mathematica</Ind>
```

17. <Ind>United Kingdom</Ind>
18. <Ind>1910</Ind>
19. </Atom>
20. <Atom>
21. <Rel>Book</Rel>
22. <Ind>Semantic Web</Ind>
23. <Ind>Brazil</Ind>
24. <Ind>2005</Ind>
25. </Atom>
26. <Atom>
27. <Rel>hasAuthor</Rel>
28. <Ind>The Description Logic Handbook</Ind>
29. <Ind>Franz Baader</Ind>
30. </Atom>
31. <Atom>
32. <Rel>hasAuthor</Rel>
33. <Ind>The Description Logic Handbook</Ind>
34. <Ind>Diego Calvanese</Ind>
35. </Atom>
36. <Atom>
37. <Rel>hasAuthor</Rel>
38. <Ind>Principia Mathematica</Ind>
39. <Ind>Bertrand Russell</Ind>
40. </Atom>
41. <Atom>
42. <Rel>hasAuthor</Rel>
43. <Ind>Principia Mathematica</Ind>
44. <Ind>Alfred Whitehead</Ind>
45. </Atom>
46. <Atom>
47. <Rel>hasAuthor</Rel>
48. <Ind>Semantic Web</Ind>
49. <Ind>Karin Breitman</Ind>
50. </Atom>
51. <Atom>
52. <Rel>references</Rel>
53. <Ind>Semantic Web</Ind>
54. <Ind>The Description Logic Handbook</Ind>
55. </Atom>
56. <Atom>
57. <Rel>references</Rel>
58. <Ind>The Description Logic Handbook</Ind>
59. <Ind>Principia Mathematica</Ind>
60. </Atom>
61. **<Implies>**
62. <head>
63. <Atom>
64. <Rel>trReferences</Rel>
65. <Var>x</Var>
66. <Var>y</Var>
67. </Atom>
68. </head>

```
69. <body>
70. <Atom>
71. <Rel>References</Rel>
72. <Var>x</Var>
73. <Var>y</Var>
74. </Atom>
75. </body>
76. </Implies>
77. <Implies>
78. <head>
79. <Atom>
80. <Rel>trReferences</Rel>
81. <Var>x</Var>
82. <Var>y</Var>
83. </Atom>
84. </head>
85. <body>
86. <And>
87. <Atom>
88. <Rel>references</Rel>
89. <Var>x</Var>
90. <Var>z</Var>
91. </Atom>
92. <Atom>
93. <Rel>trReferences</Rel>
94. <Var>z</Var>
95. <Var>y</Var>
96. </Atom>
97. </And>
98. </body>
99. </Implies>
100. <Implies>
101. <head>
102. <Atom>
103. <Rel>UKBook</Rel>
104. <Var>x</Var>
105. </Atom>
106. </head>
107. <body>
108. <Atom>
109. <Rel>Book</Rel>
110. <Var>x</Var>
111. <Ind>United Kingdom</Ind>
112. <Var>y</Var>
113. </Atom>
114. </body>
115. </Implies>
116. </Assert>
117. </RuleML>
```

SWRL

Η Semantic Web Rule Language(SWRL) επεκτείνει το σύνολο των OWL αξιωμάτων για να περιλάβει Datalog-like clauses.Σε αυτό το κομμάτι παρουσιάζουμε σύντομα ενα RDF συγκεκριμένο συντακτικό για την SWRL, χρησιμοποιώντας τα ιδια παραδείγματα με τα προηγούμενα κομμάτια αλλά τώρα τα θεωρούμε σαν μια OWL οντολογία.

Αφού η SWRL ειναι μια απευθείας επέκταση της OWL , πρέπει να μοντελοποιήσουμε την βάση γνώσης σαν μια OWL οντολογία, με τις κλάσεις και τις ιδιότητές τους στον πίνακα 2 και τα namespaces τους στον πίνακα 3.

Resource	**OWL Object**	**Intended Interpretation**
`Book`	Simple Class	the class of all books
`Title`	datatype property with domain `Book` and range `string`	maps books into `their` titles
`CountryOfPub`	datatype property with domain `Book` and range `string`	maps books into their country of publication
`YearOfPub`	datatype property with domain `Book` and range `gYear`	maps books into their year of publication
`has Author`	datatype property with domain `Book` and range `string`	maps books into their author's names
`references`	object property with domain `Book` and range `Book`	maps books into the books they reference
`trReferences`	object property with domain `Book` and range `Book`	transitive closure of `references`
`UKBook`	Complex Class	the class of all books published in the UK

Πίνακας 2. Concepts της BOOKS οντολογίας

Namespace	Abbreviation
the SWRL vocabulary `http://www.w3.org/2003/11/swrl#`	`swrl`
the XML Schema vocabulary `http://www.w3.org/2001/XMLSchema`	`xsd`
the BOOKS vocabulary `http://books.org/terms`	`bk`

Πίνακας 3.SWRL XML concrete syntax namespaces

Χρησιμοποιώντας Datalog-like συμβολισμό,ενα άτομο(atom) στην SWRL ειναι μια εκφραση απο μια από τις μορφές:

C(x)	C ειναι μια OWL περιγραφή κλάσης ή πεδίο τιμών
P(x,y)	P ειναι μια OWL ιδιότητα
sameAs(x,y)	sameAs ανήκει στο OWL λεξιλόγιο
differentFrom(x,y)	differentFrom ανήκει στο OWL λεξιλόγιο
builtIn(r,x,...)	r ειναι μια built-in σχέση

όπου χ και y ειναι ή μεταβλητές, OWL individuals ή OWL τιμές δεδομένων(data values).

Γι' αυτό οι ορισμοί των ατόμων(atoms) στην SWRL διαφέρει απ'αυτήν στην RuleML σε δύο σημεία.Πρώτα τα άτομα περιλαμβάνουν μόνο μοναδιαία κατηγορήματα (για OWL περιγραφές κλάσεων ή πεδία τιμών) ή δυαδικά κατηγορήματα(για OWL ιδιότητες,sameAs και differentFrom), ενώ στην RuleML τα κατηγορήματα μπορεί να έχουνε οποιαδήποτε arity.Δεύτερο οι OWL περιγραφές κλάσεων μπορούν να ιδωθούν σαν μοναδιαία κατηγορήματα εξ'ορισμού , μια έννοια που δεν υποστηρίζεται στην RuleML.

Ατυπα ενα άτομο C(x) ισχύει εαν και μόνο εάν το x ειναι ενα στιγμιότυπο της περιγραφής κλάσης ή του πεδίου τιμών C. Ενα άτομο P(x,y) ισχύει εαν και μόνο εάν το x συνδέεται με το y με την ιδιότητα P.Ενα άτομο sameAs(x,y) ισχύει εαν και μόνο εάν το x διερμηνεύεται σαν το ίδιο αντικείμενο με το y.Ενα άτομο differentFrom(x,y) ισχύει εαν και μόνο εάν το x και το y διερμηνεύονται σαν διαφορετικά αντικείμενα και builtIn(r,x,...) ισχύει εαν και μόνο εάν η built-in σχέση r ισχύει για τις διερμηνείες των ορισμάτων.

Σαν παράδειγμα θεωρήστε το Datalog άτομο

Book("Semantic Web","Brazil","2005").

Για να αναπαραστήσουμε αυτό το άτομο σε SWRL RDF συγκεκριμένο συντακτικό, πρώτα τα ξαναγράφουμε σαν τέσσερα διαφορετικά άτομα που ταιριάζουν με τους ορισμούς OWL της BOOKS βάσης γνώσης.

1. Book(B3).
2. Title(B3,"Semantic Web").
3. CountryOfPub(B3,"Brazil").
4. YearOfPub(B3,"2005").

Μετα τα μεταφράζουμε αυτά τα τέσσερα άτομα σε OWL παίρνοντας

```
1. <Book rdf:about="B3">
2. <Title>Semantic Web</Title>
3. <CountryOfPub>Brazil</CountryOfPub>
4. <YearOfPub>2005</YearOfPub>
1. </Book>
```

Η SWRL επιτρέπει μια σύνθετη έννοια να οριστεί σαν μια σύνθετη κλάσητης λεγόμενης OWL οντολογίας, στην οποία περίπτωση προσφέρει μια πιό πλούσια λύση απο ότι η RuleML.Για παράδειγμα αντί για χρησιμοποίηση ενός κανόνα μπορούμε να συμπεριλάβουμε τον ορισμό της σύνθετης έννοιας UKBooks στην BOOKS οντολογία όπως ακολούθως

```
1. <owl:Class rdf:about="UKBook">
2.      <owl:intersectionOf rdf:parseType="Collection">
3.              <owl:Class rdf:about="Book"/>
4.                      <owl:Restriction>
5.                      <owl:onProperty rdf:resource="CountryOfPub"/>
6.                      <owl:hasValue rdf:datatype="&xsd;string">
7.                      United Kingdom
8.                      </owl:hasValue>
9.                      </owl:Restriction>
10.     </owl:intersectionOf>
1. </owl:Class>
```

Ενας κανόνας στην SWRL μπορεί να έχει μια λίστα απο άτομα σαν consequent. Αλλά αυτό δεν αυξάνει την εκφραστική δύναμη της γλώσσας γιατί οποιοσδήποτε τέτοιος κανόνας μπορεί ισοδύναμα να αντικατασταθεί από ενα σύνολο κανόνων με μόνο ενα άτομο σαν consequent.Για παράδειγμα ο επόμενος Datalog κανόνας

$$trReferences(x,z) \leftarrow references(x,y), trReferences(y,z).$$

γράφεται σε SWRL RDF concrete syntax σαν

```
1. <swrl:Imp rdf:about="Rule-1">
2. <swrl:head>
3. <swrl:AtomList>
4. <rdf:first>
5.              <swrl:IndividualPropertyAtom>
6.              <swrl:propertyPredicate rdf:resource="trReferences"/>
7.              <swrl:argument1>
8.                      <swrl:Variable rdf:about="x"/>
9.              </swrl:argument1>
10.             <swrl:argument2>
11.                     <swrl:Variable rdf:about="z"/>
12.             </swrl:argument2>
13.             </swrl:IndividualPropertyAtom>
14. </rdf:first>
15. <rdf:rest rdf:resource="&rdf;nil"/>
16. </swrl:AtomList>
17. </swrl:head>
18. <swrl:body>
19.             <swrl:AtomList>
20.             <rdf:first>
21. <swrl:IndividualPropertyAtom>
```

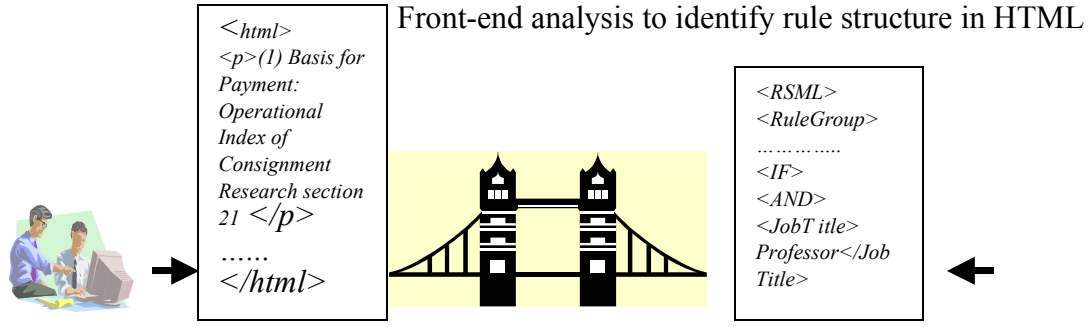
```
22. <swrl:propertyPredicate  rdf:resource="references"/>
23.          <swrl:argument1 rdf:resource="x"/>
24.          <swrl:argument2>
25.          <swrl:Variable rdf:about="y"/>
26.          </swrl:argument2>
27. </swrl:IndividualPropertyAtom>
28. </rdf:first>
29. <rdf:rest>
30.    <swrl:AtomList>
31.          <rdf:first>
32.          <swrl:IndividualPropertyAtom>
33.          <swrl:propertyPredicate rdf:resource="trReferences"/>
34.          <swrl:argument1 rdf:resource="y"/>
35.          <swrl:argument2 rdf:resource="z"/>
36.          </swrl:IndividualPropertyAtom>
37.          </rdf:first>
38.          <rdf:rest rdf:resource="&rdf;nil"/>
39.     </swrl:AtomList>
40. </rdf:rest>
41. </swrl:AtomList>
42. </swrl:body>
43. </swrl:Imp>
```

Extensible Rule Markup Language

RIML

RIML is a bridge between unstructured HTML and strudtured rule!

HTML **RIML** **RSML**



Identification of relevant aspect of rule structure including titles, variables and values

DTD

Τα DTD παρέχουν κρίσιμη πληροφορία που επιτρέπει στους επεξεργαστές XML να κάνει parsing τον κώδικα και να κάνει σίγουρο οτι περιέχει όλη την πληροφορία που χρειάζεται η εφαρμογή σε μια μορφή που η εφαρμογή ειναι έτοιμη να δεχτεί. Το DTD παρέχει εναν κρίσιμο σύνδεσμο μεταξύ των αρχείων δεδομένων που δίνονται στον XML επεξεργαστή και στα δεδομένα που μεταφέρονται απο τον XML επεξεργαστή στην εφαρμογή.Τα DTD βοηθάνε τους υπολογιστές να καταλαβαίνουν δομές που μπορεί να φαίνονται προφανείς στον άνθρωπο. Το επόμενο DTD κομμάτι ορίζει την top-level δομή του RIML εγγράφου.

<!ELEMENT RIML (Rule+)>
<!ATTLIST RIML Version CDATA #Required)>

<!ELEMENT Rule (RuleTitle, (variablei, valuei*)>*

<!ELEMENT RuleTitle(#PCDATA)>
<!ELEMENT value(#PCDATA)>
<!ELEMENT variable(#PCDATA)>

Στοιχεία και ιδιότητες της RIML

Αυτό το στοιχείο κουβαλάει μια υποχρεωτική ιδιότητα(Version) που δηλώνει την έκδοση του specification που αυτό το έγγραφο συμπλέει.Οι περισσότερες δηλώσεις στοιχείων θα περιέχουν λίστα απο στοιχεία, καθορίζοντας κανόνες για τους οποίους στοιχεία είναι υποχρεωτικά, την σειρά με την οποία θα εμφανίζονται, και πόσες φορές μπορούν να εμφανίζονται. Παράμετροι οντοτήτων(parameter entities) μπορεί να εμφανίζονται στη λίστα κάνοντας δυνατόν στους developers να δημιουργήσουν πολλαπλές όμοιες δομές. Ο πίνακας 1 δείχνει μερικά σύμβολα που παρέχουν κανόνες και χρησιμοποιούν στοιχεία.

Πίνακας 1

Symbol	Symbol Type	Description	Example	Example Notes
|	Vertical bar	Indicates or condition	thisone|thatone	Either thisone or thatone must appear
,	Comma	Requires appearance in specified sequence	thisone, thatone	thisone must appear, followed by thatone
?	Question Mark	Makes optional, but only one may appear	thisone?	thisone may appear
	No symbol	One, and only one, must appear	thisone	thisone must appear
*	Asterisk	Allows any number to appear, even zero	thisone*	thisone may be present; multiple appearance of thisone are

				acceptable
+	Plus sign	Requires at least one to appear, but may appear multiple times	thisone+	thisone must be present; multiple appearance of thisone are acceptable
()	Parentheses	Group elements	(thisone|thatone) whichone	Either thisone or thatone must appear, followed by whichone

Στοιχεία και ιδιότητες της RIML(1)

Το στοιχείο ρίζας ενός HTML/RIML εγγράφου ειναι το "RIML".Κάθε RIML έγγραφο περιέχει τουλάχιστον ενα απο "Rule".Στον πίνακα 2 φαίνεται η σημασία των στοιχείων και των ιδιοτήτων της RIML.

Πίνακας 2

Element/Attribute	Description
Rule	The element "Rule" envelops the body of rule. The presence of this element indicates that this is contains at least one variable, value, and operator, etc. At least one must be present within RIML.
RuleTitle	It is the title of the Rule. It can be left out should a specific rule be not included in rule group.
variable*i*	The literal element "variable" represents variable names, where i=1, 2,….., n.
value*j*	It is the literal element that represents value names, where j=1, 2,….., m.

Μετά το στοιχείο που ονομάστηκε ,ένας ορισμός ιδιότητας ή μια λίστα απο ορισμούς ιδιοτήτων μπορεί να ακολουθούν. Ενας ορισμός αποτελείται απο το όνομα της ιδιότητας , τον τύπο, και την default τιμή του. Τα ονόματα των ιδιοτήτων πρέπει να ακολουθούν τους ίδιους κανόνες όπως τα ονόματα των οντοτήτων και των στοιχείων. Οι τύποι των ιδιοτήτων δεν έχουν καμιά σχέση περίπου με τις δομές που έχουμε δεί ως τώρα και ορίζουν τα είδη δεδομένων που επιτρέπονται σε ιδιότητες όταν χρησιμποιούνται σε ένα element source.

Παράδειγμα

<html>
<title>KAIST Research Info.</title>

<RIML Version="1.0">

* Execution of Research Fund *

```
<Rule>
    <p>A. <RuleTitle>Payment of Consignment Research Fund</RuleTitle>
    </p>
     <p>(1) Basis for Payment: Operational Index of Consignment Research
  section 21 </p>
……
    <p>(3)Payment Scope</p>
        <p>(A) <value1>Professor</value1>
            <ul><li>Less than <value3>standard payment *1.5</value3> with
              regard to number of researches </ul>
     <p>* <variable2>Sponsored</variable2> project by <value2>Ministry of
          Science and Technology</value2> or < value2>Ministry of
          Information and Communication</value2> is exclude.</p>

        <p>(B) Student
            <ul type="circle">
                <li><value1>Bachelor Course</value1>: Less than
                <value3>$200</value3>
                <li><value1>Master Course</value1>: Less than
                <value3>$400</value3>
                <li><value1>Ph. D. Course<value1>: Less than  <value3>
                $600</value3>
                <li><value1>Greater equal 5 years at Ph. D.
                Course<value1> Less than <value3>$1,000</value3></ul>
</Rule>
</RIML>
</html>
```

RSML

DTD

Αυτό που χρειαζόμαστε τώρα ειναι η ενδιάμεση αναπαράσταση των κανόνων που μπορούν αυτόματα να μετασχηματιστούν σε δομημένους κανόνες και εύκολα να συσχετίσουν την RIML. Η RSML στην όλη XRML αρχιτεκτονική υποστηρίζει τον software agent περνώντας τον κανόνα που θα χρησιμοποιήσει η KBS(Knowledge Based System) για inference. Εκτελεί εναν σημαντικό ρόλο να συντηρεί την συνέπεια μεταξύ ιστοσελίδας και βάσης κανόνων με την ισχυρή δέσμευση με της RIML τύπους κανόνων.Για να το κάνει αυτό ο υπάρχον κανόνας πρέπει να εκφραστεί σε μορφή που να μπορεί να ανταλαχθεί στο Internet. Εδω προτείνουμε την RSML σαν ενα σχήμα που μπορεί να παρουσιάσει την δομικότητα του κανόνα στο Internet. Η RSML ορίζεται σαν μια XML που μπορεί να εκφράσει την δομή ενος condition-action τύπο κανόνα. Για να διατηρήσουμε την δομικότητα του κανόνα προτείνουμε ιεραρχικό DTD που ταιριάζει για την αναπαράσταση της RSML. Το παρακάτω DTD κομμάτι αναπαριστάνει την δομή των RSML κανόνων.

```
<!ELEMENT RSML (Rule+)>
   <!ATTLIST RSML Version CDATA #Required)>

<!ELEMENT Rule ( RuleTitle, (IF, THEN)*)>

<!ELEMENT RuleTitle (#PCDATA)>
<!ELEMENT IF (AND|OR)*>
<!ELEMENT THEN (AND|OR)*>
<!ELEMENT AND((AND|OR)*)>
<!ELEMENT OR (AND|OR)*)>
<!ELEMENT variable (#PCDATA)>
<!ELEMENT value (#PCDATA)>
```

Στοιχεία και ιδιότητες της RSML

Το νόημα και η ιεραρχία των στοιχείων και των ιδιοτήτων που ορίζονται στην RSML φαίνονται στον πίνακα 3

Πίνακας 3

Element/Attribute	Description
RSML	Root element of RSML
Rule	The element envelops the body of rule. At least one must be present within RSML. Rule encompasses elements such as RuleTitle, IF, and THEN
RuleTitle	It is the title of individual rule included in Rule group. It is the identifier of RSML rule.
IF/THEN	The element represents structure of Rule. It uses the condition-action type representation. It can either use or omit operators like AND and OR but it must have at least one variable, value and operator, which represents relationship between the variable and value.
AND/OR	These have the same operator as their names, and operator, which represents relationship between variable and value as low hierarchical elements.
variable	The literal element represents variable names.
value	It is the literal element that represents value names.

Παράδειγμα

```
<RSML Version="1.0">
  <Rule>
      <IF>
        <AND>
          <Job Title>Professor</Job Title>
          <NOT>
            <OR>
              <Sponsord>Ministry of Science and Technology</Sponsord>
                <Sponsord>Ministry of Information and Communication
                </Sponsord>
            </OR>
          </NOT>
        </AND>
      </IF>

      <THEN>
        <LE/>
          <Payment Level>Standard Payment Level*1.5</Payment Level>
      </THEN>

          <!-- The other rules are represented in the Appendix  -->

    </Rule>
</RSML>
```

RTML

DTD

Η RTML είναι μια γλώσσα που ενσωματώνεται στα προγράμματα εφαρμογών όπως οι φόρμες σε workflow management, software agents, και γενικώς σε οποιοδήποτε πρόγραμμα. Τώρα πρέπει να ορίσουμε ένα σύνολο από συγκεκριμένες εντολές σχετικά με το πότε να ενεργοποιήσουμε το inference, ποιούς κανόνες να χρησιμποποιήσουμε, πως να χρησιμοποιήσουμε το παραγόμενο αποτέλεσμα, και ούτω καθεξής.Τα RTML tags εδώ ειναι χρήσιμα για να αναγνωρίζουμε τα σχετικά tags στην RIML και στην RSML και τα αρχεία δεδομένων σε XML μορφή.Το DTD της RTML ακολουθεί

```
<!ELEMENT RTML (WhenToTrigger, Bring, Result)+>
      <!ATTLIST RTML Version CDATA #Required)>

<!ELEMENT WhenToTrigger (AND|OR|NOT)>
<!ELEMENT NOT (AND|OR)>
<!ELEMENT AND (budgetary  source, requisition)>
<!ELEMENT OR (budgetary_source, requisition)>
<!ELEMENT budgetary_source (#PCDATA)>
```

<!ELEMENT requisition (on|off)>
<!ELEMENT on (#PCDATA)>
<!ELEMENT off (#PCDATA)>

<!ELEMENT Bring (RuleTitle, DataFile)>
<!ELEMENT RuleTitle (#PCDATA)>
<!ELEMENT DataFile (#PCDATA)>

<!ELEMENT Result (expenditure)>
<!ELEMENT expenditure (permitted|denied)>
<!ELEMENT permitted (#PCDATA)>

Στοιχεία και ιδιότητες της RTML

Το root στοιχείο ειναι το RTML

- WhenToTrigger: Αυτό το στοιχείο ορίζει το inference triggering timing. Μπορεί να περιέχει τα ακόλουθα υποστοιχεία: AND, or OR.
- Bring: Αυτο το στοιχείο ορίζει τον κανόνα που θα χρησιμοποιηθεί για το inference. Πρέπει να περιέχει τα ακόλουθα υποστοιχεία: <RuleTitle> and <DataFile>.
- Result: Δηλώνει πως να χρησιμοποιηθεί το παραγόμενο αποτέλεσμα.
- AND/OR: Αυτό το στοιχείο δηλώνει σχέσεις μεταξύ υποστοιχείων. Πρέπει να περιέχει τα ακόλουθα υποστοιχεία: requisition and budgetary_source.
- requisition: Αυτό το στοιχείο αναπαριστάνει κατάσταση της διεργασίας. Εχει ON και OFF σαν τιμές.
- Budgetary_source: Ειναι το literal element που αναπαριστάνει τον τύπο του budgetary για να γίνει η δαπάνη(expenditure).
- ON/OFF: Ειναι το literal element που αναπαριστάνει requisition status.
- RuleTitle: Ειναι ο τίτλος του κάθε κανόνα που θα χρησιμοποιηθεί απο τον ευφυή πράκτορα..
- DataFile: Ειναι ο τίτλος του αρχείου δεδομένων που θα χρησιμοποιηθεί απο τον ευφυή πράκτορα...
- expenditure/ budgetary_source: The literal element

Παράδειγμα

```
<RTML Version="1.0">
  <WhenTrigger>
    <AND>
       <requisition>on</requisition>
       <budgetary_ source>type-P research fund</budgetary_ source>
     </AND>
  </WhenTrigger>

  <Bring>
    <RuleTitle>Restriction of Type-P Research Fund Expenditure</RuleTitle>
     <DataFile>Research Fund Accounts</DataFile>
  </Bring>

  <Result>
    <expenditure>permitted</expenditure >
  </Result>
</RTML>
```

Simple Rule Markup Language(SRML)

Η SRML περιγράφει μια γενική γλώσσα κανόνων που αποτελείται απο το υποσύνολο των δομικών στοιχείων της γλώσσας που βρίσκονται στις δημοφιλές μηχανές κανόνων forward-chaining. Επειδή δεν χρησιμοποιεί δομικά στοιχεία απο proprietary vendor γλώσσες, οι κανόνες που δηλώνονται με αυτό το XML DTD μπορούν εύκολα να μεταφραστούν και να εκτελεστούν σε οποιαδήποτε συμπλέουσα μηχανή κανόνων κάνοντάς την χρήσιμη σαν γλώσσα για ανταλλαγή κανόνων μεταξύ java μηχανών κανόνων. Το java rule engine API που δηλώνεται στο JSR-000094 ορίζει γενικευμένη υποστήριξη API για parsing απο rulesets που αναπαριστώνται με XML, αλλά δεν ορίζει ενα πρότυπο για γλώσα κανόνων. Το DTD της γλώσσας SRML φαίνεται παρακάτω

```
<!-- SRML (Simple Rule Markup Language) DTD -->
<?xml version="1.0" encoding="ISO-8859-1"?>
<!ELEMENT ruleset (rule*)>
<!ATTLIST ruleset
  name NMTOKEN #IMPLIED>
<!ELEMENT rule ( priority?, conditionPart, actionPart ) >
<!ATTLIST rule
  name NMTOKEN #REQUIRED>
<!ELEMENT priority (%expression;)>
<!ELEMENT conditionPart (%condition;)+>
```

```
<!ELEMENT actionPart (%action;)* >
<!ENTITY % condition "(simpleCondition | notCondition)">
<!ENTITY % action "(assignment | bind | assert | assertobj
                | modify | retract)">
<!ELEMENT simpleCondition (%expression;)*>
<!ATTLIST simpleCondition
  className CDATA #REQUIRED
  objectVariable NMTOKEN #IMPLIED>
<!ELEMENT notCondition (%expression;)*>
<!ATTLIST notCondition
  className CDATA #REQUIRED>
<!ELEMENT assert (assignment | bind)* >
<!ATTLIST assert
  className CDATA #REQUIRED>
<!ELEMENT assertobj (%expression;)>
<!ELEMENT retract (variable)>
<!ELEMENT modify (variable, (assignment | bind)+)>
<!ENTITY % expression
 "(%assignable; | constant | unaryExp | binaryExp | naryExp)">
<!ELEMENT unaryExp (%expression;)>
<!ATTLIST unaryExp operator (plus | minus | not) #REQUIRED>
<!ELEMENT binaryExp (%expression;,%expression;)>
<!ATTLIST binaryExp
  operator (eq | neq | lt | lte | gt | gte) #REQUIRED>
<!ELEMENT naryExp (%expression;)+>
<!ATTLIST naryExp
  operator (add | subtract | multiply | divide |
          remainder | and | or) #REQUIRED>
<!ELEMENT assignment (%assignable;,%expression;)>
<!ELEMENT bind (%expression;) >
<!ATTLIST bind
  name NMTOKEN #REQUIRED>
<!ELEMENT constant EMPTY>
<!ATTLIST constant
  type (string | boolean | byte | short | char | long
       | int | float | double | null) #REQUIRED
  value CDATA #REQUIRED>
<!ENTITY % assignable "(variable | field)">
<!ELEMENT variable EMPTY>
<!ATTLIST variable
  name NMTOKEN #REQUIRED>
<!ELEMENT field (%expression;)?>
<!ATTLIST field
  name NMTOKEN #REQUIRED>
```

R2ML – H REWERSE I1 Rule Markup Language

Η R2ML είναι μια κατανοητή και φιλική στον χρήστη XML μορφή κανόνων που επιτρέπει

- την ανταλλαγή κανόνων μεταξύ διαφορετικών συστημάτων και εργαλείων
- τον πλουτισμό των οντολογιών με κανόνες
- σύνδεση των συστημάτων κανόνων σας με τα σε R2ML βασισμένα εργαλεία για vizualization, verbalization, verification και validation.

Η R2ML είναι κατανοητή με την έννοια οτι ολοκληρώνει

την *Object Constraint Language (OCL)*
ενα πρότυπο που χρησιμοποιείται στο information systems engineering και στο software engineering,
την *Semantic Web Rule Language (SWRL)*
μια πρόταση για επέκταση του σημασιολογικού ιστού γλώσσας οντολογίας OWL προσθέτοντας implication axioms,
την *Rule Markup Language (RuleML)*
μια πρόταση βασισμένη στην Datalog/Prolog,

και οεριέχει τέσσερις κατηγορίες κανόνων: derivation rules, production rules, integrity rules και ECA/reaction rules.

Scope

Μια σε XML βασισμένη γλώσσα κανόνων
Υποστήριξη για Integrity Rules, Derivation Rules, Production Rules και Reaction rules
Ολοκλήρωση λειτουργικών γλωσσών (όπως η OCL) με τις γλώσσες Datalog (όπως η SWRL)
Σειριοποίηση και ανταλλαγή κανόνων απο εξειδικευμένα software εργαλεία
Ολοκλήρωση reasoning κανόνων με πραγματικές τεχνολογίες server-side
Deploying, publishing και communicating κανόνων σε ένα δίκτυο

Αρχές Σχεδίασης

Μοντελοποίηση χρησιμοποιώντας MDA
Εννοιες κανόνων ορισμένες με τη βοήθεια της MOF/UML
Απαίτηση να συμπεριλάβει
Εννοιες web naming όπως URIs και XML namespaces
Την οντολογική διαφορά μεταξύ αντικειμένων και τιμών δεδομένων
Τις έννοιες datatype της rdf και user-defined datatypes
Actions
Events
EBNF αφηρημένο συντακτικό
Συγκεκριμένο συντακτικό βασισμένο σε XML validated απο ενα XML schema
Να επιτρέπει διαφορετική σημασιολογία για κανόνες

Για περισσότερα μπορείτε να δείτε στις δημοσιεύσεις στο link

http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/3

URML(UML-based Rule Modeling Language)

Η γλώσσα μοντελοποίησης που χρησιμοποιείται στην επικοινωνία μεταξύ business/domain αναλυτών και business/domain ειδικών για την ανάλυση και τεκμηρίωση των απαιτήσεων του συστήματος δεν πρέπει να είναι τεχνική αλλά πρέπει να επιτρέπει semi-visual και/η εκφράσεις κανόνων φυσικής γλώσσας τα οποία μπορούν να κατανοηθούν απο business/domain ειδικούς χωρίς εκτεταμένη τεχνική εκπαίδευση.

Η UML προσφέρει μια semi-visual διαγραμματική γλώσσα για μοντελοποίηση πληροφορίας επιτρέποντας εναν περιορισμένο αριθμό ενσωματωμένων τύπων κανόνων(συγκεκριμένα integrity constraints και derivation rules).Η Object Constraint Language(OCL) μπορεί να χρησιμοποιηθεί για να εκφράσουμε integrity rules και derivation rules μέσα σε UML διαγράμματα κλάσεων αλλά ειναι καθαρώς κειμενική και δεν επιτρέπει οπτική αναπαράσταση κανόνων.Για να κάνει δυνατή την μοντελοποίηση κανόνων η REWERSE Working group I1 ανέπτυξε την URML που επιτρέπει την οπτική μοντελοποίηση κανόνων βασισμένα σε UML μοντέλα κλάσεων.Ακολουθεί η βασική περιγραφή των URML metamodel στοιχείων.

Περιγραφή Γλώσσας

Η URML υποστηρίζει την μοντελοποίηση των derivation rules, production rules και reaction rules.Ενας κανόνας αναπαριστάται γραφικά σαν ένας κύκλος με ενα αναγνωριστικό κανόνα.Βέλη εισόδου αναπαριστάνουν συνθήκες κανόνων ή triggering events , τα βέλη εξόδου αναπαριστάνουν rule conclusions ή παραγόμενες ενέργειες(produced actions).

Τα **condition arrows** αναφέρονται σε conditioned στοιχεία μοντέλου που ειναι ενας classifier ή ενα association. Μπορεί να έρχεται με μια έκφραση φίλτρου επιλέγοντας στιγμιότυπα απο την προέκταση του condition classifier και με μια συγκεκριμένη μεταβλητή αντικειμένου(ή object variable tuple, σε περίπτωση μιας association) με πεδίο τιμών πάνω από την παραγόμενη συλλογή στιγμιοτύπων.

Τα **negated condition arrows** σταυρώνονται στην αρχή τους.Δηλώνουν μια αρνητική συνθήκη η οποία πρέπει να συζευχθεί (conjoined) με ενα ή περισσότερα βέλη θετικών συνθηκών ετσι ώστε οι μεταβλητές τους να καλύπτονται απο αυτά.

Τα **derivation rules** αναπαριστώνται γραφικά σαν ενας κύκλος με μια εσωτερική ετικέττα "DR" και ενα αναγνωριστικό κανόνα που προσκολλάται σε αυτό.Τα βέλη εισόδου αναπαριστάνουν συνθήκες και τα βέλη εξόδου αναπαριστάνουν conclusions.

- Τα **conclusion arrows** επίσης αναφέρονται σε ενα classifier model element.Η έννοιά του είναι να δηλώσει οτι το κατηγόρημα που αναπαριστάται απο τον conclusion classifier εφαρμόζεται σε οποιοδήποτε στιγμιότυπο που ικανοποιεί όλες τις συνθήκες των κανόνων

Τα **production rules rules** αναπαριστώνται γραφικά σαν ενας κύκλος με μια εσωτερική ετικέττα "PR" και ενα αναγνωριστικό κανόνα που προσκολλάται σε αυτό. Τα βέλη εισόδου αναπαριστάνουν συνθήκες και τα βέλη εξόδου με διπλή κεφαλή βέλους αναπαριστάνουν actions.

- Τα **action arrows** αναφέρονται είτε στην κλάση (σε περίπτωση μιας create, delete, assign ή invoke ενέργεια(action)) είτε σε μια activity.

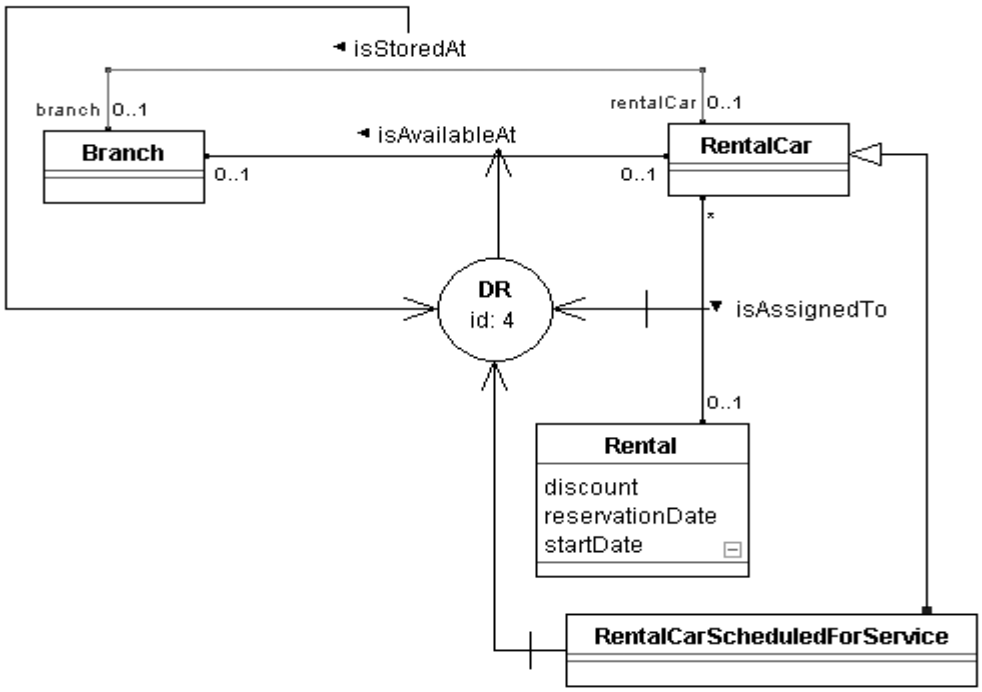
Τα **reaction rules** αναπαριστώνται γραφικά σαν ενας κύκλος με μια εσωτερική ετικέττα "RR" και ενα αναγνωριστικό κανόνα που προσκολλάται σε αυτό.Υπάρχουν δύο είδη βελών εισόδου(condition arrows και event arrows) και δύο είδη βελών εξόδου(action arrows και postcondition arrowsκαι τα δύο έχουν διπλό arrowhead).

- Τα **event arrows** αναφέρονται σε έναν κανόνα και μπορεί να αναφέρονται και σε μιά κλάση.

Rule Modeling Παραδείγματα

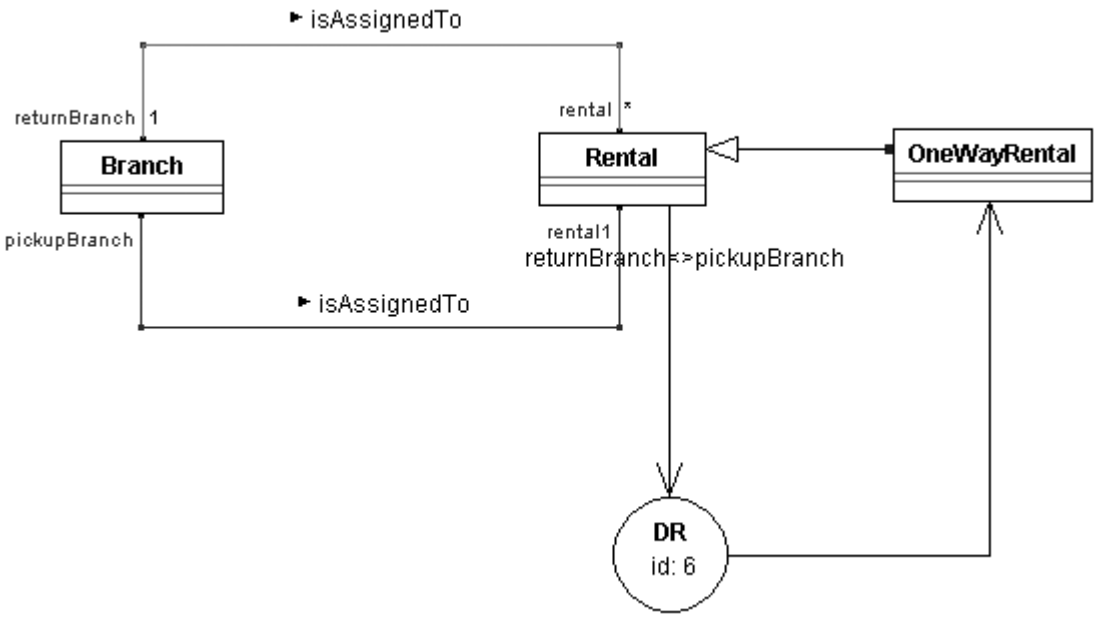
Μοντελοποιώντας ***ενα derivation rule για τον ορισμό μιας Association***

If a rental car is stored at a branch, is not assigned to a rental and is not scheduled for service, then the rental car is available at the branch.



Μοντελοποιώντας ***ενα derivation rule για τον ορισμό μιας Subclass***

If the return branch of a rental is different from the pickup branch, then the rental is a one way rental.



ARML(Active Rule Markup Language)

Η ARML δίνει την δυνατότητα στο business logics που ορίζονται σε διάφορες γλώσσες κανόνων να διαμοιράζονται και να επαναχρησιμοποιούνται μεταξύ διαφορετικών συστημάτων.Με την εύκολη στον ορισμό περιγραφών κανόνων δυνατότητα χρησιμοποιώντας την XML η ARML παρέχει μια ομοιόμορφη ECA πλατφόρμα για ετερογενή ενεργά πληροφοριακά συστήματα. Η ARML με το πλεονέκτημα του XML προτύπου με την ευκολία της κατανόησης, την τέλεια εκφραστικότητα και την δύναμη αναπαράστασης στον ιστό ειναι απλή στη χρήση και φτηνή στην υλοποίηση χρησιμοποιώντας καλά αναπτυγμένα XML APIs και XML utilities.Οι ενεργοί κανόνες μπορούν να αναπαρασταθούν με διάφορους τρόπους χρησιμοποιώντας την XSL eXtensible Stylesheet Language.

Στον πίνακα φαίνεται το DTD της ARML

```
<!ELEMENT rule (ruleDef, event ,
condition ,action , coupling,
precedence,
info)>
<!ELEMENT ruleDef
(ruleName, table?, ruleSet?)>
<!ELEMENT event (eventName |
algebra)?>
<!ELEMENT condition
(methodCall | boolean | algebra)?>
<!ELEMENT action (methodCall+)>

<!ELEMENT coupling (ec? , ca?)>
<!ELEMENT precedence (after?,
before?)>
<!ELEMENT info
(designer*, description?, category?)>
<!ELEMENT algebra (and | or | seq)>
```

```
<!ELEMENT designer (#PCDATA)>
<!ELEMENT description
(#PCDATA)>
<!ELEMENT category (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT ec (#PCDATA)>
<!ELEMENT ca (#PCDATA)>

<!ELEMENT methodCall
(methodName, params)>
<!ELEMENT methodName
(#PCDATA)>
<!ELEMENT params (param*)>
<!ELEMENT param (value)>

<!ELEMENT value (i4 | int | boolean |
string |
dateTime.iso8601 | double | base64 |
variable | struct | array)>
```

```
<!ELEMENT and
((eventName | methodCall)+ ,
algebra?)>
<!ELEMENT or
((eventName | methodCall)+ ,
algebra?)>
<!ELEMENT seq
((eventName | methodCall)+ ,
algebra?)>
<!ELEMENT before (ruleList?)>
<!ELEMENT after (ruleList?)>
<!ELEMENT ruleList (ruleName)*>
<!ELEMENT ruleName
(#PCDATA)>
<!ELEMENT table (#PCDATA)>
<!ELEMENT ruleSet (#PCDATA)>
<!ELEMENT eventName
(#PCDATA)>
```

```
<!ELEMENT i4 (#PCDATA)>
<!ELEMENT int (#PCDATA)>

<!ELEMENT boolean (#PCDATA)>
<!ELEMENT string (#PCDATA)>

<!ELEMENT double (#PCDATA)>
<!ELEMENT variable (#PCDATA)

<!ELEMENT dateTime.iso8601
(#PCDATA)>
<!ELEMENT data (value*)>
<!ELEMENT base64 (#PCDATA)

<!ELEMENT array (data)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT member (name, value)>

<!ELEMENT struct (member*)>
```

Πίνακας DTD της ARML

BRML(Business Rule Markup Language)

Η BRML ειναι η καθιερωμένη γλώσσα κανόνων για επικοινωνία πρακτόρων βασισμένη στο Courteous/Ordinary λογικά προγράμματα.Χρησιμοποιείται σε σύνδεση με την CommonRules απο την IBM και αναπτύχθηκε σε συνδυασμό με το project Business Rules για E-commerce της IBM.

Απο τα εργαστήρια AlphaWorks της IBM τα CommonRules ειναι ενα σύνολο απο « web-based διαλειτουργικούς e- business κανόνες με κοινό πλαίσιο αναπαράστασης με διαχείρηση συγκρούσεων(conflict handling). Οι CommonRules ειναι μια java βιβλιοθήκη που παρέχει λειτουργικότητα για business κανόνες ειδικά για την διαλειτουργικότητά τους.Ο ρόλος του είναι να συμπληρώσει και να επαυξήσει την λειτουργικότητα των διαφόρων rule-based συστημάτων και προγραμματιστικών μηχανισμών που υπάρχουν ήδη στην αγορά.

Η τρέχουσα έκδοση της BRML εκφράζει τα courteous λογικά προγράμματα που επικαλύπτονται απο ενα ευρύ υποσύνολο της KIF.Η BRML συμπληρώνει και επεκτείνει την ANSI-draft KIF (Knowledge Interchange Format) και παρέχει την πρώτη XML κωδικοποίηση για KIF. Η BRML συνεχίζει πιό πέρα απο την KIF για να υποστηρίξει λογική μή μονοτονικότητα συμπεριλαμβανομένου και της negation-as failure, την πιό σημαντική πρακτικά μορφή της άρνησης και με προτεραιότητα τον διαχείρηση των συγκρούσεων.Τελικά οι CommonRules ειναι ενα πλαίσιο για ανάπτυξη rule-based εφαρμογών με πρωτεύουσα έμφαση στο διαχωρισμό business logic και δεδομένων, διαχείρηση συγκρούσεων και διαλειτουργικότητα με κανόνες.

Περισσότερα στους συνδέσμους

http://www.alphaworks.ibm.com/tech/commonrules

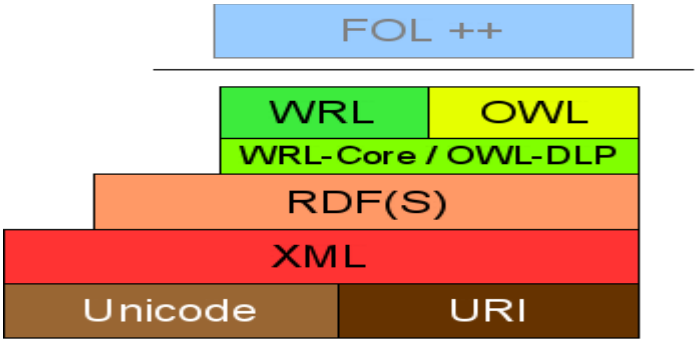
και

http://www.research.ibm.com/rules/home.html

WRL(Web Rule Language)

Η Web Rule Language αναπτύχθηκε απο το κομμάτι οντολογίας της WSML(Web Service Modeling Language).Η WRL ειναι μια βασισμένη σε κανόνες γλώσσα οντολογιών ενώ η OWL ειναι βασισμένη σε περιγραφική λογική γώσσα οντολογιών. Η WRL συμπλέει με το εννοιολογικό μοντέλο για οντολογίες που αναπτύχθηκε με την προσπάθεια της WSMO, που είναι ανεξάρτητη απο οποιοδήποτε paradigm λογικής γλώσσας.Το βασικό μεταμοντέλο οντολογίας της WRL αποτελείται απο concepts, relations, instances, και axioms.

Η WRL εχει ενα XML συντακτικό ανταλλαγής το οποίο βασίζεται στην RuleML. Αναμένεται οτι η WRL, WSML, SWSL και η RuleML θα συγκλίνουν στο XML συντακτικό.



Πίνακας: Αυτή ειναι η θέση της WRL στο σημασιολογικό ιστό

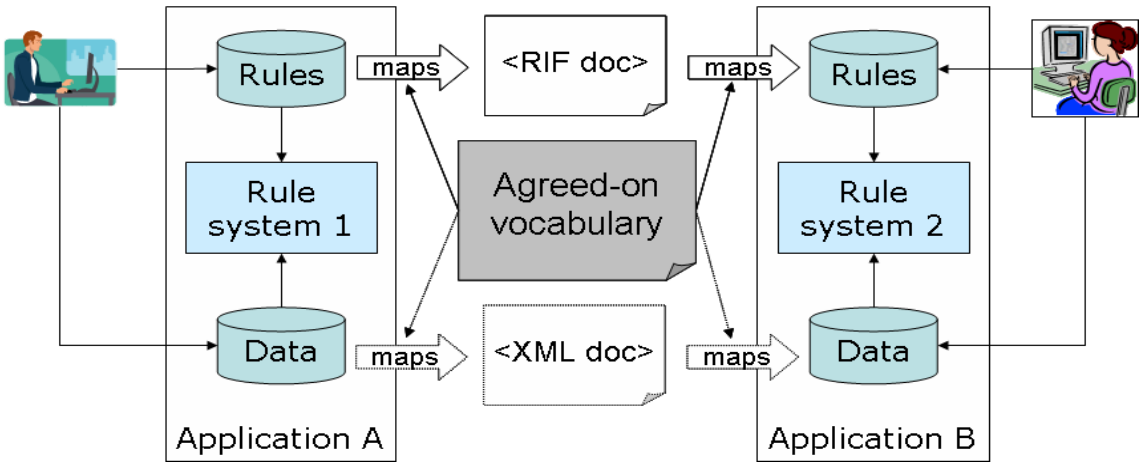
Το εννοιολογικό συντακτικό της WRL έχει frame-like στυλ.Η πληροφορία σχετικά με μια κλάση και των ιδιοτήτων της, μια σχέση και των παραμέτρων της, και ενα στιγμιότυπο και των τιμών των ιδιοτήτων της δηλώνεται σαν ενα μεγάλο συντακτικό δομικό στοιχείο αντί να διαιρείται σε εναν αριθμό απο ατομικά κομμάτια.Ειναι δυνατόν να διαχύσουμε την πληροφορία σχετικά με μια κλάση, σχέση, στιγμιότυπο ή αξίωμα σε πολλά δομικά στοιχεία αλλά δεν το συνιστούμε. Στην πραγματικότητα η WRL ειναι σαν την OIL η οποία επίσης προσφέρει την δυνατότητα της συνάθροισης περιγραφών μαζί σε πλαίσια ή διεσπαρμένες τις περιγραφές στο έγγραφο.
Μια WRL οντολογία διαχωρίζεται σε δύο μέρη.Το πρώτο μέρος παρέχει μετα-πληροφορία σχετικα με την οντολογία που αποτελείται απο τέτοια πράγματα όπως WRL variant identification, namespace references, non-functional properties (annotations), import των οντολογιών, και τον τύπο του specification.Το δεύτερο μέρος του specification που αποτελείται απο στοιχεία όπως concepts, attributes, relations, δεν ειναι διατεταγμένο.

Το specification της WRL βρίσκεται στη διεύθυνση

http://www.w3.org/Submission/2005/SUBM-WRL-20050909/

RIF(Rule Intechange Format)

Η RIF ειναι ειναι ενα πλαίσιο για ανταλλαγή κανόνων μεταξύ εφαρμογών στο διαδίκτυο.Αναπτύσσεται απο το RIF Working Group.Για περισσότερες πληροφορίες μπορείτε να δείτε στη Σελίδα
http://www.w3.org/2005/rules/wiki/RIF_Working_Group
Ενα RIF use case γενικό φαίνεται στο παρακάτω σχήμα.

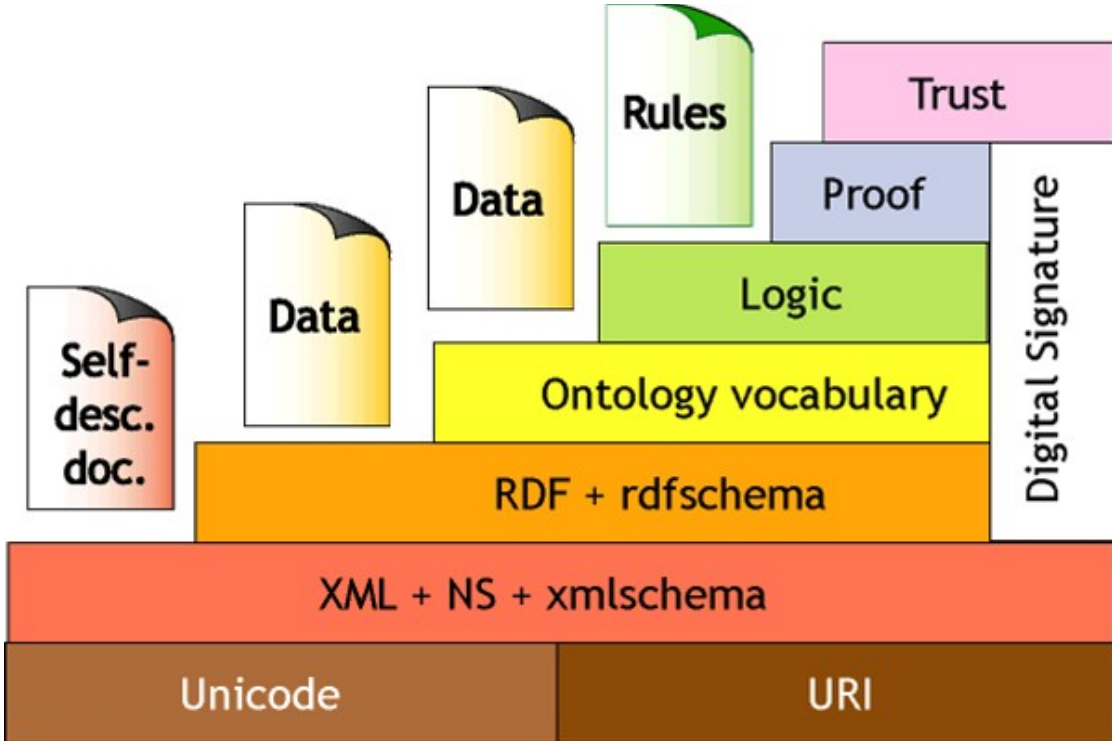


Βιβλιογραφία -Αναφορές

1).Fallside, D.C.; Walmsley, P. (Eds) (2004) XML Schema Part 0: Primer Second Edition. W3C Recommendation, 28 October 2004. Available at: http://www.w3.org/TR/xmlschema-0/.
2).ISO (2003) Information and documentation - The Dublin Core metadata element set. ISO Draft International Standard. ISO 15836:2003(E) / ISO TC 46/SC 4 N515. Available at:
http://www.niso.org/international/SC4/n515.pdf.
3).Manola, F.; Miller, E. (Eds) (2004) RDF Primer. W3C Recommendation, 10 February 2004.
Available at: http://www.w3.org/TR/rdf-primer/.
4).W3 Schools (2006a) XML Schema Tutorial. Available at: http://www.w3schools.com/schema/default.asp.
5).W3 Schools (2006b) XML Tutorial. Available at: http://www.w3schools.com/xml/default.asp
6).Carroll, J.J.; De Roo, J. (Eds) (2004) OWL Web Ontology Language Test Cases. W3C Recommendation, 10 February 2004. Latest version available at: http://www.w3.org/TR/owl-test/.
7).Dean, M.; Schreiber, G. (Eds) (2004) OWL Web Ontology Language Reference. W3C Recommendation, 10 February 2004. Latest version available at: http://www.w3.org/TR/owl-ref/.
8).Heflin, J. (Ed) (2004) OWL Web Ontology Language Use Cases and Requirements. W3C Recommendation, 10 February 2004. Latest version available at: http://www.w3.org/TR/webont-req/.
9).Horridge, M.; Knublauch, H. ; Rector, A.; Stevens, R.; Wroe, C. (2004) A Practical Guide To Building OWL Ontologies Using The Protıgı-OWL Plugin and CO-ODE Tools Edition 1.0, The University Of Manchester, Stanford University.
10).McGuinness, D.L.; Harmelen, F.V. (Eds) (2004) OWL Web Ontology Language Overview.W3C Recommendation, 10 February 2004. Latest version available at:http://www.w3.org/TR/owl-features/.
11).Patel-Schneider, P.F.; Hayes, P.; Horrocks, I. (Eds.) (2004) OWL Web Ontology Language Semantics and Abstract Syntax. W3C Recommendation, 10 February 2004. Latest version available at: http://www.w3.org/TR/owl-semantics/.

12).W3 Schools (2006) Introduction to OWL. Available at: http://www.w3schools.com/rdf/rdf_owl.asp
13). Dean, M.; Schreiber, G. (Eds) (2004) OWL Web Ontology Language Reference. W3C Recommendation, 10 February 2004. Latest version available at: http://www.w3.org/TR/owl-ref/.
14).Heflin, J. (Ed) (2004) OWL Web Ontology Language Use Cases and Requirements. W3C Recommendation, 10 February 2004. Latest version available at: http://www.w3.org/TR/webont-req/.
15).Horridge, M.; Knublauch, H. ; Rector, A.; Stevens, R.; Wroe, C. (2004) A Practical Guide To Building OWL Ontologies Using The Protıgı-OWL Plugin and CO-ODE Tools Edition 1.0, The University Of Manchester, Stanford University.
16).McGuinness, D.L.; Harmelen, F.V. (Eds) (2004) OWL Web Ontology Language Overview.W3C Recommendation, 10 February 2004. Latest version available at:http://www.w3.org/TR/owl-features/.
17).Patel-Schneider, P.F.; Hayes, P.; Horrocks, I. (Eds.) (2004) OWL Web Ontology Language Semantics and Abstract Syntax. W3C Recommendation, 10 February 2004. Latest version available at: http://www.w3.org/TR/owl-semantics/.
18).Martin, D.; Burstein, M.; Hobbs, J,; Lassila, O.; McDermott, D.; McIlraith, S.; Narayanan, S.;
19).Paolucci, M.; Parsia, B.; Payne, T.; Sirin, E.; Srinivasan, N.; Sycara, K. (2006) OWL-S:Semantic markup for Web services. Available at: http://www.daml.org/services/owl-s/1.2/Service.owl.
20).Berners-Lee, T., et al. cwm – a general purpose data processor for the semantic web.http://www.w3.org/2000/10/swap/doc/cwm
21). Boag, S., Chamberlin, D., Fernandez, M.F., Florescu, D., Robie, J., Simeon, J. XQuery 1.0: An XML Query Language. W3C Working Draft 23 July 2004. http://www.w3.org/TR/xquery/
22). Boley, H., Dean, M., Grosof, B., Sintek, M., Spencer, B., Tabet, S., Wagner, G. First-Order-Logic RuleML. http://www.ruleml.org/fol/
23). Dean, M. DAML Expense Reconciliation. http://www.daml.org/2001/06/expenses/
24). Dean, M. Subway Maps in OWL. http://www.daml.org/2003/05/subway/
25). Dean, M. Use of SWRL for Ontology Translation. WWW2004 Developer Day Rules on the Web Track. http://www.daml.org/2004/05/swrl-translation/Overview.html
Semantic Web Rules: Covering the Use Cases 5
26). Decker, S., Dean, M., McGuinness, D. Requirements and Use Cases for a Semantic Web Rule Language. http://www.isi.edu/~stefan/20030224/rulesusecases.html
27). Fensel, D. A rule language for the semantic web. SDK cluster meeting, Lausanne, 14 June 2004. http://www.wsmo.org/papers/presentations/SDK-cluster/lausanne/
28). Fikes, R., McGuinness, D.L. An Axiomatic Semantics for DAML+OIL. Stanford KSL Technical Report KSL-01-01. http://www.ksl.stanford.edu/people/dlm/daml-semantics/abstract-axiomatic-semantics.html
29). Golbreich, C. Reasoning with Ontologies and Rules Based on the Current Semantic Web Standards and Tools. Proc. Rules and Rule Markup Languages for the Semantic Web,Hiroshima, November 2004. http://2004.ruleml.org/
30). Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission 21 May 2004. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/
31). Jena 2. http://www.hpl.hp.com/semweb/jena2.htm
32). Kifer, M., et al. FORUM Working Group and forum-flogic email archive. http://projects.semwebcentral.org/projects/forum/
33). Kopena, J. OWLJessKB: A Semantic Web Reasoning Tool.

http://edge.cs.drexel.edu/assemblies/software/owljesskb/
34).Brachman, R.J.; Levesque, H.J. (2004) *Knowledge Representation and Reasoning*. Morgan Kaufmann, Los Altos, CA, USA.
35).Haarslev, V.; Moller, R. (2001) RACER system description. In: *Proceedings of the Sixth International Joint Conference on Automated Reasoning*, Siena, Italy. Lecture Notes in Computer Science, Vol. 2083. Springer-Verlag, Berlin, Germany, pp. 701–705.
36).Horrocks, I. (1998) Using an expressive description logic: FaCT or fiction? In: *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning*, pp. 636–647.
37).Horrocks, I. (2003) The FaCT System. Available at: http://www.cs.man.ac.uk/~horrocks/FaCT/
38).Huth, M.; Ryan, M. (2004) *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, Cambridge, UK.
39). Nardi, D.; Brachman, R.J. (2003) An introduction to description logics. In: Baader, F.;Calvanese, D.; McGuiness, D.L.; Nardi, D.; Patel-Schneider, P.F. (Eds) *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University.Press, Cambridge, UK.
40). W3 Schools (2006) Introduction to OWL. Available at: http://www.w3schools.com/rdf/rdf_owl.asp



The Semantic Web Stack