ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση Ανακάλυψης Υπηρεσιών Σημασιολογικού Ιστού με το πρότυπο SAWSDL

Χαράλαμπος Ε. Φραντζής

Επιβλέπων: Ευστάθιος Χατζηευθυμιάδης, Επίκουρος Καθηγητής ΕΚΠΑ

ΑΘΗΝΑ

ΜΑΡΤΙΟΣ 2008

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Υλοποίηση Ανακάλυψης Υπηρεσιών Σημασιολογικού Ιστού με το πρότυπο SAWSDL

Φραντζής Ε. Φραντζής

Α.Μ.: 878

ΕΠΙΒΛΕΠΩΝ:
Ευστάθιος Χατζηευθυμίαδης, Επίκουρος Καθηγητής ΕΚΠΑ

Μάρτιος 2008

ΠΕΡΙΛΗΨΗ

Σε αυτή την διατριβή μελετούμε την διαδικασία της ανακάλυψης υπηρεσιών στο Διαδίκτυο με τη χρήση σημασιολογικών αναφορών. Προκειμένου να αντιμετωπίσουμε τα μειονεκτήματα της αναζήτησης υπηρεσιών που στηρίζονται στο ταίριασμα λέξεων κλειδιών, χρησιμοποιούμε οντολογίες, ιεραρχίες δηλ. αντικειμένων που περιγράφουν έναν τομέα γνώσης. Χρησιμοποιώντας διάφορα εργαλεία λογισμικού όπως το WSMO Studio και το Protégé εμπλουτίζουμε τα αρχεία περιγραφής των υπηρεσιών (WSDL documents) με σημασιολογικές σημειώσεις (semantic annotations) που παραπέμπουν σε αντικείμενα (concepts) των οντολογιών κι έτσι δημιουργούμε έγγραφα SAWSDL στα οποία θα στηριχτεί η αναζήτηση. Έτσι ο χρήστης μπορεί να αναζητήσει υπηρεσίες βασισμένος σε έννοιες προσυμφωνημένων οντολογιών που επιθυμεί. Η διαδικασία της αναζήτησης, ψάχνει σε όλες τις υπηρεσίες που διαθέτει σε μια βάση και προσπαθεί να τις βαθμολογήσει ανάλογα με τον βαθμό που ταιριάζουν με τις απαιτήσεις του χρήστη Έτσι ακόμη και αν δεν βρεθεί υπηρεσία που να ταιριάζει ακριβώς (exact matching), επιστρέφεται στον χρήστη ένα σύνολο από υπηρεσίες μαζί με ένα βαθμό ταιριάσματος (degree of match) για κάθε μία. Ο χρήστης στη συνέχεια μπορεί να επιλέξει κάποια υπηρεσία, να συμπληρώσει τις εισόδους της και να προκαλέσει την εκτέλεσή της.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Υπηρεσίες Σημασιολογικού Ιστού

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: οντολογία, βαθμός ταιριάσματος, υπηρεσία διαδικτύου, σημασιολογία, SAWSDl, WSDL, αναζήτηση

ABSTRACT

In this thesis we study the procedure of discovering web services using semantic annotations. In order to confront the drawback of discovering web services using key word-based matching, we exploit ontologies, i.e. conceptual hierarchies describing a domain of knowledge. We use software tools such as WSMO Studio and Protege, in order to embed semantic annotations into WSDL documents,that refer to concepts of the ontologies. This way we create Semantic Web Services, expressed as SAWSDL documents. Hence the user can discover web services based on concepts their ontological semantics. The discovery process searches all the services stored in a database and mark each service according to the user request, based on certain criteria. Even if no service matches exactly with the user request, a ranked list of services (with a degree of match) is returned to the user. Then the user can provide the inputs of the selected service and invoke its execution.

SUBJECT AREA: Semantic Web Services Discovery

KEYWORDS : ontology, web service, degree of match, SAWSDL, WSDL, semantics, discovery

Στον Γιώργη, όπου κι αν βρίσκεται...

I

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Η διπλωματική αυτή εργασία γίνεται στο πλαίσιο της Κατεύθυνσης «Νέες Τεχνολογίες και Τηλεπικοινωνίες» του Μεταπτυχιακού Προγράμματος Πληροφορικής του Πανεπιστημίου Αθηνών. Η εγγραφή μου στο Μεταπτυχιακό Πρόγραμμα έγινε τον Σεπτέμβριο του 2006, ενώ υπηρετούσα ως καθηγητής δευτεροβάθμιας εκπαίδευσης στο Γυμνάσιο Νισύρου. Τα μαθήματα του Προγράμματος αφορούν τις τελευταίες τεχνολογίες στο χώρο της Πληροφορικής και των Τηλεπικοινωνιών.

Ο Σημασιολογικός Ιστός και η Ανακάλυψη Υπηρεσιών Σημασιολογικού Ιστού μπορεί να αλλάξει ριζικά τον τρόπο που αντιλαμβανόμαστε το Διαδίκτυο και να προσφέρει ουσιαστικές και ολοκληρωμένες λύσεις στα σημερινά προβλήματα του Διαδικτύου. Χρησιμοποιήσαμε τελευταίες τεχνολογίες και προσπαθήσαμε να παρουσιάσουμε ένα ολοκληρωμένο σύστημα μέσα από το οποίο αυτές οι τεχνολογίες θα αλληλεπιδρούν.

Θέλω να ευχαριστήσω ιδιαίτερα τον Βασίλη Τσέτσο για την πολύτιμη συνεργασία και καθοδήγησή του και την γυναίκα μου Χαρά για την αμέριστη συμπαράστασή της

Χαράλαμπος Ε. Φραντζής

ΚΕΦΑΛΑΙΟ 1

ΥΠΗΡΕΣΙΕΣ ΔΙΑΔΙΚΤΥΟΥ (WEB SERVICES)

1.1 Βασικές Έννοιες των Web Services

Ο Παγκόσμιος Ιστός (World Wide Web) έχει αλλάξει τον τρόπο με τον οποίο οι άνθρωποι επικοινωνούν μεταξύ τους και αποτελεί ένα μέσο προβολής και δημοσιοποίησης των δραστηριοτήτων των επιχειρήσεων. Τα τελευταία χρόνια πολλές επιχειρήσεις παγκοσμίως άρχισαν να χρησιμοποιούν το Διαδίκτυο για να παρέχουν υπηρεσίες προς τους πελάτες τους αλλά και προς τις άλλες επιχειρήσεις. Μια τεχνολογία που ήδη έχει θεσπιστεί από το world wide web consortium (www.w3c.org) με το όνομα Υπηρεσία Διαδικτύου (web service), ήρθε να δώσει λύση στα προβλήματα ευχρηστίας και λειτουργικότητας των ηλεκτρονικών υπηρεσιών. Μέσα από την αρχιτεκτονική των web services καθορίζονται οι προδιαγραφές και οι κανόνες με τους οποίους είναι δυνατή η δημιουργία και η παροχή ηλεκτρονικών υπηρεσιών μέσα από το Διαδίκτυο με αρκετά απλό τρόπο. Τα πλεονεκτήματα αυτής της αρχιτεκτονικής είναι πολλά, μερικά από τα οποία αναφέρονται παρακάτω:

Διαλειτουργικότητα: ένα web service παρέχει ανεξαρτησία τόσο από το λειτουργικό σύστημα όσο και από το hardware. Οποιοδήποτε πρόγραμμα που συμβαδίζει με αυτήν την τεχνολογία μπορεί να προσπελάσει μια τέτοια υπηρεσία.

Ενσωμάτωση: για την δημιουργία ενός web service σε ένα υπάρχον λογισμικό σύστημα που λειτουργεί μέσα στο Διαδίκτυο, δεν απαιτούνται αλλαγές στο μηχανισμό του συστήματος

Διαθεσιμότητα και δημοσίευση: Οι πληροφορίες για τα web services δημοσιεύονται οπότε είναι εύκολο να βρεθούν και να χρησιμοποιηθούν.

Επέκταση: Ένα έτοιμο web service είναι δυνατό να ανανεωθεί με εύκολο τρόπο παρέχοντας επιπρόσθετες υπηρεσίες στους χρήστες του.

Μικρό κόστος δημιουργίας και χρήσης: Αν σε κάποιο λογισμικό σύστημα υπάρχει ήδη έτοιμη κάποια διαδικασία που χρειάζεται να επεκταθεί σε on-line υπηρεσία, η δημιουργία του web service είναι πολύ εύκολη υπόθεση και κοστίζει ελάχιστα. Η ενσωμάτωση ενός web service σε κάποιο website ή σε δικτυακή εφαρμογή έχει πολύ χαμηλό κόστος.

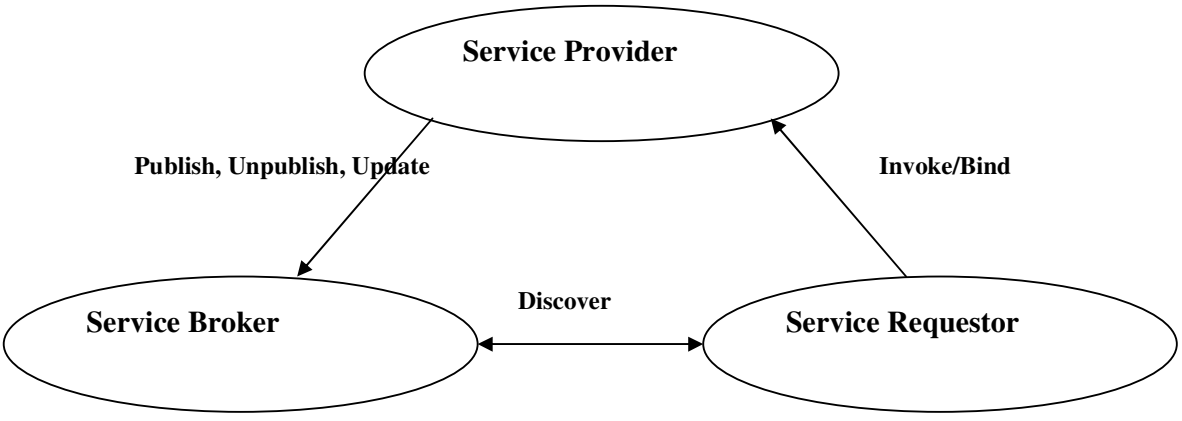
Χρήση λογισμικών συστημάτων: Όλα τα λογισμικά συστήματα και ειδικότερα τα websites που χρησιμοποιούν έτοιμες υπηρεσίες γίνονται πιο λειτουργικά και πιο φιλικά αφού παρέχουν περισσότερες υπηρεσίες στους χρήστες.

1.2 Το μοντέλο των web services

Οποιαδήποτε αρχιτεκτονική προσανατολισμένη στις υπηρεσίες (service oriented architecture) θα πρέπει να υποστηρίζει τις παρακάτω βασικές δραστηριότητες:

- Δημιουργία (Creation)
- Περιγραφή (Description)
- Δημοσίευση σε Intranet και / ή Internet καταλόγους για την εύρεσή τους από πιθανούς χρήστες
- Εύρεση (Discovery) από πιθανούς χρήστες
- Ενεργοποίηση (Invocation)
- Απόσυρση (Unpublishing) σε περίπτωση που δεν είναι πλέον διαθέσιμη ή δεν χρειάζεται ή σε περίπτωση που πρέπει να ανανεωθεί ώστε να ικανοποιεί νέες απαιτήσεις

Προκειμένου να εκμεταλλευτούμε πλήρως την αρχιτεκτονική των web services, υπάρχουν επιπρόσθετες δραστηριότητες να υλοποιηθούν, όπως είναι η σύνθεση (composition), η διαχείριση (management), η παρακολούθηση (monitoring), η τιμολόγηση (billing) και η ασφάλεια (security). Το βασικό πάντως μοντέλο των web services έχει τουλάχιστον τα ακόλουθα βασικά συστατικά :



Σχήμα 1
Web Service Model

*Παροχέας Υπηρεσίας (*Service Provider): Είναι ο συμμετέχων (party) που παρέχει το λογισμικό εφαρμογών για συγκεκριμένες ανάγκες ως υπηρεσίες. Δημοσιεύει, αποσύρει και ενημερώνει τις υπηρεσίες του έτσι ώστε να είναι διαθέσιμες στο Διαδίκτυο. Από την επιχειρηματική σκοπιά είναι ο ιδιοκτήτης της υπηρεσίας, ενώ από την αρχιτεκτονική πλευρά είναι η πλατφόρμα που αναλαμβάνει την υλοποίηση της υπηρεσίας.

*Αιτών την Υπηρεσία (*Service Requestor*)* : Είναι ο συμμετέχων που έχει μια ανάγκη / απαίτηση που μπορεί να ικανοποιηθεί από ένα web service. Από την επιχειρηματική σκοπιά είναι η επιχείρηση που απαιτεί να ικανοποιηθεί μια συγκεκριμένη λειτουργία της, ενώ από την αρχιτεκτονική πλευρά είναι η εφαρμογή που αναζητά και προκαλεί την εκτέλεση ενός web service. Ο αιτών βρίσκει την απαιτούμενη υπηρεσία μέσω του μεσίτη υπηρεσιών (service broker) που συνδέεται στις υπηρεσίες μέσω του παροχέα υπηρεσιών.

Μεσίτης Υπηρεσιών (Service Broker): Αυτός ο συμμετέχων παρέχει ένα κατάλογο (repository) με δυνατότητες αναζήτησης, όπου οι παροχείς υπηρεσιών δημοσιεύουν τις υπηρεσίες τους και οι αιτούντες εντοπίζουν τις υπηρεσίες και τις σχετικές πληροφορίες σύνδεσης (binding) με αυτές. Παράδειγμα μεσίτη είναι το UDDI Registry.

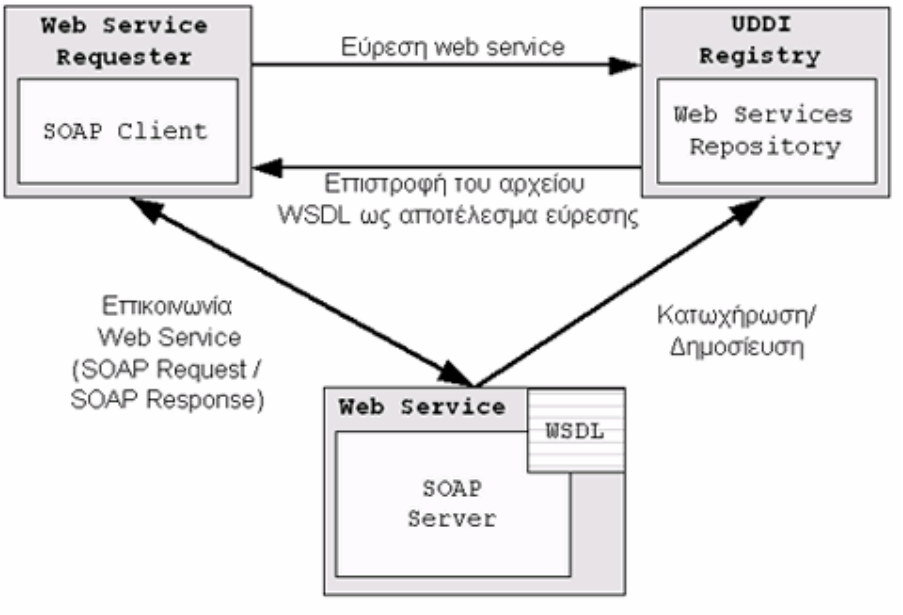
1.3 Πρότυπα των Web Services

Τα web services είναι XML αναπαραστάσεις προγραμμάτων, αντικειμένων ή κειμένων που είναι προσπελάσιμα μέσω Διαδικτύου για απ' ευθείας αλληλεπίδραση μεταξύ των εφαρμογών. Οι υπηρεσίες διαδικτύου μπορούν να προσπελαστούν με χρήση browser και παρέχουν ένα ανεξάρτητο από δεδομένα μηχανισμό παρουσίασης των υπηρεσιών της επιχείρησης με χρήση XML πρωτοκόλλων.

Οι βασικές τεχνολογίες που χρησιμοποιούνται συμπεριλαμβάνουν:

- XML, που περιλαμβάνει τη βασική XML και XML schemas
- SOAP, (Simple Object Access Protocol) που αποτελεί ένα πρωτόκολλο επικοινωνίας εφαρμογών βασισμένο σε XML
- WSDL (Web Services Description Language) που είναι ένα XML schema για την περιγραφή των μηνυμάτων, των λειτουργιών και των αντιστοιχήσεων των πρωτοκόλλων των web services
- UDDI (Universal Description Discovery and Integration) που είναι ο χώρος αποθήκευσης για την καταχώρηση και αναζήτηση των περιγραφών των web services

Ο συνδυασμός των ανωτέρω τεχνολογιών στα πλαίσια μιας αρχιτεκτονικής Web Services φαίνεται στο Σχήμα 2



Σχήμα 2
Το μοντέλο και η χρήση των Web Services

1.4 Web Services Description Language (WSDL)

Ένα από τα μεγάλα πλεονεκτήματα που έχουν οι web services έναντι των παραδοσιακών προσεγγίσεων είναι ότι μπορούν να περιγραφούν με τα έγγραφα WSDL, τα οποία είναι XML έγγραφα που περιέχουν όλες τις πληροφορίες που χρειάζονται για να συνδεθούμε με κάποια web service αλλά και να εκτιμήσουμε αν αυτή η υπηρεσία ικανοποιεί τις ανάγκες μας. Η χρησιμοποίηση της XML είναι μια καλή επιλογή γιατί εφαρμογές βασισμένες στην XML μπορεί εύκολα να είναι κατανοητές και από τον άνθρωπο (human readable) αλλά και από την μηχανή (machine readable). Είναι αρκετά απλό για έναν χρήστη να διαβάσει το WSDL έγγραφο και να το καταλάβει, όπως επίσης και για έναν προγραμματιστή να χρησιμοποιήσει έναν XML parser για να εξαγάγει δεδομένα από ένα τέτοιο έγγραφο. Κάθε πρόγραμμα που θέλει να χρησιμοποιήσει κάποια web service πρέπει να ανακτήσει το WSDL έγγραφο της υπηρεσίας για να μάθει πώς θα συνδεθεί με την υπηρεσία. Ο δημιουργός του web service δημιουργεί το WSDL έγγραφο και το εκδίδει (publish) στέλνοντας το ίδιο (ή το URL του) σε κάποια registry. Αν κάποιος θέλει να χρησιμοποιήσει την υπηρεσία θα πρέπει να απευθυνθεί σ' αυτήν την registry και να ανακτήσει το WSDL έγγραφο της υπηρεσίας.

Το WSDL έγγραφο αποτελείται από δύο μέρη: την συγκεκριμένη και την αφηρημένη περιγραφή (concrete and abstract description). Η συγκεκριμένη περιγραφή

περιλαμβάνει τα στοιχεία που αφορούν τον τρόπο με τον οποίον θα συνδεθούμε με την υπηρεσία, ενώ η αφηρημένη αφορά τα στοιχεία που περιγράφουν τις δυνατότητες του web service.

Υπάρχουν 4 αφηρημένα XML στοιχεία που είναι:

- <wsdl : types>
- <wsdl : message>
- <wsdl : operation>
- <wsdl : portType>

Υπάρχουν 3 συγκεκριμένα XML στοιχεία:

- <wsdl : service>
- <wsdl : port>
- <wsdl : binding>

Το στοιχείο <wsdl : types> χρησιμοποιείται για να δηλώνει τύπους δεδομένων, από τους πιο απλούς (π.χ. ακέραιος) μέχρι πιο σύνθετους όπως ο παρακάτω (κάτι ανάλογο με τις κλάσεις στον αντικειμενοστραφή προγραμματισμό)

```
<wsdl:types>
<xsd:schema targetNamespace=”http://www.stevepotts.com/customerType.xsd”
xmlns:xsd=”http://www.w3.org/2001/XMLSchema”>
<xsd:element name=”customer”>
<xsd:complexType>
<xsd:sequence>
<xsd:element name=”customerID” type=”xsd:string”/>
<xsd:element name=”lastName” type=”xsd:string”/>
<xsd:element name=”firstName” type=”xsd:string”/>
<xsd:element name=”address” type=”xsd:string”/>
<xsd:element name=”city” type=”xsd:string”/>
<xsd:element name=”state” type=”xsd:string”/>
<xsd:element name=”zip” type=”xsd:string”/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
</wsdl:types>
```

Το επόμενο στοιχείο στο WSDL είναι το <wsdl : message>. Τα messages ανταλλάσσονται μεταξύ των υπολογιστών και περιγράφουν λογικά τα μηνύματα. Π.χ. σε ένα τυπικό σενάριο request / response ένα μήνυμα αποστέλλεται και ένα άλλο μήνυμα λαμβάνεται. Το παρακάτω αποτελεί ένα παράδειγμα μηνύματος για την εγγραφή ενός πελάτη στο web service:

```
<wsdl:message name=”addCustomer”>
<wsdl:part name=”customerInfo” element=”tns:customer”/>
<wsdl:/message>
```

και ένα μήνυμα που περιέχει έναν ακέραιο θα σταλεί πίσω για επιβεβαίωση :

```
<wsdl:message name=”confirmation”>
<wsdl:part name=”response” element=”xsd:integer”/>
<wsdl:/message>
```

Το στοιχείο <wsdl : operation> είναι ανάλογο με τις *μεθόδους* στον αντικειμενοστραφή προγραμματισμό. Σε ένα operation επιτρέπονται μόνο τρία μηνύματα:

- Input Message, καθορίζει τα δεδομένα που το web service περιμένει να δεχτεί.
- Output Message, καθορίζει τα δεδομένα που το web service περιμένει να στείλει.
- Fault Message, καθορίζει τα μηνύματα λάθους

Παράδειγμα operation αποτελεί το παρακάτω:

```
<wsdl:operation name=”createNewCustomer”>
<wsdl:input message=”addCustomer”>
<wsdl:output message=”confirmation”>
<wsdl:fault message=”exceptionMessage”>
<wsdl:/operation>
```

Το portType element αποτελεί το σύνολο όλων των λειτουργιών (operations) που ένα web service μπορεί να δεχτεί. Αποτελεί ένα σημείο στο οποίο ο πελάτης μπορεί να αποκτήσει πληροφορίες για όλες τις λειτουργίες (operations) που παρέχει ένα web service. Παράδειγμα:

```
<wsdl:portType name=”newCustomerPortType”>
<wsdl:operation name=”createNewCustomer”>
<wsdl:input message=”addCustomer”/>
<wsdl:output message=”confirmation”/>
<wsdl:fault message=”exceptionMessage”/>
</wsdl:operation>
</wsdl:portType>
```

Το στοιχείο σύνδεσης (binding) παρέχει πληροφορίες όπως είναι το πρωτόκολλο και η διεύθυνση του web service. Π.χ.

```
<wsdl:binding name=”newCustomerBinding” type=”newCustomerPortType”>
<soap:binding style=”rpc”
transport=”http://schemas.xmlsoap.org/soap/http” />
<wsdl:operation name=”createNewCustomer”>
<soap:operation
soapAction=”http://www.stevepotts.com/createNewCustomer”/>
<wsdl:input>
<soap:body use=”encoded”
namespace=”http://www.stevepotts.com/customer”
encodingStyle=
“http://schemas.xmlsoap.org/soap/encoding/”/>
<wsdl:input>
<wsdl:output>
<soap:body use=”encoded”
namespace=”http://www.stevepotts.com/customer”
encodingStyle=
“http://schemas.xmlsoap.org/soap/encoding/”/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
```

Το επόμενο στοιχείο port μας παρέχει την πραγματική IP διεύθυνση και την θύρα (port) του web service που αντιπροσωπεύεται από το WSDL. Π.χ.

```
<wsdl:port binding=”newCustomerBinding” name=”newCustomerPort”>
<soap:address
location=”http://www.stevepotts.com:1776/soap/servlet/rpcrouter”>
</wsdl:port>
```

Το στοιχείο service περιέχει όλα τα ports που αντιπροσωπεύονται από το WSDL έγγραφο:

```
<wsdl:service name=”newCustomerService”>
<wsdl:documentation>
This is for adding new customers.
</wsdl:documentation>
<wsdl:port binding=”newCustomerBinding” name=”newCustomerPort”>
<soap:address
location=”http://www.stevepotts.com:1776/soap/servlet/rpcrouter”/>
</wsdl:port>
</wsdl:service>
```

Το τελευταίο στοιχείο definitions αποτελεί το στοιχείο ρίζα (root) του WSDL εγγράφου και περιέχει ένα σύνολο από namespaces που χρησιμοποιούνται στο έγγραφο :

<wsdl:definitions name=”customerExample”
targetNamespace=”http://www.stevepotts.com/customer.wsdl”
xmlns:soap=”http://www.schemas.xmlsoap.org/wsdl/soap/”
xmlns:wsdl=”http://www.schemas.xmlsoap.org/wsdl/”
xmlns=”http://www.stevepotts.com/customer.xsd”>
Το υπόλοιπο έγγραφο εμφανίζεται εδώ
</wsdl:definitions>

1.5 Ο Σημασιολογικός Ιστός (Semantic Web)

Όπως αναφέραμε και προηγούμενα το World Wide Web (WWW) άλλαξε ριζικά τον τρόπο με τον οποίο οι άνθρωποι επικοινωνούν και οι επιχειρήσεις κάνουν τις δουλειές τους. Αυτή την στιγμή το Διαδίκτυο βρίσκεται στην καρδιά της εξέλιξης προς μία κοινωνία της πληροφορίας. Κι ενώ στην αρχή χρησιμοποιούσαμε τους υπολογιστές για αριθμητικούς υπολογισμούς, και έπειτα για επεξεργασία πληροφοριών (επεξεργασία κειμένου, εφαρμογές Βάσεων Δεδομένων και παιχνίδια), σήμερα θεωρούμε τους υπολογιστές σαν τα σημεία εισόδου προς τις λεωφόρους της πληροφορίας.

Το σημερινό περιεχόμενο του Διαδικτύου είναι προσανατολισμένο στην κατανόηση από τον άνθρωπο (human consumption). Έτσι μπορεί σήμερα κάποιος χρήστης να ψάξει και να ανακαλέσει πληροφορίες από το Διαδίκτυο, να έρθει σε επικοινωνία με άλλους χρήστες, να περιηγηθεί σε ηλεκτρονικά καταστήματα και να παραγγείλει προϊόντα κ.λ.π. Μεγάλη ώθηση στο Διαδίκτυο και την χρησιμοποίησή του έδωσαν οι μηχανές αναζήτησης (search engines) όπως Yahoo, AltaVista,Google, οι οποίες στηρίζονται σε μια αναζήτηση βασισμένη σε λέξεις-κλειδιά (keyword-based search engines). Παρόλη την μεγάλη εξέλιξη των μηχανών αναζήτησης και την ευρεία χρήση τους από πολλούς χρήστες, υπάρχουν κάποια σοβαρά προβλήματα που πρέπει να ξεπεραστούν, όπως :

- Μεγάλη ανταπόκριση αλλά χαμηλή ακρίβεια: μπορεί να πάρουμε πολλά αποτελέσματα στην ερώτησή μας αλλά αρκετά από αυτά δεν έχουν και μεγάλη σχέση με αυτήν
- Χαμηλή ή καθόλου ανταπόκριση: παρόλο που αυτό είναι ένα μάλλον σπάνιο φαινόμενο σήμερα, μπορεί να μην πάρουμε καθόλου αποτελέσματα
- Αποτελέσματα που εξαρτώνται από το λεξιλόγιο (vocabulary), δηλ. από τις συγκεκριμένες λέξεις που θα χρησιμοποιήσουμε. Γι' αυτό πολλές φορές για να

έχουμε καλύτερα αποτελέσματα, χρησιμοποιούμε στην ερώτησή μας παρόμοιες σημασιολογικά λέξεις.

- Τα αποτελέσματα είναι απλές Web pages: Αν χρειαζόμαστε πληροφορίες οι οποίες διασκορπίζονται σε πολλά έγγραφα, τότε πρέπει να κάνουμε και πολλές ερωτήσεις για να συγκεντρώσουμε τα σχετικά έγγραφα

Ακόμα και αν η αναζήτησή μας είναι επιτυχημένη και έχουμε πάρει αρκετά αποτελέσματα, ο ίδιος ο χρήστης είναι υπεύθυνος για την ανάκληση των πληροφοριών από αυτά, διαδικασία αρκετά χρονοβόρα. Πολλοί διαμαρτύρονται με τον όρο ανάκτηση πληροφοριών (information retrieval) και αντιπροτείνουν τον όρο εύρεση τοποθεσιών (location finder). Το κυριότερο εμπόδιο για καλύτερη υποστήριξη των χρηστών στις αναζητήσεις τους, είναι το γεγονός ότι το περιεχόμενο του Διαδικτύου δεν μπορεί να επεξεργαστεί αυτοματοποιημένα από μηχανή (machine accessible) . Παρόλο που υπάρχει λογισμικό για την ανάλυση των προτάσεων και τον χωρισμό τους σε λέξεις (parsers), εντούτοις τα περιθώρια στενεύουν όταν θέλουμε να εξάγουμε πληροφορία από τις προτάσεις. Είναι δύσκολο για κάποιο λογισμικό να αντιληφθεί την διαφορά ανάμεσα στις δύο προτάσεις :

" Είμαι καθηγητής της Πληροφορικής "

" Θα σκέφτεσαι ότι είμαι καθηγητής της Πληροφορικής "

Μια λύση σ' αυτό θα μπορούσε να δοθεί από την ανάπτυξη πιο εξελιγμένων εργαλείων για ανάλυση κειμένου που να βασίζονται σε προηγμένες αρχές της Τεχνητής Νοημοσύνης. Αυτή η προσέγγιση παρόλο που ακολουθήθηκε στο παρελθόν δεν προσέδωσε τα αναμενόμενα αποτελέσματα.

Μια εναλλακτική προσέγγιση είναι η αναπαράσταση του περιεχομένου του Web με τρόπο ευκολότερα επεξεργάσιμο από μηχανή (machine processable or machine understandable) Αυτή η προσέγγιση οδηγεί προς την ανάπτυξη του Σημασιολογικού Ιστού (Semantic Web) [1], μιας προσπάθειας που έχει υιοθετηθεί από το World Wide Web Consortium (W3C) [18] και εμπνευστής της είναι ο δημιουργός του WWW την δεκαετία του 80 Tim Berners-Lee. Στόχος αυτής της προσπάθειας αποτελεί ο εμπλουτισμός του Διαδικτύου με στοιχεία ‘ευφυΐας’ , όπως θα φαίνεται στον απλό χρήστη. Δηλ. η αναζήτηση στο Διαδίκτυο δεν θα στηρίζεται πλέον στην περιορισμένης αξίας συντακτική ανάλυση των λέξεων, αλλά στην σημασιολογία των λέξεων και τα αποτελέσματα θα βρίσκονται πιο κοντά σ’ αυτό που πραγματικά επιθυμεί ο χρήστης.

Παρόμοια ζητήματα με την συνολική οργάνωση του Διαδικτύου προκύπτουν και για τις υπηρεσίες Διαδικτύου. Παρόλη την μεγάλη εξάπλωση της χρήσης των υπηρεσιών Διαδικτύου, υπάρχουν ακόμη πολλά ζητήματα ανοικτά τα οποία πρέπει να λυθούν. Ένα σημαντικό πρόβλημα είναι και το παρακάτω:

Σκεφτείτε ότι ζητάτε μια υπηρεσία που να σας λέει το τι καιρό θα κάνει στο νησί της Νισύρου. Βάζετε λοιπόν σε μια μηχανή αναζήτησης υπηρεσιών τις λέξεις κλειδιά: Νίσυρος, Δωδεκάνησα συν τις όποιες άλλες λέξεις χρησιμοποιήσετε για να βρείτε υπηρεσίες μετεωρολογίας. Αν όμως κάποια υπηρεσία παρέχει τον καιρό για την περιοχή του Νοτιοανατολικού Αιγαίου δεν θα βρεθεί μ' αυτές τις λέξεις κλειδιά που χρησιμοποιήσαμε. Κι αν σκεφτείτε ότι μπορεί να μην επιστραφεί καμιά υπηρεσία μ' αυτές τις λέξεις κλειδιά, τότε το πρόβλημα αμβλύνεται.

Η παρούσα εργασία εστιάζει στο συγκεκριμένο πρόβλημα και προσπαθεί να δώσει λύσεις χρησιμοποιώντας οντολογίες, δηλ. κατάλληλες ιεραρχίες εννοιών (concepts) και τις σχέσεις μεταξύ τους π.χ.

Ελλάδα

…..Νοτιοανατολικό Αιγαίο

…..Δωδεκάνησα

…….Νίσυρος

έτσι ώστε ακόμα και αν δεν βρούμε μια κατάλληλη υπηρεσία για την Νίσυρο, να βρούμε μία για τα Δωδεκάνησα και/ή για το Νοτιοανατολικό Αιγαίο κ.ο.κ. Οι υπηρεσίες που προκύπτουν από μια τέτοια προσέγγιση ονομάζονται Υπηρεσίες Σημασιολογικού Ιστού (Semantic Web Services). Στο πλαίσιο των Semantic Web Services χρησιμοποιούμε βασικά στοιχεία του Σημασιολογικού Ιστού, όπως οι οντολογίες και η Λογική. Πριν περιγράψουμε αναλυτικά το περιεχόμενο της παρούσας εργασίας, θα δώσουμε μια σύντομη εισαγωγή σε αυτά τα στοιχεία.

1.5.1. Οντολογίες

Ο όρος Οντολογία προέρχεται από την αρχαία ελληνική φιλοσοφία και ασχολείται με τις έννοιες (κλάσεις) και τα αντικείμενα τα οποία πραγματικά υπάρχουν κι την περιγραφή τους. Για παράδειγμα, η παρατήρηση ότι ο κόσμος αποτελείται από συγκεκριμένα αντικείμενα τα οποία μπορούν να ομαδοποιηθούν σε κλάσεις βασισμένα σε κοινές ιδιότητες, αποτελεί στοιχείο της Οντολογίας.

Στην Πληροφορική, ένας ορισμός της οντολογίας (σύμφωνα με τον Studer [10]) είναι:

Η οντολογία είναι ένας σαφής και τυπικός ορισμός μιας ιδεατής αντίληψης

Γενικά, μια οντολογία περιγράφει ένα ορισμένο πεδίο και περιέχει τους όρους αλλά και τις σχέσεις μεταξύ αυτών των όρων. Οι όροι υποδηλώνουν τις έννοιες ή κλάσεις (concepts) του πεδίου. Για παράδειγμα σε μια οντολογία για την άγρια ζωή στην Αφρική βασικά θέματα αποτελούν οι κλάσεις: σαρκοβόρα ζώα και τα φυτοφάγα ζώα.

Οι σχέσεις περιλαμβάνουν ουσιαστικά ιεραρχίες κλάσεων. Μια ιεραρχία καθορίζει ότι η κλάση Α είναι υποκλάση της Α' αν κάθε αντικείμενο της Α ανήκει και στην Α'. Για παράδειγμα όλα τα λιοντάρια είναι και σαρκοβόρα.

Οι οντολογίες μπορεί επίσης να περιλαμβάνουν:

- Ιδιότητες (το λιοντάρι *τρώει* αντιλόπες)
- Περιορισμούς τιμών (μόνο τα σαρκοβόρα τρώνε κρέας)
- Disjointness axioms (τα λιοντάρια και οι αντιλόπες είναι disjoint δηλ. δεν μπορεί ένα ζώο να ανήκει στις κλάσεις λιοντάρι και αντιλόπη ταυτόχρονα)
- Καθορισμός λογικών σχέσεων μεταξύ των αντικειμένων (κάθε σαρκοβόρο ζώο πρέπει να τρώει ένα τουλάχιστον φυτοφάγο)

Έτσι μια οντολογία προσφέρει μια κοινά κατανοητή περιγραφή ενός πεδίου η οποία μπορεί να ξεπεράσει τις διαφορές στην ορολογία. Σε μια εφαρμογή η λέξη «λιοντάρι» θα πρέπει να μπορεί να ταυτιστεί με τη λέξη «λέων» μιας άλλης εφαρμογής.

Όσον αφορά στο πεδίο των υπηρεσιών, οι οντολογίες βελτιώνουν την ακρίβεια στην ανακάλυψη των υπηρεσιών και μπορούν να εκμεταλλευτούν την ιεραρχία των αντικειμένων για την ανάκτηση υπηρεσιών που προσφέρουν παρεμφερείς λειτουργίες με αυτές που ζητά ο χρήστης.

Η χρησιμοποίηση της ίδιας οντολογίας και από τον service requestor αλλά και από τον service provider, παρόλο που δεν είναι υποχρεωτική απλοποιεί την διαδικασία του ταιριάσματος (matching algorithm), εξαλείφοντας την ανάγκη για ενδιάμεσες μετατροπές. Αυτό, βέβαια, το σενάριο είναι πολύ αισιόδοξο στο ανοικτό περιβάλλον του Διαδικτύου.

Ο κλάδος της Τεχνητής Νοημοσύνης έχει να προσφέρει πολλά στην ανάπτυξη γλωσσών οντολογίας (ontology languages). Οι πιο ευρέως χρησιμοποιούμενες γλώσσες που χρησιμοποιούνται για τον ορισμό οντολογιών στο Web είναι:

- XML : προσφέρει το υπόβαθρο (σύνταξη) για δομημένα έγγραφα αλλά δεν υπάρχουν σημασιολογικά στοιχεία σε αυτά
- XML Schema : είναι μια γλώσσα που περιγράφει την δομή XML documents
- Resource Description Framework (RDF): είναι ένα μοντέλο δεδομένων για αντικείμενα (resources) και των σχέσεων μεταξύ τους
- RDF Schema : είναι μια γλώσσα για την περιγραφή ιδιοτήτων και κλάσεων RDF resources με την χρήση ιεραρχιών αντικειμένων
- Web Ontology Language (OWL): αποτελεί την πιο πλούσια γλώσσα περιγραφής οντολογιών και μ' αυτή θα ασχοληθούμε περισσότερο σε επόμενο κεφάλαιο.

1.5.2. Λογική

Η Λογική είναι ο επιστημονικός κλάδος ο οποίος μελετά τις αρχές του συμπερασμού (reasoning) , της διαδικασίας δηλ. να βγάζουμε συμπεράσματα από υπάρχοντα στοιχεία, και οι ιστορικές αρχές της φτάνουν μέχρι τον Αριστοτέλη.

Ένα σημαντικό στοιχείο της Λογικής είναι ότι μας προμηθεύει με αυτοματοποιημένους αιτιολογητές (automated reasoners) οι οποίοι μπορούν να εξάγουν συμπεράσματα από δεδομένη γνώση και να κάνουν την υπονοούμενη πληροφορία ρητή και σαφή. Ας υποθέσουμε ότι όλοι οι Αθηναίοι είναι και κάτοικοι Αττικής, οι κάτοικοι Αττικής είναι και κάτοικοι Στερεάς Ελλάδας και ο Χαράλαμπος είναι κάτοικος Αττικής. Στην κατηγορηματική λογική αυτές οι πληροφορίες μπορούν να εκφραστούν ως εξής:

ΚάτοικοςΑθήνας(X) → ΚάτοικοςΑττικής(X)

ΚάτοικοςΑττικής(X) → ΚάτοικοςΣτερεάς(X)

ΚάτοικοςΑθήνας(Χαράλαμπος)

Από τα παραπάνω βγάζουμε τα συμπεράσματα :

ΚάτοικοςΑττικής(Χαράλαμπος)

ΚάτοικοςΣτερεάς(Χαράλαμπος)

ΚάτοικοςΑθήνας(X) → ΚάτοικοςΣτερεάς(X)

Το παραπάνω παράδειγμα αφορά γνώση που μπορεί να βρίσκεται εκφρασμένη σε έννοες οντολογίες (π.χ. έννοια *ΚάτοικοςΑττικής*, κλπ.). Η λογική μας βοηθά να ανακαλύψουμε γνώση υπονοούμενη και να βρούμε σχέσεις αλλά και αναντιστοιχίες μεταξύ των concepts σε μια οντολογία.

Προκειμένου να χρησιμοποιήσουμε reasoners με αποτελεσματικό τρόπο θα πρέπει να παρουσιάσουμε τις ήδη υπάρχουσες πληροφορίες με δομημένο τρόπο. Ένας από αυτούς τους τρόπους που έχει υιοθετηθεί και στο περιβάλλον του Web μέσω της OWL είναι οι Περιγραφικές Λογικές (Description Logics - DLs), οι οποίες διαθέτουν ένα λεξιλόγιο για την περιγραφή των concepts και των roles , των σχέσεων δηλ. μεταξύ των στιγμιοτύπων (individuals of concepts). Βασικό χαρακτηριστικό των DL περιγραφών είναι το inclusion axiom (subsumption relation) δηλ. αν ένα concept εμπεριέχει ένα άλλο, το οποίο μας βοηθά να δημιουργήσουμε IS-A ιεραρχίες (taxonomies) από τα concepts. Μπορούμε να δημιουργήσουμε πολύπλοκες περιγραφές καθώς και δηλώσεις γύρω από τον τρόπο με τον οποίο τα concepts σχετίζονται το ένα με το άλλο. (Πίνακας 1, Πίνακας 2) [19]

Πίνακας 1 Main DL constructors (C,D : concepts - R : role)

Constructor	**DL syntax**	**Example**	**Meaning**
Intersection	C⊓D	Young ⊓ Male	Όλα τα στιγμιότυπα τα οποία είναι Young and Male
Union	C⊔D	Young ⊔ Male	Όλα τα στιγμιότυπα τα οποία είναι είτε Young είτε Male
Value restriction	∀R.C	∀ hasInterest.Movies	Όλα τα στιγμιότυπα τα οποία ενδιαφέρονται only in Movies
Existential role quantification	∃R.C	∃ hasInterest.Sports	Όλα τα στιγμιότυπα τα οποία έχουν ενδιαφέρον in Sports (μπορεί να ενδιαφέρονται και για άλλα πράγματα)
Atomic negation	¬C	¬Male	Κάθε στιγμιοτυπο που δεν είναι Male

Πίνακας 2 DL axioms

Axiom	**DL syntax**	**Example**	**Meaning**
Inclusion (subsumption)	C⊑D	Young⊑Person	Κάθε στιγμιότυπο τύπου Young είναι και τύπου Person
Equality	C≡D	Young≡ Teenager	Κάθε Young είναι και Teenager και το αντίστροφο
Disjoint	C⊑¬D	Teenager⊑¬Adult	Κάποιος δεν μπορεί να είναι Teenager και Adult ταυτόχρονα

Παράδειγμα

Η παρακάτω DL περιγραφή αφορά μια περιγραφή ενός primitive concept (δηλ. ενός concept που αποτελείται από αναγκαίες συνθήκες):

C ⊑ Young ⊓ Male ⊓ ∃hasInterest.(Interest ⊓ Music) ⊓ ∀dislikes.(Movies)

Η παραπάνω πρόταση περιγράφει ότι αν ένας άνθρωπος ανήκει στην κατηγορία C τότε είναι νεαρός άνδρας, ένα από τα ενδιαφέροντα του είναι η μουσική, αλλά αντιπαθεί όλων των ειδών τις ταινίες. Το αντίστροφο δεν ισχύει. Στην παρακάτω πρόταση, που

υπάρχει το σύμβολο της ταυτότητας ορίζεται ένα defined concept (που περιλαμβάνει ικανές και αναγκαίες συνθήκες) και ισχύει και το αντίστροφο:

C ≡ Young ⊓ Male ⊓ ∃hasInterest.(Interest ⊓ Music) ⊓ ∀dislikes.(Movies)

Χρησιμοποιώντας τέτοιες περιγραφές ένας reasoner μπορεί να αποφανθεί αν οι πληροφορίες μας είναι συνεπείς (consistent) δηλ. δεν περιέχουν αντιφάσεις και αναντιστοιχίες και μπορεί να αποφασίσει αν ένα concept A είναι πιο γενικό από ένα concept B οπότε λέμε ότι «A subsumes B». Επίσης μπορεί να εκτελέσει classification, κατατάσσοντας ένα καινούριο concept στην κατάλληλη θέση στην ιεραρχία των concepts.

1.6 Διάρθρωση της εργασίας

Στην παρούσα εργασία θα προσπαθήσουμε να εμβαθύνουμε στα Semantic Web Services και στις τεχνολογίες πάνω στις οποίες στηρίζονται. Στο Κεφάλαιο 2 δίνονται δύο χαρακτηριστικά παραδείγματα ανάγκης επέκτασης της ήδη εφαρμοσμένης τεχνολογίας με σημασιολογικές αναφορές, ώστε να έχουμε καλύτερα αποτελέσματα. Προχωράμε με την μελέτη της διαδικασίας ανακάλυψης των Semantic Web Services αλλά και της τροποποιημένης αρχιτεκτονικής που χρησιμοποιούν σε σχέση μα τα παραδοσιακά Web Services.

Το Κεφάλαιο 3 παρουσιάζει αναλυτικά την τεχνολογία SAWSDL για την προσθήκη σημασιολογικών αναφορών σε έγγραφα WSDL και την γλώσσα περιγραφής οντολογιών OWL, που θα χρησιμοποιήσουμε εκτενώς. Επίσης, δίνεται μια σύντομη περιγραφή και για τις εναλλακτικές τεχνολογίες υλοποίησης Semantic Web Services: OWL-S και WSMO framework.

Στο Κεφάλαιο 4 δίνεται μια αναλυτική περιγραφή των προγραμμάτων που αναπτύξαμε για την εύρεση υπηρεσιών Σημασιολογικού Ιστού με το πρότυπο SAWSDL.

Στο κεφάλαιο 5 παρουσιάζονται μια σύντομη σύνοψη της εργασίας, συμπεράσματα, προβλήματα κατά την ανάπτυξη και σκέψεις για την εξέλιξη των Semantic Web Services.

.

ΚΕΦΑΛΑΙΟ 2

ΥΠΗΡΕΣΙΕΣ ΣΗΜΑΣΙΟΛΟΓΙΚΟΥ ΙΣΤΟΥ (SEMANTIC WEB SERVICES)

2.1 Εισαγωγή

Το Διαδίκτυο υποστηρίζει τις κατανεμημένες εφαρμογές και ο όρος αρχιτεκτονική προσανατολισμένη στις υπηρεσίες (service-oriented architecture, SOA) δημιουργήθηκε σαν ένας γενικός όρος περιγραφής της τεχνολογίας των Web Services. Το Διαδίκτυο αποτελεί μια τεράστια δεξαμενή δεδομένων και οι Web Services απλώνονται ταχύτατα. Για να αποκτήσουν όμως νόημα και αξία όλα αυτά τα δεδομένα και οι υπηρεσίες, αναπτύχθηκε ο Σημασιολογικός Ιστός ο οποίος στηρίζεται στη Λογική και στην αναπαράσταση της γνώσης, έτσι ώστε ο χρήστης να μπορεί να βρει την πληροφορία που πραγματικά χρειάζεται Η ανακάλυψη και ο συνδυασμός πληροφορίας αποτελούν μόνο το ένα σκέλος του οράματος του Σημασιολογικού Ιστού. Οι υπολογιστές θα πρέπει να μπορούν να βρίσκουν και να συνδυάζουν υπηρεσίες, ώστε να διευκολύνουν τον χρήστη και να εκμεταλλεύονται τον τεράστιο πλούτο υπηρεσιών και δεδομένων του Διαδικτύου.

Τα Semantic Web Services στοχεύουν σε μια αναπτυγμένη τεχνολογία επόμενης γενιάς του Web συνδυάζοντας τεχνολογίες Semantic Web και Web Services , μετατρέποντας το Διαδίκτυο από μία τεράστια δεξαμενή αποθηκευμένων πληροφοριών κατανοητών από τον άνθρωπο, σε ένα παγκόσμιο σύστημα κατανεμημένων υπολογιστικών μονάδων.

2.2 Παραδείγματα Semantic Web Services

2.2.1. B2C Ηλεκτρονικό Εμπόριο

Ένα τυπικό σενάριο Ηλεκτρονικού Εμπορίου αφορά έναν χρήστη ο οποίος επισκέπτεται διάφορα ηλεκτρονικά καταστήματα, ψάχνει τις προσφορές τους, επιλέγει και παραγγέλνει προϊόντα. Αν όλα αυτά γίνουν χειροκίνητα τότε η διαδικασία είναι πολύ χρονοβόρα. Έτσι έχουν αναπτυχθεί έξυπνοι πράκτορες (agents) οι οποίοι αναλαμβάνουν να αυτοματοποιήσουν όλη αυτή τη διαδικασία. Μειονέκτημα αποτελεί η αναζήτηση με βάση λέξεις-κλειδιά π.χ. η αναζήτηση της τιμής ενός προϊόντος με βάση την λέξη "price" δεν αποτελεί και την καλύτερη τακτική, αφού κάποιο κατάστημα μπορεί να χρησιμοποιήσει την φράση "amount of money" ή κάτι παρόμοιο για να δηλώσει την τιμή του προϊόντος. Εξαιτίας αυτού περιορίζονται και οι πληροφορίες που εξάγονται.

Πληροφορίες όπως έξοδα αποστολής, χρόνοι παράδοσης, επίπεδο ασφάλειας κ.α. συνήθως δεν μπορούν να αναχθούν. Η εφαρμογή του Semantic Web μπορεί να επιτρέψει την ανάπτυξη πρακτόρων (software agents) που να είναι ικανοί να αναλύουν και να εξάγουν πληροφορίες.

Ανάλογες εφαρμογές μπορεί να προκύψουν και για το B2B Ηλεκτρονικό εμπόριο, όπου το Semantic Web θα επιτρέψει στις επιχειρήσεις να αναπτύξουν τις εμπορικές τους σχέσεις. Οι διαφορές στην ορολογία θα λυθούν με την χρήση αφηρημένων μοντέλων δεδομένων και τα δεδομένα θα ανταλλάσσονται χρησιμοποιώντας υπηρεσίες μετάφρασης.

2.2.2. Έξυπνοι πράκτορες (Semantic web agents)

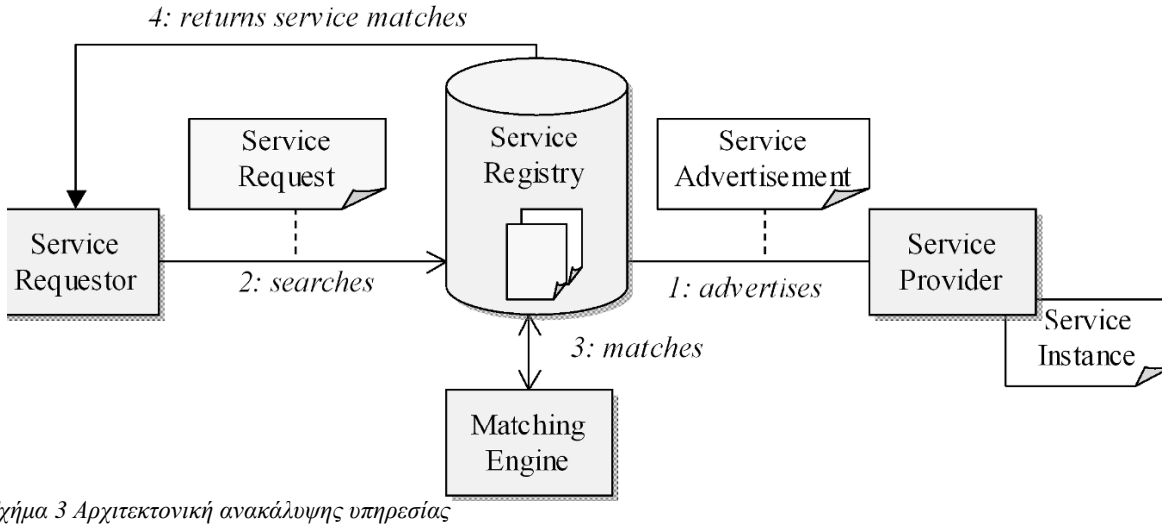
Σκεφτείτε το παρακάτω σενάριο, το οποίο δεν είναι επιστημονικής φαντασίας, αλλά επιμέρους λύσεις σε όλα τα σημαντικά ζητήματα που θέτει έχουν δοθεί :

Ο κ. Παπαδόπουλος θέλει να κλείσει ένα ραντεβού με γιατρό για ένα πρόβλημα που αντιμετωπίζει στο γόνατό του. Αναθέτει, λοιπόν στον πράκτορα που μόλις έχει εγκαταστήσει στον υπολογιστή του να του κλείσει ένα ραντεβού. Ο πράκτορας ελέγχει για όλους τους ορθοπεδικούς που είναι συμβεβλημένοι με την ασφάλεια του πελάτη του και βρίσκονται σε κοντινή απόσταση από αυτόν (σε ακτίνα 10 χλμ) και εξετάζει την ικανότητα καθενός απ' αυτούς στηριζόμενος σε ανεξάρτητες υπηρεσίες αξιολόγησης. Μετά προσπαθεί να ταιριάξει τις ελεύθερες ώρες του γιατρού με αυτές του κ. Παπαδόπουλου. Μέσα σε λίγα λεπτά έχει επιστρέψει δύο προτάσεις. Ο κ. Παπαδόπουλος ζητά εξηγήσεις από τον πράκτορά του για το πώς ανέκτησε τον βαθμό ικανότητας του γιατρού, πώς διαπραγματεύτηκε την αμοιβή του γιατρού κ.α. Φυσικά ο πράκτορας είναι έτοιμος να αιτιολογηθεί μέσα σε λίγα λεπτά.

2.3 Ανακάλυψη Υπηρεσιών Σημασιολογικού Ιστού (SWS Discovery)

Το πιο σημαντικό ίσως κομμάτι μιας υπηρεσίας Διαδικτύου (web service) είναι το στάδιο της ανακάλυψης. Μια υπηρεσία δεν μπορεί να χρησιμοποιηθεί από έναν χρήστη αν δεν βρεθεί πρώτα και αν δεν πειστεί ο χρήστης ότι ικανοποιεί τις ανάγκες του.

Οι περιγραφές των web services στηρίζονται σε λέξεις κλειδιά και την ίδια τακτική ακολουθούν και οι αιτήσεις. Εφόσον, λοιπόν, η ανακάλυψη των υπηρεσιών στηρίζεται σε συγκρίσεις λέξεων κλειδιών, δεν αποτελεί κι έναν έξυπνο τρόπο ανάκτησης πληροφορίας Το παρακάτω σχήμα εξηγεί την τυπική αρχιτεκτονική ανακάλυψης υπηρεσιών



Σχήμα 3 Αρχιτεκτονική ανακάλυψης υπηρεσίας

Ο service provider προσφέρει κάποιες υπηρεσίες (service instances) τις οποίες φροντίζει να διαφημίσει (service advertisement) σε έναν κατάλογο υπηρεσιών (service registry) που μπορεί να προσπελαστεί από τους χρήστες. Ο κατάλογος αυτός μπορεί να είναι δημόσιος ή ιδιωτικός. Ο service requestor ψάχνει αυτόν τον κατάλογο και ανακτά τις διαφημίσεις των υπηρεσιών που τον ενδιαφέρουν. Αυτό γίνεται με την χρήση αλγορίθμων ταιριάσματος(matchmaking algorithms) που προσπαθούν να ταιριάξουν τα service requests με τα service advertisments. Εάν βρεθεί κάποια υπηρεσία τότε επιστρέφεται στον requestor ένα σύνολο πληροφοριών για να μπορέσει να την εκτελέσει(π.χ. την διεύθυνση και την θύρα της υπηρεσίας, το πρωτόκολλο επικοινωνίας κ.α). Προφανώς ο αλγόριθμος ταιριάσματος παίζει σημαντικό ρόλο στην αποτελεσματικότητα της ανακάλυψης υπηρεσιών.

Όπως αναφέραμε και σε προηγούμενο κεφάλαιο η περιγραφή της υπηρεσίας επιτυγχάνεται με το WSDL document , το οποίο αποθηκεύεται σε registries ,πρότυπο των οποίων αποτελεί το Universal Description,Discovery and Integration(UDDI) [2]. Κάθε provider είναι ελεύθερος να περιγράψει την λειτουργικότητα της υπηρεσίας του με αδόμητο ή ημί-δομημένο τρόπο και μπορεί να φτάσει σε οποιοδήποτε επίπεδο λεπτομέρειας επιθυμεί. Η χρήση keyword based matching δεν οδηγεί σε ποιοτικά αποτελέσματα διότι θα πρέπει να χρησιμοποιηθεί ακριβώς η ίδια ορολογία στην διαφήμιση και στην αίτηση για την υπηρεσία, και να αντιμετωπιστούν τα προβλήματα της πολυσημίας (μία λέξη να έχει πολλές σημασίες) και της ασάφειας. Επίσης μ' αυτό τον τρόπο αναζήτησης δεν λαμβάνονται υπόψη περιορισμοί και μη λειτουργικά

χαρακτηριστικά της υπηρεσίας όπως Quality of Service (QoS) ενώ δεν υποστηρίζεται το έμμεσο ταίριασμα (indirect matching) Δηλ. δεν μπορούν να δημιουργηθούν αλυσίδες υπηρεσιών όταν καμία υπηρεσία δεν μπορεί από μόνη της να ικανοποιήσει την αίτηση του χρήστη παρόλο που μια τέτοια ίσως μπορεί.

Προκειμένου να αυτοματοποιηθεί ο συνολικός κύκλος ζωής μια υπηρεσίας από τη διαφήμισή της μέχρι και την ανάκτησή της, μπορούν να χρησιμοποιηθούν σημασιολογικές σημειώσεις (semantic service annotations). Στην αρχιτεκτονική SWS discovery τα στοιχεία WS discovery διατηρούνται, αλλά υλοποιούνται με διαφορετικό τρόπο για να προσφέρουν επιπρόσθετη λειτουργικότητα.

Η διαφήμιση της υπηρεσίας στηρίζεται σε συγκεκριμένα Service Annotation Ontologies τα οποία καθορίζουν σημασιολογικά μοντέλα για την περιγραφή διαφόρων διαστάσεων της υπηρεσίας (λειτουργικότητα, λεπτομέρειες εκτέλεσης). Οι όροι που χρησιμοποιούνται για την περιγραφή στην διαφήμιση της υπηρεσίας «δανείζονται» από κοινά διαμοιραζόμενα λεξιλόγια, που αποτελούν οντολογίες πεδίου ενδιαφέροντος (domain ontology).

Τα Service annotation ontologies μπορεί να θεωρηθούν σαν περιγραφικά μοντέλα των σημασιολογικών εννοιών μιας υπηρεσίας. Αυτές οι οντολογίες καθορίζουν ένα σύνολο από χαρακτηριστικών ιδιοτήτων της υπηρεσίας με σημαντικότερες τις λεγόμενες IOPE (Inputs, Outputs, Preconditions, Effects) :

- Inputs : καθορίζουν την πληροφορία που χρειάζεται η υπηρεσία για την εκτέλεσή της
- Outputs: καθορίζουν την πληροφορία που παράγεται από την εκτέλεση της υπηρεσίας
- Preconditions: καθορίζουν τις συνθήκες που πρέπει να ικανοποιηθούν προκειμένου να εκτελεστεί με επιτυχία η υπηρεσία
- Effects: καθορίζουν τις επιπτώσεις από την εκτέλεση της υπηρεσίας

Οι τιμές των ιδιοτήτων IOPE καθορίζονται επίσης με τη χρήση domain ontologies.

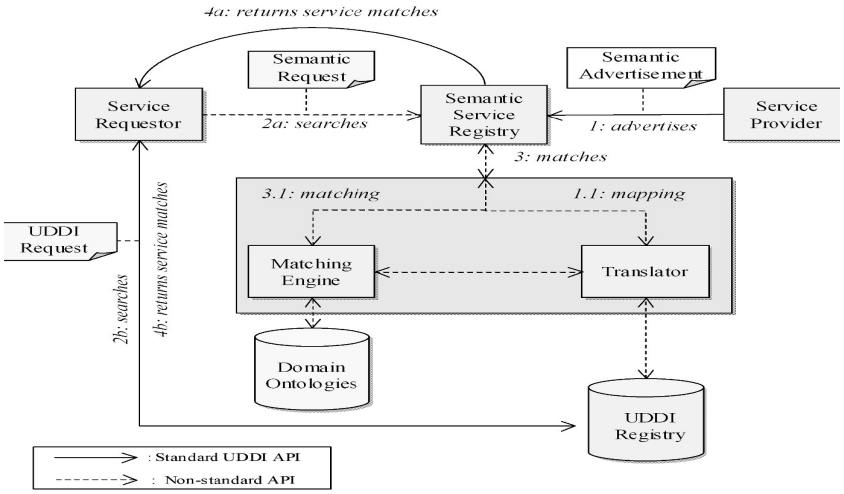
Οι πιο πολλές Service annotation ontologies χωρίζονται σε τρία λογικά μέρη :

- Service profile: μια μορφή απλού κειμένου που περιγράφει την υπηρεσία, μια λειτουργική περιγραφή της υπηρεσίας (σημασιολογικές πληροφορίες σχετικά με τα IOPE χαρακτηριστικά) και επιπρόσθετες παράμετροι που περιγράφουν χαρακτηριστικά της υπηρεσίας όπως QoS ή διάρκεια εκτέλεσης της υπηρεσίας.

- Service model: λεπτομερής περιγραφή της λειτουργίας της υπηρεσίας
- Service grounding: λεπτομερής περιγραφή των μηνυμάτων που χρειάζονται για την πρόσβαση στην υπηρεσία (πρωτόκολλο επικοινωνίας, μορφή μηνυμάτων)

2.3.1. Αρχιτεκτονική για SWS Discovery

Στην εργασία αυτή θα χρησιμοποιήσουμε μια κεντρικοποιημένη αρχιτεκτονική ανακάλυψης [1]. Η αρχιτεκτονική αυτή στηρίζεται στην ιδέα της προσάρτησης σημασιολογικών περιγραφών σε στοιχεία του standard UDDI (όπως είναι τα έγγραφα WSDL) . Σημαντικό στοιχείο μιας τέτοιας αρχιτεκτονικής αποτελεί η μηχανή ταιριάσματος (matching machine) και ο μεταφραστής (translator) , οι οποίοι δουλεύουν ως εξής: όταν λαμβάνουν ένα service advertisement που περιέχει σημασιολογικές σημειώσεις ο μεταφραστής το μεταφράζει σε τυπικό στοιχείο UDDI (tModel entry : WSDL) έτσι ώστε να μπορεί να αποθηκευτεί στην UDDI registry. Όταν έρχεται ένα service request η μηχανή ταιριάσματος παίρνει την σημασιολογική περιγραφή της αίτησης και με την υποστήριξη της κατάλληλης οντολογίας προσπαθεί να βρει μια αποθηκευμένη υπηρεσία που να ταιριάζει με την αίτηση. Η προτεινόμενη αρχιτεκτονική συνδυάζει δύο τρόπους για ανακάλυψη υπηρεσιών: χρησιμοποιώντας το τυπικό UDDI keyword-based matching αλλά κι ένα matching που εκμεταλλεύεται τις σημασιολογικές σημειώσεις.



Σχήμα 4 Κεντρικοποιημένη αρχιτεκτονική ανακάλυψης [1]

Στην παρούσα εργασία θα γίνει η εξής αλλαγή: η UDDI registry θα αντικατασταθεί από μια σχεσιακή Βάση Δεδομένων στην οποία θα αποθηκεύονται οι υπηρεσίες , τα operations, τα inputs και τα outputs κάθε operation μαζί με τις σημασιολογικές τους αναφορές. Επιπλέον η αίτηση θα γίνεται απευθείας στη μηχανή ταιριάσματος απλώς συμπληρώνοντας ο χρήστης τις εισόδους και τις εξόδους που επιθυμεί από μια

υπηρεσία, σε απλά πεδία μιας φόρμας (ή επιλέγοντας από προκαθορισμένες εισόδους και εξόδους) στηριζόμενος σε κάποιες οντολογίες.

Ο αλγόριθμος ταιριάσματος εκμεταλλεύεται το service profile των SWS δηλ. προσπαθεί να ταιριάξει τις IOPE ιδιότητες (Inputs, Outputs, Preconditions, Effects) του advertisement με του request (στην παρούσα υλοποίηση χρησιμοποιούμε μόνο τα Inputs και Outputs, κάτι που συνηθίζεται και στη βιβλιογραφία χάριν απλότητας). Αυτό είναι απόλυτα λογικό, αφού υπηρεσίες με τις όμοιες IOPE ιδιότητες μπορούν να χαρακτηριστούν γενικά όμοιες. Στον matchmaking algorithm δεν είναι ρεαλιστικό να περιμένουμε ακριβή αντιστοίχηση μεταξύ αίτησης και διαφήμισης, αλλά εισάγουμε ένα καινούριο μέγεθος, τον Βαθμό Ταιριάσματος (Degree of Match, DoM). Ο DoM μπορεί να χρησιμοποιηθεί για να μας δείξει πόσο παρόμοιες μπορεί να είναι δυο οντότητες χρησιμοποιώντας κάποια μετρική. Αυτές οι οντότητες μπορεί να είναι υπηρεσίες, IOPE ιδιότητες ή και operations. Έτσι ένας matchmaking algorithm υπολογίζοντας το DoM για διάφορες οντότητες μπορεί να βαθμολογήσει μια υπηρεσία ως προς την ομοιότητά της με μια αιτούμενη υπηρεσία.

Υπάρχουν δύο κατηγορίες για matchmaking algorithm:

- Άμεση : η αίτηση υπηρεσίας συγκρίνεται με μία απλή διαφήμιση υπηρεσίας.
- Έμμεση : η αίτηση υπηρεσίας συγκρίνεται από ένα σύνολο υπηρεσιών που μπορούν να δημιουργούν αλυσίδες υπηρεσιών.

Ο αλγόριθμος που θα εφαρμόσουμε υποστηρίζει το DoM και θεωρεί ότι τόσο η διαφήμιση της υπηρεσίας όσο και η αίτηση της υπηρεσίας χρησιμοποιούν την ίδια οντολογία. Θα στηριχτεί στην ιδέα ότι αν μια διαφήμιση ταιριάζει με μια αίτηση, τότε όλες οι έξοδοι της διαφήμισης ταιριάζουν με τις εξόδους της αίτησης και όλες οι είσοδοι της διαφήμισης ταιριάζουν με τις εισόδους της αίτησης. Λαμβάνουμε υπόψη δηλ. μόνο τις εισόδους και τις εξόδους. Στα πλαίσια της εργασίας και σύμφωνα με κοινές πρακτικές ο βαθμός ταιριάσματος (DoM) ανάμεσα σε δύο εισόδους ή εξόδους εξαρτάται από την σχέση στην ιεραρχία της οντολογίας των concepts που αντιστοιχούν σ' αυτές τις εισόδους ή εξόδους. Ο παρακάτω πίνακας δείχνει τον Βαθμό Ταιριάσματος (DoM) για τις εισόδους request input (req.i) και advertisement input (adv.i) και τις εξόδους request output (req.o) και advertisement output (adv.o). [1]

Πίνακας 3 Βαθμοί Ταιριάσματος

Subsumption relation	**Meaning/Potential problems**

req.i subsumes adv.i	Για την εκτέλεση της υπηρεσίας χρειάζεται περισσότερο εξειδικευμένη πληροφορία εισόδου
adv.i subsumes req.i	Η αίτηση περιλαμβάνει όλη την απαραίτητη πληροφορία για την εκτέλεση της υπηρεσίας
req.o subsumes adv.o	Η έξοδος της υπηρεσίας περιέχει ένα μέρος από τα επιθυμητά αποτελέσματα. Για να ικανοποιηθούν πλήρως οι ανάγκες του αιτούντος θα χρειαστεί μια σύνθεση τέτοιων υπηρεσιών
adv.o subsumes req.o	Η υπηρεσία ίσως να μην επιστρέφει τα επιθυμητά αποτελέσματα, και σε ακραίες καταστάσεις εντελώς άσχετα.

Ο αλγόριθμος συγκρίνει κάθε req.i με adv.i και κάθε req.o με adv.o χρησιμοποιώντας το subsumption matching δηλ. ένα concept εμπεριέχει ένα άλλο αν το πρώτο είναι πιο γενικό από το δεύτερο και από αυτή την σύγκριση προκύπτει ένας βαθμός (DoM). Την βαθμολογία αυτή την έχουμε επιλέξει αυθαίρετα και αποτελείται από 4 βαθμούς: 0, 5, 7 και 10 όπως φαίνεται από τον Πίνακα 4

Πίνακας 4 Subsumption relations μεταξύ εισόδων εξόδων

Degree of Match	**Συνθήκες σύγκρισης (αντίστοιχη αριθμητική τιμή για DoM)**
EXACT	Αν το req.o είναι ισοδύναμο με το adv.o (10)
PLUGIN	Αν το adv.o subsumes req.o (7)
SUBSUMES	Αν το req.o subsumes adv.o (5)
FAIL	Αν δεν υπάρχει subsumption relationshipμεταξύ req.o και adv.o (0)

Θα πρέπει να λάβουμε επίσης υπόψη μας ότι το output matching είναι πιο σημαντικό από το input matching, αφού ο requestor ψάχνει υπηρεσίες που να του παρέχουν συγκεκριμένες λειτουργίες σαν outputs και δεν ενδιαφέρεται και πολύ για τις εισόδους που δέχονται αυτές οι υπηρεσίες.

Ας θεωρήσουμε για παράδειγμα μία υπηρεσία που παρέχει την θερμοκρασία σε βαθμούς Κελσίου στην περιοχή της Νισύρου. Αν ο χρήστης ζητήσει την θερμοκρασία για την περιοχή των Δωδεκανήσων, τότε επειδή η Νίσυρος ανήκει (εμπεριέχεται) στα Δωδεκάνησα η υπηρεσία θα βαθμολογηθεί με 5 (η έξοδος της αίτησης, τα Δωδεκάνησα, εμπεριέχουν την έξοδο της υπηρεσίας ,την Νίσυρο). Αν όμως ο χρήστης ζητήσει τη

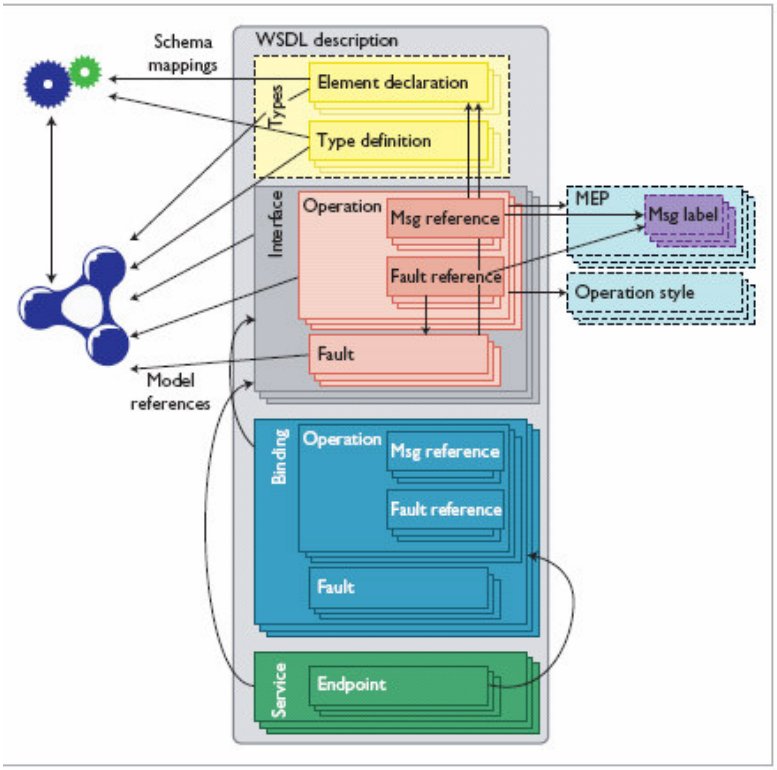
θερμοκρασία για την περιοχή του Μαντρακίου που είναι η πρωτεύουσα του νησιού, τότε η υπηρεσία θα βαθμολογηθεί με 7. Φυσικά αν ζητηθεί η θερμοκρασία για την Νίσυρο θα έχουμε απόλυτη ταύτιση και βαθμό 10, ενώ αν ζητηθεί η θερμοκρασία για την Αθήνα ο βαθμός ταιριάσματος θα είναι 0.

ΚΕΦΑΛΑΙΟ 3

ΤΕΧΝΟΛΟΓΙΕΣ ΓΙΑ ΤΗΝ ΥΛΟΠΟΙΗΣΗ SEMANTIC WEB SERVICES

3.1 SAWSDL (Semantic Annotations for WSDL)

Το SAWSDL αποτελεί ένα σύνολο ιδιοτήτων που μπορούμε να προσθέσουμε στο Web Services Description Language and XML Schema definition language [14], έτσι ώστε να ενσωματώσουμε αναφορές σε σημασιολογικά μοντέλα όπως είναι οι οντολογίες. Το SAWSDL δεν αποτελεί μια ξεχωριστή γλώσσα αλλά μας παρέχει τους μηχανισμούς για να συσχετίσουμε στοιχεία του WSDL εγγράφου με concepts σημασιολογικών μοντέλων (π.χ. οντολογίες) όπως φαίνεται στο παρακάτω σχήμα. Αυτές οι σημασιολογικές αναφορές μπορούν να ξεκαθαρίσουν τις ασάφειες στην περιγραφή των Web Services κατά την διαδικασία της ανακάλυψης και της σύνθεσής τους.



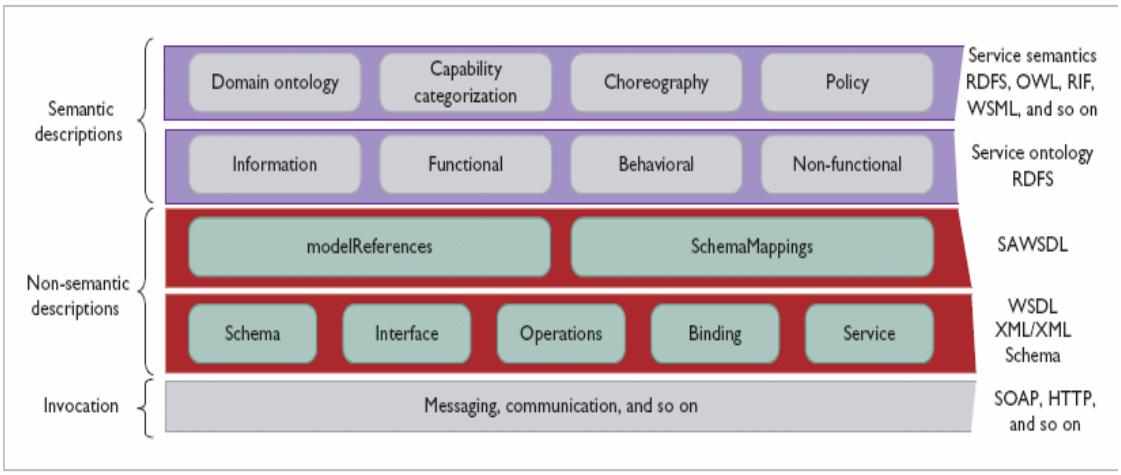
Σχήμα 5 Η WSDL με σημασιολογικές σημειώσεις [14]

Το SAWSDL αποτελεί το πρώτο και σημαντικότατο βήμα του World Wide Web Consortium (W3C) για την προτυποποίηση της τεχνολογίας των SWS[11] . Η διαδικασία της προτυποποίησης από το W3C απαιτεί κάθε προτεινόμενη τεχνολογία να ελέγχεται για την εφαρμοσιμότητά της. Κάθε χαρακτηριστικό της νέας τεχνολογίας θα πρέπει να είναι λειτουργικό σε τουλάχιστον μία υλοποίηση. Η ομάδα εργασίας που επιθυμεί να

προωθήσει μία τεχνολογία θα πρέπει να δημιουργήσει μία αναφορά υλοποίησης. Η ομάδα εργασίας για το SAWSDL έχει δημιουργήσει μια τέτοια αναφορά η οποία δείχνει πολλές κατηγορίες υλοποίησης.

Οι άμεσες κατηγορίες υλοποίησης αποτελούνται από API για parsers που χρησιμοποιούν οι εφαρμογές για τις σημασιολογικές αναφορές, και από εργαλεία για τον εμπλουτισμό των εγγράφων wsdl με σημασιολογικές σημειώσεις. Στην πρώτη κατηγορία ανήκουν η Woden API for WSDL 2.0 και η WSDL4J API for WSDL 1.1 [9], η οποία και χρησιμοποιείται από αυτή την εργασία. Στην δεύτερη κατηγορία ανήκουν δύο GUI εργαλεία : το Radiant [20] από το Πανεπιστήμιο της Georgia και το WSMO Studio από την Ontotext [13]

Σαν πρότυπο το SAWSDL[3] αποτελεί μια σταθερή βάση πάνω στην οποία μπορεί να αναπτυχθούν άλλα SWS πλαίσια εργασίας (frameworks), όπως το Web Service Modeling Ontology (WSMO, www.wsmo.org) και το OWL-S (www.daml.org/services/owl-s). Οι βασικές τεχνολογίες υλοποίησης για τα Web Services είναι η SOAP και η WSDL. Η SAWSDL προσθέτει δείκτες σε σημασιολογίες οι οποίες είναι σημαντικές για να επιτύχουμε αυτοματισμό στα διάφορα στάδια του κύκλου ζωής μιας υπηρεσίας. Η παρακάτω εικόνα δείχνει την στοίβα των επιπέδων περιγραφής των Web Services. Η βάση της στοίβας είναι η WSDL και οι συνδέσεις της με τις τεχνολογίες επικοινωνίας χαμηλού επιπέδου, ιδιαίτερα την HTTP και την SOAP. Πάνω από την WSDL βρίσκονται οι σημασιολογικές σημειώσεις που χρησιμοποιούνται από τις οντολογίες της υπηρεσίας. Στην κορυφή βρίσκονται η εφαρμογή, domain ontologies και άλλες σημασιολογικές περιγραφές.



Σχήμα 6 Η στοίβα ορισμού των επεκταμένων με σημασιολογικές σημειώσεις υπηρεσιών Διαδικτύου [14]

Μπορούμε να προσθέσουμε πληροφορίες κατηγοριοποίησης σε WSDL interfaces και operations, να προσθέσουμε σημασιολογικές σημειώσεις (annotations) σε schema types ώστε να χρησιμοποιηθούν στο web service discovery και composition και να αντιστοιχίσουμε schema types με οντολογίες. Περιληπτικά η ορολογία που χρησιμοποιεί η συγκεκριμένη τεχνολογία, είναι η παρακάτω:

- Σημασιολογικό μοντέλο (Semantic model) : είναι ένα σύνολο αναπαραστάσεων κατανοητών από μηχανή που μοντελοποιεί μια περιοχή γνώσης
- Concept : ένα στοιχείο του σημασιολογικού μοντέλου
- Σημασιολογική σημείωση (Semantic annotation) : επιπρόσθετη πληροφορία που εμπεριέχεται σε ένα έγγραφο και χρησιμοποιεί τα concepts ενός σημασιολογικού μοντέλου για την περιγραφή τμημάτων του εγγράφου
- Σημασιολογία (Semantics) : το σύνολο των concepts που χρησιμοποιούνται στις σημειώσεις

Ένα παράδειγμα ενός αρχείου SAWSDL είναι ένα έγγραφο WSDL αλλά στο πεδίο Types υπάρχουν σημασιολογικές αναφορές [Παράρτημα 1] :

```
<xs:element name="getPasta">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="0" name="param0"
                            nillable="true"
                            sawsdl:modelReference="http://www.owl-
ontologies.com/2008/Ontology1209488954.owl#Fish" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
```

Οι δύο βασικοί μηχανισμοί για σημασιολογικές σημειώσεις είναι οι παρακάτω:

- modelReference : είναι οι γραμμοσκιασμένες περιοχές του παραπάνω εγγράφου και καθορίζουν μια αντιστοιχία μεταξύ ενός στοιχείου WSDL ή XML Schema και ενός concept σε μια οντολογία
- οι ιδιότητες liftingSchemaMapping και loweringSchemaMapping οι οποίες προστίθενται στις δηλώσεις στοιχείων XML Schema και στους ορισμούς τύπων για να καθορίζουν αντιστοιχίες μεταξύ semantic data και XML.

Πίνακας 5 Βασικοί μηχανισμοί για Semantic Annotation

Όνομα	**Περιγραφή**
modelRefernce	Μια λίστα από concepts σημασιολογικών μοντέλων
liftingSchemaMapping	Μια λίστα από δείκτες σε εναλλακτικούς data lifting μετασχηματισμούς
loweringSchemaMapping	Μια λίστα δεικτών σε εναλλακτικούς data lowering μετασχηματισμούς
attrExtensions	Προσαρτά επιπλέον ιδιότητες

Στην παρούσα εργασία θα χρησιμοποιήσουμε μόνο τον πρώτο μηχανισμό. Στο παραπάνω παράδειγμα μία χρήση του modelReference μηχανισμού είναι η εξής:

```
<xs:element name="getVegetableResponse">
    <xs:complexType>
        <xs:sequence>
            <xs:element minOccurs="0" name="return"
                nillable="true"
                sawsdl:modelReference="http://www.owl-
ontologies.com/2008/Ontology1209488954.owl#Vegetables" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Το στοιχείο *getVegetableResponse* είναι τύπου String και αντιστοιχεί στο concept *Vegetables* της οντολογίας http://www.owl-ontologies.com/2008/Ontology1209488954.owl. Αν μια αίτηση, για παράδειγμα, έχει σαν είσοδο της το ίδιο concept της ίδιας οντολογίας τότε θα έχουμε ένα ακριβές ταίριασμα.

3.2 Εναλλακτικοί τρόποι περιγραφής SWS

Όπως ήδη αναφέρθηκε,η τεχνολογία που θα χρησιμοποιήσουμε στην παρούσα εργασία για να περιγράψουμε SWS είναι η SAWSDL, η οποία προσθέτει σημασιολογικές σημειώσεις σε WSDL documents με αναφορές σε εξωτερικές οντολογίες και περιγράψαμε με αρκετές λεπτομέρειες στο προηγούμενο κεφάλαιο. Εδώ θα αναφερθούμε σε κάποιες άλλες εναλλακτικές τεχνολογίες για την περιγραφή των SWS για λόγους πληρότητας

Διάφορες υψηλού επιπέδου οντολογίες για την περιγραφή των υπηρεσιών έχουν ήδη προταθεί. Η πρώτη απ' αυτές ήταν η DAML-S [17] η οποία βασίστηκε στην γλώσσα καθορισμού οντολογίας DAML+OIL . Παρόλα αυτά, η ευρεία αποδοχή της Web Ontology Language (OWL) , οδήγησε στην αντικατάσταση της προηγούμενης με την OWL-S [16]. Την ίδια στιγμή διάφορες ομάδες επιστημόνων έχουν καταλήξει στις

τεχνολογίες WSDL-S, και SWSO. Ας ρίξουμε μια ματιά στις δύο κυριότερες τεχνολογίες, αυτές των WSMO και OWL-S

3.2.1 WSMO

Το WSMO (Web Service Modeling Ontology) είναι μια οντολογία για να περιγράφουμε σημασιολογικά τις υπηρεσίες Διαδικτύου [15]. Στηριγμένο πάνω στο WSMF (Web Service Modeling Framework), επεκτείνει αυτό το πλαίσιο εργασίας και δημιουργεί μια τυποποιημένη γλώσσα και οντολογία. Το WSMF αποτελείται από 4 διαφορετικά κύρια στοιχεία: ι) οντολογίες που παρέχουν τα αντικείμενα (concepts) και τις σχέσεις τους (relationships) ιι) σκοποί και επιδιώξεις του χρήστη, όπως τα προβλήματα που πρέπει να λυθούν με τα Web Services ιιι) περιγραφές των Web Services και ιν) ενδιάμεσο λογισμικό (mediator) ώστε να επιλυθούν προβλήματα διαλειτουργικότητας (interoperability). Το WSMO στηρίζεται στις ακόλουθες βασικές αρχές:

- Συμμόρφωση με το Web (Web Compliance): χρησιμοποιεί το URI για την διακριτή αναγνώριση των δεδομένων (resources) και υιοθετεί τα Namespaces για να δηλώνει σταθερούς χώρους πληροφοριών
- Βασίζεται στις οντολογίες: όλες οι περιγραφές των resources και τα δεδομένα που ανταλλάσσονται κατά την διάρκεια της υπηρεσίας βασίζονται σε οντολογίες
- Χρήση Ενδιάμεσου (Mediation): προκειμένου να αντιμετωπιστεί η ετερογένεια στα δεδομένα, στις οντολογίες, στα πρωτόκολλα και στις διεργασίες που είναι φυσικό να εμφανίζονται σε ανοικτά περιβάλλοντα, γίνεται χρήση ενδιάμεσων
- Διαχωρισμός των οντολογικών ρόλων: δηλ. οι χρήστες μπορεί να βρίσκονται και να αναζητούν διαφορετικό θεματικό περιεχόμενο από αυτό που προσφέρει η υπηρεσία, οπότε και το WSMO πρέπει να τα διαφοροποιήσει
- Περιγραφή - Υλοποίηση: Το WSMO διαφοροποιεί την διαδικασία περιγραφής της υπηρεσίας από την διαδικασία της υλοποίησης της

Τα παρακάτω στοιχεία δημιουργούν το μοντέλο του WSMO:

- Οντολογίες : παρέχουν την ορολογία και τα σημασιολογικά στοιχεία
- Web Services: παρέχουν μια σημασιολογική περιγραφή των υπηρεσιών, συμπεριλαμβανομένων των λειτουργικών και μη λειτουργικών ιδιοτήτων (functional and non-functional properties) και των θεμάτων που αφορά την διαλειτουργικότητά τους (interoperability)

- Οι σκοποί δηλ. οι επιδιώξεις και οι απαιτήσεις του χρήστη από μία υπηρεσία
- Οι ενδιάμεσοι (mediators) : οι συνδετικοί ιστοί μεταξύ ετερογενών συστατικών

Σε συνδυασμό με τα παραπάνω το WSMO προσφέρει και ένα σύνολο από μη λειτουργικές ιδιότητες πυρήνα που μπορεί να χρησιμοποιηθούν από όλα τα στοιχεία του μοντέλου και αφορούν θέματα όπως είναι η αξιοπιστία, η ασφάλεια και η εμπιστοσύνη.

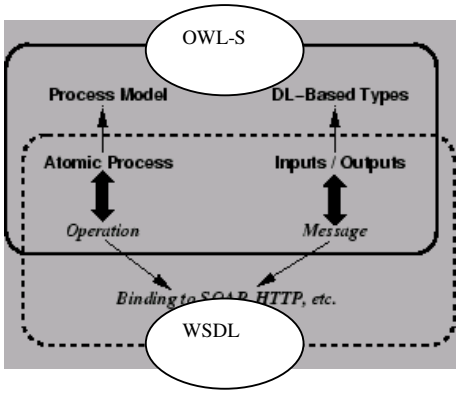
3.2.2 OWL-S

Η OWL-S είναι μία οντολογία βασισμένη στο πλαίσιο εργασίας της OWL για το Semantic Web.[16]. Βοηθά τους χρήστες αλλά και τους πράκτορες λογισμικού (software agents) να ανακαλύψουν, να ανακαλέσουν και να συνθέσουν υπηρεσίες από το Διαδίκτυο. Μια ανώτερη οντολογία (upper level ontology) θα πρέπει να παρέχει τις παρακάτω πληροφορίες για μια υπηρεσία:

- Τι χρειάζεται από την χρήστη μια υπηρεσία για να λειτουργήσει και τι του προσφέρει πίσω. Η απάντηση σ' αυτό το ερώτημα είναι το service profile, το οποίο μας πληροφορεί για το τι κάνει μια υπηρεσία
- Πώς δουλεύει μια υπηρεσία, δηλ. το service model , έτσι ώστε ο χρήστης να μπορεί να καταλάβει αν μια υπηρεσία καλύπτει τις ανάγκες του, και να συνθέτει περιγραφές υπηρεσιών για πιο σύνθετους σκοπούς
- Πώς χρησιμοποιείται μια υπηρεσία (service grounding) δηλ. το πρωτόκολλο επικοινωνίας, οι μορφές των μηνυμάτων και άλλες λεπτομέρειες όπως ο αριθμός της θύρας για την επικοινωνία με την υπηρεσία.

Γενικά λοιπόν μιλώντας, το service profile προσφέρει τις πληροφορίες που χρειάζονται από έναν πράκτορα για να ανακαλύψει την υπηρεσία, ενώ το service model και το service grounding τις πληροφορίες για να χρησιμοποιήσει την υπηρεσία.

Η OWL-S χρησιμοποιεί WSDL documents για να προσφέρει πληροφορίες grounding. Και οι δύο γλώσσες, η OWL-S και η WSDL , χρειάζονται γι' αυτές τις πληροφορίες, επειδή οι δύο γλώσσες δεν αλληλοκαλύπτονται. Όπως φαίνεται και από το παρακάτω σχήμα, οι δύο γλώσσες αλληλοκαλύπτονται σε δεδομένα που η WSDL αποκαλεί 'αφηρημένους τύπους' ,δηλ. στην περιγραφή των εισόδων και εξόδων της υπηρεσίας. Αλλά η WSDL δεν μπορεί να εκφράσει την σημασιολογία των OWL κλάσεων, αλλά ούτε και η OWL-S μπορεί να παρέχει τις πληροφορίες σύνδεσης (binding) που παρέχει η WSDL



Σχήμα 7 Αντιστοιχία μεταξύ OWL-S και WSDL

3.3 OWL (Web Ontology Language)

Η ανάγκη για μια ισχυρή γλώσσα μοντελοποίησης οντολογιών οδήγησε στην ανάπτυξη της DAM+OIL [17] η οποία αποτελεί τον πρόγονο της OWL, μιας γλώσσας που τείνει να γίνει το standard πρότυπο για την ανάπτυξη και περιγραφή οντολογιών. Μια τέτοια γλώσσα πρέπει να ικανοποιεί τις εξής απαιτήσεις :

- ένα καλά καθορισμένο συντακτικό
- καλή καθορισμένη σημασιολογία
- αποτελεσματική υποστήριξη σε συμπερασμό (reasoning)
- ικανοποιητική εκφραστική δύναμη
- ευχέρεια στην έκφραση

Μια χρήση μιας καλά καθορισμένης σημασιολογίας είναι και η δυνατότητα για συμπερασμό (reasoning) που παράγει νέα γνώση [10]. Ο συμπερασμός μπορεί να αφορά:

- class membership: εάν χ είναι ένα στιγμιότυπο της κλάσης Α, η Α είναι υποκλάση της Β τότε μπορούμε να συμπεράνουμε ότι ο χ είναι στιγμιότυπο και της Β.
- equivalence of classes: εάν η κλάση Α είναι ισοδύναμη με την κλάση Β και η Β με την κλάση Γ , τότε και η Α είναι ισοδύναμη με την Γ
- consistency: ας υποθέσουμε ότι ο χ είναι στιγμιότυπο της κλάσης Α. Τότε αν Α είναι υποκλάση της τομής Β και Γ, η Α είναι υποκλάση της Δ και η Δ με την Β είναι disjoint , τότε έχουμε ασυνέπεια, γιατί η Α θα πρέπει να είναι άδεια αλλά έχει το χ

- classification: αν έχουμε δηλώσει ότι κάποιες ιδιότητες είναι ικανές συνθήκες για μια κλάση Α, αν ένα στιγμιότυπο ικανοποιεί αυτές τις συνθήκες τότε μπορούμε να συμπεράνουμε ότι ο χ ανήκει στην Α

Με την διαδικασία του συμπερασμού (reasoning) μπορούμε να πετύχουμε τα εξής:

- να ελέγξουμε την συνέπεια μιας οντολογίας
- να ελέγξουμε τις υπονοούμενες σχέσεις μεταξύ των κλάσεων
- να κατατάξουμε στιγμιότυπα σε κατηγορίες

Έχουν αναπτυχθεί τρεις υπο-γλώσσες της γλώσσας OWL:

- OWL Full: πλήρης και ισχυρή γλώσσα οντολογιών με πολλές δυνατότητες
- OWL DL: προκειμένου να κερδίσουμε σε υπολογιστική αποτελεσματικότητα χρησιμοποιούμε ένα υποσύνολο της παραπάνω
- OWL Lite: ένας αυστηρότερος περιορισμός στο παραπάνω σύνολο

Στην παρούσα εργασία θα χρησιμοποιήσουμε την OWL DL σε συνδυασμό με το πρόγραμμα Protégé για την ανάπτυξη των οντολογιών που κρίθηκαν απαραίτητες.

Ένα έγγραφο OWL ξεκινά με ένα σύνολο από namespaces :

```
<rdf:RDF
xmlns:owl ="http://www.w3.org/2002/07/owl#"
xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd ="http://www.w3.org/2001/XLMSchema#">
```

ακολουθεί ένα σύνολο από ορισμούς κλάσεων :

```
<owl:Class rdf:ID="associateProfessor">
<rdfs:subClassOf rdf:resource="#academicStaffMember"/>
</owl:Class>
```

Μπορούμε να δηλώσουμε ότι μια κλάση είναι ισοδύναμη ή disjoint με κάποια άλλη :

```
<owl:Class rdf:about="associateProfessor">
<owl:disjointWith rdf:resource="#professor"/>
<owl:disjointWith rdf:resource="#assistantProfessor"/>
</owl:Class>
```

Για ισοδυναμία αντί του disjointWith χρησιμοποιούμε τον όρο equivalentClass.

Στην OWL υπάρχουν δύο ειδών ιδιότητες: Object properties (συσχετίζει ένα στοιχείο με κάποιο άλλο στοιχείο) και datatype properties(συσχετίζει ένα στοιχείο με κάποια τιμή).

```
<owl:ObjectProperty rdf:ID="isTaughtBy">
<owl:domain rdf:resource="#course"/>
<owl:range rdf:resource="#academicStaffMember"/>
<rdfs:subPropertyOf rdf:resource="#involves"/>
</owl:ObjectProperty>
```

Στο παραπάνω παράδειγμα η ιδιότητα isTaughtBy έχει domain την κλάση course (μαθήματα) και παίρνει τιμές (range) από την κλάση academicStaffMember.

Στο παρακάτω παράδειγμα δηλώνουμε ότι τα μαθήματα του πρώτου έτους πρέπει να διδάσκονται μόνο από καθηγητές (professors):

```
<owl:Class rdf:about="#firstYearCourse">
        <rdfs:subClassOf>
        <owl:Restriction>
                <owl:onProperty rdf:resource="#isTaughtBy"/>
                <owl:allValuesFrom rdf:resource="#Professor"/>
        </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

Αντί για το allValuesFrom που δηλώνει ότι η ιδιότητα isTaughtBy παίρνει τιμές αποκλειστικά και μόνο από την κλάση Professor, μπορούμε να χρησιμοποιήσουμε την owl:hasValue που δηλώνει ότι η ιδιότητα isTaughtBy μπορεί να πάρει κάποιες τιμές από ένα σύνολο στιγμιοτύπων της κλάση Professor .

Οι ιδιότητες μπορούν να ανήκουν στις εξής κατηγορίες:

- owl – TransitiveProperty : η μεταβατική ιδιότητα, όπως ‘ψηλότερος από’
- owl – SymmetricProperty : η συμμετρική ιδιότητα, όπως ‘είναι αδερφός του’
- owl – FunctionalProperty: η ιδιότητα έχει μία τιμή για κάθε αντικείμενο
- owl – InverseFunctionalProperty: δύο διαφορετικά αντικείμενα δεν μπορούν να έχουν την ίδια τιμή.

Παράδειγμα των παραπάνω είναι το εξής :

```
<owl:ObjectProperty rdf:ID="hasSameGradeAs">
<rdf:type rdf:resource="&owl;TransitiveProperty" />
<rdf:type rdf:resource="&owl;SymmetricProperty" />
<rdfs:domain rdf:resource="#student" />
<rdfs:range rdf:resource="#student" />
</owl:ObjectProperty>
```

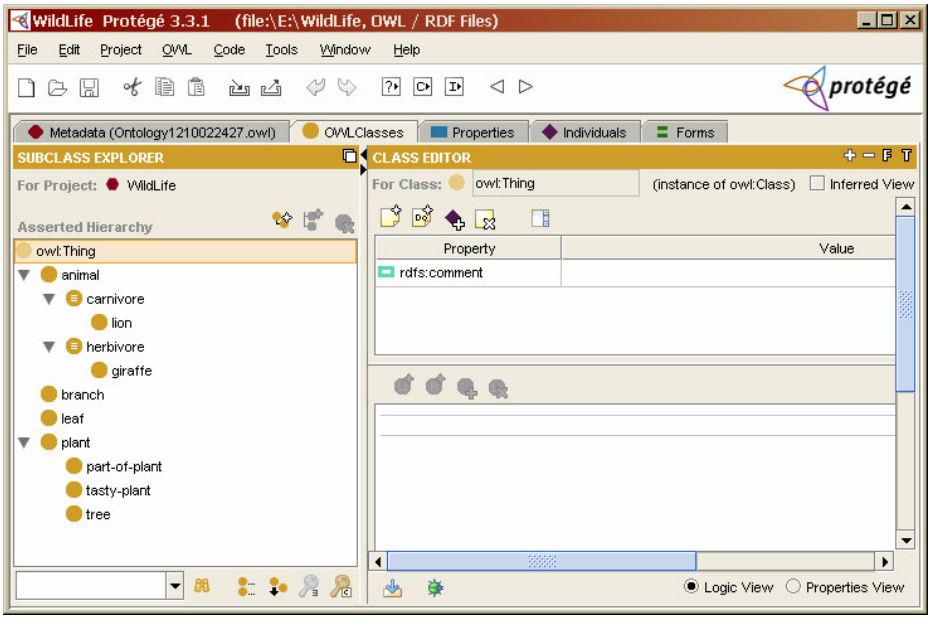
Με τις ιδιότητες μπορούμε να χρησιμοποιήσουμε και λογικούς συνδυασμούς, όπως η τομή, η ένωση και το συμπλήρωμα. Ένα παράδειγμα ένωσης (αντίστοιχα είναι και τα άλλα χρησιμοποιώντας τον τελεστή owl:complementof ή owl:intesectionof για συμπλήρωμα ή τομή αντίστοιχα) είναι:

```
<owl:Class rdf:ID="peopleAtUni">
<owl:unionOf rdf:parseType="Collection">
<owl:Class rdf:about="#staffMember"/>
<owl:Class rdf:about="#student"/>
</owl:unionOf>
</owl:Class>
```

Μπορούμε να δηλώσουμε και στιγμιότυπα κλάσεων, όπως:

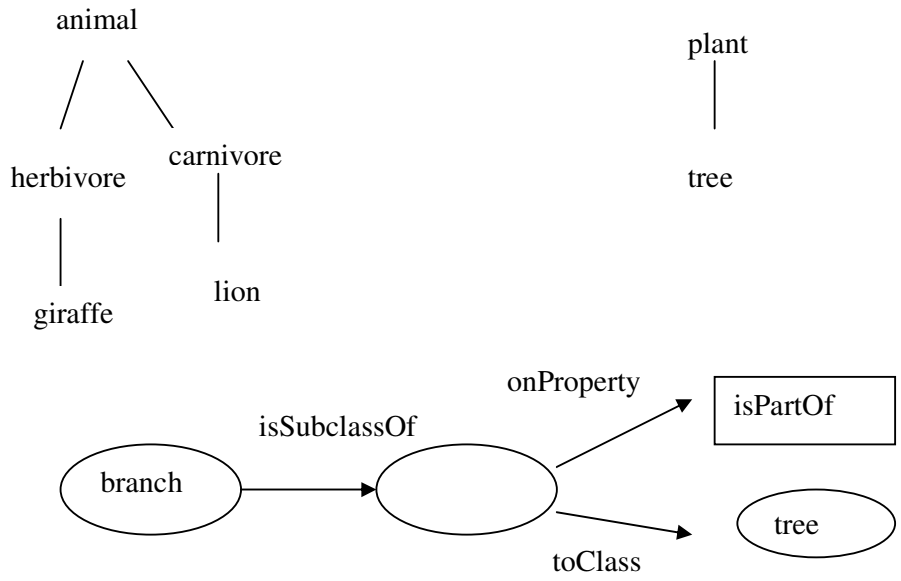
```
<rdf:Description rdf:ID="949352">
<rdf:type rdf:resource="#academicStaffMember"/>
</rdf:Description>
```

Το πρόγραμμα που θα χρησιμοποιήσουμε για να διαχειριστούμε τις οντολογίες και να δημιουργήσουμε αρχεία OWL με φιλικό προς τον χρήστη γραφικό τρόπο είναι το Protégé [8], όπως φαίνεται στην Εικόνα 1.



Εικόνα 1 Το περιβάλλον του Protege

Ως ένα παράδειγμα οντολογίας που θα χρησιμοποιήσουμε είναι η Οντολογία για την Άγρια Ζωή στην Αφρική, όπως φαίνεται και στο σχήμα :



Σχήμα 8 Η σχηματική αναπαράσταση της οντολογίας WildLife

Για μια πλήρη περιγραφή της οντολογίας WildLife ο αναγνώστης μπορεί να δει το Παράρτημα 2.

Το πρόγραμμα Protégé έχει διάφορες καρτέλες με τι οποίες μπορούμε να δημιουργήσουμε κλάσεις(OWL Classes) , ιδιότητες (Attributes) και στιγμιότυπα κλάσεων (Individuals). Ένα ενδιαφέρον σημείο του προγράμματος είναι και το API που διαθέτει

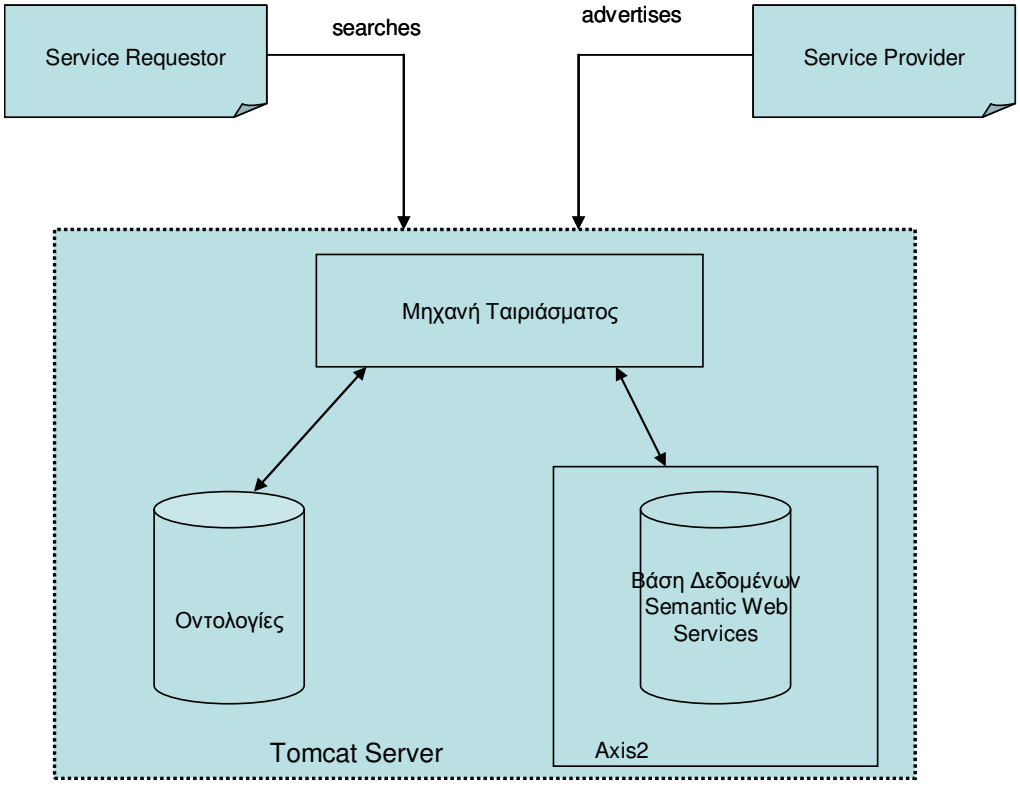
για τη δημιουργία και τη διαχείριση οντολογιών. Το χρησιμοποιούμε σε μεγάλο βαθμό και στα προγράμματα JAVA που αναπτύξαμε για την εργασία .

ΚΕΦΑΛΑΙΟ 4

ΥΛΟΠΟΙΗΣΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

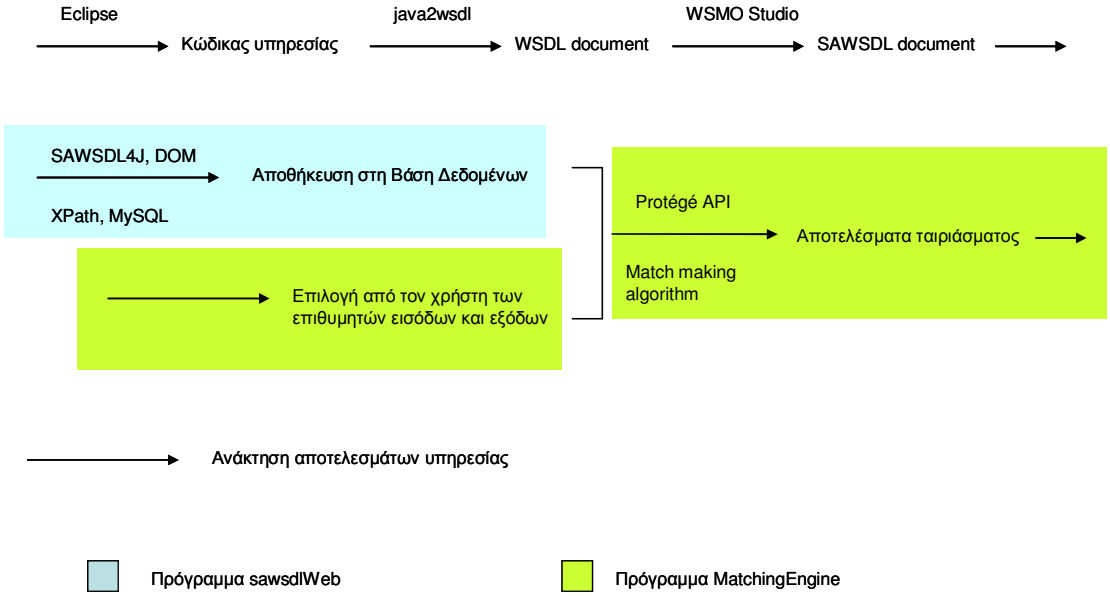
Εισαγωγή

Μέχρι τώρα έχουμε σκιαγραφήσει το θέμα αυτής της εργασίας και έχουμε αναφέρει τις βασικές θεωρητικές παραμέτρους μέσα στα πλαίσια των οποίων θα κινηθούμε. Η υλοποίηση του συστήματος απαιτεί την οργάνωση των επιμέρους στοιχείων . Το σχήμα 7 παρουσιάζει την αρχιτεκτονική του συστήματος



Σχήμα 9 Η αρχιτεκτονική του συστήματος

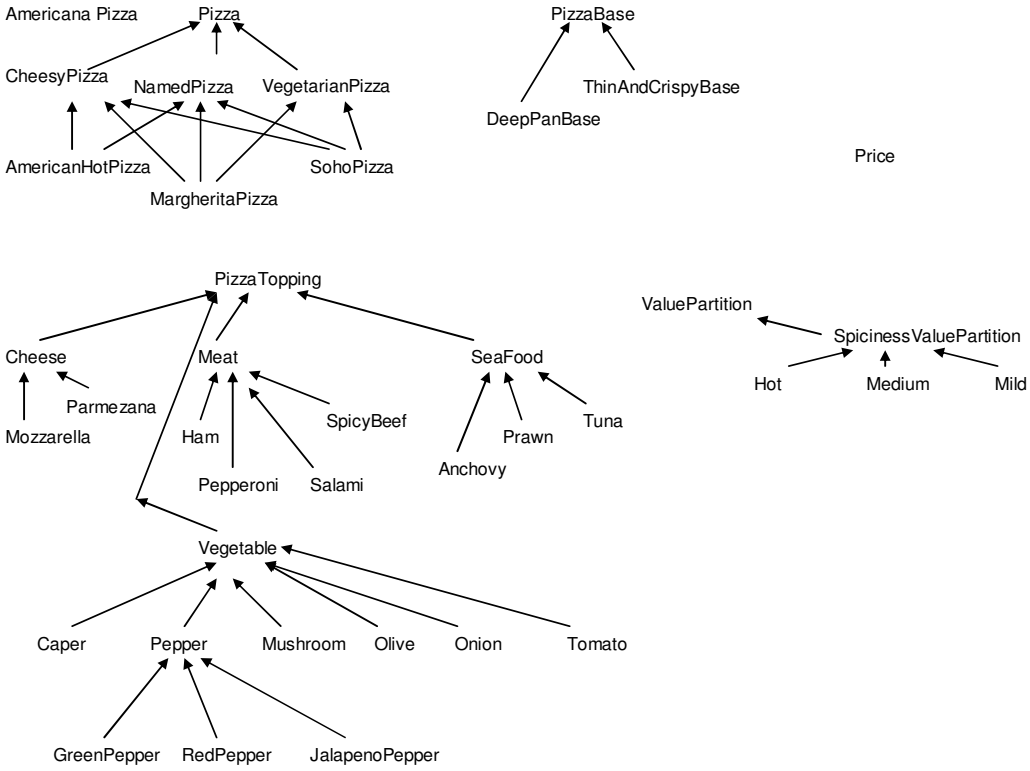
Το σχήμα 8 παρουσιάζει την ροή των εργασιών στο σύστημά μας, ξεκινώντας από την δημιουργία υπηρεσιών, την συσχέτιση τους με οντολογίες, την αποθήκευση τους σε Βάση Δεδομένων, τον αλγόριθμο ταιριάσματος και την ανάκτησή τους



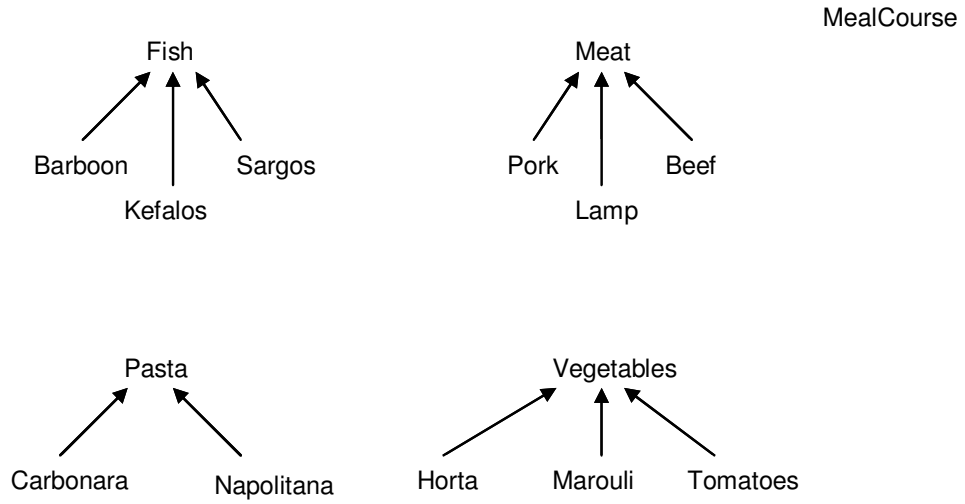
Σχήμα 10 Η ροή των εργασιών του συστήματος

4.1 Οντολογίες

Χρησιμοποιώντας το Protégé, εκτός από την οντολογία WildLife , δημιουργήσαμε δύο ακόμη οντολογίες: Pizza και Food. Η αναπαράσταση των οντολογιών φαίνεται στα παρακάτω σχήματα.



Σχήμα 11 Οι κλάσεις της οντολογίας Pizza



Σχήμα 12 Οι κλάσεις της οντολογίας Food

Τα OWL αρχεία μπορούν να βρεθούν στο CD που συνοδεύει την εργασία.

4.2 Web Services

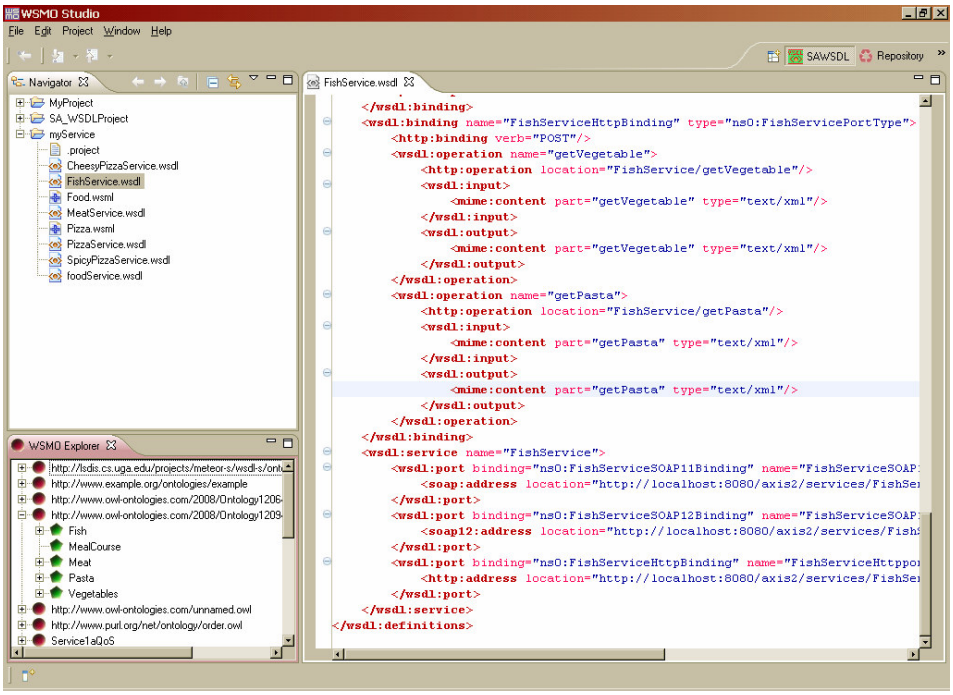
Τα αρχεία των υπηρεσιών που αναπτύξαμε για τον σκοπό της εργασίας γράφτηκαν σε Java και είναι τα εξής:

- PizzaService : έχει τις μεθόδους getPizza getPrice. Η πρώτη παίρνει σαν εισόδους αντικείμενα PizzaTopping και SpicinessValuePartition και επιστρέφει ένα αντικείμενο Pizza, ενώ η δεύτερη παίρνει εισόδους PizzaTopping και SpicinessValuePartition και επιστρέφει ένα αντικείμενο Price. Στην getPizza δίνουμε το πάνω μέρος της πίτσας που θέλουμε και το αν την επιθυμούμε πικάντικη ή όχι, και μας προτείνει ποιο είδος πίτσας ταιριάζει. Στην getPrice με τις ίδιες εισόδους μας επιστρέφει την τιμή της πίτσας
- SpicyPizzaService: έχει τις μεθόδους getTop (με inputs ένα αντικείμενο Pizza και output PizzaTopping) και getPizza (με input SpicinessValuePartition και output Pizza). Στην getTop δίνουμε σαν είσοδο το είδος της πίτσας και μας επιστρέφει το τι πάνω μέρος έχει. Στην getPizza δίνουμε σαν είσοδο το αν θέλουμε την πίτσα πικάντικη ή όχι και μας επιστρέφει το είδος της πίτσας
- MeatService: με μεθόδους την getPasta (με input Meat και output Pasta) και την getVegetable (με input Meat και output Vegetable). Στην getPasta δίνουμε σαν είσοδο το είδος του κρέατος και μας επιστρέφει το είδος της μακαρονάδας που ταιριάζει μ' αυτό. Στην getVegetable με την ίδια είσοδο επιστρέφει τα λαχανικά που ταιριάζουν με αυτό το είδος κρέατος.

- Foodservice: με μεθόδους getMeat (με input Meat και output Pasta) και την getFish (με input Fish και output MealCourse). Στην getMeat με είσοδο το είδος του κρέατος, επιστρέφει την μακαρονάδα που ταιριάζει. Στην getFish με είσοδο το είδος ψαριού, επιστρέφει την δίαιτα που περιλαμβάνει αυτό το ψάρι
- FishService:με μεθόδους getPasta (με input Fish και output Pasta) και την getVegetable (με input Fish και output Vegetable). Στην getPasta με είσοδο το είδος ψαριού, επιστρέφει την μακαρονάδα που έχει σάλτσα με αυτό το ψάρι. Στην getVegetable με είσοδο το είδος ψαριού, επιστρέφει τα λαχανικά που ταιριάζουν μαζί του.
- CheesyPizzaService: με μεθόδους την getBase (με input Pizza και output PizzaBase) και την getTopping (με input Pizza και output PizzaTopping). Στην getBase δίνουμε είσοδο το είδος της πίτσας και μας επιστρέφει την βάση της πίτσας. Στην getTopping με την ίδια είσοδο, μας επιστρέφει το πάνω μέρος της πίτσας.
- AnimalService: με μέθοδο την getFood (με input animal και output plant). Στην getFood εισάγουμε το ζώο και μας επιστρέφει τα φυτά που τρώει.

4.3 Δημιουργία αρχείων SAWSDL

Το πρόγραμμα που χρησιμοποιήσαμε για να προσθέσουμε σημασιολογικές σημειώσεις σε ένα WSDL document version 1.1 είναι το WSMO Studio[13]. Σ' αυτό μπορούμε να προσθέσουμε οντολογίες (αρχεία OWL) και αρχεία WSDL και με μεταφορά και απόθεση να αντιστοιχίσουμε στοιχεία του WSDL εγγράφου με αντικείμενα της οντολογίας.



Εικόνα 2 WSMO Studio

Το πρόγραμμα μετατρέπει εσωτερικά τα αρχεία οντολογιών σε ένα δικό του format αρχείων WSML . Ανοίγουμε σε δικό τους παράθυρο το WSDL αρχείο και το WSML αρχείο της οντολογίας. Με μεταφορά και απόθεση συσχετίζουμε στοιχεία του WSDL αρχείου με concepts από το WSML αρχείο.

4.4 Διαμόρφωση Βάσης Δεδομένων

Τα δεδομένα των υπηρεσιών αποθηκεύονται σε μια σχεσιακή βάση δεδομένων MYSQL με όνομα semantic. Αποτελείται από 4 πίνακες στους οποίους αποθηκεύουμε αντίστοιχα τις υπηρεσίες(services), τις λειτουργίες(μεθόδους operations), τις εισόδους(inputs) και τις εξόδους (outputs).

Ο πίνακας services αποθηκεύει τις υπηρεσίες κι έχει την παρακάτω δομή:

- uriID : αύξων αριθμός για την υπηρεσία
- uri: η διεύθυνση της υπηρεσίας (π.χ. http://localhost:8082/axi2/services/foodService)

Ο πίνακας operations περιγράφει τις μεθόδους για κάθε υπηρεσία και έχει τα πεδία:

- opID: αύξων αριθμός
- uriID: κωδικός για να σχετίζεται με τον προηγούμενο πίνακα
- operation: το όνομα της μεθόδου

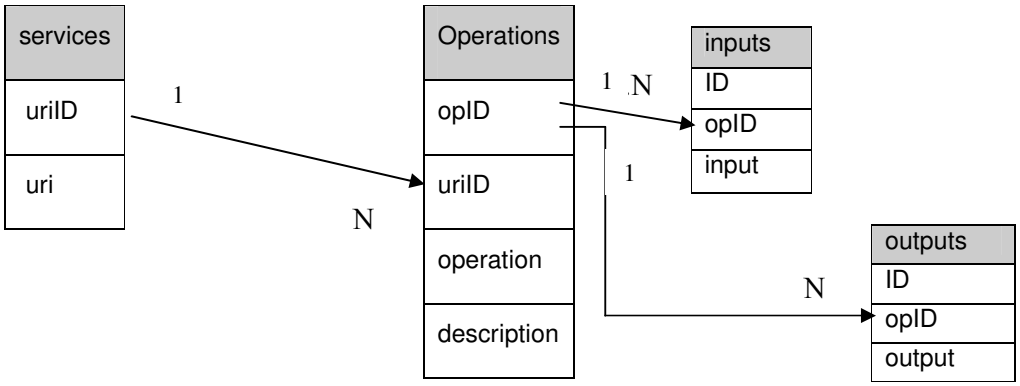
- description: μια σύντομη περιγραφή της μεθόδου

Ο πίνακας inputs περιγράφει τις εισόδους κάθε μεθόδου με πεδία:

- ID αύξων αριθμός
- opID κωδικός για να σχετίζεται με τον πίνακα operations
- input: το αντικείμενο της οντολογίας που σχετίζεται με την συγκεκριμένη είσοδο

Ο πίνακας outputs περιγράφει τις εξόδους κάθε μεθόδου με πεδία :

- ID αύξων αριθμός
- opID κωδικός για να σχετίζεται με τον πίνακα operations
- output: το αντικείμενο της οντολογίας που σχετίζεται με την συγκεκριμένη έξοδο



Σχήμα 13 Η οργάνωση της βάσης δεδομένων MySQL. Τα σύμβολα 1 N δείχνουν μια σχέση ένα προς πολλά

4.5 Διαμόρφωση διακομιστή

Στην συγκεκριμένη εργασία διαμορφώσαμε τον διακομιστή Apache Tomcat 6.0.14 server [12] να τρέχει σαν υπηρεσία, ενώ για την εκτέλεση των υπηρεσιών χρησιμοποιήσαμε τον διακομιστή Axis2 [5] μεταφέροντας τον φάκελο axis2 που κατεβάσαμε από το Διαδίκτυο στον φάκελο webapps του Tomcat. (π.χ. C:\Program Files\Apache Software Foundation\Tomcat 6.0\webapps\axis2) Για κάθε διεργασία, αφού γράφαμε το πρόγραμμα σε Java, το μεταγλωττίζαμε, το τοποθετούσαμε στον δικό του φάκελο στον διακομιστή Axis2(π.χ. C:\Program Files\Apache Software Foundation\Tomcat 6.0\webapps\axis2\WEB-INF\services\FoodService) και κατόπιν δημιουργούσαμε το WSDL document με το εργαλείο : java2wsdl του Axis2

%AXIS2_HOME%/bin/java2wsdl –cp . –cn samples.nisyros.foodService –of foodservice.wsdl

όπου AXIS2_HOME ο φάκελος που αποσυμπιέσαμε τον axis2 (στον υπολογιστή που δουλέψαμε C:/axis2-1.3) ενώ samples.nisyros το πακέτο(package) στο οποίο ανήκει η υπηρεσία foodservice

4.6 Ανάπτυξη προγραμμάτων

Το επόμενο βήμα είναι η ανάπτυξη των προγραμμάτων της εργασίας. Ουσιαστικά δημιουργήσαμε δύο projects με το περιβάλλον ανάπτυξης εφαρμογών Eclipse. Το πρώτο, με όνομα sawsdlWeb, κάνει parsing σε ένα έγγραφο SAWSDL (version 1.1) και αποθηκεύει στην Βάση Δεδομένων στοιχεία του εγγράφου. Το δεύτερο ,με όνομα MatchingEngine υλοποιεί τον αλγόριθμο SWS discovery ζητώντας από τον χρήστη να επιλέξει τις εισόδους και τις εξόδους της επιθυμητής υπηρεσίας, από ένα σύνολο από αντικείμενα οντολογιών. Αφού ο χρήστης επιλέξει, του επιστρέφεται μια αναφορά με τις μεθόδους(operations) κάθε υπηρεσίας και τον βαθμό ταιριάσματος (degree of match) κάθε μεθόδου. Με διπλό κλικ σε κάθε μέθοδο, μπορεί να συμπληρώσει τις εισόδους και να πάρει ως αποτέλεσμα την έξοδο της μεθόδου.

4.6.1 Πρόγραμμα sawsdlWeb

Το πρόγραμμα αποτελείται από 4 κλάσεις της Java [6]:

- DBConnect:Η κλάση αυτή περιέχει την δημόσια μέθοδο getConnection με την οποία συνδεόμαστε στην Βάση Δεδομένων semantic.:

```
String url="jdbc:mysql:///semantic";
Class.forName("com.mysql.jdbc.Driver");
con=DriverManager.getConnection(url,"root","nisyros69");
```

Ακόμη περιέχει την δημόσια μέθοδο insertToServices, που παίρνει σαν είσοδο το όνομα της υπηρεσίας και την αποθηκεύει στον πίνακα services,την μέθοδο insertToOperations για να εισάγουμε τις μεθόδους(operations) κάθε υπηρεσίας στον πίνακα operations, την μέθοδο insertToInputs για την εισαγωγή των εισόδων(που αντιστοιχούν σε αντικείμενα οντολογιών) κάθε μεθόδου στον πίνακα inputs, και την μέθοδο insertToOutputs για την εισαγωγή των εξόδων κάθε μεθόδου στον πίνακα outputs.

- BasicDom: διαθέτει την μέθοδο parseXMLFile για να πάρουμε ένα αντικείμενο Document από το SAWSDL document, ώστε να το χρησιμοποιήσουμε αργότερα για να εξάγουμε σημασιολογικά στοιχεία από αυτό.

- Η βοηθητική κλάση getModelReference.class η οποία περιέχει τις μεθόδους: modelRef: δέχεται το όνομα ενός στοιχείου (element) που βρίσκεται μέσα στο στοιχείο Types του SAWSDL document και επιστρέφει μια λίστα με τις τιμές όλων των ιδιοτήτων sawsdl:modelReference του στοιχείου. Π.χ. σε ένα SAWSDL document υπάρχει το στοιχείο :

```
……..<xs:element name="getPasta">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="0" name="param0"
                            nillable="true"
                            sawsdl:modelReference="http://www.owl-
ontologies.com/2008/Ontology1209488954.owl#Fish" type="xs:string"/> ……..
```

Η μέθοδος χρησιμοποιεί την γλώσσα XPath [7] για να ψάξει το έγγραφο και να βρει ένα στοιχείο με ιδιότητα (attribute) getPasta (δίνεται σαν παράμετρος στην μέθοδο ένα String που αντιπροσωπεύει αυτή την ιδιότητα):

```
String xpath="//element[@name='"+s1+"']//element";

NodeList nodeList=org.apache.xpath.XPathAPI.selectNodeList(doc, xpath);
```

Το παραπάνω XPath ερώτημα ψάχνει για ένα στοιχείο με όνομα element και attribute το String s1 (=getPasta στο παράδειγμά μας) το οποίο περιέχει ένα άλλο στοιχείο element και αφού το βρει το βάζει σε μια λίστα κόμβων (NodeList).

Η μέθοδος **public** String getPosition(Document doc) χρησιμοποιεί ένα άλλο ερώτημα XPath για να βρει την διεύθυνση (URL) της υπηρεσίας :

```
String xpath2="//service/port/address";

Node myNode=org.apache.xpath.XPathAPI.selectSingleNode(doc, xpath2);
```

Ψάχνει να βρει οπουδήποτε στο έγγραφο ένα στοιχείο service το οποίο περιέχει ένα στοιχείο port, το οποίο με τη σειρά του περιέχει ένα στοιχείο address

Η μέθοδος **public** ArrayList getParts(Definition def,QName myName,Document doc) ψάχνει σε ένα στοιχείο message, το οποίο έχει την μορφή :

```
….<wsdl:message name="getVegetableResponse">
    <wsdl:part element="ns0:getVegetableResponse" name="parameters"/>
</wsdl:message>….
```

για να βρει ένα στοιχείο part και να πάρει την ιδιότητα του με όνομα element. Χρησιμοποιούμε το μοντέλο SAWSDL4J [9] για να την τιμή της ιδιότητας element του στοιχείου part, το οποίο θα χρησιμοποιήσουμε στην μέθοδο modelRef για να πάρουμε τις σημασιολογικές σημειώσεις του στοιχείου. Με την παρακάτω εντολή ανακτούμε το στοιχείο message με όνομα myName

```
Message semanticMessage=def.getSemanticMessage(myName);
```

και αποκτούμε πρόσβαση στα στοιχεία parts:

```
Map parts=semanticMessage.getParts();
```

και για κάθε αντικείμενο part παίρνουμε την τιμή της ιδιότητας του element

```
for(Object partKey:parts.keySet()){
   Part semanticPart=semanticMessage.getSemanticPart((String)partKey);
   String s1=semanticPart.getElementName().getLocalPart();
```

Τέλος καλούμε την μέθοδο modelRef και παίρνουμε μια λίστα με τις τιμές όλων των ιδιοτήτων sawsdl:modelReference των στοιχείων με το ίδιο όνομα με την τιμή της προηγούμενης ιδιότητας.

- Τέλος χρησιμοποιούμε την κλάση SAWSDLExampler η οποία περιέχει και την κεντρική μέθοδο main . Δίνουμε σαν είσοδο στην μέθοδο την διεύθυνση ενός αρχείου SAWSDL (π.χ. C:\\Program Files\\WSMO-Studio-0.7.2\\workspace\\myService/AnimalService.wsdl). Το πρόγραμμα ανακτά το αντικείμενο document του εγγράφου χρησιμοποιώντας την βοηθητική κλάση BasicDom και την διεύθυνση url της υπηρεσίας την οποία αποθηκεύουμε στην βάση δεδομένων. Κατόπιν αποκτούμε το στοιχείο definition του εγγράφου με τις εντολές :

```
System.setProperty("javax.wsdl.factory.WSDLFactory",
"edu.uga.cs.lsdis.sawsdl.impl.factory.WSDLFactoryImpl");
File wsdlURI= new File(WSDLFileName);
Definition def=SAWSDLUtility.getDefinitionFromFile(wsdlURI);
```

Αφού ανακτήσουμε το definition , αποκτούμε πρόσβαση και στα στοιχεία portTypes του εγγράφου :

```
…Map portTypes=def.getPortTypes();
      for(Object key:portTypes.keySet()){
      PortType semanticPortType =def.getSemanticPortType((QName) key);…
```

οπότε παίρνουμε το όνομα καθενός operation

```
List operations=semanticPortType.getOperations();
      for(Object operation : operations){
            String opName=((Operation) operation).getName();
```

αλλά και το στοιχείο message του operation και στις εισόδους

```
…. Operation oper=(Operation) operation;
      QName myName=oper.getInput().getMessage().getQName();
      myListIn=new getModelReference().getParts(def, myName, doc); …
```

και εξόδους του operation :

```
……myName=oper.getOutput().getMessage().getQName();
myListOut=new getModelReference().getParts(def, myName, doc); ……
```

Αφού αποκτήσουμε τις εισόδους και εξόδους (δηλ. τις σημασιολογικές σημειώσεις τους) του operation τις αποθηκεύουμε στην βάση δεδομένων :

```
DBConnect dbc=new DBConnect();
dbc.insertToInputs(myListIn, opID);
dbc.insertToOutputs(myListOut, opID);
```

Για καλύτερη κατανόηση της διαδικασίας παραθέτουμε και ένα κομμάτι από το sawsdl document και μια σηματοδότηση της ροής απόκτησης της σημασιολογικής πληροφορίας (οι γραμμοσκιασμένες περιοχές χρησιμοποιούνται από το πρόγραμμα)

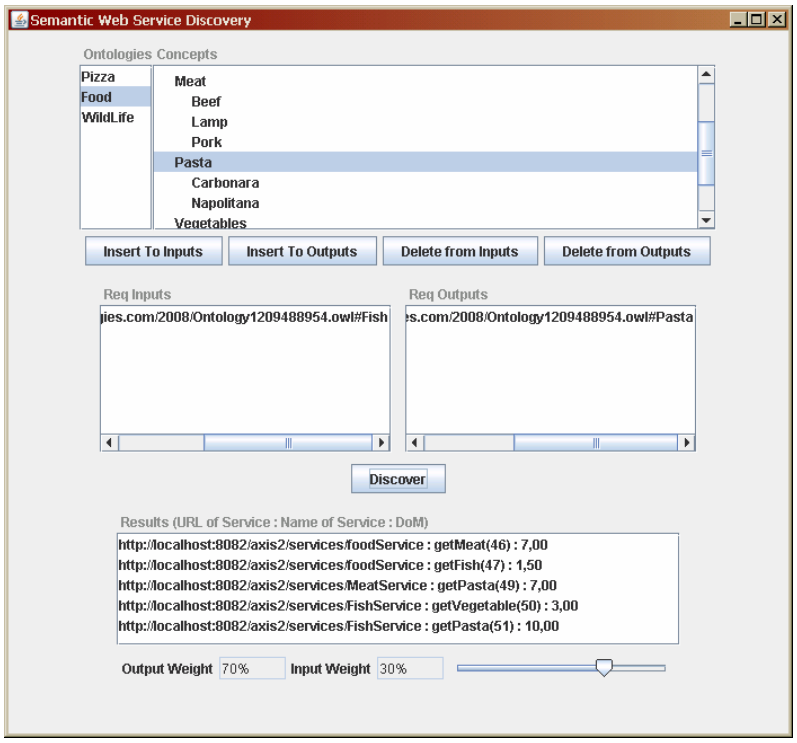
```
…… <wsdl:portType name="FishServicePortType">

      <wsdl:operation name="getVegetable">
            <wsdl:input message="ns0:getVegetableRequest"
            wsaw:Action="urn:getVegetable"/>
            <wsdl:output message="ns0:getVegetableResponse"
            wsaw:Action="urn:getVegetableResponse"/>
       </wsdl:operation>
</wsdl:portType> ……

……<wsdl:message name="getVegetableRequest">
        <wsdl:part element="ns0:getVegetable" name="parameters"/>
    </wsdl:message>
    <wsdl:message name="getVegetableResponse">
        <wsdl:part element="ns0:getVegetableResponse" name="parameters"/>
    </wsdl:message> ……

      ……<xs:element name="getVegetable">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="0" name="param0"
                            nillable="true"
                            sawsdl:modelReference="http://www.owl-
ontologies.com/2008/Ontology1209488954.owl#Fish" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
         </xs:element>

        ……<xs:element name="getVegetableResponse">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="0" name="return"
                            nillable="true"
                            sawsdl:modelReference="http://www.owl-
ontologies.com/2008/Ontology1209488954.owl#Vegetables" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
           </xs:element> ……
```

4.6.2 Πρόγραμμα MatchingEngine

Το project χρησιμοποιεί την βοηθητική κλάση DBConnect του προηγούμενου project για να αποκτήσουμε πρόσβαση στην βάση δεδομένων με την βοήθεια της μεθόδου public Connection getConnection().

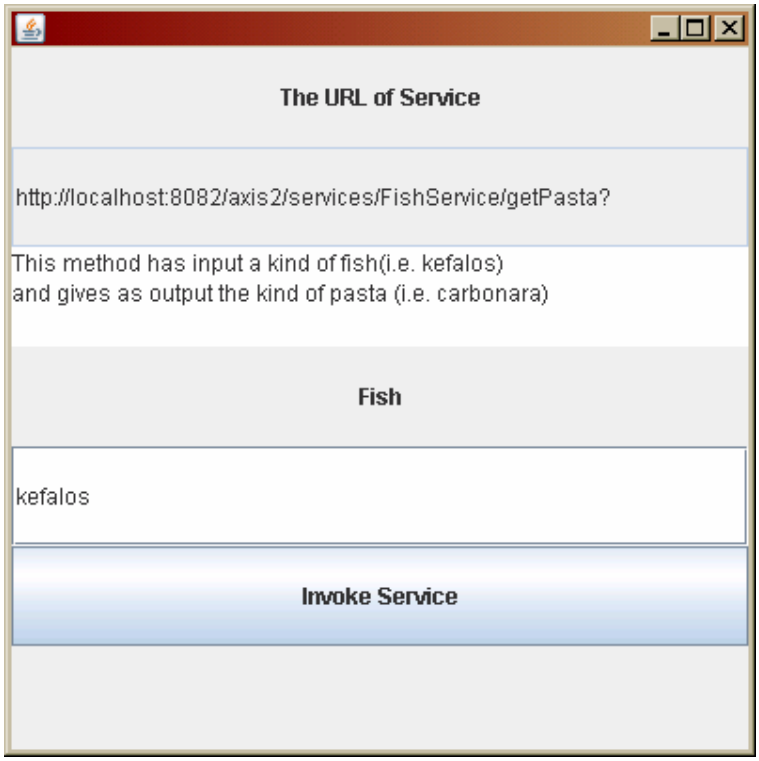
Το γραφικό περιβάλλον του προγράμματος δημιουργήθηκε με την τεχνολογία Swing της Java στην κλάση ListPanel .

Στο πρώτο επίπεδο βλέπουμε δυο λίστες. Η πρώτη(αριστερά) περιέχει τις οντολογίες που χρησιμοποιούμε και βρίσκονται αποθηκευμένες στον διακομιστή Tomcat, ενώ αν ο χρήστης επιλέξει κάποια, η δεξιά λίστα γεμίζει με τις κλάσεις της οντολογίας. Φυσικά ο χρήστης μπορεί να επιλέξει εισόδους και εξόδους από διαφορετικές οντολογίες. Κατόπιν ο χρήστης μπορεί να επιλέξει από την λίστα τις κλάσεις που επιθυμεί για εισόδους και εξόδους της υπηρεσίας και να πατήσει τα αντίστοιχα κουμπιά (Insert To Inputs, Insert To Outputs), οπότε και γεμίζουν οι λίστες Req Inputs, Req Outputs αντίστοιχα. Υπάρχουν και τα βοηθητικά κουμπιά Delete from Inputs, Delete from Outputs αν γίνει κάποιο λάθος κατά την εισαγωγή ή για αλλαγή των εισόδων και εξόδων. Στο κάτω μέρος του παραθύρου υπάρχει ο μηχανισμός slider και δύο πλαίσια κειμένου ,ένα για έξοδο και το άλλο για είσοδο. Αυτά περνούν στο πρόγραμμα μια παράμετρο για το ποσοστό βαρύτητας των εισόδων και εξόδων. Για παράδειγμα ένα ποσοστό εξόδου 70 σημαίνει ότι ο βαθμός ταιριάσματος των εξόδων της διαφήμισης της υπηρεσίας με τις εξόδους της επιθυμητής υπηρεσίας από τον χρήστη θα συμμετέχει με ποσοστό 70% στον τελικό βαθμό ταιριάσματος (Degree of Match, DoM). Τέλος αφού ρυθμίσουμε και τα ποσοστά όπως τα επιθυμούμε, πατώντας το κουμπί Update εμφανίζονται τα αποτελέσματα στο πλαίσιο κειμένου Results, δηλ. εμφανίζεται το url της υπηρεσίας, το όνομα της μεθόδου και ο DoM της μεθόδου, όπως φαίνεται και στην Εικόνα 3:



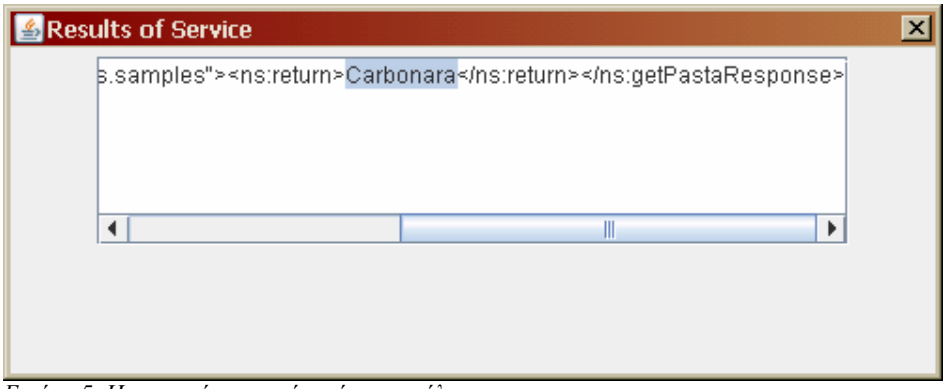
Εικόνα 3 Τα αποτελέσματα της αναζήτησης

Ο χρήστης αφού πάρει κάποιο αποτέλεσμα μπορεί να κάνει διπλό κλικ πάνω σ' αυτό και θα εμφανιστεί το παρακάτω παράθυρο για να συμπληρώσει τις εισόδους



Εικόνα 4 Εισαγωγή των εισόδων του operation

και να πάρει το αποτέλεσμα :



Εικόνα 5 Η υπηρεσία επιστρέφει ένα αποτέλεσμα

Ας δούμε όμως πώς υλοποιούνται όλες αυτές οι διαδικασίες.

Η κλάση ListPanel φροντίζει για το γραφικό περιβάλλον του προγράμματος. Ένα σημείο που πρέπει να σταθούμε είναι στο τι συμβαίνει όταν ο χρήστης επιλέξει μια οντολογία. Τότε δημιουργούμε το μοντέλο της οντολογίας

model = ProtegeOWL.*createJenaOWLModelFromURI*(uri);

και ζητάμε από έναν reasoner να κάνει ανάλυση του μοντέλου, ώστε να ανακαλυφθούν νέες σχέσεις μεταξύ των κλάσεων. Ο reasoner είναι ένα πρόγραμμα που τρέχει στο παρασκήνιο και 'ακούει' στην θύρα 8080:

```
ReasonerManager reasonerManager = ReasonerManager.getInstance();

ProtegeOWLReasoner r = reasonerManager.getReasoner(model);

r.setURL("http://localhost:8080");

r.classifyTaxonomy(null);

r.computeInferredIndividualTypes(null);

r.computeEquivalentConcepts(null);

r.computeInferredHierarchy(null);
```

Αφού, λοιπόν, κάνουμε μια ανάλυση του μοντέλου μπορούμε να πάρουμε τις κλάσεις της οντολογίας χρησιμοποιώντας την μέθοδο printClassTree:

```
private  void printClassTree(RDFSClass cls,String indentation){
      if(cls.getName().indexOf(":")==-1 ){
            System.out.println(indentation+cls.getName());
            inputs.addElement(indentation+cls.getName());
      }
      for(Iterator it=cls.getNamedSubclasses().iterator();it.hasNext();){
                  RDFSClass subclass=(RDFSClass) it.next();
                  printClassTree(subclass, indentation+"     ");
      }
```

```
}
```

και να τις εμφανίσουμε σε δεντροειδή μορφή στην λίστα Inputs.

Η καρδιά όλης της εργασίας βρίσκεται στην κλάση OWLAPIApplication. Η κλάση αυτή είναι υπεύθυνη να εφαρμόσει τον αλγόριθμο ταιριάσματος και να επιστρέψει τα αποτελέσματα στον χρήστη. Ο αλγόριθμος υλοποιείται στην μέθοδο degreeOfMatch(ArrayList adv,ArrayList req), που δέχεται σαν είσοδο την λίστα με τις εισόδους της διαφήμισης και την λίστα με τις εισόδους της αίτησης του χρήστη. Για κάθε μία είσοδο της διαφήμισης δημιουργείται ένας πίνακας degrees βαθμού όσος και ο αριθμός των εισόδων του χρήστη. Για κάθε είσοδο του χρήστη προστίθεται ένας βαθμός στον πίνακα, ανάλογα με την σχέση των δύο εισόδων βάσει του Πίνακα 4. Θα πρέπει οι δύο είσοδοι να αναφέρονται στην ίδια οντολογία και αυτό το πετυχαίνουμε συγκρίνοντας τα αρχικά τμήματα των εισόδων. Π.χ. μια είσοδος μπορεί να είναι η παρακάτω:

```
http://www.owl-ontologies.com/2008/Ontology1209488954.owl#Fish
```

όπου το αρχικό κομμάτι μέχρι το σύμβολο # μας δείχνει σε ποια οντολογία ανήκει, ενώ το τελευταίο κομμάτι ,π.χ. το `Fish`, μας δείχνει το αντικείμενο (concept) της οντολογίας. Αν, λοιπόν οι δύο είσοδοι ανήκουν στην ίδια οντολογία, τότε εξετάζεται η σχέση τους στην οντολογία. Δημιουργούμε, όπως και προηγούμενα, το μοντέλο της οντολογίας με τον reasoner , και συγκρίνουμε την είσοδο της διαφήμισης (name) με την είσοδο του χρήστη (reqInput). Χρησιμοποιούμε το API του Protégé και παίρνουμε το αντικείμενο της οντολογίας που αντιστοιχεί στην είσοδο της διαφήμισης :

```
OWLNamedClass cls=model.getOWLNamedClass(name);
```

και στην είσοδο του χρήστη:

```
OWLNamedClass reqNamed=model.getOWLNamedClass(reqInput);
```

κατόπιν παίρνουμε μια λίστα με τις υπερκλάσεις αυτού του αντικειμένου και αν σε αυτές υπάρχει η είσοδος του χρήστη τότε βάζουμε στον πίνακα degrees στην κατάλληλη θέση τον βαθμό 5:

```
Collection supClasses=cls.getNamedSuperclasses(true);

if(supClasses.contains(reqNamed)){
      degrees[j]=Math.max(5, degrees[j]);
 }
```

Αν , πάλι, υπάρχει ταύτιση των δύο εισόδων τότε βαθμολογούνται με 10:

```
if(name.equalsIgnoreCase(reqInput))
      degrees[j]=10;
```

ενώ αν είσοδος του χρήστη ανήκει στις υποκλάσεις της εισόδου της διαφήμισης τότε βαθμολογούνται με 7 :

```
Collection subClasses=cls.getNamedSubclasses(true);

if(subClasses.contains(reqNamed))
      degrees[j]=Math.max(7, degrees[j]);
```

Στο τέλος δημιουργούμε ένα πίνακα result ο οποίος κρατά τις μεγαλύτερες τιμές κάθε θέσης των πινάκων degrees και από τον οποίο υπολογίζουμε τον μέσο όρο, που αποτελεί και την βάση για τον υπολογισμό του DoM της μεθόδου της υπηρεσίας

Τέλος χρησιμοποιούμε την μέθοδο με πρωτότυπο **public** ArrayList getInOut() η οποία αναλαμβάνει να διατρέξει την Βάση Δεδομένων των υπηρεσιών. Για κάθε υπηρεσία ανακτά το σύνολο των μεθόδων που προσφέρει, και για κάθε μέθοδο το σύνολο των εισόδων και εξόδων της. Για κάθε τέτοιο σύνολο εφαρμόζει την μέθοδο degreeOfMatch. Αφού πάρει και τους δύο βαθμούς ταιριάσματος :

```
int maxIn=degreeOfMatch(advInputs,reqInputs);
int maxOut=degreeOfMatch(advOutputs,reqOutputs);
```

χρησιμοποιεί το ποσοστό βαρύτητας εξόδων (outPer) και εισόδων (inPer) για τον υπολογισμό του τελικού βαθμού ταιριάσματος:

```
double finalDegree=outPer*maxOut+inPer*maxIn;
```

και αποθηκεύει μια διαμορφωμένη συμβολοσειρά με πληροφορίες για την υπηρεσία , την μέθοδο και τον βαθμό ταιριάσματος σε μια λίστα, ώστε να την εμφανίσει αργότερα στο γραφικό περιβάλλον της εργασίας:

```
if(finalDegree>=1.0){
      DecimalFormat theFormat=new DecimalFormat("###0.00");
      String deg=theFormat.format(finalDegree);
      str=serviceUri+" : "+operation+"("+opID+") : "+deg+"\n";
      myList.add(str);

}
```

ΚΕΦΑΛΑΙΟ 5

ΣΥΜΠΕΡΑΣΜΑΤΑ

Αφού λοιπόν ολοκληρώσαμε αυτήν εργασία, μπορούμε να αναφέρουμε διάφορα συμπεράσματα, σκέψεις αλλά και προβλήματα που προέκυψαν κατά την διάρκειά της. Το θέμα του Σημασιολογικού Ιστού και της ανακάλυψης σημασιολογικών υπηρεσιών βρίσκεται στην ακμή της τεχνολογίας κι αποτελεί ένα θέμα που συνεχώς ερευνάται. Η προσπάθεια να προσθέσουμε ένα είδος λογικής σε μηχανές, αποτελεί μια μεγάλη πρόκληση για τους επιστήμονες και σκιαγραφεί το μέλλον του Παγκόσμιου Ιστού.

Αυτή η εργασία, αποτελεί μία από τις πρώτες πρωτότυπες υλοποιήσεις για την ανακάλυψη Υπηρεσιών Σημασιολογικού Ιστού με το πρότυπο SAWSDL. Αρχικά δημιουργήσαμε τις υπηρεσίες που θα χρησιμοποιήσουμε και τις αποθηκεύσαμε στο διακομιστή Axis2 του Tomcat. Εμπλουτίσαμε τα WSDL αρχεία των υπηρεσιών με σημασιολογικές σημειώσεις από οντολογίες που είτε δημιουργήσαμε μόνοι μας με το εργαλείο Protégé είτε κατεβάσαμε από το Διαδίκτυο, με την βοήθεια του εργαλείου WSMO Studio. Με το πρόγραμμα sawsdlWeb που αναπτύξαμε «σαρώσαμε» τα αρχεία SAWSDL και αποθηκεύσαμε σε μια Βάση Δεδομένων MySQL στοιχεία και σημασιολογικές σημειώσεις των εισόδων και εξόδων κάθε υπηρεσίας. Με το πρόγραμμα MatchingEngine μπορεί ο χρήστης να επιλέξει τις εισόδους και εξόδους που επιθυμεί από τα θέματα (concepts) των οντολογιών και να πάρει τον βαθμό ταιριάσματος γι' αυτά από κάθε υπηρεσία. Κατόπιν μπορεί να καλέσει την υπηρεσία και να πάρει τα ανάλογα αποτελέσματα.

Ακριβώς όμως επειδή αποτελεί ακόμη αντικείμενο μεγάλης έρευνας, προκύπτουν και διάφορα προβλήματα ασυμβατότητας μεταξύ των προτεινόμενων τεχνολογιών. Για παράδειγμα πολλές από τις έτοιμες οντολογίες που κατεβάσαμε από το Διαδίκτυο δεν μπορούσε να τις επεξεργαστεί το Protégé ή άνοιγαν με πολλά λάθη. Τα αρχεία WSDL που δημιουργήσαμε και κατόπιν μετατρέψαμε σε SAWSL αρχεία, πρέπει να ήταν έκδοσης 1.1, γιατί αλλιώς οι συναρτήσεις του API SAWSDL4J, δεν μπορούσαν να κάνουν parsing τα αρχεία SAWSL. Βέβαια μπροστά σε κάθε νέα τεχνολογία παρουσιάζονται τέτοια θέματα, τα οποία λύνονται με τον χρόνο και ίσως και με τις συμμαχίες που κάθε εταιρεία μπορεί να κλείσει προκειμένου να προωθήσει την τεχνολογία της.

Παρόλα αυτά έχουν γίνει μεγάλα βήματα προόδου και αρκετές τεχνολογίες έχουν προτυποποιηθεί, ώστε να αποτελέσουν την βάση για παραπέρα έρευνα. Η δημιουργία και η ανακάλυψη Semantic Web Services θα βρίσκονται για πολλή καιρό ακόμα στην κορυφή της έρευνας και θα αποτελέσουν ένα σημαντικό κέρδος όχι μόνο για τις εταιρείες αλλά και για τους απλούς χρήστες.

ΠΑΡΑΡΤΗΜΑ 1

Το πλήρες έγγραφο του εγγράφου SAWSDL για την υπηρεσία FishService

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://nisyros.samples"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:ns0="http://nisyros.samples"
    xmlns:ns1="http://org.apache.axis2/xsd"
    xmlns:sawsdl="http://www.w3.org/ns/sawsdl"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <wsdl:types>
        <xs:schema attributeFormDefault="qualified"
            elementFormDefault="qualified"
            targetNamespace="http://nisyros.samples"
xmlns:ns="http://nisyros.samples">
            <xs:element name="getPasta">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="0" name="param0"
                            nillable="true"
                            sawsdl:modelReference="http://www.owl-
ontologies.com/2008/Ontology1209488954.owl#Fish" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="getPastaResponse">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="0" name="return"
                            nillable="true"
                            sawsdl:modelReference="http://www.owl-
ontologies.com/2008/Ontology1209488954.owl#Pasta" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="getVegetable">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="0" name="param0"
                            nillable="true"
                            sawsdl:modelReference="http://www.owl-
ontologies.com/2008/Ontology1209488954.owl#Fish" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="getVegetableResponse">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="0" name="return"
                            nillable="true"
                            sawsdl:modelReference="http://www.owl-
ontologies.com/2008/Ontology1209488954.owl#Vegetables" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:schema>
    </wsdl:types>
```

```
    <wsdl:message name="getVegetableRequest">
        <wsdl:part element="ns0:getVegetable" name="parameters"/>
    </wsdl:message>
    <wsdl:message name="getVegetableResponse">
        <wsdl:part element="ns0:getVegetableResponse" name="parameters"/>
    </wsdl:message>
    <wsdl:message name="getPastaRequest">
        <wsdl:part element="ns0:getPasta" name="parameters"/>
    </wsdl:message>
    <wsdl:message name="getPastaResponse">
        <wsdl:part element="ns0:getPastaResponse" name="parameters"/>
    </wsdl:message>
    <wsdl:portType name="FishServicePortType">
        <wsdl:operation name="getVegetable">
            <wsdl:input message="ns0:getVegetableRequest"
wsaw:Action="urn:getVegetable"/>
            <wsdl:output message="ns0:getVegetableResponse"
wsaw:Action="urn:getVegetableResponse"/>
        </wsdl:operation>
        <wsdl:operation name="getPasta">
            <wsdl:input message="ns0:getPastaRequest"
wsaw:Action="urn:getPasta"/>
            <wsdl:output message="ns0:getPastaResponse"
wsaw:Action="urn:getPastaResponse"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="FishServiceSOAP11Binding"
type="ns0:FishServicePortType">
        <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="getVegetable">
            <soap:operation soapAction="urn:getVegetable" style="document"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getPasta">
            <soap:operation soapAction="urn:getPasta" style="document"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:binding name="FishServiceSOAP12Binding"
type="ns0:FishServicePortType">
        <soap12:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="getVegetable">
            <soap12:operation soapAction="urn:getVegetable"
style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
```

```
        <wsdl:operation name="getPasta">
            <soap12:operation soapAction="urn:getPasta" style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:binding name="FishServiceHttpBinding"
type="ns0:FishServicePortType">
        <http:binding verb="POST"/>
        <wsdl:operation name="getVegetable">
            <http:operation location="FishService/getVegetable"/>
            <wsdl:input>
                <mime:content part="getVegetable" type="text/xml"/>
            </wsdl:input>
            <wsdl:output>
                <mime:content part="getVegetable" type="text/xml"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="getPasta">
            <http:operation location="FishService/getPasta"/>
            <wsdl:input>
                <mime:content part="getPasta" type="text/xml"/>
            </wsdl:input>
            <wsdl:output>
                <mime:content part="getPasta" type="text/xml"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="FishService">
        <wsdl:port binding="ns0:FishServiceSOAP11Binding"
name="FishServiceSOAP11port_http">
            <soap:address
location="http://localhost:8080/axis2/services/FishService"/>
        </wsdl:port>
        <wsdl:port binding="ns0:FishServiceSOAP12Binding"
name="FishServiceSOAP12port_http">
            <soap12:address
location="http://localhost:8080/axis2/services/FishService"/>
        </wsdl:port>
        <wsdl:port binding="ns0:FishServiceHttpBinding"
name="FishServiceHttpport">
            <http:address
location="http://localhost:8080/axis2/services/FishService"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

ΠΑΡΑΡΤΗΜΑ 2

Ένα πλήρες έγγραφο OWL που περιγράφει την οντολογία WildLife

```
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:owl ="http://www.w3.org/2002/07/owl#"
        xmlns="http://www.mydomain.org/african">
        <owl:Ontology rdf:about="">
                <owl:VersionInfo>
                        Η άγρια ζωή της Αφρικής
                </owl:VersionInfo>
        </owl:Ontology>
        <owl:Class rdf:ID="animal">
                <rdfs:comment>Animals form a class</rdfs:comment>
        </owl:Class>
        <owl:Class rdf:ID="plant">
                <rdfs:comment>
                        Plants form a class disjoint from animals
                </rdfs:comment>
                <owl:disjointWith="#animal"/>
        </owl:Class>
        <owl:Class rdf:ID="tree">
                <rdfs:comment>Trees are a type of plants</rdfs:comment>
                <rdfs:subClassOf rdf:resource="#plant"/>
        </owl:Class>
        <owl:Class rdf:ID="branch">
                <rdfs:comment>Branches are parts of trees </rdfs:comment>
                <rdfs:subClassOf>
                        <owl:Restriction>
                                <owl:onProperty rdf:resource="#is-part-of"/>
                                        <ow:allValuesFrom rdf:resource="#tree"/>
                        </owl:Restriction>
                </rdfs:subClassOf>
        </owl:Class>
        <owl:Class rdf:ID="leaf">
                <rdfs:comment>Leaves are parts of branches</rdfs:comment>
                        <rdfs:subClassOf>
                                <owl:Restriction>
                                        <owl:onProperty rdf:resource="#is-part-of"/>
                                        <owl:allValuesFrom rdf:resource="#branch"/>
                                </owl:Restriction>
                        </rdfs:subClassOf>
        </owl:Class>
        <owl:Class rdf:ID="herbivore">
                <rdfs:comment>
                Herbivores are exactly those animals that eat only plants,
                        or parts of plants
                </rdfs:comment>
                <owl:intersectionOf rdf:parsetype="Collection">
                        <owl:Class rdf:about="#animal"/>
                                <owl:Restriction>
                                        <owl:onProperty rdf:resource="#eats"/>
                                                <owl:allValuesFrom>
                                        <owl:unionOf rdf:parsetype="Collection">
                                                <owl:Class rdf:about="#plant"/>
                                                <owl:Restriction>
                                        <owl:onProperty rdf:resource="#is-part-of"/>
                                        <owl:allValuesFrom rdf:resource="#plant"/>
                                                </owl:Restriction>
                                        </owl:unionOf>
```

```
                                        </owl:allValuesFrom>
                                </owl:Restriction>
                        </owl:intersectionOf>
                </owl:Class>
                <owl:Class rdf:ID="carnivore">
                        <rdfs:comment>Carnivores are exactly those animals
                                that eat also animals</rdfs:comment>
                                <owl:intersectionOf rdf:parsetype="Collection">
                                        <owl:Class rdf:about="#animal"/>
                                                <owl:Restriction>
                                                        <owl:onProperty rdf:resource="#eats"/>
                                                <owl:someValuesFrom rdf:resource="#animal"/>
                                                        </owl:Restriction>
                                </owl:intersectionOf>
                </owl:Class>
                <owl:Class rdf:ID="giraffe">
                        <rdfs:comment>Giraffes are herbivores, and they
                                eat only leaves</rdfs:comment>
                        <rdfs:subClassOf rdf:type="#herbivore"/>
                                <rdfs:subClassOf>
                                        <owl:Restriction>
                                                <owl:onProperty rdf:resource="#eats"/>
                                                <owl:allValuesFrom rdf:resource="#leaf"/>
                                        </owl:Restriction>
                                </rdfs:subClassOf>
                </owl:Class>
                <owl:Class rdf:ID="lion">
                        <rdfs:comment>Lions are animals that eat
                                only herbivores</rdfs:comment>
                        <rdfs:subClassOf rdf:type="#carnivore"/>
                                <rdfs:subClassOf>
                                        <owl:Restriction>
                                                <owl:onProperty rdf:resource="#eats"/>
                                        <owl:allValuesFrom rdf:resource="#herbivore"/>
                                        </owl:Restriction>
                                </rdfs:subClassOf>
                </owl:Class>
                <owl:Class rdf:ID="tasty-plant">
                        <rdfs:comment>Tasty plants are plants that are eaten
                                both by herbivores and carnivores</rdfs:comment>
                        <rdfs:subClassOf rdf:resource="#plant"/>
                                <rdfs:subClassOf>
                                        <owl:Restriction>
                                        <owl:onProperty rdf:resource="#eaten-by"/>
                                                <owl:someValuesFrom>
                                                <owl:Class rdf:about="#herbivore"/>
                                                </owl:someValuesFrom>
                                        </owl:Restriction>
                                </rdfs:subClassOf>
                        <rdfs:subClassOf>
                                <owl:Restriction>
                                        <owl:onProperty rdf:resource="#eaten-by"/>
                                                <owl:someValuesFrom>
                                                        <owl:Class rdf:about="#carnivore"/>
                                                </owl:someValuesFrom>
                                </owl:Restriction>
                        </rdfs:subClassOf>
                </owl:Class>
                <owl:TransitiveProperty rdf:ID="is-part-of"/>
                <owl:ObjectProperty rdf:ID="eats">
                        <rdfs:domain rdf:resource="#animal"/>
                </owl:ObjectProperty>
```

```
        <owl:ObjectProperty rdf:ID="eaten-by">
                <owl:inverseOf rdf:resource="#eats"/>
        </owl:ObjectProperty>
</rdf:RDF>
```

ΑΝΑΦΟΡΕΣ

1. Jorge Cardoso “SEMANTIC WEB SERVICES Theory, Tools and Applications” , Information Science Reference 2007, Ch. XI, p.240.
2. Stephen Potts, Mike Kopack “SAMS Teach Yourself Web Services in 24 Hours”, SAMS 2003 , p.139.
3. Semantic Annotations for WSDL and XML Schema, 27/6/2007 ; www.w2.org/2002/ws/sawsdl/spec/SAWSDL.html.
4. Web Services Description Language (WSDL), 26 June 2007; www.w3.org/TR/wsdl20-primer/
5. The Apache Software Foundation AXIS2 , 13/8/2007; http://ws.apache.org/axis2/1_3/toc.html.
6. Java Tutorials ; http://java.sun.com/docs/books/tutorial/index.html.
7. XPath Tutorial; www.w3schools.com/xpath/default.asp.
8. Protégé - owl api programmer’s guide , 21 September 2006; http://protege.stanford.edu/plugins/owl/api/guide.html.
9. SAWSDL4J Home ; http://knoesis.wright.edu/opensource/sawsdl4j/index.html.
10. Grigoris Antoniou, Frank van Harmelen “A Semantic Web Primer” , MIT Press , Ch 1, 2, 3 , p. 15,37,77.
11. Michael Uschold, Michael Gruninger “Ontologies and Semantics for Seamless Connectivity” , www.sigmod.org/sigmod/record/issues/0412/12.uschold-9.pdf
12. Jason Brittain with Ian F. Darwin “Tomcat The Definitive Guide”, O’ REILLY, October 2007, Ch 2, p. 38.
13. WSMO Studio Users Guide v.127 ; www.wsmostudio.org/doc/wsmo-studio-ug.pdf.
14. Jasek Kopecky and al. “SAWSDL:Semantic Annotations foe WSDL and XML Schema” , http://doi.ieeecomputersociety.org/10.1109/MIC.2007.134.
15. Web Service Modeling Ontology(WSMO) – An ontology for Semantic Web Services, www.w3.org/2005/04/FSWS/Submissions/1/wsmo_position_paper.html .
16. OWL-S: Semantic Markup for Web Services, “www.daml.org/services/owl-s/1.0/owl-s.html .

17. OWL-S Relationship to Selected Other Technologies, “www.daml.org/services/owl-s/1.1/related.html.

18. World Wide Web consortium, www.w3.org.

19. Paolucci, Kawamura & Sycara. Semantic matching of Web Services capabilities “First International Semantic Web Conference on the Semantic Web”, Sardinia, (LNCS2342, pp.333-347).

20. METEOR-S: Semantic Web Services and Processes, http://lsdis.cs.uga.edu/projects/meteor-s/ .