SESSION

SEMANTIC WEB

Chair(s)

TBA

Semantic Discovery and Retrieval of Relevant Medical Knowledge

A. Asmita Rahman[1], B. Budak Arpinar[1], and C. Singaram Sundar[1]
[1]Computer Science Department, University of Georgia, Athens, Georgia, USA 30602

Abstract - *In the fast growing world of information, the amount of medical knowledge is growing at an exponential level. It has now become a very difficult task for an ordinary person to keep up with all the new discoveries and updates in this domain. This paper describes an approach to semantically discover and retrieve relevant medical data/information for respective health records (people). This system comprises of sample Electronic Health Records (EHRs) and Health Publications from PubMed. Our system implements a semantic matchmaking algorithm to find the relevant publications in PubMed for any particular health record (profile) using BioPortal Ontologies and UMLS. It then displays the results to the user. Our system empowers the users and enables them to discover hidden but relevant information. The result of the evaluation clearly proves that our system retrieves the relevant information better than syntactic searches.*

Keywords: Semantic Matchmaking, Matchmaking algorithm, Knowledge Discovery, Electronic Health Records, Health Publications, Ontology.

1 Introduction

We all know that today the knowledge in the medical domain is growing at a very fast pace. It is becoming harder and almost impossible for a normal person to keep up with all the updates in this field. Every day there are several new drugs coming to market, several new treatment options are being introduced, many old medications are being replaced, several new discoveries are being made etc. In this fast moving world, there is barely any time left for a normal person to read and research about the new updates in the medical industry. Our research is going to contribute in this field by making relevant information easily available.

1.1 Motivating Scenario

Mr. Burton is a patient of Dr. Brown. Mr. Burton has had a heart attack in 2005. Dr. Brown has prescribed the drug Plavix to reduce the risk of future heart attacks. As Plavix leads to acid reflux, the doctor has also prescribed the drug Prilosec to lower acidity. Note that until recently this has been the standard treatment regimen for patients with heart attack histories. In March 2009, a study appeared in the Journal of American Medical Association, which indicated that combination of drugs Clopidogrel (Plavix is the brand name of Clopidogrel) and proton pump inhibitor (PPI) Prilosec is one of the PPIs) in patients with previous histories of heart attacks can actually double the risk of second heart attack. This research finding has direct implication on the treatment regimen of Mr. Burton as it puts him in high-risk category for a second heart attack. Currently, there are a few ways in which Dr. Brown can learn about the discovery: (a) searching and browsing relevant web sites (e.g., PubMed); (b) attending a conference/ professional meeting where recent research findings are discussed; or (c) through colleagues who may have knowledge about the new discoveries. However, in all of these methods, there could be significant delays between publishing of new information and Dr. Brown becoming aware of the information. Even after Dr. Brown becomes aware of the study, his staff has to search through patients' medical records to identify the patients who are on Plavix and Prilosec simultaneously which can be difficult process [20]. Here is the diagram that illustrates this scenario:

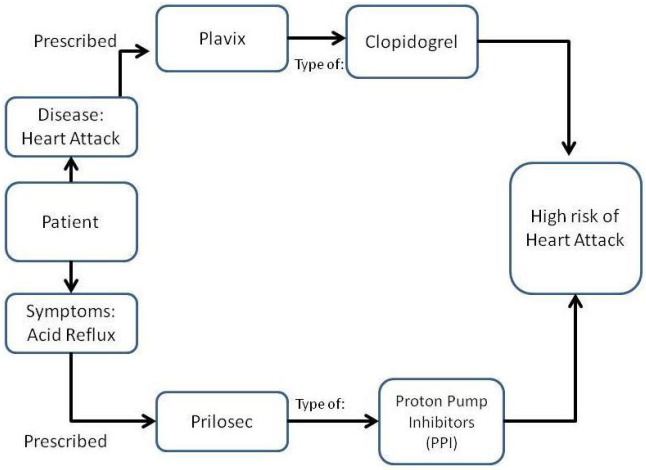


Figure 1: Test case scenario diagram.

Since the matchmaking in our system is done on the semantics rather than the syntax, the knowledge discovery enables the system to find such relevant publications and provide the results to the patient.

2 Related Work

Related works include research done in the field of health and science for clinical trials. It also includes researches that use semantic graphs and relationships for retrieval of information.

TrialX: It is a system, a third party tool that is built for recruiting related health records for clinical trials. As one must realize that before any medication becomes available to the market, there are clinical trials performed to measure the efficiency and side effects of the same. However, this process

of clinical trials currently takes over years due to the fact that finding appropriate people for testing the drug is a laborious process. However, TrialX makes it easier for the people to find the clinical trials related to their health record. It performs a matchmaking algorithm and finds the related clinical trials for any particular health record.

Semantically Connected Named Entities and Relationships (SCOONER): Domain specific searches comprises of knowledge about the domain which serves as the basis of the search. However, there are three major concerns about such available knowledge: (1) exists only for few well known broad domains; (2) is of a basic nature: either purely hierarchical or involves only few relationship types; and (3) is not always kept up-to-date and is missing insights from recently published results. Kno.e.sis at WSU developed a framework addressing the above concerns. Their implementation provides an up-to-date knowledge based search system called SCOONER. The knowledge is extracted from recent bioscience abstracts. It uses a populated ontology (also called knowledge base) for semantic metadata extraction.

Retrieval of Similar Electronic Health Records Using UMLS Concept Graphs [17]. Physicians are often faced with a decision making challenge, in which case they can use the information available to them about the previous clinical trials. However, since the amount of information in this field is large, exhaustive search is unfeasible. This paper proposes an approach to deal with this issue. They propose an approach for the retrieval of similar clinical cases, based on mapping the text onto UMLS concepts and representing the patient records as semantic graphs. They also did a thorough evaluation of the proposed method and the results show that their method correlates well with the expert judgments and outperforms remarkably the traditional term-vector space model.

3 Our Approach

This system consists of the following two major parts; semantic Matchmaking and Semantic Ranking. This research paper focuses on the semantic Matchmaking. The matchmaking performs all the core operations of finding the relevant publications for any particular health record. Once the results are found, the Semantic Ranking provides us a way of calculating the relevance of the publications to a particular record.

The matchmaking and the ranking process are performed semantically where the system uses ontology mappings, synonyms calculation and hierarchy verification for calculating relevant results. Here is a diagram showing the overview of the functionality of our system:

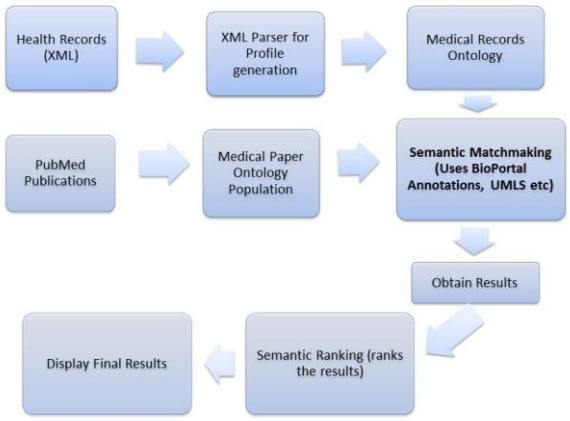


Figure 2: Overview of the System.

Our health record consists of the following personal information: (1) Name, (2) Address, (3) ID, (4) Age, (5) KnownDisease, (6) Medications (7) Gender, (8) Symptoms, (9) PrimaryPhysician, (10) PhysicianId, (11) PrimaryPharmacy and (12) PrimaryPharmacyId.

A sample template was used for generating test health records for our system. Since there was no standard found for generating health records, we used Google health's format as the reference. The health record information is then parsed to create a patient profile with all the pertinent information. Once the profiles have been generated, all the data is populated into ontology for semantic matchmaking. On the other hand, the PubMed publications were downloaded and an ontology was populated with all the information about the medical papers. Once both the ontologies have been populated with health records and medical publications information respectively, the system can begin the matchmaking procedure.

One of the most important parts of matchmaking is to be able to annotate the unstructured text. We need annotations for populating both the ontologies. The publications from the PubMed including their title and abstract would be annotated. Also, the same process would be followed to annotate the information in the health records. Once both the annotations are received, both the ontologies are updated with the respective annotations and matchmaking procedure advances to the next step.

4 Building Blocks

In order to understand the matchmaking process completely, we must examine the following building blocks of the system. These components played a key role in the implementation of the system; (a) Electronic Health Records (b) PubMed (c) UMLS (d) NCBO BioPortal.

4.1 Electronic Health Records

In order to be able to test the system, one must realize the need of health records. However, due to the sensitivity of health records and the information within, it is nearly impossible to be able to work with real records. In order to deal with this shortcoming, sample health records were created for testing purposes based on Google Health's format (Google Health is discontinued now). Here is a sample of the Google Health Record [9]:

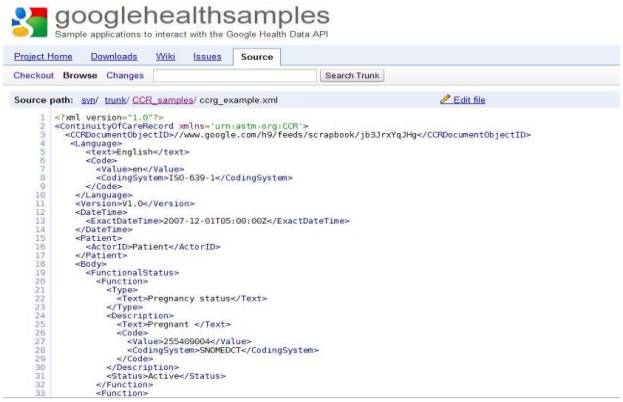


Figure 3: Google Health Samples [9]

Sample health records (75) in XML format, similar to Google Health records for testing purposes were generated. These sample health records would enable the application to work properly even when fed with real health records. Here are a few samples of the health records that were generated:

<Patient>
<Name>Robin Woods</Name>
<Address>1563 South Milton st</Address>
<City>Tuscon</City>
<State>AZ</State>
<Zip>92009</Zip>
<Country>United States</Country>
<Id>1235</Id>
<Age>25</Age>
<KnownDisease>Asthma</KnownDisease>
<Medications>Aerobid, Alvesco</Medications>
<Gender>Male</Gender>
<symptoms>Vomiting</symptoms>
<PrimaryPhysician>Dr Smith</ PrimaryPhysician>
<PhysicianId>dc1247</PhysicianId>
<PrimaryPharmacy>Walgreens</PrimaryPharmacy>
<PrimaryPharmacyId>247Phar</PrimaryPharmacyId>
</Patient>

4.2 PubMed

PubMed comprises more than 21 million citations for biomedical literature. The sources of these citations are MEDLINE, life science journals, and online books. These citations are a combination of both links to full-text content from PubMed Central and from publisher web sites [10]. PubMed is maintained by the National Center for Biotechnology Information (NCBI), at the U.S. National Library of Medicine [11].

PubMed is a free resource and it provides an easy to use search interface to search the publications via the title, journal name, names of authors, specific citations, keywords etc. We have used PubMed as the knowledge resource in this research. About a couple hundred research publications (Abstracts) were downloaded, annotated and then the knowledgebase (Ontology) was populated. This allows the system to do accurate matchmaking and display relevant results.

4.3 UMLS

UMLS stands for Unified Medical Language System and it is a system that brings together health vocabularies, biomedical terms and standards. It enables to enhance and develop applications with use of such information and promotes interoperability. It is a source of a large number of national and international vocabularies and classifications (over 100) and provides a mapping structure between them [13]. The UMLS can be used to design information retrieval for patient record systems, to facilitate the communication between different systems, or to develop systems that parse the biomedical literature. UMLS consists of three knowledge sources [14]: (a) Metathesaurus, (b) Semantic Network and (c) SPECIALIST Lexicon and Lexical Tools.

4.4 NCBO BioPortal

NCBO (National Center for Biomedical Ontology) offers a BioPortal, which can be used to access and share ontologies that are actively used on the biomedical community. By using the BioPortal, one can search the ontologies, search biomedical resources, obtain relationship between terms in different ontologies, obtain ontology based annotations of the text etc. Bio portal is a web- based application [4]. It can be used for browsing, finding, filtering, searching ontologies. It can also be used for submitting new ontologies and for exploring mapping between ontologies.

BioPortal provides access to one of the largest repositories of biomedical ontologies. We can access these by web browsers or via web services (RESTful services). The BioPortal library consists of the following:

Total number of ontologies: 173
Number of classes/types: 1,438,792

4.4.1 NCBO Annotator:

The NCBO annotator provides us with a web service that we can use to process text, to recognize relevant biomedical ontology terms. The NCBO Annotator annotates or "tags" free-text data with terms from BioPortal and UMLS ontologies. It can be accessed via the browser or via the web

service. The web service is flexible enough to allow for customizations particular to any application[5]. For example we can limit results to a particular ontology (e.g. Anatomical entity Ontology) or to a certain UMLS semantic type (e.g. T017 for 'Anatomical Structure').

The annotations are performed in two steps; first is the direct annotations by matching the raw text with the preferred name and then expanding the annotations by considering the ontology mappings and hierarchy. The expanded semantic annotations are obtained by considering the transitive closure, semantic distance and ontology mappings. Here is the workflow of the annotator web service:

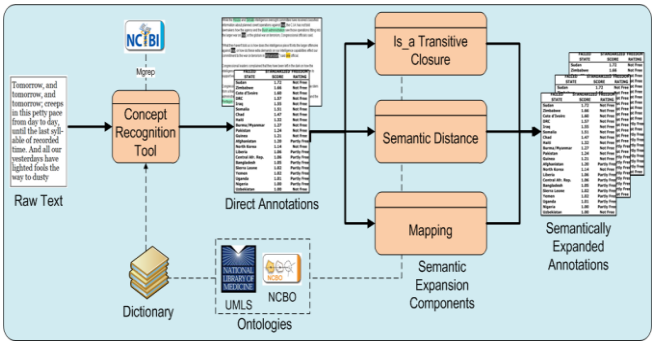


Figure 4: Annotator Workflow [16]

5 Semantic Matchmaking

Matchmaking is a process by which we calculate or compute the related results with respect to a certain entity. For example, if the entity in question was entity A, by applying a matchmaking algorithm, we would search and obtain all the entities and resources related to entity A. This list of results should be calculated based on the semantics of the entity A as well as the semantic annotations of the resulting resources. With respect to our domain, our purpose of matchmaking in this paper is to obtain relevant publications to a particular patient (health record). We perform the matchmaking between the health record and paper publications to obtain relevant results. Semantic matchmaking is different from any other matchmaking in a way that the results are obtained in light of a shared conceptualization for the knowledge domain at hand, which we call ontology.

In order to obtain relevant results, we must ensure that the semantic annotations are accurate. Also, the underlying ontology used should be appropriate, relevant and should provide us with all the possible outputs. One can also use more than a single Ontology to obtain better results. In our matchmaking process we are using UMLS for obtaining the annotations.

5.1 Health Record Ontology

This ontology contains all the patients information with all the results obtained after the annotation process as follows:

(a) Name, (b) ID (unique), (c) Age, (d) Gender, (e) Known disease, (f) Medications, (g) Symptoms, (h) Annotations results for known disease (with synonyms), (i) Annotations results for medications (with synonyms), (j) Annotations results for symptoms (including synonyms)

5.2 Paper Publication Ontology

This ontology contains all the paper publications information. About 150 publication abstracts were downloaded from PubMed for testing purposes. Since the entire paper consists of figures, images, calculations etc. which results in excessive and/or unnecessary annotations, we choose to use only the abstracts for the annotations. This enabled us to get precise annotations and thus better results. Similar to the health records; annotations were obtained to supply better results for the matchmaking. This ontology contains the following information:

(a) Title, (b) Abstract, (c) Publication date, (d) Authors names, (e) Annotations for title, (f) Annotations for abstract

6 Matchmaking Algorithm

As seen in the Figure 5, the matchmaking algorithm starts from the two ontologies. One is for health records and the other one for PubMed Publications. Once the ontologies are populated, matchmaking is performed based on the data and annotations obtained. Here is the workflow indicating the flow of information and the matchmaking process:

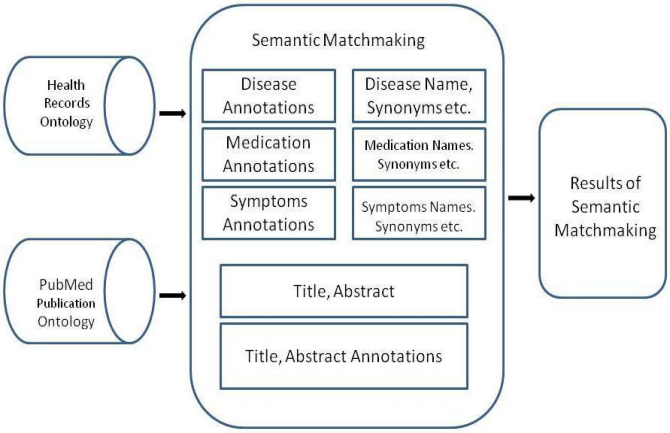


Figure 5: Matchmaking Workflow

The system performs matchmaking of the health records and publications based on the following information:

For the Heath Records: (a) Disease name, (b) Annotations and synonyms of the disease names, (c) Medications, (d) Annotations and synonyms of the medication names, (e) Symptoms and (f) Annotations and synonyms of the medication names.

For the Publications: (a) Title of the paper, (b) Abstract of the paper, (c) Annotations of the title (Considering semantic hierarchy. i.e. strength of the concepts), (d)Annotations of the abstract (Considering semantic hierarchy i.e. strength of the concepts)

Our system now performs the matchmaking and provides the results accordingly. In this process, the system not only performs the keyword matching, but also takes into consideration the semantic hierarchy, transitive closure, ontology mappings, semantic distance, synonyms, annotations etc.

Once the matchmaking is done semantically, it goes above and beyond the keyword matches. This enables the user to get the relevant results regardless of the "word" or the "term" they enter. For example, a person has a symptom of vomiting, however, is unaware of the disease. Suppose that there is a new discovery about people having symptoms of Bilious attack and this discovery is found in one of the new research publications. However, if that person were to search a normal keyword search from their symptoms they would not be able to locate the paper, which discusses about the new discovery with symptoms of Bilious attack. However, with this system and with the underlying ontologies that person will get the results of this new discovery even if the paper does not have the word "vomiting" in it.

7 Testing the Matchmaking Algorithm

Let us consider motivating scenario mentioned in Motivation section. Our system performs the semantic matchmaking and thus provides the following results. It clearly identifies the semantic relationship between the two drugs and thus shows the paper indicating the effects of both drugs when taken together.

Results: Here are the results related to: Mathew burton
Patient Record Number: 1284
Disease: Heart Attack
Rank is: 8
Link is: http://www.ncbi.nlm.nih.gov/pubmed/22053225
Rank is: 7
Link is: http://www.ncbi.nlm.nih.gov/pubmed/22053219
Rank is: 6
Link is: http://www.ncbi.nlm.nih.gov/pubmed/21944415
Rank is: 6
Link is: http://www.ncbi.nlm.nih.gov/pubmed/21573267
Rank is: 5
Link is: http://www.ncbi.nlm.nih.gov/pubmed/21884023
Rank is: 4
Link is: http://www.ncbi.nlm.nih.gov/pubmed/20729752

Here is a Snapshot of the User Interface results:


Name: Mathew burton
Patient Record Number: 1284
Disease: Heart Attack

Here are all the related paper, please click the corresponding links:

Rank : 6
Link: http://www.ncbi.nlm.nih.gov/pubmed/21944415
Preview: There is increasing concern regarding a possible adverse interaction between proton pump inhibitors (PPIs) and clopidogrel that could lead to reduced cardiovascular protectio performed a literature search for relevant original studies and systematic reviews. PPIs likely affect the antiplatelet activity of clopidogrel as measured in vitro, and this may be a class effe the pharmacodynamic effect has not been translated into any clinically meaningful adverse effect. PPI cotherapy reduces the incidence of recurrent peptic ulcer and of upper gastrointestir patients on clopidogrel.

Title: No
Medication: Yes
Symptoms: Yes
Disease: No

Rank : 6
Link: http://www.ncbi.nlm.nih.gov/pubmed/21573267
Preview: The patient with haematemesis and melaena Bleeding from the upper gastrointestinal (GI) tract is a common medical emergency, with an incidence of between 50-150 cases pe A recent audit by the British Society of Gastroenterology showed the mortality rate from upper GI bleeds has fallen from 14%2 in 1993 to 10% in 2007.3 However, despite the use of p (PPIs), admission rates for peptic ulcer haemorrhage have increased in older age groups,4 probably related to increased use of antiplatelet agents such as aspirin and clopidogrel and anti coronary syndromes, stroke and atrial fibrillation. The rising age of the population may also have offset further reductions in mortality and morbidity that may have otherwise come about supportive and endoscopic care.

Figure 6: Snapshot of results of test case

8 Comparison with Syntactic Matchmaking

8.1 Advance Ontological Search

The semantic matchmaking enables the system to perform advance search based on the ontology concepts and hierarchy, which is not possible by a syntactic matchmaking process. This enables the user to be able to discover and retrieve results that would not be found by a simple keyword search. This is an efficient way to discover hidden but important information.

8.2 Discovery of Medication Side Effects

Our system enables a user to not only get the related publications based on the disease that they are suffering from, but also enables them to discover any side effects of the medications and drugs they are taking. The results are not just limited to the disease's name because of using UMLS and 173 ontologies during the matchmaking process. For example if a person is on some medication for a long time and if that drug or medication has some long term side effects; such publications should be displayed to the user. Our system does the same. It gives the user, publications related to the effects of the drugs or medications they are on. For example; a query that was ran on a record suffering from breast cancer, the following result was not only retrieved but also given a good rank:

Rank is: 8
Link is: http://www.ncbi.nlm.nih.gov/pubmed/21993405
Title: Second cancer after radiotherapy 1981-2007

In our system, the side effects of drugs are discovered whether they appear directly or indirectly as it checks the annotations, synonyms etc. This is something that cannot be achieved by syntactic matchmaking.

8.3 Extended Search via Profile

Our system enables the user to retrieve publications that are not only related to his current disease or medications but also papers, which may have some synonyms of the current

medications or symptoms. This search goes beyond the keyword search and retrieves the papers semantically. For example, if we like to conduct matchmaking for someone with the symptoms of vomiting. Also, let's suppose that the patient does not suffer from any disease currently. In a syntactic search we will be able to receive all the results related to vomiting. However, with the help of semantic matchmaking the user will get results pertaining to vomiting including Haematemesis, Bilious attack and Throwing up etc. This enables the user to retrieve complete results regardless of the search term. When searched for symptoms "vomiting" we get the following results:

Rank is:6
Link is: http://www.ncbi.nlm.nih.gov/pubmed/12207199
Title: Vomiting

Rank is:6
Link is: http://www.ncbi.nlm.nih.gov/pubmed/21573267
Title: The patient with haematemesis and melaena

Rank is:6
Link is: http://www.ncbi.nlm.nih.gov/pubmed/21359665
Title: Gastric duplication cysts as a rare cause of haematemesis

8.4 Knowledge Discovery without Specific Input

Our system allows a user to discover the papers related to them without having particular information about the disease that they might be suffering from. Since the search can be done with any one of the parameters (medications, disease, symptoms etc.), the complete information is not mandatory. A person might search based on his symptoms without knowing the name of the disease or a person might just search without having any symptoms but on some particular medication. This enables them to retrieve and discover hidden knowledge. For example with our test case scenario number 2, the two drugs together had side effects which we were able to detect since we took the semantic relationship of both the drugs into consideration.

9 Preliminary Evaluation

In order to evaluate the functionality of our system, we compared our results with the results of PubMed. PubMed provides a user interface to search for publications related to the terms entered. We use the same interface to enter the disease name, symptoms or medications and retrieve results. On the other hand, we use our system and find related papers to a particular record (patient), who is suffering from the same disease, symptoms and takes the same medications. This allowed us to do a comparison on both the results obtained and conclude the results. We used our test scenario number 2 that was explained in the above section for the evaluation purposes. Here is snapshot of the results obtained from PubMed:

User Profile:
Name: Mathew Burton
Known Disease: Heart Attack
Symptoms: Arm pain, Acidity
Medications: Prilosec, Plavix, Alprenolol

Query 1: PubMed Input: Heart Attack, Arm pain, Acidity, Prilosec, Plavix, Alprenolol
PubMed Output: No items found.

Query 2: Prilosec, Plavix, Alprenolol
PubMed Output: No items found.

Query 3: Heart Attack, Arm pain, Acidity
PubMed Output: No items found.

Query 4: Heart Attack
PubMed Output:

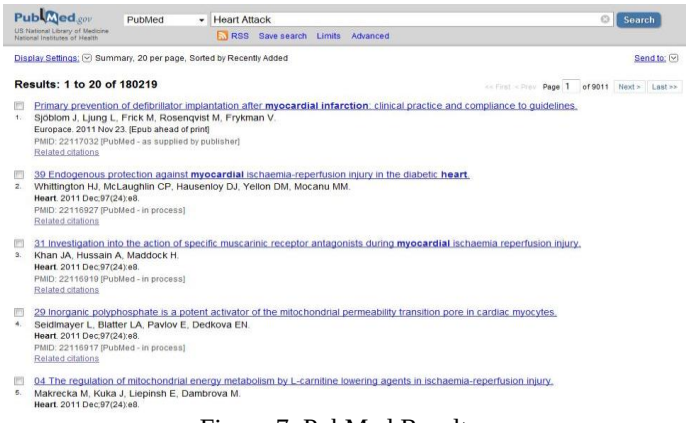


Figure 7: PubMed Results

Here is snapshot of the results obtained from our system:

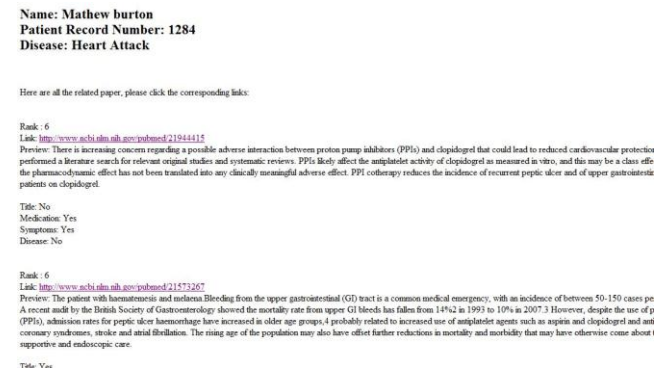


Figure 8: Results Snapshot

We can see that our system, gave the results of papers discussing the combined effects of both the drugs Prilosec and Plavix together, while there was no implicit information given. Our system was able to discover the semantic

relationship between the two drugs and thus showed the related papers in the result, which were not found in the PubMed results.

From the above example it is evident that our system performs better than the searches done at PubMed. Our system not only allows us to search based on the profile or keyword, but it also takes the semantic relationships between the provided information into consideration. Thus in the above example, we did not get results related only to Heart attack, but also results related to symptoms, medications, side effects of medication, combined effect of two medications etc.

10 Conclusions and Future Work

The amount of knowledge in the medical domain is growing exponentially. With this growth, it is becoming a very hard for physicians or the patients to keep track of all the new discoveries. Our system addresses this issue and makes this knowledge discovery easier. Our system performs semantic matchmaking for knowledge discovery. This can be used by physicians or by patients to discover resources related to their Personal Health Record. Since the system performs semantic matchmaking, the results are more precise and accurate. As seen in the above two motivating examples; our system enables the user to discover papers/knowledge that would not have been possible to discover via syntactic matchmaking.

Future works on this system might include an extended evaluation in form of usability studies can be done with the help of doctors and physicians to identify the accuracy of the results.

11 References

[1] Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini. "A Non-Monotonic Approach to Semantic Matchmaking and Request Refinement in E-Marketplaces" International Journal of Electronic Commerce, 2008.

[2] Jonquet, Clement. "Semantic Annotations of BioMedical Data". http://www.slideshare.net/jonquet/semantic-annotation-of-biomedical-data-7281656. March 2011.

[3] Good, Benjamin. "NCBO Annotator versus MetaMap on GO concept detection". http://i9606.blogspot.com/2010/12/ncbo-annotator-versus-metamap-on-go.html. December 02, 2010.

[4] Jonquet, Clement. Musen, Mark A and Shah, Nigam. "A System for Ontology-Based Annotation of Biomedical Data". Proceedings of the 5th international workshop on Data Integration in the Life Sciences. "http://bmir.stanford.edu/file_asset/index.php/1316/Article-DILS08_Jonquet_Musen_Shah_published.pdf." 2008.

[5] Patricia L. Whetzel, Nigam H. Shah, Natalya F. Noy, Clement Jonquet, Adrien Coulet, Nicholas Griffith, Cherie Youn, Michael Dorf and Mark A. "Ontology Web Services for Semantic Applications". Stanford University, Stanford CA, USA. "http://www2.lirmm.fr/IC//Supports/FMIN113-ProjetTutore/2010_11/ExemplesPosters/CJ2.pdf." 2008.

[6] U.S. National Library of Medicine. "MetaMap". "http://www.nlm.nih.gov/research/umls/implementation_resources/metamap.html" . September 2011.

[7] Alan R. Aronson, PhD and François M. Lang, MSE. "The Evolution of MetaMap, a Concept Search Program for Biomedical Text". Lister Hill National Center for Biomedical Communications. "http://www.lhncbc.nlm.nih.gov/lhc/docs/published/2009/pub2009041.pdf". 2009.

[8] Aronson, Alan R. Lang, François-Michel. "An overview of MetaMap: historical perspective and recent advances" J Am Med Inform Assoc (JAMIA)."http://www.lhncbc.nlm.nih.gov/lhc/docs/published/2010/pub2010033.pdf". 2010

[9] Google Health Samples. "http://code.google.com/p/googlehealthsamples/source/browse/trunk/CCR_samples/". Retrieved on September 2011.

[10] PubMed.com "http://www.ncbi.nlm.nih.gov/pubmed/" 2011.

[11] PubMed Quick Start. U.S. National Library of Medicine National Institutes of Health. "http://www.ncbi.nlm.nih.gov/books/NBK3827/#pubmedhelp.PubMed_Quick_Start". Retrieved on Aug 2011.

[12] PubMed FAQS. U.S. National Library of Medicine National Institutes of Health. "http://www.ncbi.nlm.nih.gov/books/NBK3827/#pubmedhelp.FAQs". Retrieved on Aug 2011.

[13] UMLS. Open Clinical Knowledge Management for Medical Care. "http://www.openclinical.org/medTermUmls.html " Retreived on Aug 2011.

[14] UMLS. Wikepedia. "http://en.wikipedia.org/wiki/Unified_Medical_Language_System" Retrieved on Aug 2011.

[15] Chintan Patel, Sharib Khan, and Karthik Gomadam. "TrialX: Using semantic technologies to match patients to relevant clinical trials based on their Personal Health Records". In Proceedings of the 8th International Semantic Web Conference, 2009.

[16] OBA Service Workflow. BioOntology Wiki. http://www.bioontology.org/wiki/index.php/File:OBA_service_workflow.png" retrieved on Sep 2011.

[17] Laura Plaza, Alber Diaz. "Retrieval of Similar Electronic Health Records Using UMLS Concept Graphs". Proceedings of the Natural language processing and information systems, 2010.

[18] Barbara Hayes, William Aspray. "Fighting Diabetes with Information: Where Social Informatics Meets Health Informatics". iConference, 2010.

[19] Semantic Network, U.S. National Library of Medicine National Institutes of Health. "http://www.ncbi.nlm.nih.gov/books/NBK9679/". Retrieved on Nov 2011.

[20] L. Ramaswamy, and I. B. Arpinar, "Semantics-enabled Proactive and Targeted Dissemination of New Medical Knowledge", CSHALS 2011: Conference on Semantics in Healthcare and Life Sciences, Feb 2011, Cambridge/Boston MA.

The Role of Semantics in Testing Semantic Web Services

Hazlifah Mohd Rusli[1], **Suhaimi Ibrahim**[2], **Mazidah Puteh**[1] **and Naz'ri Mahrin**[2]
[1]Faculty of Computer Science & Mathematics, Universiti Teknologi MARA, Terengganu, Malaysia
[2] Advanced Informatics School, Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia

Abstract – *Semantic Web Services are Web Services that are semantically annotated in order to make the services machine understandable, thus allowing service discovery, selection, composition, and invocation to be done automatically or with minimum human intervention. The Semantic Web services research community has been focusing on how these semantics can facilitate service discovery, selection, composition, and invocation. As of late, there have been some growing research interests in the area of Semantic Web services testing. However, it is not stated how the semantic annotations in the Web services description can help improve testing and how different it is from testing normal Web services. This paper discusses current ongoing research on testing Semantic Web services and classifies how testing uses the semantics of the Semantic Web services description.*

Keywords: A Semantic Web Service; Software Testing;

1 Introduction

More and more web based applications are being developed according to the service oriented architecture (SOA) framework. It is estimated that by the year 2015, 80% of web based applications will be developed using this architectural strategy [1]. One way of implementing SOA is by using Web services. Web services are service providing web applications whose service descriptions are advertised in a repository such as Universal Description, Discovery and Integration Protocol (UDDI) registry. Web service enables interoperability between heterogeneous web applications by leveraging on standards such as Extensible Markup Language (XML), Web Service Definition Language (WSDL), Simple Object Access Protocol (SOAP) and, UDDI. While these standards allow interoperability between applications developed using different languages and on different platforms, they do not facilitate the automation of Web services tasks such as service discovery, selection, composition, and invocation. Semantic Web services (SWS) were introduced to provide solutions to the automation of these tasks. Adding semantics to the Web service descriptions enables the descriptions to be understood by machines, thus allowing automation of Web services tasks. Although there are many research conducted in the area of SWS discovery, selection, composition and, mediation, research in the areas of Web services testing, particularly, SWS testing is still new and starting to garner interest.

This paper will focus on Semantic Web services testing and offers an overview and classification of state-of-the-art approaches. The question is how semantics in the service description is used in testing Semantic Web services? Therefore the objective of this article is to find out the role of semantics in testing Semantic Web services and to provide a classification of the testing approaches used. It is hoped that the result of this paper will be able to provide an essential perspective on how testing of Semantic Web services differ from that of normal syntax based Web services.

The remainder of this paper is organized as follows. A brief overview of current syntax based Web services testing is discussed in Section 2. Section 3 briefly describes current prominent Semantic Web services approaches as well as the rule language used. A classification of the SWS testing approaches is presented in Section 4. Section 5 summarizes and discuses the testing approaches. Section 6 concludes the review.

2 Web Services Testing

At the very basic, testing a Web service consists of generating SOAP message request to be sent to the service provider from the service client. The SOAP message response is then analyzed to determine whether it is the same as what is expected of it. Testing a Web service involves having a test data generator, SOAP message generator, message executor, test oracle generator and, a test analyzer at the very least. Generating test cases based solely on the WSDL files have been researched by Sneed [2] and Bartolini [3]. For composite services, several approaches [4, 5] have been proposed for testing using Business Process Execution Language (BPEL) [6].

Both WSDL and BPEL are syntax based Web service description. Syntax based description lacks the necessary information to facilitate the automation of tasks such as selection, discovery, composition and, mediation. Semantic Web services were introduced to provide solution to the problem. While numerous researches catered to finding solutions to service selection, discovery, composition, and mediation, not many were focused on Web services testing. Only recently, there has been a growing research interest in testing Semantic Web services. This paper will discuss several

[2] Suhaimi Ibrahim and Naz'ri Mahrin are supported by Research University Grant Vot 00H68

Semantic Web services testing approaches and classify them, but before that, a brief description on Semantic Web services approaches is provided in the following section.

3 Semantic Web Services Approach

Semantic Web services are Web services that are semantically annotated using ontology and rule languages. The word semantics itself means meaning and the semantic annotations provide meaning to the Web service description. Several initiatives have been proposed to implement Semantic Web services such as OWL-S [7], WSMO [8] and, WSDL-S [9]. This section briefly discusses the initiatives, focusing on the elements of the initiatives that are used in testing.

3.1 OWL-S

OWL-S is an ontology for describing Web services, hence the name Web Ontology Language for services. OWL-S ontology includes three primary sub-ontologies which are service profile, process model and grounding. The profile ontology describes what the service does, the process model ontology describes how the service is used, and finally the grounding ontology describes how to interact with it. While the service profile provides a way to describe the services offered by the providers and the services needed by the requesters, the process model describes the interaction protocol between a Web service and its client. It is organized as a workflow of processes where each processes is described by three components which are inputs, preconditions and results. Results specify the output and effect based on the process condition. The inputs and outputs use Web Ontology Language (OWL) [10] as representation whilst the precondition and postcondition use rule languages such as Semantic Web Rule Language (SWRL) [11].

3.2 WSMO

WSMO stands for Web Service Modeling Ontology. WSMO identifies four main top level elements which are ontologies, Web services, goals and mediators. Ontologies provide the terminology used by the other elements of the WSMO. Web services are computational entities that provide some value in a specific domain. Goals describe a user's desire or objectives when consulting Web services. Mediators describe elements that handle interoperability problems between WSMO elements. Web services and goals have two main sub-elements which are capability and interface. The capability of the service contains a set of axioms that describes the precondition, postcondition, assumption and effect. The interface of a service describes how to interact with the web service (choreography) and how the service provides its functionality by making use of other services (orchestration). Both choreography and orchestration are defined in terms of their state signature and transition rules. A state signature defines the state ontology and the transition rules change the state according to the given condition.

3.3 WSDL-S

WSDL-S or Web Service Semantics is a semantic web services approach that builds on existing Web services standards by annotating WSDL with ontological information for the domain model along with pre-conditions and effects for each operation in the service. Unlike OWL-S and WSMO, it does not duplicate descriptions in the existing WSDL, but rather enhances it. WSDL-S also allows developers to use any semantic language of their choice and not fixed to just OWL, WSML or UML. In WSDL-S the WSDL operations, input and output are annotated with semantics. The Web service operations are associated with pre and post conditions using modelReference annotation on WSDL portType. Similar to OWL-S, the preconditions and postconditions can be described using rule languages such as SWRL [11] and Object Constraint Language (OCL) [12].

4 Classification of SWS Testing

Literature search on Semantic Web services testing resulted in the discovery of sixteen research papers with the earliest publication in 2005. The research papers were searched from IEEE Xplore, Springer Link, Science Direct as well as Google scholar using search strings "Semantic Web services testing", "OWL-S and testing", "WSMO and testing" and "WSDL-S and testing". Once a research paper has been identified as a SWS testing paper, other relevant papers were also discovered by looking at the references of the identified paper as well searching for other papers that have cited the identified paper. Based on the issues discussed in the selected papers, we have classified the papers into three categories which are test case generation (12 papers), mutation testing (3 papers) and, test selection (1 paper). The research papers worked on SWS initiatives or specification languages such as OWL-S (8 papers), WSMO (3 papers) and WSDL-S (3 papers). The remaining two papers did not specify the exact SWS used but rather, they used the general notion of Input, Output, Precondition and Effect (IOPE) that is inherent in OWL-S, WSMO and WSDL-S. Table 1 describes the test issues addressed by the papers as well as the description language used. The following section will further discuss the issues highlighted in Table 1.

Table 1: SWS Testing Research Paper Breakdown

	SWS Approach			
Test Issues	**OWL-S**	**WSMO**	**WSDL-S**	**General IOPE**
Test Case Generation	4	3	3	2
Mutation Testing	3	-	-	-
Test Selection	1	-	-	-

4.1 Test Case Generation

Test cases are a set of test inputs and expected results created to exercise a particular program path or to verify compliance to a specification. Manually creating test cases can be tedious as well as time consuming. Furthermore, creating test cases manually does not support the Semantic Web services objective which is to facilitate the automation of Web services usage. Generating test input data from web services description is done by selecting data from a database that corresponds to the input parameter type. The intelligence of the test generator can be improved by selecting data that not only corresponds to the parameter type, but also satisfy the precondition of the Web service [13].

Table 2: SWS Test Case Generation Techniques

Test Case Generation Technique	SWS Approach			
	OWL-S	WSMO	WSDL-S	General IOPE
EFSM based	-	-	1	1
Petri-Net based	3	-	-	-
Model Checking	1	2	-	-
Planner based	-	-	-	1
Decision Table	-	-	1	-
Equivalence Partition & Boundary Condition testing	1	1	-	-
Pair Wise Testing	-	-	1	-

Several testing techniques have been employed in generating test cases such as Extended Finite State Machine (EFSM) based, Petri-net based testing, model checking, planner based, decision table, equivalence partitioning and boundary condition testing and, pair wise testing. Table 2 shows the number of research done for each test generation techniques as well as the semantic web services approaches used. The following section will discuss the different techniques in generating test cases using different semantic web services approaches.

4.1.1 Extended Finite State Machine Based (EFSM)

Finite state machines (FSM) are behavioural model used in designing computer programs and is often used for defining the temporal order of Web service interaction. However it is not sufficient to describe all aspect of a Web service as Web service have input and output messages with data parameter. In order to better describe Web service behavior, EFSM extends FSM with variables, statements and conditions [14]. Semantic Web services testing based on EFSM involves creating an EFSM representation of the Web service behaviour and applying existing EFSM test case generation to the created EFSM model [15, 16].

Sinha and Paradkar [15] proposed an algorithm that translates WSDL-S behavioural specification of a single Web service into its equivalent EFSM representation. The preconditions and effect of the WSDL-S operations are expressed using SWRL. Sinha and Paradkar suggested several EFSM based test generation techniques that can be applied such as Full predicate coverage, BZ-TT method and mutation based techniques. However there was no further elaboration on which test generation was selected and applied. Ramollari [16] did not focus on any particular SWS description but rather described how IOPE elements of SWS operations encoded in Rule Interchange Format–Production Rule Dialect (RIF-PRD) and OWL could be used to generate Stream X-machine model (SXM), which is a type of an EFSM.

4.1.2 Petri-Net Based

In testing composite Web services, it is not sufficient to just evaluate the preconditions and postconditions of the service operations, as the internal execution process needs to be tested as well. Dai [13] and Wang [17] proposed the use of Petri-Net model to represent the structure and operational semantics of composite Web services due to its ability to analyze and verify properties such as reachability, liveness and deadlocks. The Perform construct in OWL-S process model is represented by Petri-Net transition, the input and preconditions mapped to Petri-Net input places and output and effects mapped to Petri-Net output places. The Petri-Net model is then traversed and test cases are generated to cover all branches of the Petri-Net. A Petri-Net ontology that catered for the IOPE semantics was created to enhance the modeling capability of the derived Petri-Net model. The derived Petri-Net ontology can be used to generate test data based on ontology reasoning.

4.1.3 Model Checking

Model checking technique has been used in generating test cases based on semantic web services description by Huang [18] and Jokhio [19, 20]. The issue in using model checking approach for generating test cases for semantic web services is how to convert or translate the existing specification language into the input language of the model checker, taking into account the semantic description. Another issue is the derivation of trap properties. In all approaches the trap properties are embedded into the input specification language as assertions. In [19], mapping rules were created to map the goal capability and interface into B elements in order to create a B specification. Like WSMO, B language is based on abstract state machine, The state signatures are mapped to B variable types, transition rules are mapped to B operations and state ontology is mapped to sets in B machine. Huang [18] converted the OWL-S process model control construct and IOPE logical formulas expressed in Planning Domain Definition Language (PDDL) into BLAST's C-like specification input language.

4.1.4 Planner Based

Planning, or more specifically AI (artificial intelligence) planning is a process that chooses and organizes actions by anticipating the expected outcomes in order to change the state of the system. The final result of planning is a plan. Given the initial state of the world and a goal state, the planner is able to come up with a plan that consists of a sequence of actions that corresponds to a sequence of state transitions. Paradkar [21] created a planner that is able to generate a sequence of web service invocations as a test case. The test goals are generated by refining the preconditions from each Web service using a set of fault models. These goals are then fed into the planner, together with the initial state of the world and web service definitions in order to generate test cases. Similar to Ramollari's approach [16], Paradkar does not state the actual SWS used in the approach.

4.1.5 Decision Table

Decision table is a black box testing technique used to represent complex logical relationship, where a number of combinations of actions are taken based on varying sets of conditions. A decision table has four portions, which are condition stub, action stub, condition entries and action entries. A column in the entry portion is a rule. Each rule indicate actions to be taken for the conditional circumstances inidicated in the condition part of the rule. Generating test cases from decision tables involves indentifying the input conditions (causes) and action (effect), generating a cause-effect graph and converting it into a decision table and finally convert the decision table rules into test cases. Noikajana [22] used this technique to generate test cases from WSDL-S where the preconditions and effects of the operations in WSDL-S were described using SWRL. The idea is to automatically populate the condition and action entries using SWRL's antecedent and consequents. Conditions are obtained from antecedents of precondition and post-condition whilst actions are obtained from consequents of precondition and post-condition.

4.1.6 Equivalence Partitioning & Boundary Condition Testing

In equivalence partition testing, the program's input domain is divided into sub-domains where the sub-domains have properties that cause that program under test to either produce incorrect answer for every input element in the sub-domain or correct answers for every input element in the sub-domain. One of the issues of equivalence partitioning testing is the generation of equivalence class. It is mostly performed manually based on heuristics. Bai [23] tries to provide a solution to this problem by using the ontology information of the input parameter in OWL-S to generate input partitions. The input partitions are derived by analyzing the relationship, property and restriction of the input parameter class. Similarly, Jokhio [24] proposed to identify equivalence class using the transition rules of WSMO goals.

The guarded condition of the transition rules are obtained from the goal's choreography interface. Jokhio [24] also proposed the use of precondition's logical predicate to identify boundary condition for test data generation. Boundary value condition testing is an extension of equivalence partition testing in which values at the equivalence class boundaries are selected as test input.

4.1.7 Pair-Wise Testing

Exercising a set of all possible input combination of software under test is not practical as the test case would be too large to be executed exhaustively. Pair wise testing is an economical alternative to testing all possible input combinations, based on the observation that most faults are caused by interactions of at most two factors. In pair-wise testing a tester needs to first select data values for the system input variables, after which test cases are generated by covering all combinations of the selected test data values for each pair of input variables. As is it is very tedious to generate these reduced test cases combinations by hand, several pair wise testing algorithms exists to automate the process.

Noikajana [25] uses OCL to specify pre and post conditions of Web services operations described in WSDL-S. Based on the service rules and operations definitions derived from both WSDL-S and OCL file analysis, an input parameter model (IPM) is generated. The IPM consist of a set of input parameters and a set of values for each parameter. A pair-wise testing technique is then applied to the IPM. However, it is not stated which pair-wise testing technique is used.

4.2 Mutation Testing

Mutation testing is a technique where two or more mutant programs are executed together with the same test cases in order to determine the ability of the test cases to detect the mutants. Lee [26] proposed mutation testing based on OWL-S specification where the typical errors that might occur are incorrect use of ontology class in the input/output parameter, mistake in the rules defining the precondition and postconditions, control flow and data flow error in the OWL-S composite process. Based on these possible errors, Lee have identified four categories of mutants which are input/output data, condition, control flow and data flow mutation. The input/output data mutant operators were generated from two perspectives which are input type and ontology class definition, and they were the only mutants discussed in the paper.

Similar to Lee' work [26], Wang [27] and Wang [28] also conducted research on mutation testing based on OWL-S. However, instead of using existing specification in OWL-S, the authors proposed extensions to OWL-S to accommodate interaction requirements in terms of the temporal, invocation time, application data and response time properties. The mutation operators were then generated from these extensions which use Future Time Linear Temporal Logic (FTLTL) and SWRL to describe the extended properties.

4.3 Test Selection

Although comprehensive testing is necessary to ensure the quality of a software system, executing all possible test cases can be expensive as it takes up valuable time, machine and tester resources. Bai [29] proposed an ontology based risk assessment approach to test selection where test cases for service features with high risks are given priority. Software risk assessment identifies crucial parts of the system which have a high failure probability rate or which causes serious consequences due to its failure. Thus the service features' risks are assessed based on their failure probability and importance. Bai [29] estimates the failure probability of ontology class and service's interface which is obtained from domain experts or historical data. The failure probability for ontology class is adjusted according to the failure probability of its dependencies (parent and property class) whilst the failure probability of service's interface is adjusted according to its input and output parameters. Failure probability of a composite service can be estimated based on the control flow analysis obtained from the process model of OWL-S.

Table 3: Summary of SWS Testing Approaches

Reference	Year	Semantic Description Utilization	Approach	SWS Description	Rule Langugae
Sinha [15]	2006	Describe service behaviour	EFSM	WSDL-S	SWRL
Ramollari [16]	2009	Describe service behaviour	EFSM	General IOPE	RIF-PRD
Dai [13],Wang [17]	2007	Describe service orchestration	Petri Net	OWL-S	Not mentioned
Huang [18]	2005	Describe service orchestration	Model Checking	OWL-S	Not mentioned
Jokhio [19, 20]	2009	Describe service behaviour	Model Checking	WSMO	Axioms
Paradkar [21]	2007	Generate test goals for test planner input	Planner Based	General IOPE	Not mentioned
Noikajana [22]	2008	Determine test partitions for test data selection	Decision Table	WSDL-S	SWRL
Bai [23]	2008	Determine test partitions for test data selection	Equivalence Partition & Boundary Condition	OWL-S	Not mentioned
Jokhio [24]	2009	Determine test partitions for test data selection	Equivalence Partition & Boundary Condition	WSMO	Not mentioned
Noikajana [25]	2009	Determine test partitions for test data selection	Pair Wise	WSDL-S	OCL
Lee [26]	2008	Generate test mutants	Existing semantic desc.	OWL-S	Not mentioned
Wang [27, 28]	2008, 2009	Generate test mutants	Extension to OWL-S	OWL-S	Not mentioned
Bai [29]	2009	Calculate service feature risk	Test selection	OWL-S	Not mentioned

5 Discussion

A summary of the test issues discussed in the previous section has been provided in Table 3. As mentioned previously, a majority of the test issues involves test case generation, which involves test data generation and for composite services, test process generation is included as well. In the approaches, the semantic elements of the services description were utilized for 1) describing the service behavior in terms of a test model 2) describing the service behavior of a composite service in terms of a test model 3) determining test partitions for test data selection 4) calculating service feature risks 5) generating test mutants.

Sinha [15], Ramollari [16] and Jokhio [19, 20] developed test models to represent behavior of single services and used the developed test models to generate test cases using existing testing tools associated to the test models. The test cases developed consist of a sequence of input and outputs that goes through all possible state transitions. All four approaches focused on how to create formal representation of the service behavior using the IOPE information, a task which normally requires a developer's or tester's intuitive, experience as well as understanding of the system under test.

Similarly, Dai [13], Wang [17] and Huang [18] also focused on the creation of formal test models using semantic descriptions. However, the generated test model represents the orchestration of a composite service instead of a single service behavior. Apart from using the IOPE information, these approaches also utilizes the OWL-S process model information in order to generate test cases. The generated test cases consisted of a test process that contains a sequence of service invocations as well as the input and output from one service to another in the composite service. Paradkar [21] also generated a sequence of service invocations as test cases. However, unlike Wang [17] and Huang [18], no formal test model were generated. Instead, test goals were generated by refining the service preconditions using a set of fault models which was eventually fed into a planner based test generator.

Preconditions and postconditions allow test generators to generate test input data and expected output that not only conforms to input and output parameter type, but also satisfies rules associated with them. Approaches such as those of Bai [23], Jokhio [24] and Noikajana [25] are also able to systematically select test data from a data pool according to some partition criteria such as equivalence partitioning and boundary condition testing. Again, the task of determining these partitions are usually based on tester's heuristic and understanding of the system under test. Fortunately this task can be automated to some extent via the use of input parameter ontology information, rules in IOPE, and guarded

condition of transition rules. Semantic descriptions of Web services have also been used in calculating risks associated with a service failure [29]. The result is used to select and prioritize test cases such that test cases will be tested according to their priority whenever there is limitation in testing time.

All the approaches we have discussed so far have shown how semantic descriptions of Web services can be used to improve test generator's intelligence for generating test models, determining test data partitions and calculating service feature risks. However, approaches by Lee [26], Wang [27] and Wang [28] tries to verify the proper use of the semantics elements by creating mutants out of the ontology class, rules defining preconditions and postconditions, as well as control and data flow error in composite process model.

Based on the discussions, semantic description of Web services is able to assist in increasing automation of Web service testing by enhancing the intelligence of the test generator such as deriving test model, determining test data partition, generating ontology based mutants and, calculating ontology risks. Normally these testing tasks require the experience and knowledge of a human tester in order to perform them. We believe the use of semantics can minimize human efforts in performing testing tasks. However, most of the approaches are still in their early stages with many of them only reporting results of early findings. More work needs to be done to enhance research in this promising area of Semantic Web services testing.

6 Conclusion

In conclusion, this paper has provided a review on Semantic Web services testing with the aim of understanding how testing of Semantic Web services differ from testing normal Web services and what are the advantages of using semantic descriptions of Web services in testing SWS. An introduction to syntax based WS testing is presented. This is followed by a brief description of the Semantic Web services approaches focusing on the elements of each Semantic Web services approach that are used in testing. The most prominent approaches of testing Semantic Web services are then presented and classified into several categories which are test case generation, mutation testing and test selection, yet it is not possible to claim that the list is exhaustive. A summary of the approaches are presented and the results are discussed. Result of the literature study indicates that semantic annotations of Semantic Web services can improve test intelligence by assisting with tasks that normally require human tester's experience and knowledge. We are currently in the process of studying how these semantic annotations can be used to describe the interaction behavior of Semantic Web services in order to support interaction testing of composite Web services.

7 References

[1] J. Vaughan: "Gartner: SOA will be like electricity for architects looking toward cloud computing," SOA News, http://searchsoa.techtarget.com/news/article/0,289142,sid26_gci1523670,00.html, [27 October 2011]

[2] H. M. Sneed, and S. Huang, "The design and use of WSDL-Test: a tool for testing Web services," Journal of Software Maintenance and Evolution: Research and Practice, vol. 19, September/October 2007, pp. 297-314, doi: 10.1002/smr.v19:5.

[3] C. Bartolini, A. Bertolino, E. Marchetti, and A. Polini, "WS-TAXI: A WSDL-Based Testing Tool for Web Services," Proc. International Conference on Software Testing Verification and Validation (ICST'09), IEEE Computer Society, 1-4 April 2009, pp. 326 - 335, doi:10.1109/ICST.2009.28

[4] S.-S. Hou, L. Zhang, Q. Lan, H. Mei, and J.-S. Sun, "Generating Effective Test Sequences for BPEL Testing," Proc. 9th International Conference on Quality Software (QSIC'09), IEEE Computer Society, 24-25 August 2009, pp. 331-340, doi:10.1109/QSIC.2009.50

[5] T.-D. Cao, P. Felix, and R. Castanet, "WSOTF: An Automatic Testing Tool for Web Services Composition," Proc. Fifth International Conference on Internet and Web Applications and Services (ICIW), IEEE Computer Society, 9-15 May 2010, pp. 7-12, doi:10.1109/ICIW.2010.9

[6] Oasis: "Web Services Business Process Execution Language Version 2.0," http://docs.oasis-open.org/wsbpel/2.0/Primer/wsbpel-v2.0-Primer.pdf, [27 October 2011]

[7] W3C Working Group: "OWL-S: Semantic Markup for Web Services " http://www.w3.org/Submission/OWL-S/, [27 October 2011]

[8] W3C Working Group: "Web Service Modeling Ontology (WSMO)," http://www.w3.org/Submission/WSMO/, [27 October 2011]

[9] W3C Working Group: "Web Service Semantics - WSDL-S," http://www.w3.org/Submission/WSDL-S/, [27 October 2011]

[10] W3C Working Group: "OWL Web Ontology Language," http://www.w3.org/TR/owl-features/, [27 October 2011]

[11] Protege: "SWRL Language FAQ," http://protege.cim3.net/cgi-bin/wiki.pl?SWRLLanguageFAQ, [12 October 2011]

[12] Object Management Group: "Object Constraint Language," http://www.omg.org/spec/OCL/2.0/PDF, [27 October 2011]

[13] G. Dai, X. Bai, Y. Wang, and F. Dai, "Contract-Based Testing for Web Services," Proc. 31st Annual International Computer Software and Applications Conference (COMPSAC 2007), IEEE Computer Society, 24-27 July 2007, pp. 517-526, doi:10.1109/COMPSAC.2007.100

[14] C. Keum, S. Kang, I.-Y. Ko, J. Baik, and Y.-I. Choi, "Generating Test Cases for Web Services Using Extended Finite State Machine," Proc. The 18th IFIP International Conference on Testing Communicating Systems (TestCom 2006), Springer, 16-18 May 2006, pp. 103-117, doi:10.1007/11754008_7

[15] A. Sinha, and A. Paradkar, "Model-Based Functional Conformance Testing of Web Services Operating on Persistent Data," Proc. Workshop on Testing, Analysis, and Verification of Web services and Applications (TAV-WEB'06), ACM, 17 July 2006, pp. 17-22, doi:10.1145/1145718.1145721

[16] E. Ramollari, D. Kourtesis, D. Dranidis, and A. Simons, "Leveraging Semantic Web Service Descriptions for Validation by Automated Functional Testing," Proc. 6th European Semantic Web Conference (ESWC 2009), Springer, 31 May - 4 June 2009, pp. 593-607, doi:10.1007/978-3-642-02121-3_44

[17] Y. Wang, X. Bai, J. Li, and R. Huang, "Ontology-Based Test Case Generation for Testing Web Services," Proc. Eighth International Symposium on Autonomous Decentralized Systems (ISADS '07), IEEE Computer Society, 21-23 March 2007, pp. 43-50, doi:10.1109/ISADS.2007.54

[18] H. Huang, W.-T. Tsai, R. Paul, and Y. Chen, "Automated Model Checking and Testing for Composite Web Services," Proc. Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2005), IEEE Computer Society, 18-20 May 2005, pp. 300-307, doi:10.1109/ISORC.2005.16

[19] M. S. Jokhio, G. Dobbie, and J. Sun, "A Framework for Testing Semantic Web Services Using Model Checking," Proc. Fourth South-East European Workshop on Formal Methods (SEEFM), IEEE Computer Society, 4-5 December 2009, pp. 17-24, doi:10.1109/SEEFM.2009.11

[20] M. S. Jokhio, "Goal-Based Testing of Semantic Web Services," Proc. 24th IEEE/ACM International Conference on Automated Software Engineering (ASE '09), IEEE Computer Society, 16-20 November 2009, pp. 707-711, doi:10.1109/SEEFM.2009.11

[21] A. M. Paradkar, A. Sinha, C. Williams, R. D. Johnson, S. Outterson, C. Shriver, and C. Liang, "Automated Functional Conformance Test Generation for Semantic Web Services," Proc. IEEE International Conference on Web Services (ICWS 2007), IEEE Computer Society, 9-13 July 2007, pp. 110-117, doi:10.1109/ICWS.2007.48

[22] S. Noikajana, and T. Suwannasart, "Web Service Test Case Generation Based on Decision Table," Proc. The Eighth International Conference on Quality Software (QSIC '08), IEEE Computer Society, 12-13 August 2008, pp. 321-326, doi:10.1109/QSIC.2008.7

[23] X. Bai, S. Lee, W.-T. Tsai, and Y. Chen, "Ontology-Based Test Modeling and Partition Testing of Web Services," Proc. IEEE International Conference on Web Services (ICWS '08), IEEE, 23-26 September 2008 pp. 465-472, doi:10.1109/ICWS.2008.111

[24] M. S. Jokhio, G. Dobbie, and J. Sun, "Towards Specification Based Testing for Semantic Web Services," Proc. Australian Software Engineering Conference (ASWEC'09), IEEE Computer Society, 14-17 April 2009, pp. 54-63, doi:10.1109/ASWEC.2009.38

[25] S. Noikajana, and T. Suwannasart, "An Improved Test Case Generation Method for Web Service Testing from WSDL-S and OCL with Pair-Wise Testing Technique," Proc. 33rd Annual IEEE International Computer Software and Applications (COMPSAC'09), IEEE Computer Society, 20-24 July 2009 pp. 115-123, doi:10.1109/COMPSAC.2009.25

[26] S. Lee, X. Bai, and Y. Chen, "Automatic Mutation Testing and Simulation on OWL-S Specified Web Services," Proc. 41st Annual Simulation Symposium (ANSS 2008), IEEE Computer Society, 3-16 April 2008, pp. 149-156, doi:10.1109/ANSS-41.2008.13

[27] R. Wang, and N. Huang, "Requirement Model-Based Mutation Testing for Web Service," Proc. 4th International Conference on Next Generation Web Services Practices (NWESP '08), IEEE Computer Society, 20-22 October 2008, pp. 71-76, doi:10.1109/NWeSP.2008.20

[28] X. Wang, N. Huang, and R. Wang, "Mutation Test Based on OWL-S Requirement Model," Proc. IEEE International Conference on Web Services (ICWS 2009), IEEE Computer Society, 6-10 July 2009, pp. 1006-1007, doi:10.1109/ICWS.2009.129

[29] X. Bai, and R. S. Kenett, "Risk-Based Adaptive Group Testing of Semantic Web Services," Proc. 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC'09), IEEE Computer Society, 20-24 July 2009, pp. 485-490, doi:10.1109/COMPSAC.2009.180

Benchmarking Bottom-Up and Top-Down Strategies for SPARQL-to-SQL Query Translation

Andrey Kashlev[a], Artem Chebotko[b,c], John Abraham[b], Pearl Brazier[b], and Shiyong Lu[a]
[a]Department of Computer Science, Wayne State University, Detroit, Michigan, USA
[b]Department of Computer Science, University of Texas - Pan American, Edinburg, Texas, USA
[c]Corresponding Author. E-mail: chebotkoa@utpa.edu

Abstract— *Many researchers have proposed using conventional relational databases to store and query large Semantic Web datasets. The most complex component of this approach is SPARQL-to-SQL query translation. Existing algorithms translate SPARQL queries to SQL using either bottom-up or top-down strategy and result in semantically equivalent but syntactically different relational queries. While it can be expected that relational query optimizers produce identical query execution plans for semantically equivalent bottom-up and top-down queries, is this usually the case in practice? And if not, which strategy yields faster SQL queries? To address these questions, this work studies bottom-up and top-down translations of SPARQL queries with nested optional graph patterns that yield SQL queries with left outer joins whose reordering is not always possible. This paper presents: i) a bottom-up nested optional graph pattern translation algorithm, ii) a top-down nested optional graph pattern translation algorithm, and iii) a performance study featuring SPARQL queries with nested optional graph patterns over RDF databases created in Oracle, DB2, and PostgreSQL.*

Keywords: SPARQL; SQL; translation; query; bottom-up; top-down; Semantic Web; RDF; query optimization; query performance

1. Introduction

Semantic Web technologies are finding more and more applications in solving challenging problems of intelligent data and computing resources search, discovery, sharing, and integration. Numerous RDF [1] datasets, such as UniProt, GeoNames, WordNet, DBpedia, and hundreds of others[1], have become available over the Web for use and exploration. The rapid growth of semantic datasets brings forward a new challenge - efficient management of RDF data that is crucial for supporting new semantics-enabled applications.

Many researchers have proposed using conventional relational databases to store and query large Semantic Web datasets [2]. Emerged systems, called *relational RDF databases*, share a common design pattern that uses a schema mapping algorithm to generate a relational database schema, a data mapping algorithm to insert RDF data into the database, and a query mapping algorithm to translate RDF queries into equivalent SQL queries. SPARQL-to-SQL translation is not only the most complex mapping in a relational RDF database, but also very critical to overall querying performance. Existing algorithms translate SPARQL queries to SQL using either bottom-up or top-down strategy and result in semantically equivalent but syntactically different relational queries.

To illustrate the difference between bottom-up and top-down SPARQL-to-SQL translations in the context of nested optional graph patterns, we use a sample RDF graph G in Fig. 1 that describes academic relations among professors and graduate students in a university. The graph is presented both graphically and as a set of triples. The RDF schema defines two concepts/classes (*Professor* and *GradStudent*) and two relations/properties (*hasAdvisor* and *hasCoadvisor*). Each relation has the *GradStudent* class as a domain and the *Professor* class as a range. Additionally, two instances of *Professor*, two instances of *GradStudent* and relations among these instances are defined as shown in the figure.

We design an RDF query that returns (1) every graduate student in the RDF graph; (2) the student's advisor if this information is available; and (3) the student's coadvisor if this information is available and if the student's advisor has been successfully retrieved in the previous step. In other words, while the query attempts to find students and as many advisors as possible, there is no point to return a coadvisor if no advisor is assigned to a student. The SPARQL representation of this query is as follows:

```
SELECT ?s ?a ?c
WHERE {
  ?s type GradStudent .     /* R1(s)   */
  OPTIONAL {
    ?s hasAdvisor ?a .      /* R2(s,a) */
    OPTIONAL {
      ?s hasCoadvisor ?c . /* R3(s,c) */
} } }
```

The query has three variables: *?s* for student, *?a* for advisor, and *?c* for coadvisor. There are two *OPTIONAL* clauses, where the innermost one is the nested *OPTIONAL* clause. For the purpose of illustration, let's assume that each individual triple pattern in the query is translated into

[1]W3C SWEO Linking Open Data community project, http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

$$
\begin{aligned}
G = \{\ & (John, && type, && Professor),\\
& (Artem, && type, && Professor),\\
& (Andrey, && type, && GradStudent),\\
& (Nathan, && type, && GradStudent),\\
& (Andrey, && hasAdvisor, && Artem),\\
& (Andrey, && hasCoadvisor, && John),\\
& (Nathan, && hasCoadvisor, && Artem) \quad \}
\end{aligned}
$$

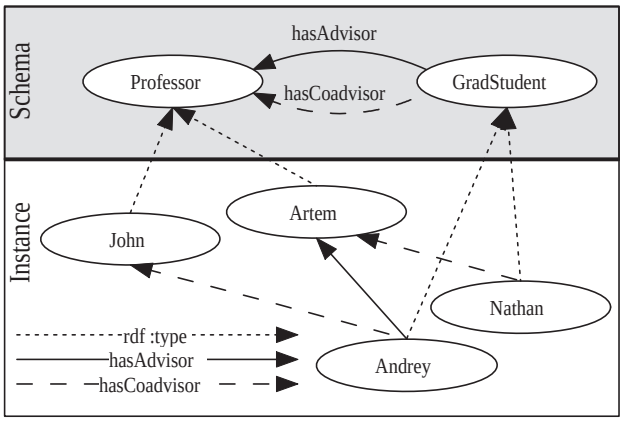


Fig. 1: Sample RDF graph.

SQL and represented by a virtual relation that captures the corresponding variable bindings: *R1(s)* ← (*?s type GradStudent*), *R2(s,a)* ← (*?s hasAdvisor ?a*), and *R3(s,c)* ← (*?s hasCoadvisor ?c*). Then, SQL queries generated for this SPARQL query using our bottom-up and top-down translations presented in [3] and [4], respectively, are:

```
/* Bottom-up query */
Select R1.s As s, a, c
From R1
Left Outer Join
   (Select R2.s As s, a, c
    From R2 Left Outer Join R3
    On (R2.s = R3.s)) R4
On (R1.s = R4.s Or R1.s Is Null
    Or R4.s Is Null)
```

and

```
/* Top-down query */
Select R4.s As s, a, c
From
   (Select R1.s As s, a
    From R1 Left Outer Join R2
    On (R1.s = R2.s)) R4
Left Outer Join R3
On (R4.s = R3.s And R4.a Is Not Null)
```

Both bottom-up and top-down SQL queries have two left outer joins, however the join order and conditions are different. The evaluation of these queries produces the same resulting relations as shown in Fig. 2.

The research that we report is motivated by the following two questions: *While it can be expected that relational query optimizers produce identical query execution plans for semantically equivalent bottom-up and top-down queries, is this usually the case in practice? And if not, which strategy yields faster SQL queries?* In our search for the answers, in this paper, we present i) a bottom-up nested optional graph pattern translation algorithm, ii) a top-down nested optional graph pattern translation algorithm, and iii) a performance study featuring SPARQL queries with nested optional graph patterns over RDF databases created in Oracle, DB2, and PostgreSQL.

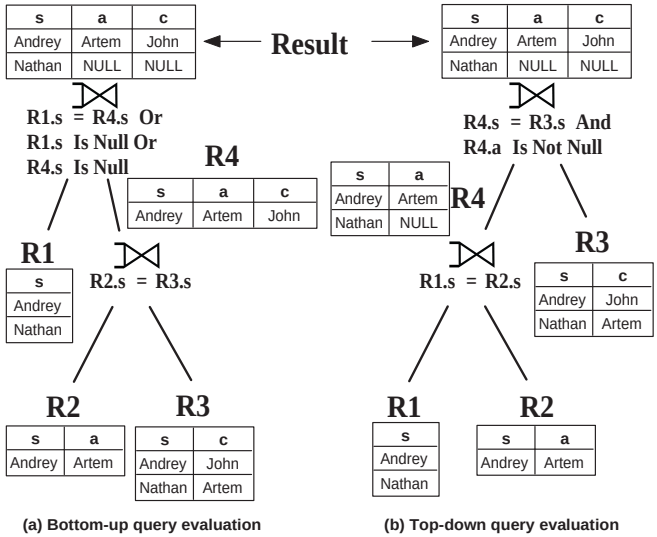


Fig. 2: Evaluation of top-down and bottom-up queries.

The organization of this paper is as follows. Related work is discussed in Section 2. Notation and preliminary definitions are introduced in Section 3. Our algorithms for bottom-up and top-down nested optional graph pattern translations are presented in Sections 4 and 5, respectively. Finally, our performance study and conclusions are reported in Sections 6 and 7.

2. Related Work

In recent years, a number of relational RDF database systems have been developed to support large-scale Semantic Web applications [2]. Representatives of such systems include Jena, Sesame, 3store, KAON, RStar, OpenLink Virtuoso, PARKA, DLDB, DBOWL, RDFSuite, RDFBroker, RDFProv, and S2ST (see [2] or [3] for a survey). While they share a common design pattern, they differ in employed database schemas, inference support and algorithms that map RDF data and queries to the relational model.

One of the most complex mappings in relational RDF databases is the SPARQL-to-SQL query mapping or translation [3], [5], [6], [4], [7], [8]. Existing algorithms translate SPARQL queries to SQL using either bottom-up or top-down strategy and result in semantically equivalent but syntactically different relational queries. To our best knowledge, this work is the first to compare bottom-up and top-down query translations in the context of complex nested optional graph

patterns. The importance of such a comparison is twofold: it gives insights to the query optimization problem of choosing a "good" translation strategy for a particular query and motivates future research on a potentially hybrid translation strategy where both bottom-up and top-down approaches are employed. While we present this work in the context of relational RDF databases, its insights are also beneficial for query optimization in non-relational RDF databases, such as emerging Hadoop and HBase based RDF data management systems in the cloud environment [9], [10].

Other related works on RDF query optimization that are complimentary to our research include containment and minimization of RDF/S query patterns [11], SPARQL query rewriting [12], and various RDF data indexing techniques [13], [14], [15], [16].

3. Notation and Preliminary Definitions

Let I, B, L, and V denote pairwise disjoint infinite sets of Internationalized Resource Identifiers (IRIs), blank nodes, literals, and variables, respectively. Let IB, IL, IV, IBL, and IVL denote $I \cup B$, $I \cup L$, $I \cup V$, $I \cup B \cup L$, and $I \cup V \cup L$, respectively. Elements of the set IBL are also called *RDF terms*. In the following, we formalize the notions of RDF triple, RDF graph, triple pattern, basic graph pattern, and nested optional graph pattern.

Definition 3.1 (*RDF triple and RDF graph*): An RDF triple t is a tuple $(s, p, o) \in (IB) \times I \times (IBL)$, where s, p, and o are a subject, predicate, and object, respectively. An RDF graph G is a set of RDF triples.

Definition 3.2 (*Triple pattern*): A triple pattern tp is a triple $(sp, pp, op) \in (IVL) \times (IV) \times (IVL)$, where sp[2], pp, and op are a subject pattern, predicate pattern, and object pattern, respectively.

Definition 3.3 (*Basic graph pattern*): A basic graph pattern bgp is a set of triple patterns $\{tp_1, tp_2, \ldots, tp_{n-1}, tp_n\}$, also denoted as $tp_1\ AND\ tp_2\ AND \cdots AND\ tp_{n-1}\ AND\ tp_n$, where AND is a binary operator that corresponds to the conjunction in SPARQL and n is the number of triple patterns in bgp.

Definition 3.4 (*Nested optional graph pattern*): A nested optional graph pattern $nogp$ has the form $bgp_1\ OPT\{\ bgp_2\ OPT\{\ \cdots\ \{bgp_{n-1}\ OPT\{\ bgp_n\}\}\ \cdots\}\}$, where OPT corresponds to the *OPTIONAL* construct in SPARQL, curly braces $\{\}$ denote nesting of graph patterns, and $n \geq 3$ represents the number of basic graph patterns in $nogp$.

Formal semantics of RDF and SPARQL are described in [17], [5], [3]. In this paper, to achieve the semantic equivalence of SQL queries that result from bottom-up and top-down SPARQL-to-SQL translations, we require nested optional graph patterns to be *well-designed* [5], such that for any sub-pattern $\{bgp_{i-1}\ OPT\{\ bgp_i \cdots\}\}$ in $nogp$, if a variable $?v$ occurs both outside this sub-pattern and inside bgp_i, then $?v$ also occurs in bgp_{i-1}.

In order to support a generic translation of SPARQL graph patterns into equivalent SQL queries over different database schemas, we need a generic representation for a relational RDF storage scheme, in which the following information will be modeled: (1) which relation is used to store RDF triples that can potentially match a triple pattern, and (2) which relational attributes of the relation are used to store the components (subjects, predicates, and objects) of triples. To capture this information, we formalize the relational RDF storage scheme as the following two RDF-to-Relational mappings α and β.

Definition 3.5 (*Mapping* α): Given a set of all possible triple patterns $TP = (IVL) \times (IV) \times (IVL)$ and a set of relations REL in a relational RDF database, a mapping α is a many-to-one mapping $\alpha : TP \rightarrow REL$, if given a triple pattern $tp \in TP$, $\alpha(tp)$ is a relation in which all the triples that may match tp are stored.

Definition 3.6 (*Mapping* β): Given a set of all possible triple patterns $TP = (IVL) \times (IV) \times (IVL)$, a set $POS = \{sub, pre, obj\}$, and a set of relational attributes ATR in a relational RDF database, a mapping β is a many-to-one mapping $\beta : TP \times POS \rightarrow ATR$, if given a triple pattern $tp \in TP$ and a position $pos \in POS$, $\beta(tp, pos)$ is a relational attribute whose value may match tp at position pos.

Examples of different storage schemes captured with α and β can be found in our prior work [3].

In addition to mappings α and β, our translation uses three auxiliary functions: (1) a function $alias$ that generates a unique alias for a relation, (2) a function $vars$ that returns a set of all variables in a graph pattern, and (3) a function $name$ that generates a unique name for a variable in V, such that the generated name conforms to the SQL syntax for relational attribute names (e.g., a variable can be "renamed" by simply removing initial '?' or '$').

Finally, for the brevity of our presentation, we assume the existence of an algorithm that translates SPARQL basic graph patterns into fully flat SQL queries. We denote such an algorithm as function BGPtoFlatSQL; a similar algorithm is presented in [18].

4. Bottom-Up Nested Optional Graph Pattern Translation

The bottom-up approach to SPARQL-to-SQL query translation is well-studied in the literature [3] and implemented in many relational RDF databases. This section presents an algorithm that implements one of our translation rules described in [3]. It should be noted that, while this paper assumes that nested *OPTIONAL* clauses contain basic graph patterns, which is sufficient for our study, in the general

[2]Note that a triple pattern can have a literal as a subject pattern, while an RDF triple cannot have a literal as a subject. This inconsistency between current RDF [1] and SPARQL [17] specifications does not affect our work and most likely will be resolved by W3C.

case, other graph patterns, such as sequential optional graph patterns and alternative graph patterns, are possible. The algorithm uses the translation rule for the general case with an additional simplification that eliminates the call of the `Coalesce` function for some attributes in projection lists. The use of `Coalesce` is redundant with only basic graph patterns assumed in *OPTIONAL* clauses; however, other simplifications on join conditions are not applied.

Our bottom-up translation function NOGPtoSQL-BU is outlined in Algorithm 1. It visits each basic graph pattern in a SPARQL nested optional graph pattern $nogp$ starting from bgp_n and going up to bgp_1. Each basic graph pattern is translated to SQL using function BGPtoFlatSQL producing a flat SQL query. During the first loop iteration, the translation of bgp_n is assigned to variable sql and the translation of bgp_{n-1} is assigned to variable sql_i. A new SQL query that computes a left outer join between virtual relations sql_i and sql is constructed. This query contains: "`($`sql_i`) $`a_1 `Left Outer Join (`$`$sql$`) $`$a_2$" in its `From` clause, where $a_1$ and $a_2$ are unique aliases; a join condition "$`$a_1$`.`$`$ra$ `= $`$a_2$`.`$`$ra$ `Or $`a_1`.`$`$ra$ `Is Null Or $`$a_2$`.`$`$ra$ `Is Null`" in its `On` clause, which requires common relational attributes in a_1 and a_2 to be equal or one of them to be `Null`; and a projection list in its `Select` clause of all attributes in a_1 and all other unique attributes in a_2. This newly constructed query is assigned to variable sql, overwriting its previous value. The following loop iteration repeats the procedure but with a new value of sql as previously described and a new value of sql_i that now holds the translation of bgp_{n-2}. After the final iteration, a value of sql represents a fully generated query and is returned.

Algorithm 1 Bottom-up translation of SPARQL nested optional graph patterns to SQL queries

1: **function** NOGPtoSQL-BU
2: **input:** nested optional graph pattern $nogp$; mappings α and β; functions $alias$, $vars$, and $name$
3: **output:** bottom-up SQL query
4: Let $nogp = bgp_1\ OPT\{\ bgp_2\ OPT\{\ \cdots\ \{bgp_{n-1}\ OPT\{\ bgp_n\}\}\ \cdots\}\}$ and $n \geq 3$
5: //Construct a bottom-up SQL query:
6: sql = BGPtoFlatSQL$(bgp_n, \alpha, \beta, alias, vars, name)$
7: **for** $i = n-1$; $i \geq 1$; $i = i-1$ **do**
8: //Construct the SQL `From` clause:
9: sql_i = BGPtoFlatSQL$(bgp_i, \alpha, \beta, alias, vars, name)$
10: $a_1 = alias()$; $a_2 = alias()$
11: $from$ = "`($`sql_i`) $`a_1 `Left Outer Join (`$`$sql$`) $`$a_2$"
12: //Construct a join condition:
13: $cond$ = "`True `"
14: **for** each relational attribute ra that appears in the `Select` clause of both sql_i and sql **do**
15: $cond$ += "`And ( $`a_1`.`$`$ra$ `= $`$a_2$`.`$`$ra$ `Or $`a_1`.`$`$ra$ `Is Null Or $`$a_2$`.`$`$ra$ `Is Null )`"
16: **end for**
17: //Construct the SQL `Select` clause:
18: $select$ = ""
19: **for** each relational attribute ra that appears in the `Select` clause of sql_i **do**
20: $select$ += "`$`a_1`.`$`$ra$ `As $`$ra$`,`"
21: **end for**
22: **for** each relational attribute ra that appears in the `Select` clause of sql but not sql_i **do**
23: $select$ += "`$`a_2`.`$`$ra$ `As $`$ra$`,`"
24: **end for**
25: sql = "`Select $`$select$ `From $`$from$ `On(`$`$cond$`)`"
26: **end for**
27: **return** sql
28: **end function**

5. Top-Down Nested Optional Graph Pattern Translation

One of the first top-down SPARQL-to-SQL query translations found in the literature is described in our unpublished report [4]. This section summarizes our solution for the case when only basic graph patterns are used in *OPTIONAL* clauses.

Our top-down translation function NOGPtoSQL-TD is outlined in Algorithm 2. The logic of this algorithm is similar to the logic described for NOGPtoSQL-BU. One obvious difference is that function NOGPtoSQL-TD visits each basic graph pattern in a SPARQL nested optional graph pattern $nogp$ starting from bgp_1 and going down to bgp_n. The other difference lies in how a join condition is generated. It encodes the following semantics: before a nested optional graph pattern can succeed, all containing optional graph patterns must have succeeded. Therefore, a join condition must check that a basic graph pattern in a containing *OPTIONAL* clause has a solution. This is achieved via a `Not Null` check on a relational attribute with special properties: this attribute must appear in the `Select` clause of sql, since the translation of the containing graph pattern is part of sql, and it must correspond to a variable that first occurred in a basic graph pattern of the containing *OPTIONAL* clause and not in any preceding basic graph pattern. If such an attribute is not readily available, a new attribute for a "dummy" variable can be introduced in a basic graph pattern to perform the check. Further details on this solution can be found in [4].

6. Performance Study

This section reports our query performance study conducted using the WordNet dataset and test SPARQL queries that were translated to SQL using the proposed bottom-up and top-down query translation algorithms and evaluated in three relational database management systems.

6.1 Experimental Setup

The experiments were conducted on a server with two 2GHz Intel Xeon E5504 Nehalem CPUs, 32GB RAM and 6TB disk array running Ubuntu 9.02 Jaunty x64. Three different database management systems, namely Oracle 10.2 Express Edition, DB2 9.7 Express-C and PostgreSQL 8.3.12, were installed on the server.

Algorithm 2 Top-down translation of SPARQL nested optional graph patterns to SQL queries

function NOGPtoSQL-TD
input: well-designed nested optional graph pattern $nogp$; mappings α and β; functions $alias$, $vars$, and $name$
output: top-down SQL query
Let $nogp = bgp_1\ OPT\{\ bgp_2\ OPT\{\ \cdots\ \{bgp_{n-1}\ OPT\{\ bgp_n\}\}\ \cdots\}\}$ and $n \geq 3$
//Construct a top-down SQL query:
sql = BGPtoFlatSQL($bgp_1, \alpha, \beta, alias, vars, name$)
for $i = 2$; $i \leq n$; $i = i + 1$ **do**
  //Construct the SQL `From` clause:
  sql_i = BGPtoFlatSQL($bgp_i, \alpha, \beta, alias, vars, name$)
  $a_1 = alias()$; $a_2 = alias()$
  $from$ = "`(`sql`)` a_1 `Left Outer Join (`sql_i`)` a_2"
  //Construct a join condition:
  Let v be a relational attribute that (1) appears in the `Select` clause of sql, (2) $v = name(?v)$ corresponds to a variable $?v$, and (3) variable $?v \in vars(bgp_{i-1}) - (vars(bgp_1) \cup vars(bgp_2) \cup \cdots \cup vars(bgp_{i-2}))$ first occurs in bgp_{i-1} but not in $bgp_1, bgp_2, \cdots, bgp_{i-2}$. If sql has no attribute that satisfies these conditions, a dummy attribute must be introduced as discussed in [4].
  $cond$ = "$v `Is Not Null`"
  for each relational attribute ra that appears in the `Select` clause of both sql and sql_i **do**
    $cond$ += " `And` $a_1.$$ra = $a_2.$$ra$"
  end for
  //Construct the SQL `Select` clause:
  $select$ = ""
  for each relational attribute ra that appears in the `Select` clause of sql **do**
    $select$ += "$a_1.$$ra `As` $ra`,`"
  end for
  for each relational attribute ra that appears in the `Select` clause of sql_i but not sql **do**
    $select$ += "$a_2.$$ra `As` $ra`,`"
  end for
  sql = "`Select` $select `From` $from `On (`$cond`)`"
end for
return sql
end function

Our algorithms were implemented in Java 6 within the S2ST[3] system; generic schema and data mapping algorithms supported by S2ST were used to generate identical database schemas in Oracle, DB2 and PostgreSQL, and to store the RDF dataset into the databases, respectively.

6.2 Dataset and Test Queries

The OWL representation of WordNet[4] was chosen for our experiments. WordNet is a lexical database for the English language, which organizes English words into synonym sets according to part of speech (e.g. noun, verb, etc.) and enumerates linguistic relations between these sets. In the *WordNet.OWL*, each part of speech is modeled as an *owl:Class*, and each linguistic relation is modeled as an *owl:ObjectProperty*, *owl:DatatypeProperty*, *owl:TransitiveProperty*, or *owl:SymmetricProperty*. The relevant statistics for the WordNet dataset is shown in Table 1. For example, *WordNet.OWL* contains 251,726 triples involving `rdf:type` as the predicate, and 140,470 of them have `wn:WordObject` as the object.

Table 1: Properties and Resources in WordNet 1.2

Property	Count
type	251,726
wordForm	195,802
glossaryEntry	111,223
hyponymOf	90,267
similarTo	22,494
antonymOf	7,115
Others	36,225
Total	714,852

Resource	Count
WordObject	140,470
Noun	75,804
Verb	13,214
AdjectiveSatellite	11,231
Adjective	7,345
Adverb	3,629
Others	33
Total	251,726

Table 2 shows 22 SPARQL queries over the WordNet dataset that were carefully selected for our experiments. In the table, `W` stands for *WHERE* and `O` stands for *OPTIONAL*; the SPARQL `SELECT` clause is omitted for brevity, and the projection includes all distinct variables of a query. Queries Q1-Q6 are constructed as all possible permutations of the three triple patterns occurring outside and inside *OPTIONAL* clauses. These queries have one nested *OPTIONAL* clause. Queries Q1′-Q6′ and Q1″-Q6″ are obtained from respective queries Q1-Q6 by restricting variable values in the first and second triple patterns, respectively. The rationale for such restrictions is to reduce cardinalities of intermediate relations resulting from first left outer joins in the queries. In particular, in terms of the intermediate relation size, Q1′-Q6′ favor the top-down approach and Q1″-Q6″ favor the bottom-up approach. We chose not to restrict variable values in the third triple pattern of the nested *OPTIONAL* clause in any of queries Q1-Q6 because the relation that results after matching the third triple pattern is always used as the right operand of a left outer join and therefore can only marginally influence the join result for the given dataset and queries. Finally, queries Q7, Q8, Q7′, and Q8′ are interesting because they only include triple patterns of the same form with the same predicate and variables as subject and object patterns. From the viewpoint of bottom-up and top-down translations, these queries are "symmetric".

6.3 Bottom-Up and Top-Down Query Performance

The S2ST system was used to generate database schemas with property relations [3] and load *WordNet.OWL* into Oracle, DB2 and PostgreSQL. The test SPARQL queries were translated to SQL using algorithms NOGPtoSQL-BU and NOGPtoSQL-TD. The resulting SQL queries were evaluated by RDBMSs. To prevent an unintentional comparison of the three RDBMSs, Fig. 3 reports the ratio of a bottom-up query evaluation time to a top-down query evaluation time for each test query. In the figure, if *ratio*

[3]S2ST: Next-Generation Relational RDF Database Management System (RRDBMS), `http://www.s2st.org`

[4]WordNet (version 1.2), a lexical database for English, `http://wordnet.princeton.edu`

Table 2: Test SPARQL Queries

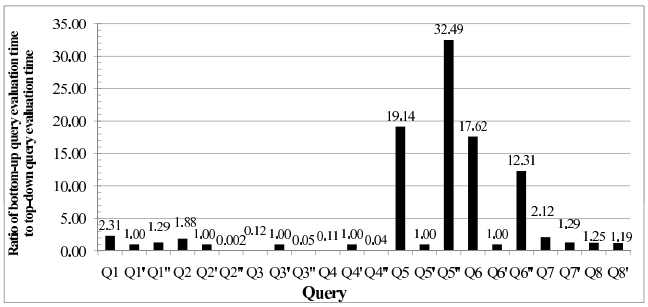
Q#	SPARQL
Q1	`W{?a rdf:type :Adjective O{?a :wordForm ?c O{?a :glossaryEntry ?b}}}`
Q2	`W{?a rdf:type :Adjective O{?a :glossaryEntry ?b O{?a :wordForm ?c}}}`
Q3	`W{?a :wordForm ?c O{?a rdf:type :Adjective O{?a :glossaryEntry ?b}}}`
Q4	`W{?a :glossaryEntry ?b O{?a rdf:type :Adjective O{?a :wordForm ?c}}}`
Q5	`W{?a :wordForm ?c O{?a :glossaryEntry ?b O{?a rdf:type :Adjective}}}`
Q6	`W{?a :glossaryEntry ?b O{?a :wordForm ?c O{?a rdf:type :Adjective}}}`
Q7	`W{?n1 :hyponymOf ?n2 O{?n2 :hyponymOf ?n3 O{?n3 :hyponymOf ?n4}}}`
Q8	`W{?n1 :hyponymOf ?n2 O{?n2 :hyponymOf ?n3 O{?n3 :hyponymOf ?n4 O{?n4 :hyponymOf ?n5 O{?n5 :hyponymOf ?n6 O{?n6 :hyponymOf ?n7}}}}}}`
Q1′-Q6′	Same as respective queries Q1 - Q6 but with one variable in the first triple pattern (the `W` clause) restricted to a URI or literal
Q1″-Q6″	Same as respective queries Q1 - Q6 but with one variable in the second triple pattern (the first `O` clause) restricted to a URI or literal
Q7′	Same as Q7 but with `?n1` and `?n4` restricted to URIs
Q8′	Same as Q8 but with `?n1` and `?n7` restricted to URIs

> 1, a top-down query was faster; if *ratio* < 1, a bottom-up query was faster; and if *ratio* $= 1$, both top-down and bottom-up queries showed the same execution times.

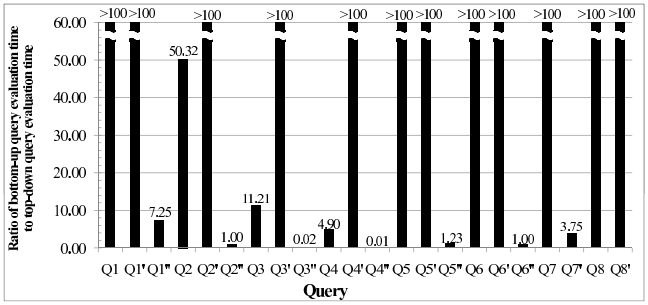
Our *first observation* was that bottom-up and top-down queries generally showed different execution times. This observation gave the definite "*No*" answer to question "*While it can be expected that relational query optimizers produce identical query execution plans for semantically equivalent bottom-up and top-down queries, is this usually the case in practice?*" in the case of SPARQL queries with nested optional graph patterns.

Our *second observation* was that different database management systems showed quite different and sometimes even "contradicting" query evaluation ratios. For example, Oracle showed much less contrast between bottom-up and top-down approaches than DB2 and PostgreSQL. Some queries, such as Q1″, Q3, Q4, Q5″, and Q6″, showed different classes of ratios (> 1, < 1, and $= 1$) in different databases. For example, for Q6″, the bottom-up approach was slower than the top-down approach in Oracle, equivalent to the top-down approach in DB2, and faster than the top-down approach in PostgreSQL.

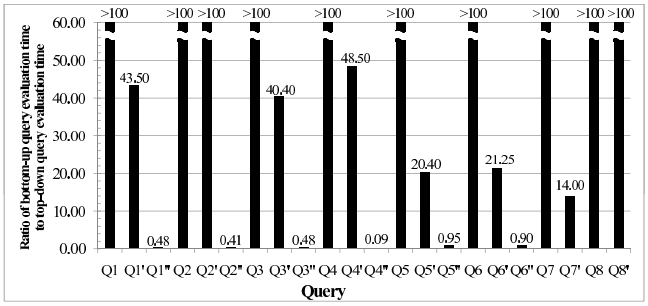
Our *third observation* was that selectivities of participating triple patterns and their occurrence in a SPARQL query had a significant impact on which SPARQL-to-SQL translation strategy won, which could be explained by a similar effect of cardinalities of join participating relations and intermediate relations on corresponding top-down and bottom-up SQL queries. In particular, top-down queries Q1 and Q2 were consistently faster in all experiments, given that the first triple pattern *?a rdf:type :Adjective* yielded the smallest result set of $7,345$ triples (the other two triple patterns yielded over 10 times larger results), and therefore the intermediate relation in the top-down queries was also small and over 10 times



(a) over an RDF database instantiated in Oracle



(b) over an RDF database instantiated in DB2



(c) over an RDF database instantiated in PostgreSQL

Fig. 3: Bottom-up and top-down query performance.

smaller than the intermediate relation in the corresponding bottom-up queries. When *?a rdf:type :Adjective* occurred in the first *OPTIONAL* clause of Q3 and Q4, the situation was opposite: the intermediate relation in the bottom-up queries was over 10 times smaller than the intermediate relation in the corresponding top-down queries. However, while all three systems showed that the ratios decreased when compared to Q1 and Q2, only Oracle showed the advantage of the bottom-up approach, and DB2 and PostgreSQL still ran top-down queries faster. Moving *?a rdf:type :Adjective* to the nested *OPTIONAL* clause in Q5 and Q6 did not favor one or the other translation strategy since the last triple pattern did not influence the size of an intermediate relation. Top-down queries Q5 and Q6 were consistently faster in all experiments. Next, restricting selectivities of the first triple pattern in Q1′-Q6′ to 1 or 2 triples, which was favorable for the top-down approach, showed that the top-down queries

were faster or as fast as the corresponding bottom-up queries. Interestingly, Oracle showed identical performance for both top-down and bottom-up queries Q1′-Q6′. Finally, Q1″-Q6″, which restricted selectivities of the second triple pattern and favored the bottom-up approach, showed a consistent performance pattern only for PostgreSQL, where bottom-up queries were faster. For Oracle and DB2, some queries showed a similar pattern: top-down queries Q1″ and Q5″ were faster and bottom-up queries Q3″ and Q4″ were faster; in addition, both bottom-up and top-down Q6″ showed identical times in DB2, top-down Q6″ was faster in Oracle, bottom-up query Q2″ was significantly faster (the smallest ratio in our experiments) in Oracle but as fast as top-down query Q2″ in DB2.

Our *fourth observation* was that "symmetric" queries Q7 and Q8 (and similarly Q7′ and Q8′), which are neutral to both top-down and bottom-up translation strategies, showed better performance of the top-down queries. The ratios were significantly larger for DB2 and PostgreSQL, while only from 1.19 to 2.12 times larger in Oracle. These "symmetric" queries showed that, in a general (with no particular bias towards one or the other translation strategy) case, the top-down approach is superior to the bottom-up approach.

Our last, *fifth observation* was that a choice of a translation strategy could have a tremendous impact on a resulting query performance. In one case of Q2″ for Oracle, the bottom-up query was over 600 times faster than the top-down query. In 12 other cases (all occurred in experiments with DB2 and PostgreSQL), the ratios were greater than $1,000$ in the favor of top-down queries.

6.4 Summary

The performance study gives the answers to the two questions of this paper. For the first question, our results imply that, in a general case, a relational RDF database designer cannot rely on a relational query optimizer to produce identical or close to identical query execution plans for semantically equivalent SQL queries resulted from bottom-up and top-down translations of SPARQL queries. To answer the second question, neither of the two approaches is universally better than its sibling. The performance of queries produced by bottom-up and top-down translation strategies depends on many factors, including selectivities of triple patterns, their order and location in a SPARQL query, and even a relational engine that evaluates translated queries. A number of important observations are made that suggest directions for choosing the best translation strategy for a particular query by a SPARQL query optimizer; the choice can have a tremendous impact on query performance.

7. Conclusions and Future Work

In this paper, we studied the bottom-up and top-down SPARQL-to-SQL translation strategies and compared them empirically in the context of SPARQL queries with nested optional graph patterns. We proposed bottom-up and top-down nested graph pattern translation algorithms and compared their resulting SQL queries in Oracle, DB2, and PostgreSQL. Our performance study suggested that the choice between bottom-up and top-down translation algorithms can have dramatic performance implications on the resulting SQL queries. This choice depends on many factors, including selectivities of triple patterns, their order and location in a SPARQL query, and even a relational engine that evaluates translated queries. In the future, we will research a formal framework for optimizing SPARQL queries and defining heuristics for choosing a "good" translation strategy for a SPARQL query.

References

[1] W3C, "Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, 10 February 2004. G. Klyne, J. J. Carroll, and B. McBride (Eds.)," 2004.

[2] A. Chebotko and S. Lu, *Querying the Semantic Web: An Efficient Approach Using Relational Databases*. LAP Lambert Academic Publishing, 2009.

[3] A. Chebotko, S. Lu, and F. Fotouhi, "Semantics preserving SPARQL-to-SQL translation," *Data & Knowledge Engineering (DKE)*, vol. 68, no. 10, pp. 973–1000, 2009.

[4] A. Chebotko, S. Lu, H. M. Jamil, and F. Fotouhi, "Semantics preserving SPARQL-to-SQL query translation for optional graph patterns," Wayne State University, Tech. Rep. TR-DB-052006-CLJF, May 2006, available from http://www.cs.wayne.edu/~artem/main/research/TR-DB-052006-CLJF.pdf.

[5] J. Perez, M. Arenas, and C. Gutierrez, "Semantics and complexity of SPARQL," *ACM Transactions on Database Systems (TODS)*, vol. 34, no. 3, pp. 16:1–16:45, 2009.

[6] R. Cyganiak, "A relational algebra for SPARQL," Hewlett-Packard Laboratories, Tech. Rep. HPL-2005-170, 2005, available from http://www.hpl.hp.com/techreports/2005/HPL-2005-170.html.

[7] F. Zemke, "Converting SPARQL to SQL," Tech. Rep., October 2006, available from http://lists.w3.org/Archives/Public/public-rdf-dawg/2006OctDec/att-0058/sparql-to-sql.pdf.

[8] S. Harris and N. Shadbolt, "SPARQL query processing with conventional relational database systems," in *Proc. of SSWS*, 2005, pp. 235–244.

[9] M. F. Husain, L. Khan, M. Kantarcioglu, and B. M. Thuraisingham, "Data intensive query processing for large RDF graphs using cloud computing tools," in *Proc. of CLOUD*, 2010, pp. 1 – 10.

[10] C. Franke, S. Morin, A. Chebotko, J. Abraham, and P. Brazier, "Distributed Semantic Web data management in HBase and MySQL Cluster," in *Proc. of CLOUD*, 2011.

[11] G. Serfiotis, I. Koffina, V. Christophides, and V. Tannen, "Containment and minimization of RDF/S query patterns." in *Proc. of ISWC*, 2005, pp. 607–623.

[12] O. Hartig and R. Heese, "The SPARQL query graph model for query optimization," in *Proc. of ESWC*, 2007, pp. 564–578.

[13] A. Harth and S. Decker, "Optimized index structures for querying RDF from the Web," in *Proc. of LA-WEB*, 2005, pp. 71–80.

[14] O. Udrea, A. Pugliese, and V. S. Subrahmanian, "GRIN: A graph based RDF index," in *Proc. of AAAI*, 2007, pp. 1465–1470.

[15] C. Weiss, P. Karras, and A. Bernstein, "Hexastore: sextuple indexing for Semantic Web data management," *Proc. of PVLDB*, vol. 1, no. 1, pp. 1008–1019, 2008.

[16] G. H. L. Fletcher and P. W. Beck, "Scalable indexing of RDF graphs for efficient join processing," in *Proc. of CIKM*, 2009, pp. 1513–1516.

[17] W3C, "SPARQL Query Language for RDF. W3C Recommendation, 15 January 2008. E. Prud'hommeaux and A. Seaborne (Eds.)," 2008.

[18] A. Chebotko, S. Lu, X. Fei, and F. Fotouhi, "RDFProv: A relational RDF store for querying and managing scientific workflow provenance," *Data & Knowledge Engineering (DKE)*, vol. 69, no. 8, pp. 836–865, 2010.

Supporting Meaningful Data Mediation Using Semantic Mediator Description

Kanmani Munusamy[1], Suhaimi Ibrahim[2], Mohd Sapiyan Baba[1], Norbik Bashah Idris[2], Mohd Naz'ri Mahrin[2] and Hazlifah Mohd Rusli[2]

[1]Faculty of Computer Science & Information Technology, Universiti Malaya (UM), Kuala Lumpur, Malaysia

[2]Advanced Informatics School (AIS), Universiti Teknologi Malaysia (UTM), Kuala Lumpur, Malaysia

Abstract - *Data mediation plays an important role in ensuring successful interaction between a service requestor and a provider in Web services. Web service messages must be interpreted by the machines that reside at the service requester and provider in order to send and receive their messages correctly. The Semantic Web Service vouches for automation in discovery, selection and composition still faces great challenges in providing mediation actions in Web service interaction. This paper examines the existing data mediation approaches in Semantic Web Services and presents the lack of accuracy in the existing mapping techniques that support data mediation. The paper also proposes the use of Semantic Mediator Description (SMD) to provide specific semantic descriptions that support mediation actions. The SMD allows the machines to understand the content of the messages and mediate them correctly according to the ontology mapping provided.*

Keywords: Semantic Web Service; Data Mediation; Process Mediation; Message Mismatches; Semantic Knowledge Representations

1 Introduction

At present many approaches use ontology as an important element to explicitly describe Web service. Ontology has helped the development of the Semantic Web Service (SWS) frameworks such as the Web Service Modelling Ontology (WSMO) [1], OWL-S [2], and the SAWSDL [3] which has been accepted by W3C. These SWS frameworks have enhanced automation in Web service discovery, selection, composition and testing by providing semantic descriptions to Web services. Approaches that have adopted these frameworks have shown proven results on how semantic descriptions can be useful in discovering, selecting and composing Web services automatically at runtime.

However, the existing semantic descriptions are unable to assure that these matched services can interact correctly during the actual invocation phase. Incompatibility in the selected Web services can lead to the termination of the Web service composition and invocation if they are not detected and corrected before the actual execution.

Data mediation handles most common type of mismatches that occur due to the usage of different terminology, format and data representation in messages that are sent or received by different Web services [4]. M. Nagarajan has classified the data level heterogeneity into three levels of incompatibility, namely the attribute, the entity, and the abstraction level of incompatibility [5]. This paper focuses on the incompatibility of the entity level that addresses semantically similar entities that may have a number of different attributes which are also known as the scheme isomorphism conflicts [5].

Data mediation is an important element in ensuring a successful interaction between Web services as well as identifying behavioural incompatibility and choreography mismatches. There are several research works [6-8] in behavioural incompatibilities that have classified the process level mismatches into five basic mismatches, namely extra messages, missing messages, one to many messages, many to many messages, and wrong order messages. In order to solve the identified mismatches, five types of mediation actions have been proposed such as stop, merge, split, generate, and reorder [7, 9]. The original messages are split, merged, or reordered according to the required communication pattern. New messages may even be generated from the original messages in order to conform to the required communication pattern.

Systematic literature review on process mediation in previous work conducted [10] shows that data level mapping and data mediation in Web services interaction needs further research work to ensure a successful invocation. It has also been highlighted that knowledge representations of the messages need to be defined clearly to transform the original messages into target messages by using mediation actions such as merging, splitting and generating new messages [10].

The existing works in data mediation [11, 12] only focus on utilizing the domain ontology in generating the mapping between the different ontologies used by service requestors and providers, and show little interest in them utilizing semantic descriptions to expose the knowledge required for data mediation between messages.

In this paper, the existing role of semantic descriptions in Web services that could be useful in increasing the level of automation during service configuration by the client is analysed. A new framework using Semantic Mediator Descriptions (SMD) to increase correctness in the existing

data mediation effort to support process mediation at runtime is proposed. It is believed that this research would be very useful in enhancing data mediation at runtime for the existing SWS initiatives.

The WSMO Semantic Web Service Framework is selected to realize this approach. The WSMO Framework [13] provides rich descriptions on all the related aspects of Web services through four important components which are the goal, the Web service, the ontology, and the mediator. WSMO uses Web Service Modelling Language (WSML) to describe all the components and also provides a modelling tool called Web Service Modelling Toolkit (WSMT) to assist the developer in developing the service descriptions.

The later part of this paper is organized as follows:- Section II illustrates a motivating scenario. Section III describes the related work in data mediation in Web services and provides an overview of the state-of-art research contribution in data mediation to support process mediation. Section IV describes the limitation of the related works and the proposed approach. Section V describes how Semantic Mediator Descriptions are able to provide data mediation knowledge to the illustrated scenario. Finally, Section VI presents the conclusions.

2 Motivating Scenario

A simple scenario is presented here to illustrate the need for data interoperability for Web service interaction. In examining the process of Bibliographic Scholarly Database (BSD) organizations such as SciVerse Scopus and ISI Thomson's interaction with Higher Learning Institutions (HLI), it has been found that in order to retrieve the publication records of an individual academician via Web services, it is necessary that publication records are found existing in the HLI. By assigning publication records to each individual academician the calculation and retrieval of citation counts and HIndex data from the BSD is allowed. Therefore, it is essential for the BSD and HLI to interact correctly.

The process begins when BSD organizations publish their Web services that allow HLI to send their publication data and retrieve information on the citation counts. The BSD organizations will be addressed as the service provider hereafter in this paper. Most public universities have their own publication systems developed in-house or bought over the shelf. In this paper, such systems addressed as University Publication System (UPS).

For the above scenario the Web service provided by ISI Thomson Reuters was studied as the BSD service provider and a local public university was chosen as a client that accesses the service provided by ISI Thomson Reuters.

2.1 Service description provided by the provider

In taking into account of the service description that is given by service providers, the WSDL file describes the requested format of the publication data in terms of the reference-type, the contributors, the titles, the pages, the volume, and other details. Due constraints in space, we only mediation is discussed as an effort in restructuring the UPS name of authors according to the needs of the service provider. The listing below shows the XML data scheme that describes the contributor's or author's requirement format.

```
<xs:complexType name="authorType" >
  <xs:attribute name="title-name" >
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="20"/>
    </xs:restriction>
</xs:attribute>
  <xs:attribute name="first-name" >
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="20"/>
    </xs:restriction>
</xs:attribute>
   <xs:attribute name="last-name">
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="35"/>
    </xs:restriction>
</xs:attribute>
  <xs:attribute name="middle-name" >
    <xs:restriction base="xs:string">
      <xs:minLength value="0"/>
      <xs:maxLength value="20"/>
    </xs:restriction>
</xs:attribute>
</xs:complexType>
```

It is clear that the service requestors need to provide the names of authors in three fields namely; the first-name, the last-name, and the middle-name. The service requestor also needs to ensure that the size of each field does not exceed the maximum length specified in the XML scheme.

2.2 Requestor's data that need mediation

The publication data resides in the database of the UPS system, which is the service requester. All the names of authors for each publication records are stored in a field that is named as 'authors'. Table 1 provides an example the names of authors for a publication record that is stored in the UPS database.

Table 1: An Example of a Publication Record Stored in a UPS Database

Fields	***Instance***
ID	101
Title	Semantic Description that Supports Data Mediation in WSMO
Authors	K. Munusamy, C.S. Cheong, S. Ibrahim, S. Nallusamy, M.N. Mahrin and H. Mohd Rusli

The developers who are involved in the task of implementing the Web Service in the university need to understand the required data and the format that is specified in the WSDL file provided by the service provider. Then, they conduct detail analysis of their internal publication system's data structure and format in order to create middleware

applications that modify the data scheme and structure in the UPS.

This middleware applications need to be developed according to the existing UPS system in each university and service description given by the providers. This process requires lots of guidance from the service providers to ensure that each of the university's development team understands the required data structure and format. The team also needs to provide additional files and descriptions to the clients, apart from the Web service WSDL files. On the other hand, the developer that resides at HLI needs to have knowledge of the author's name structures and how it can be mediated into the required structure.

The role of the developer is very limited during the Web service composition and invocation at runtime and the role of the automatic data mediator is very important in supporting automation in service discovery, selection, process mediation and composition. Therefore, it has been found that the mediation knowledge of the developer's needs to be interpreted semantically to ensure correctness and automation in the Web service data mediation.

3 Related Works

Related work on data mediation was analysed in the existing SWS Frameworks namely WSMO, OWL-S and SAWSDL. Early data mediation effort in Web services is found in [14] which introduces the mapping rules between RDF scheme in Triple Space Computing. This effort is extended by WSMO data mediation initiatives which focus on ontology alignment [12, 15, 16].

The WSMO initiatives generate mapping rules in the form of axioms based on the abstract mappings identified by the developers at the time of design. It demonstrates that source instances can be transformed into target instances via posting query and retrieving answers from the mapping rules at the time of execution. However, the data mediation effort is found to focus only on the generating alignment between attributes that are placed at different levels within the ontology. It describes implicitly the data mediation effort that involves splitting and merging messages during process mediation.

Secondly, in OWL-S, data mediation that supports the process mediation is not explained in detail and is only mentioned as an external service [17]. However, the researchers have concluded on the need of better support for data mediation in order to allow real life Web service mediation [18]. Finally, data mediation efforts in SAWSDL introduces the use of context-based data type ranking algorithm to generate scheme mapping between Web service messages [19]. However further discussion on data mediation to support process mediation actions is not provided by the researchers [20].

Apart from SWS Frameworks, it also necessary to create mapping between target and source attributes in their mediation effort [21]. There are various techniques [22] and tools that support message mapping at design time which are implemented during Web service invocation at run time [11].

These approaches focus on creating the mapping automatically; and only provide limited discussion on mediating the actual instances. On the other hand, the approach introduced in this paper, focuses on executing the provided mapping by understanding the content of the attributes. Figure 1 shows an example of ontology mapping for one-to-many message mismatches that can be generated by the existing approaches. In Figure 1, the Authors and Contributors have similar ontologies that need to be mapped but they contain different attributes. The Authors' ontology contains only one attribute which is the Full Name, while the Contributors' ontology contains three attributes which are the First Name, the Middle Name and the Last Name.

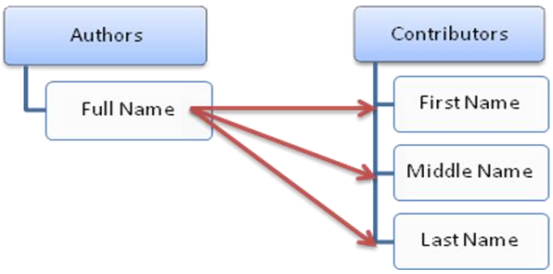


Figure 1. Ontology Mapping that support Data Mediation

4 Proposed Approach

It can be seen that ontology mapping alone is insufficient to mediate the data messages correctly at run time as it only provides the connection between the sourced messages to the targeted messages that have different data representations. However, it does not provide any knowledge to the mediator about the content of the sourced message and how it can be split or merged correctly to targeted messages as specified in the message mappings. As shown in Figure 1, the existing approaches can provide precise matches between the Full Name from the Author's ontology to the First Name, the Middle Name and the Last Name in the Contributors ontology.

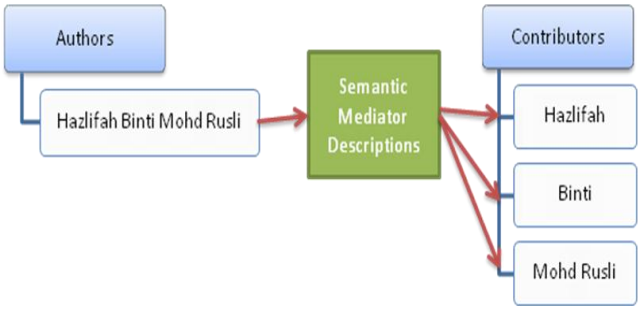


Figure 2. Semantic Mediator Description that supports data mediation

However, as to how the content of the Full Name is to be mapped remains unclear. Moreover, the developer's task does not stop at the mapping level but needs to be extended to the actual mediation task at run time. The research objective for

this paper is to provide an understanding of the content of the Full Name attribute in order to correctly map its content to the attributes of the First Name, the Middle Name, and the Last Name. For example, in a given full name "Hazlifah binti Mohd Rusli", the mediator would have to know how to split the full name into the first name, the middle name, and the last name as illustrated in Figure 2.

Within the scope of Web service, the existing approaches still require the developer's presence at run time to understand the content of the messages and execute the mediation actions such as splitting or merging the messages correctly onto the map. The approach used in this paper, adds a Semantic Mediator Descriptions component to the existing ontology mapping so as to mediate the data correctly according to the developer's mediation knowledge. Figure 2 illustrates the role of Semantic Mediator Descriptions in mediating the full name "Hazlifah Binti Mohd Rusli" correctly into the provided mapping attributes.

4.1 Semantic Mediator Description

The Semantic Mediator Descriptions (SMD) component contains the semantic descriptions of the message content, related rules and execution steps. Although a simple example that manipulates the author's name to illustrate the data mediation problem in this application but however, it is a common problem in various scenarios that involve content such as address, location, price, rate and others underlying factors. Therefore, further research on data mediation in Web services is needed to ensure successful interaction during service invocation. Figure 3 illustrates an overview of the approach described in this paper.

An additional component is proposed to the existing data mediation framework in WSMO (refer to Figure 3). The mediator can be described as another SWS that is to be discovered and selected during service invocation. The mediator functionalities are realized by using the service capability and interface. Both the service capability and interface are defined by basing them on the SMD that are created at design time with the developer's assistance.

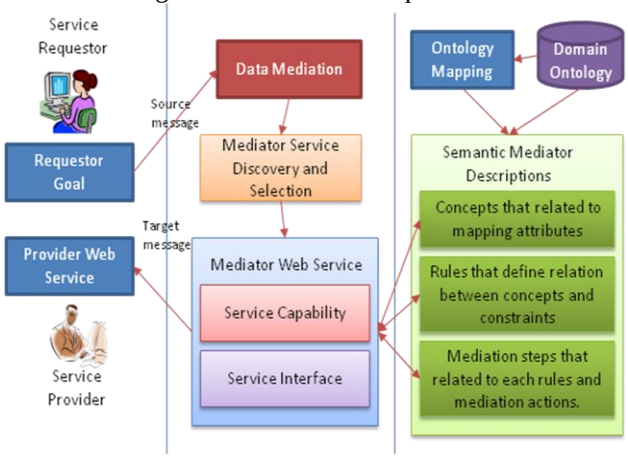


Figure 3. Overview of Our Approach

SMD semantically present the developer's mediation knowledge in the form of concepts, rules and the steps that are required for mediation actions. It uses existing ontology mapping techniques and domain ontology. The following section describes the data collection activities that interpret the developer's mediation knowledge into related concepts, relationship, and rules.

4.2 Identifying related concepts

It is required that a reference is made to the author's table that contains the full name and other details of each author, in order to provide the authors' name in the format that is requested by the service provider. Each author's short name is retrieved by splitting the data at every "," symbol. Then, the author's short name is used to map onto the author's table to retrieve the full name and other details as shown in Table 2.

Table 2 : Result of Retrieving the Full Name, Gender and Race using the Author's Short Name

Short Name	*Full Name*	*Gender*	*Race*
K. Munusamy	Kanmani A/P Munusamy	Female	Indian
C. S. Cheong	Chong Suh Cheong	Male	Chinese
S. Ibrahim	Suhaimi Bin Ibrahim	Male	Malay
S. Nallusamy	Sugumaran A/L Nallusamy	Male	Indian
M. N. Mahrin	Mohd. Nazri Bin Mahrin	Male	Malay
H. Mohd Rusli	Hazlifah Binti Mohd Rusli	Female	Malay

4.3 Identifying relationship and concepts

The author's full name has been identified in relation to the other fields such as gender and race. Below are the rules that are helpful in understanding the formation of the author's name.

4.3.1 Full name and detail description

The formation of full name of an author can be defined in many ways as preferred by the system developer or it could be extracted from existing data resources. The full name of an author can consist of the title name, the first name, the middle name, and the last name. These detailed descriptions of an author's name can be presented in different sequences and combination, depending on the country, region, race, and gender.

a) Title name: A word or abbreviation that is used before a person's name to show the person's profession or a social status. In this paper only two category of titles have been described namely; formal social titles and academic titles. Example of formal social titles are Mr, Ms, Miss and Mrs and academic titles are restricted to Dr, Associate Professor, Associate Professor Dr, Professor Dr and Professor.
b) First Name: First name refers to author's individual name.

c*) Last Name:* Last name is also referred to as surname or family name. Generally, the last name refers to the family name that is passed down from generation to generation such as within the Chinese and European communities. In some races or ethnics groups, the last name refers to the author's 'father's first name.
d) Middle Name: Generally, middle name is not defined as clearly as the first name and the last name. However, in some races, middle name can be useful to in indicating the gender of the author.

4.3.2 Name descriptions and the rules applied

It has been explained earlier of the rules that are applied in order to mediate author information among Web services. These rules are important to proceed with mediation actions such as splitting, merging and generating new data during the Web service interaction.

a) Title Name: These are defined as the formal social title for each author by assigning Mr as the title for all the male authors and, Ms for female authors who do not provide information on their marital status. Female authors with married marital status are assigned the title Mrs. Single female authors are assigned the title of Miss.
b) First Name: With reference to Table 2, the author's full name, gender and ethnic group or race is retrieved by using the author's short name. The short name consists of the first letter of the author's name and the family name (or in some cases, the father's name). In general, the first name is defined according to the following rules:

Rule 1: Some ethnic groups or races, such as Malays and Indians, use a separator to separate the first name from the last name. Words such as "Bin", "Binti", "A/L" or "A/P" which means "son of" or "daughter of" are often used as the separators. In this rule all elements that come before the separator is the first name. Below is an example:

Short Name	Full Name	First Name
K. Munusamy	Kanmani A/P Munusamy	Kanmani
H. Mohd Rusli	Hazlifah Binti Mohd Rusli	Hazlifah

Rule 2: In some situations, the full name of authors from the Malay and Indian ethnic groups also appear without any separators as explained in Rule 1. The names appear as individual names with their fathers' names. Thus, for names that only contain first names and the last names, and that do not match their fathers' names are defined as the authors' first names.

Rule 3: For authors of the Chinese ethnic group, both rules given above are not applicable since their last names or family names are placed as the first word in their names. Therefore, it is defined here that any words after the first word are regarded as their first name. Below is an example of the first name for authors from the Chinese ethnic group:

Short Name	Full Name	First Name
C. S. Cheong	Chong Suh Cheong	Suh Cheong

c) Middle Name: Similar to the rules defined for the first name, for ethnic groups such as the Malays and Indians, the middle names are the separators used in the full name, i.e. "Bin", "Binti", "A/L" or "A/P". However, no rules are found to define the middle name as in the Chinese authors' names.
d) Last Name: Last names are also known as surnames or family names. The last name is defined according to the following rules:

Rule 1: For the Malays and Indians ethnic group, the last names are all the elements that come after the separators Bin", "Binti", "A/L" or "A/P".

Rule 2: For names without separators, the element that matches the father's or family name is the author's last name. However, for authors of Chinese origin, the first word in the full name is regarded as the last name.

5 Semantic descriptions that supports the Data mediation

There are three main components of the Semantic Mediator Descriptions; concepts, rules and the mediation steps. The advantage of declarative knowledge representation is exploited in order to describe the content of the Web service messages. The main objective in this context is to translate the mediation knowledge of a developer into semantic descriptions which can be read and processed by machines. Therefore, the semantic mediation knowledge is expressed in WSML ontology language using Web Service Modelling Toolkit (WSMT).

5.1 Domain Ontologies

In this approach, two simple ontologies were created and these are the requester ontology and the provider ontology. The requestor ontology is built based on the publication records that are stored in the UPS system and named termed as the PublicationOntology. The structure of the Publication Ontology is as follows:

```
ontology PublicationOntology

concept Gender
concept Religion
concept MaritalStatus

concept Authors
hasName ofType _string
NameContent impliesType NameDescription
hasGender impliesType Gender
```

```
hasEthnic impliesType Ethnic
hasShortName ofType _string
hasNationality impliesType Nationality
hasMaritalStatus impliesType MaritalStatus

instance Married memberOf MaritalStatus
instance Single memberOf MaritalStatus
instance Divorced memberOf MaritalStatus

instance Malay memberOf Ethnic
instance Chinese memberOf Ethnic
instance Indian memberOf Ethnic
instance Male memberOf Gender
instance Female memberOf Gender
instance Malaysian memberOf Nationality
```

The requestor ontology describes all the important elements of an author. In addition, the content of the author's name as a separate concept is also described and listed below.

```
concept TitleNames

concept NameDecription subConceptOf Authors
hasTitleName impliesType TitleNames
hasFirstName ofType _string
hasMiddleName ofType _string
hasLastName ofType _string

instance Ms memberOf TitleNames
instance Miss memberOf TitleNames
instance Mrs memberOf TitleNames
instance Mr memberOf TitleNames
```

The instances of each authors is also extracted from the existing data from the UPS system. An example of the generated instances as follows.

```
instance Kanmani memberOf Authors
hasName hasValue "Kanmani A/P Munusamy"
hasGender hasValue Female
hasEthnic hasValue Indian
hasShortName hasValue "K. Munusamy"
hasNationality hasValue Malaysian
hasMaritalStatus hasValue Married
```

The provider ontology is also defined by extracting the important *complexType* and the related attribute from the XML data scheme as specified in the WSDL file and is termed as IndexedPublication. Figure 4 illustrates the concept of Contributor in the IndexedPublication.

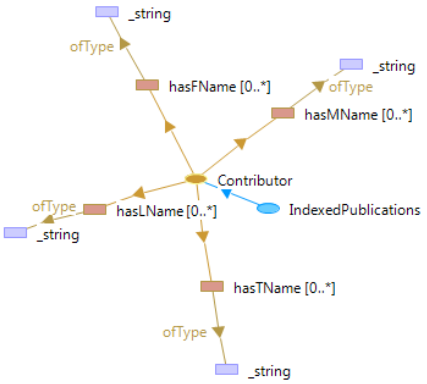


Fig. 4 Concept of Contributor in IndexedPublication

5.2 Axioms that define the rules

Rules and relation that describe the content of name are translated into axioms and the listing below shows the excerpt of axioms that describe the relationship between the middle name with the other concepts like Gender and Race. In this rule, the relationship of the concept of Nationality, have been included since all these rules are true among Malaysians.

```
axiom FemaleMalayDef
definedBy ?a memberOf FemaleMalay :- ?a memberOf
Authors and
?a[hasGender hasValue Female] and
?a[hasEthnic hasValue Malay].

axiom FemaleMalayMiddleDef definedBy
?x memberOf FemaleMalay and ?x[hasNationality
hasValue Malaysian]
implies ?x[hasMiddleName hasValue "Binti"].

axiom MaleIndianMiddleDef definedBy
?x memberOf Authors and ?x[hasGender hasValue
Male] and
?x[hasEthnic hasValue Indian] and
?x[hasNationality hasValue Malaysian]
implies ?x[hasMiddleName hasValue "A/L"].
```

5.3 Testing the Semantic Mediation Descriptions and mediation steps

The provider ontology and axioms are tested to ensure that the all the developer knowledge that required for mediating the Web service messages is represented correctly into semantic descriptions. The IRIS Reasoner is used to execute the queries and to generate the middle name and the title for each author based on the provided descriptions and rules. Figure 5 shows a simple query that retrieves the middle name of the described authors. These queries are called in termed as transition rules that describe the steps involved in the mediation action. The transition rules are not explained further due to the space limitations.

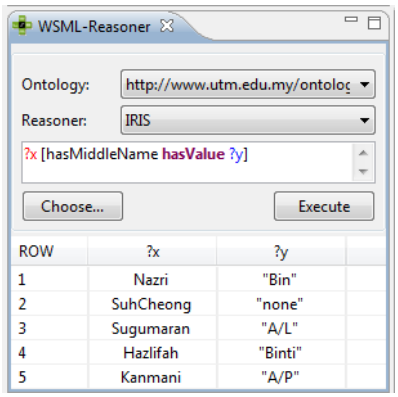


Figure 5 Executing Queries using IRIS Reasoner

6 Conclusions

In this paper, the importance of understanding the content of the messages in order to mediate the messages correctly according to the provided mapping has been discussed. A new component which is termed as the Semantic Mediator Description that expresses developers' mediation knowledge in the form of semantic knowledge representations has been proposed. Moreover, it has been illustrated in the proposed framework and further explained that each element in the Semantic Mediator Description component uses a real life scenario. In this research, generating the rules that define content of messages tends to be very challenging. The rules and relations between the concepts require all the possible descriptions of a message from various perspectives. The rules and mediation steps that are created at design time with the developer's assistance must be tested using large data sets before the actual Web service invocation to ensure its accuracy.

7 References

[1] Bruijn, J.d., et al. *Web Service Modeling Ontology (WSMO)*. 2005 3 June 2005 [cited 28 June 2011]; Available from: http://www.w3.org/Submission/WSMO/.

[2] Martin, D., et al. *OWL-S: Semantic Markup for Web Services* 2004 22 Nov 2004 [cited 28 June 2011]; Available from: http://www.w3.org/Submission/OWL-S/.

[3] Farrell, J. and H. Lausen, *Semantic Annotations for WSDL and XML Schema (SAWSDL)*, in *W3C Recommendation*. 2007.

[4] Fensel, D., M. Kerrigan, and M. Zaremba, *Mediation*, in *Implementing Semantic Web Services*. 2008. p. 211-231.

[5] Nagarajan, M., et al., *Ontology driven data mediation in Web services.* International Journal of Web Services Research, 2007. **4**(4): p. 104-126.

[6] Fensel, D. and C. Bussler, *The web service modeling framework WSMF*. Electronic Commerce Research and Applications, 2002. **1**(2): p. 113-137.

[7] Motahari Nezhad, H.R., et al. *Semi-automated adaptation of service interactions*. in *16th International World Wide Web Conference, WWW2007*. 2007. Banff, AB.

[8] Wei, S., et al. *Toward a model-based approach to dynamic adaptation of composite services*. in *Proceedings of the IEEE International Conference on Web Services, ICWS 2008*. 2008. Beijing.

[9] Li, X., et al., *A pattern-based approach to protocol mediation for web services composition.* Information and Software Technology, 2009. **52**(3): p. 304-323.

[10] Munusamy, K., et al., *A Comparative Evaluation of State-of-Art Web Services to Support Protocol Mismatches.* International Journal on Web Service Computing (IJWSC), 2011. **2**(3).

[11] Euzenat, J., A. Mocan, and F. Scharffe, *Ontology Alignments*, in *Ontology Management*. 2008. p. 177-206.

[12] Mocan, A., E. Cimpian, and M. Kerrigan, *Applying Reasoning to Instance Transformation*, in *5th European Semantic Web Conference (ESWC)*. 2009: Tenerife Spain.

[13] Fensel, D., M. Kerrigan, and M. Zaremba, *WSMO and WSML*, in *Implementing Semantic Web Services*, D. Fensel, M. Kerrigan, and M. Zaremba, Editors. 2008. p. 43-65.

[14] Shafiq, O., et al. *Data Mediation Support for Triple Space Computing*. in *2006 International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom*. 2006. Atlanta, GA, United states: Inst. of Elec. and Elec. Eng. Computer Society.

[15] Mocan, A. and E. Cimpian, *An ontology-based data mediation framework for semantic environments.* Semantic Web and Information Systems, 2007. **3**(2): p. 69-98.

[16] Mocan, A. and E. Cimpian. *WSMX Data Mediation*. 2005 11 Oct 2005 [cited 28 June 2011]; Available from: http://www.wsmo.org/TR/d13/d13.3/v0.2/20051011/.

[17] Vaculin, R., R. Neruda, and K. Sycara, *The process mediation framework for semantic Web services.* International Journal of Agent-Oriented Software Engineering, 2009. **3**(1): p. 27-58.

[18] Vaculin, R. and K. Sycara. *Towards automatic mediation of OWL-S process models*. in *Proceedings - 2007 IEEE International Conference on Web Services, ICWS 2007*. 2007. Salt Lake City, UT.

[19] Wu, Z.X., et al., *Automatic composition of semantic web services using process mediation*, J. Cardoso, J. Cordeiro, and J. Filipe, Editors. 2007, LSDIS lab, University of Georgia. p. 453-461.

[20] Gomadam, K., et al., *A Declarative Approach using SAWSDL and Semantic Templates Towards Process Mediation*, in *Semantic Web Services Challenge*. 2009. p. 101-118.

[21] Zhe, S., A. Kumar, and P. Grefen. *Towards Integrated Service Adaptation A New Approach Combining Message and Control Flow Adaptation*. in *Web Services (ICWS), 2010 IEEE International Conference on*. 2010.

[22] Euzenat, J.r.m. and P. Shvaiko, *Classifications of ontology matching techniques Ontology Matching*. 2007, Springer Berlin Heidelberg. p. 61-72.

PERFORMANCE EVALUATION OF A RELATION BASED PAGE RANKING ALGORITHM USED FOR IMPROVING SEARCH RESULTS

Sumera Hayat [1] MS CS, Nadeem Qazi [2] Associate Professor

[1 2] Faculty of Engineering, Sciences & Technology. IQRA University, Karachi, Pakistan.

***Abstract*— Search engines help us mine the web and get what we are looking for. But the problem with today's search engines is that, it gives us most reputable results but not really the most relevant ones. The search criteria of these traditional web search engines is based on keyword matching or to some higher extent (like in Google), matching words similar or in the same context to the query word and then ranking the matched pages with PageRank Algorithm, based purely on link analysis. PageRank despite being quiet efficient in sorting the results to be presented to the user, still results in a number of irrelevant pages. To overcome this problem and provide relevant results, the Relation Based Page Ranking Algorithm has been introduced recently. In this paper, we evaluate the performance of the Relation Based Page Ranking Algorithm on some academic web pages annotated with an Ontology specifically built for the web page's domain. Ranking of web pages through this algorithm is based upon a probability measure that checks the relevancy of a page to the query. Finally, we compare the results of our experiments with that of Google's results generated for the same query.**

Keywords: **Semantic web, PageRank, Ontology, Relevance Ranking, Annotations.**

1. INTRODUCTION

Search Engines are developed to help the users search for what they are looking for in the web's data repository. Search Engines work with the help of predefined automated software programs that are known as Spiders or bots and are used for crawling, that is, going through the entire website (all the pages) and records down the content in the form of databases known as indexes. These search engine indexes are invoked to display the webpage content, when the user defines a query in the search engine interface and the query is parsed and checked against the search engine database to get the best results possible [1]. The matching criteria of the existing search engines is usually based on keyword matching, that is, matching the query keywords with the web page's keywords stored in the indexes of the search engines. There is no architecture on the web 2.0 [14] to store the relations a user has in mind while typing the query [2]. At the time of query processing, the relations are lost and the given keywords are treated as individual keywords, hence creating the major problem of isolated keyword matching. Even the ranking of the retrieved web pages has no account for relations and is purely based on link analysis like in Google's PageRank [3][4] and some on page relevance factors [5]. There should be some way to measure these relations and to make keywords search more meaningful that is, incorporating Semantics into the search engines working. Though Google has injected some of the very important semantic web technologies [6] such as latent semantic indexing [15] and displaying rich snippets [16], but it still processes and displays some irrelevant results on the very first page.

In this paper, we prove that the Relation Based Page Ranking Algorithm proposed in [8] is a solution to the above mentioned inconsistencies of the existing search engines. The Ranking strategy is based on the ontology data, user query and the page annotations, which will be exploited to measure the relevance of a web page to a given query. The Ranking criterion proposed specifically for the Semantic Search Engines, in no ways eliminates the use of the existing ranking algorithms like PageRank, which checks for the repute of pages. In fact, this Relation Based Ranking Algorithm can be used in conjunction with the PageRank algorithm to give most reputable and most relevant results on a semantic search engine.

2. RELATED WORKS IN RANKING ALGORITHMS FOR THE SEMANTIC WEB

The aim of this paper is to make use of the relations embedded as annotations within a web page and the query concepts for creating a ranking strategy capable of assigning a ranking score better than the ones done in today's search engine's ranking algorithms. This idea of making use of the ontology based semantic meta data for ranking web pages is not new [9] [10] [11]. How ever, these previous approaches did not consider the semantic relations which are said to be the key component of the Semantic Web. To make optimum use of the Semantic Web content marked up, there should be approaches available that takes into account the relations and the semantic web associations between content in annotated web pages that can be used for ranking and retrieving data [2][12][13].

The most relevant work with respect to this paper would be [8] and [2]. The basic idea of [2] is that if a graph based web page annotation can be provided, where concepts and relations are modeled as vertices and weighted edges, respectively, it becomes possible to define a series of cuts removing less relevant concepts from the graph. This allows for the generation of a so

called candidate relation –keyword set (CRKS) to be submitted to the annotated database, which can reduce the presence of uninteresting pages in the result set. How ever, the effectiveness of this approach is strongly limited as there is no kind of ranking strategy involved. In [8] we see an extension to this approach which relies on the assumption that for providing effective ranking, the search engine logic should only need to know the structure of the underlying ontology and the web page to be ranked in order to compute the relevance score.

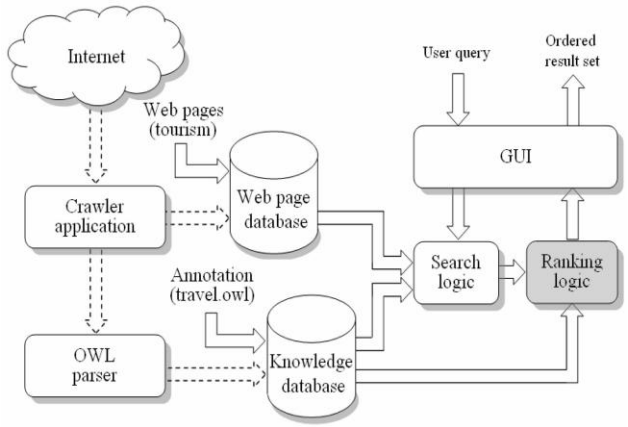
3. Overview of the Ranking Strategy

For the Semantic web to work optimally and give the results actually anticipated by the users, we need to not only add in the so called semantic web plug ins, but to change the entire infrastructure of the current web search engines. The Relation Based Page Ranking Algorithm for the Semantic Search Engine is proposed to solve the issues related to the ranking techniques of today's web 2.0, like PageRank used by Google. To prove that this Algorithm works better than the existing search engines ranking systems we have created a data set to reproduce the ranking results generated by Relation Based Page Ranking Algorithm in [8]. We first present the real time environment in which this algorithm will be applicable in future.

3.1 Infrastructure of a Semantic Search Engine

The Relation Based Page Ranking Algorithm cannot be used along with the current web search engines infrastructure as the current search engines lack to store the relations between query keywords and the annotated web documents which are the basis of Relation Based Page Ranking Algorithm. In order to understand the ranking algorithm itself, we first need to understand the environment where it will actually function in real time scenarios. The prototype of the Relation Based Page Ranking Algorithm presented in [8] is shown in figure 3.1.
In this paper, we have not focused on how to create this tiny Semantic web environment, but to evaluate the performance of the Ranking Logic of this Semantic Web, hence limiting our focus to the Ranking Module of the figure below.

Figure 3.1 Prototype Architecture for the Relation Based Page Ranking Algorithm



3.1 Scenario for the Evaluation of the Algorithm

When searched through Google (on 12 January 2012, at 3:44 pm) about Dr Alam Raza's Research work who is working at IQRA University with the following keywords, "Dr Alam Raza" "Research" "IQRA University", the search results displayed the web page (Faculty Web page of IQRA university) on top of the list and gives the profile page on 7th position in the list, which actually had the research details of Dr Alam Raza. The third page which is also less relevant than the profile page but is much closer to the query than the faculty web page is ranked 3rd and the least relevant page that contains nothing about Dr Alam Raza's Research work is ranked 2nd by Google. The keywords were merely matched on the basis of keywords density and latent semantic indexing, with no importance given to the relations in between the keywords which a user had in mind. For instance in this case, the relations in our mind were Dr Alam Raza *has Research* some Research, who is *working* at IQRA University. The ranking of these web pages was based of the link analysis PageRank Algorithm which gave no importance to the relations. The faculty web page having a PageRank 4/10 [18], was ranked first as the other pages had PageRank less than this.

3.2 Creating OWL ontology

Ontology is a way of representing things of the world, called entities, into a concept based method which defines a kind of taxonomy or hierarchy. It gives a common vocabulary for the people and machines to share knowledge of a particular domain [19]. So we created an ontology using Protégé 4.1[20][21] for manually annotating some web pages of IQRA University. The ontology shown by a graphical option (Ontoviz) in Protégé 4.1 can be seen in figure 3.2.

Figure 3.2 IQRA University Ontology Onto viz view in Protégé 4.1

The above graphical representation of the ontology can be written in an ontology language [22] understood by machines, which will be the language for annotating the web pages as well. We have extracted the RDF/XML code of this ontology written in a web ontology language [23] from Protégé 4.1 after its creation and verification by a reasoner. A portion of the RDF/XML code of the IQRA University ontology can be seen in figure 3.

3.3 Manual Annotation of web pages

The task performed to make web page data available and readable to the machines is known as Annotations or Semantic Markup. The term annotate simply means to attach data to some other piece of data [24]. Semantic annotations can be given to a web document in many ways. The traditional way is with the help of the tools like OntoMat annotizer[25], Annotea [26], SMORE [27] etc. Other ways are through Semantic Wikis, Semantic blogs, tagging with the help of RDFa , microformats and embedding RDF [7] meta data with the help of ontology vocabulary. With the help of our IQRA University Ontology we manually annotated 4 web pages of IQRA University and the RDF/ XML code generated for one of these web pages using Protégé Ontology RDF/XML code is shown in figure 3.4.

```
<!-- http://www.iqra-ontology.owl#Teaching_Staff -->

<owl:Class rdf:about="#Teaching_Staff">
  <rdfs:subClassOf rdf:resource="#Staff"/>
    <rdfs:subClassOf>
      <owl:Restriction>
  <owl:onProperty rdf:resource="#Graduated_From"/>
  <owl:someValuesFrom rdf:resource="#Universities"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
  <owl:onProperty rdf:resource="#HasResearchArea"/>
  <owl:someValuesFrom rdf:resource="#Research"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
  <owl:onProperty rdf:resource="#HasPublication"/>
  <owl:someValuesFrom rdf:resource="#Publications"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
  <owl:onProperty rdf:resource="#IsAdvisorOf"/>
  <owl:someValuesFrom rdf:resource="#Students"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
  <owl:onProperty rdf:resource="#Is_working_at"/>
  <owl:someValuesFrom rdf:resource="#Universities"/>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
```

Figure 3.3 Part of the IQRA University Ontology in RDF/XML Syntax

```
WEB PAGE 1 "Faculty" (http://www.iqra.edu.pk/?page_id=1210)

<!-- http://www.iqra-ontology.owl#Dr_Alam_Raza -->

<owl:NamedIndividual rdf:about="#Dr_Alam_Raza">
  <rdf:type rdf:resource="#Assistant_ProfStaff"/>
    <rdf:type rdf:resource="#Teaching_Staff"/>
  <#Is_working_at rdf:resource="#IQRA_University"/>
  <#Master:s_From rdf:resource="#Karachi_University"/>
      </owl:NamedIndividual>
<!-- http://www.iqra-ontology.owl#IQRA_University -->

<owl:NamedIndividual rdf:about="#IQRA_University">
    <rdf:type rdf:resource="#Universities"/>
  <#HasResearchCenter rdf:resource="#IURC"/>
      </owl:NamedIndividual>
```

Figure 3.4 Part of annotations of web page 1

3.4 Graph Based Notation and Methodology

Starting from the ontology defined for a domain, a graph based representation can be designed where OWL classes are mapped into graph vertices and OWL relation properties are mapped into graph edges. A link between concepts in the graph shows a relation between the concepts and has some weight on the basis of the given relations. Like wise, we formulate the query and annotated web pages into their respective graphs using the graph based notations given below in table 3.1. Given an ontology graph G and a query sub graph GQ, it is possible to define a ranking strategy capable of assigning each page including queried concepts a relevance score based on the semantic relations available among concepts within the page itself (thus neglecting the contribution of the remaining Web pages). The proposed ranking strategy assumes that given a query Q, for each page p, it is possible to build a page sub graph GQ,p using a methodology that is similar to the one used for G and GQ and exploiting the information available in page annotation A. All these graphs will help us implement the Relation Based Page Ranking Algorithm.

Symbol	Definition
$G(C,R)$	Ontology graph
$C=\{c_1,c_2,\ldots,c_n\}$	Set of concepts constituting the vertices of the ontology graph
$R=\{R_{ij} \mid i=1,\ldots,n,\ j=1,\ldots,n,\ j>i\}$	Set of relations constituting the edges of the ontology graph
$R_{ij}=\{r_{ij}^1,r_{ij}^2,\ldots,r_{ij}^m,\ m<n\}$	Set of relations between concepts c_i and c_j in $G(C,R)$
$Q=\{(k_i,c_i)\}$	Query as a collection of pairs (keyword, concept)
$G_Q(C_Q,R_Q)$	Query subgraph for query Q
$C_Q=\{c_i \mid (k_i,c_i)\in Q\}$	Set of concepts constituting the query subgraph
$R_Q=\{\bar{R}_{ij} \mid 1\le i\le n,\ 1\le j\le n,\ j>i\}$	Set of relations constituting the query subgraph
$\bar{R}_{ij}=\{\bar{r}_{ij} \mid c_i,c_j\in C_Q, \lvert R_{ij}\rvert\ge 1\}$	Set of relations between c_i and c_j in the query subgraph
$\eta_{ij}=\lvert\bar{R}_{ij}\rvert$	Number of relations between c_i and c_j in $G_Q(C_Q,R_Q)$
$A=(AC,AR)$	Graph-based annotation (AC and AR are the sets of concepts and relations)
$AR_{ij}=\{r_{ij}^d \mid r_{ij}^d\in AR,\ 1\le d\le m\}$	Set of relations between c_i and c_j in AR
$G_{Q,p}(C_{Q,p},R_{Q,p})$	Page subgraph for page p given the query Q
$C_{Q,p}=\{c_i \mid c_i\in C_Q\cap AC\}$	Set of concept of page subgraph $G_{Q,p}(C_{Q,p},R_{Q,p})$
$R_{Q,p}=\{\bar{r}_{ij} \mid c_i,c_j\in G_{Q,p}\}$	Set of relations of page subgraph $G_{Q,p}(C_{Q,p},R_{Q,p})$
$\delta_{ij}=\lvert AR_{ij}\rvert$	Number of relations c_i and c_j in $G_Q(C_Q,R_Q)$
$\tau_{ij}=P(\bar{r}_{ij},p)=\delta_{ij}/\eta_{ij}$	Relation probability for $\bar{r}_{ij}$ in page p given the query Q
$SF_{Q,p}(l)$	Set of spanning forests (l edges) for page p and query Q
$SF_{Q,p}^f(l)$	f-th spanning forest (l edges) for page p and query Q
$\sigma_{Q,p}(l)=\lvert SF_{Q,p}(l)\rvert$	Number of spanning forests (l edges) for page p and query Q
$P(Q,p,l)$	Constrained relevance score for page p , query Q , relevance class l
$ps_{Q,p}$	Relevance score of page p for a given query Q

Table 3.1 Showing definitions of symbols used.

3.5 Graph Based Formulization

So now using the graph theory we formulate the ontology graph for the IQRA University ontology created in section 3.2. We start with the ontology graph to be built over the part of IQRA University Ontology shown below in figure 3.5 (a)(b):

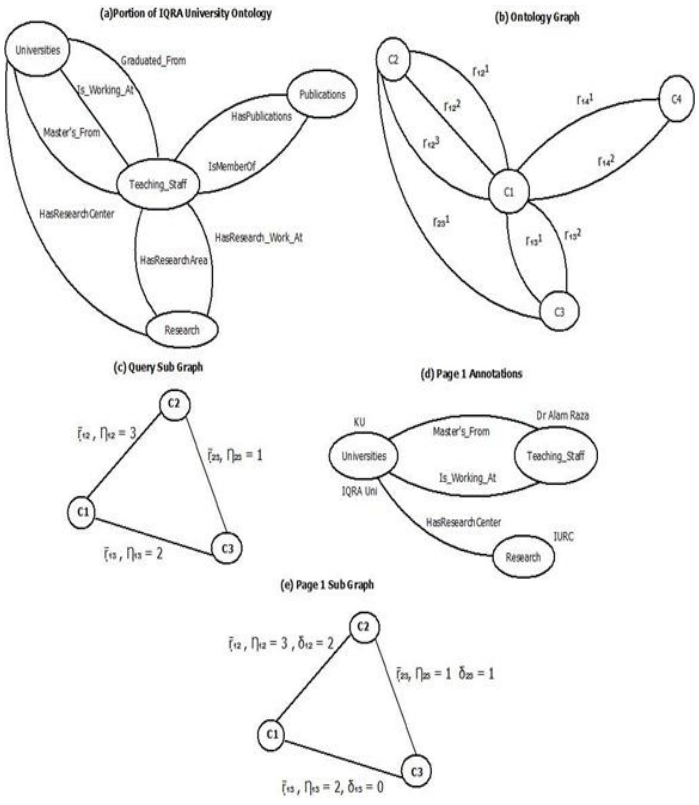


Figure 3.5 (a)(b) Part of IQRA University ontology and its Graph. (c) Web page 1 annotations graph. (d) query sub graph. (e) page sub graph

Assuming a Semantic Search Engine, we pass a query containing three keywords and three concepts associated with it. Keywords: "Dr Alam Raza" "Research" "IQRA University" and the associated Concepts: "Teaching_Staff" "Research" "Universities". So now we formulate the query sub graph based on the query and web page 1's sub graph based on the query sub graph in figure 3.5 (d) and (e).

3.6 Page Relevance Score and Ranking

For each page sub graph, all the possible spanning forests [28][29] (both constrained and unconstrained) are generated with either progressively removing edges of all the spanning forests to get constrained forests or by adding new edges to all the constrained forest of length 1 until a spanning forest is obtained [30]. We use the first approach for our experiments. Total number of spanning forests can be identified with the Cayley's formula [31] = $\mathbf{n^{n-2}}$. Considering page 1's sub graph in figure 3.5 (e) , we find out the total spanning trees of this sub graph, $\mathbf{n^{n-2} = 3^{3-2} = 3}$

So we have total 3 spanning trees for page 1's sub graph with edges n-1 =2 edges.

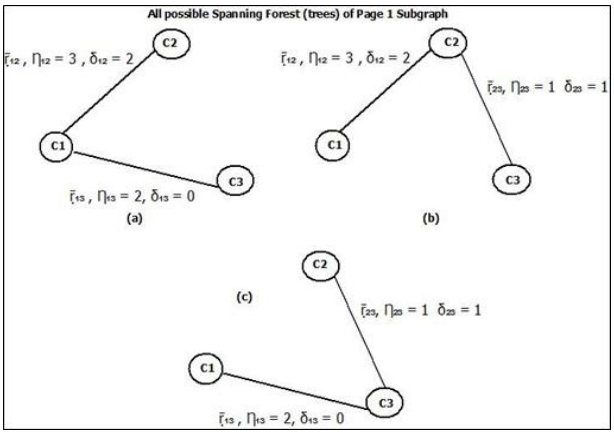


Figure 3.6 All possible spanning trees of Page 1 Sub graph

With the help of the original page spanning forest we can generate all the constrained page spanning forests. The constrained page spanning forests has edges less than the original spanning forest and if equal than it will be a spanning forest not a constrained forest. So all the possible constrained page spanning forests with edges = l = 1 are in figure 3.7.

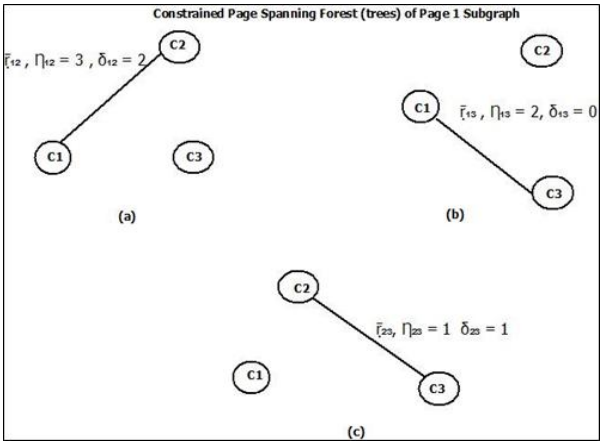


Figure 3.7 All possible constrained page spanning forests

We find the probability P(Q, p, l) of a web page with constrained page spanning forest of length l in the following way:

Constrained Page Relevance Score of Page 1

P (Q,p, l) =

$$\sum_{f=1}^{SF^f_{Q,p(l)}} \prod_{\bar{r}ij \,\varepsilon\, SF^f_{Q,p(l)}} P\,(\bar{r}ij, p).\, P\; SF^f_{Q,p(l)})$$

P(Q, p1, 1) = $P(\bar{r}_{12}, p_1).P(SF^1_{Q,p_1}(1)) + P(\bar{r}_{13},p_1).P(SF^2_{Q,p_1}(1)) + P(\bar{r}_{23},p_1).P(SF^3_{Q,p_1}(1))$ =

$$\frac{2}{3} \mathbf{x} \frac{1}{3} + \frac{0}{2} \mathbf{x} \frac{1}{3} + \frac{1}{1} \mathbf{x} \frac{1}{3} = \frac{5}{9} = \mathbf{0.5555}$$

Similarly, we compute the relevance score for the other web pages as well. The results of the ranking are presented in the table below with each page showing its ranks of Google and the

Relation Based Page Ranking Algorithm, along with the constrained page relevance score.

Google Ranking	Our Ranking	Web page details	Relevance Score
1	3	Faculty | Iqra University www.iqra.edu.pk/?page_id=1210 ... University Ranking. Click here for details Business School's Research Ranking of Pakistan ... Dr. Alam Raza. PhD, (Eco). MSc (Eco.) .... Iqra University, Karachi ...	0.5555
2	4	Program Teams | Iqra University www.iqra.edu.pk/?page_id=2070 Click here for details Business School's Research Ranking of Pakistan ... Dr. Alam Raza; Mr. Muhammad Ahsan ullah Khan Durrani; Mr. Faisal K Qureshi	0.1111
3	2	IURC Organogram | Iqra University www.iqra.edu.pk/?page_id=1667 Latest University Ranking. Click here for details Business School's Research Ranking of Pakistan ... DR. ALAM RAZA, Social Sciences. DR. ASADULLAH LARIK ...	0.6111
7	1	IU Learning Management System iulms.edu.pk/profile/publicprofile.php?UserId=353-08-1005 Picture of Dr. Alam Raza ... At present, working at Iqra University, Defence View Campus (Business Administration Department), Karachi, Pakistan, as an ...	0.6666

Table 3.2 Accuracy of the Ranking Algorithm checked over 4 web pages

4. Conclusion

It can be clearly seen that the profile page which had the research details about Dr Alam Raza has been ranked 1st by the Relation Based Page Ranking Algorithm applied. Hence it is proved that using the relations between concepts embedded in a web page (as semantic annotations) a ranking criterion formulated known as the Relation Based Page Ranking Algorithm, results in much more accurate results than Google and matched user needs to a greater extent.

To benefit from the accuracy of Relation Based Page Ranking Algorithm, we need to opt for the Semantic Web Environment, which needs to be built on top of the existing web and not by just adding the semantic web technologies into the existing web. The search engines today are becoming more and more semantic and are trying to provide users with accurate results. But in order to implement this novel Ranking Strategy, the existing search engines would require this module to be implemented in a Web 3.0 environment, as it's the best platform for this ranking strategy to work to its full potential.

REFERENCES

[1] Searching the Web. Arvind Arasu Junghoo Cho Hector Garcia-Molina Andreas Paepcke Sriram Raghavan Computer Science Department, Stanford University.
[2] A Relation-Based Search Engine in Semantic Web Yufei Li, Yuan Wang, and Xiaotao Huang.
[3] The Anatomy of a Large-Scale Hypertextual Web Search Engine. Sergey Brin and Lawrence Page.
[4] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Stanford Digital Library Technologies Project, 1998.
[5] Search Engine Ranking Variables and Algorithms. Sean A. Golliher – Publisher, SEMJ.org
[6] Google Rolls out Semantic Search Capabilities By Juan Carlos Perez, IDG News. http://www.pcworld.com/businesscenter/article/161869/google_rolls_out_semantic_search_capabilities.html
[7] A Developer's Guide to the Semantic Web. Liyang Lu
[8] A Relation-Based Page Rank Algorithm for Semantic Web Search Engines. Fabrizio Lamberti, Member, IEEE, Andrea Sanna, and Claudio Demartini, Member, IEEE
[9] XSEarch: A Semantic Search Engine for XML. Sara Cohen Jonathan Mamou Yaron Kanza Yehoshua Sagiv
[10] Swoogle: A Semantic Web Search and Metadata Engine. Li Ding, Tim Finin, Anupam Joshi, Yun Peng, R. Scott Cost, Joel Sachs, Rong Pan, Pavan Reddivari, Vishal Doshi
[11] Semantic Search. R. Guha, Rob McCool, Eric Miller
[12] SemRank: Ranking Complex Relationship Search Results on the Semantic Web. Kemafor Anyanwu, Angela Maduko, Amit Sheth.
[13] Semantic Association Identification and Knowledge Discovery for National Security Applications. Amit Sheth, Boanerges Aleman-Meza, I. Budak Arpinar, Chris Halaschek, Cartic Ramakrishnan, Clemens Bertram, Yashodhan Warke, David Avant, F. Sena Arpinar, Kemafor Anyanwu, Krys Kochut.
[14] Web 2.0 http://en.wikipedia.org/wiki/Web_2.0
[15] An Overview of Latent Semantic Indexing. Jason Hong, University of California, Berkeley.
[16] Introducing Rich Snippets. http://googlewebmastercentral.blogspot.com/2009/05/introducing-rich-snippets.html
[17] H. Knublauch, Protégé, Stanford Medical Informatics, http://protege.cim3.net/file/pub/ontologies/travel/travel.owl
[18] Page Rank Checker. http://www.prchecker.info/check_page_rank.php
[19] Ontology Development 101: A Guide to Creating Your First Ontology. Natalya F. Noy and Deborah L. McGuinness
[20] what is protégé? http://protege.stanford.edu/overview/
[21] A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition 1.1

Matthew Horridge, Simon Jupp, Georgina Moulton, Alan Rector, Robert Stevens, Chris Wroe
The University Of Manchester.
[22] Choosing an Ontology Language. Anna V. Zhdanova, Uwe Keller
[23] Web Ontology Language: OWL. Grigoris Antoniou$_1$ and Frank van Harmelen$_2$ Department of Computer Science, University of Crete, ga@csd.uoc.gr$_1$ Department of AI, Vrije Universiteit Amsterdam$_2$,
[24] What are Semantic Annotations?? Eyal Oren$_1$, Knud Hinnerk M¨oller$_1$, Simon Scerri1, Siegfried Handschuh$_1$, and Michael Sintek$_2$. $_1$ Digital Enterprise Research Institute, National University of Ireland, Galway {eyal.oren, knud.moeller, simon.scerri, siegfried.handschuh}@deri.org $_2$ German Research Center for Artificial Intelligence (DFKI) sintek@dfki.uni-kl.de
[25] Onto Mat Annotizer
http://annotation.semanticweb.org/Members/cobu/AnnotationTool.2004-07-28.1138
[26]Annotea.http://annotation.semanticweb.org/Members/lago/AnnotationTool.2003-08-25.1258
[27] SMORE.
http://annotation.semanticweb.org/Members/lago/AnnotationTool.2003-08-25.4401
[28] Spanning trees and Spanning forests
www.cafed.sssup.it/~giulio/software/spanntree/spanning_tree.html
[29] Spanning Trees and Optimization Problems. Bang Ye Wu, Department of Computer Science and Information Engineering, Shu-Te University, Yen-Chau, Kaohsiung County, Taiwan 824. Kun-Mao Chao, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan 106.
[30] RELATION BASED SEMANTIC WEB SEARCH ENGINE S. Raja Ranganathan1, Prof. M. Sadish Sendil2, Dr. S. Karthik3
[31] A. Cayley. A theorem on trees. Quart. J. Math., 23:376–378, 1889.

Integration of Semantic Web and Knowledge Management for creating dynamic environment

Molood Barati[1] ,Seyedjamal Zolhavarieh [1]

[1] Faculty of Creative Multimedia (FCM), Multimedia University, Cyberjaya, Malaysia

Abstract - *The Semantic Web is organized in a semantic direction so that it is significant to computers as well as to humans. The principal purpose of Semantic Web is to encrypt semantic repositories in computer language framework due to obtaining or sharing knowledge anytime and anywhere. Connectivity, accountability, and liability of knowledge management systems are the main component to the future generation of web services. The challenge of semantic web is the arrangement of distributed valid information and knowledge with well-defined meaning which could be applicable for different portions. Supplying semantic Web services based on the Web service modeling and semantic web ontology which has capability to dynamically explore and invoke is one of the conventional topic in Semantic Web technology. The main purpose of this paper is to present the relatedness challenge of Semantic Web Service (SWS) technologies to Knowledge Management System (KMS) by preparing dynamic environment. Additionally we argue about how to unite Knowledge management methods to SWS in order to create dynamic architecture in web.*

Keywords: Ontology modeling, Collaborative environment, Semantic Web Service (SWS), semantic gap, Content Management System, knowledge based

1 Introduction

Information retrieval (IR) has revealed several techniques to make human search easier in resources. The field of this study involved in searching for metadata in documents, observing structured storage and preparing optimized connection between various databases [9]. Additionally Semantic Web supplies collaborative environment that enlarges in frequent formats in World Wide Web. The Semantic Web disposes to convert unstructured information into a "web of data" that is constructed on the Resource Description Framework (RDF) [8].

In semantic web technology a knowledge base is a special form of database for knowledge management. A knowledge base is a basic section of each semantic web repository to collect, sorted, distributed, and search information and terms. Machine-readable knowledge bases are a term about collecting information and knowledge in a computer readable form that should be in a logical coherent structure. Some machine readable knowledge bases are exploited with artificial intelligence [3]. Combining information in the form of object attribute value is called triplets. These triplets can be semantically processed, adapted, analyzed and systematically arranged by machine agents. In addition, the agent can exploit this information with other machine agents due to create Semantic Web more real [10].

The first semantic classifier is Latent semantic analysis (LSA) that has technical method in processing and analyzing associations between repositories of information and knowledge in the web. LSA structure builds on this hypothesis that words which are close in meaning will happen close together in text documents [15]. A matrix containing word counts per paragraph (rows expresses unique words and columns expresses each paragraph) is built from a large part of text and mathematical method called Singular Value Decomposition (SVD). SDV declines the number of columns with comparable structure between rows. Words are compared by taking the cosine of the angle between the two vectors formed by any two rows. Values near to 1 express very similar word while values near to 0 show very dissimilar words in context [5].

Corporate Semantic Web (CSW) elucidates the application of Semantic Web technology and Knowledge Management methodology to unify environments. The initial framework of a Semantic Web encounters to many problems such as scalability, lack of stimulus to annotate sources, and comprehensive adoption of shared ontologies (Corby & Faron-Zucker, 2002). Moreover privacy qualification and trust issues are the other essential of a perfect Semantic Web Service (SWS) [1]. CSW regards to semantic improvement of information which is conveyed to subscribers as well as semantic applications. CSW is aimed to promote the unification of information in heterogeneous sources, improving information retrieval by reducing information overload, providing decision making support, dispersing ambiguities in terminology association, and identifying relevant information [14].

CSW is constructed based three fundamental areas: ontology engineering, semantic applications, and collaboration. The web life cycle of Corporate Semantic is illustrated in figure 1. Ontology engineering considers to efficiency and effectiveness of ontology extension toward ontology growth and maintenance. Semantic application analyzes applications to evaluate what range could attain benefit from semantic technology. Collaboration emphasizes on the human centered aspects of knowledge management in corporate concepts. Extracting explicit knowledge from the amateur user activities in building collaborative ontology could be one of the examples of collaborative environment [14].

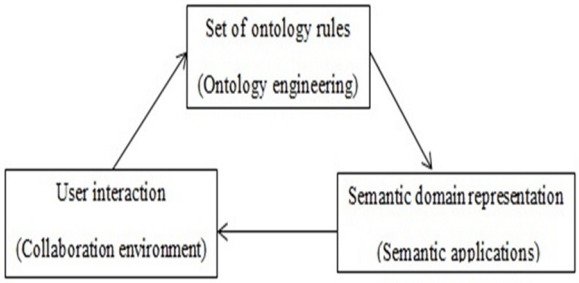


Fig.1: The web life cycle of Corporate Semantic

Providing dynamic distributed semantic web that has the capability to support difference possibilities has created many issues in computer science world. In this paper we present how to unify new knowledge management method into SWS to create dynamic environment.

2 The review of Dynamic environment in SWS

The concept of Semantic web Service in Dynamic Environments pertains to exploring, arguing, classifying, executing and handling dynamically to realize development steps of workflow. Methods which are suggested in this area include in many various concepts. Dynamic Semantic Wed (DSW) is based on the techniques, methods and paradigms of the emerging Semantic Web movement and its applications. DSW has the structure of reducing inter-ontological transactions (translations, mappings, navigation) among various ontologies and taxonomies. What is required to extensive and perfect framework are the ability to manage Virtual Organizations workflow processing, to change organization workflow due to collect service-oriented tasks, and alters these tasks from accessible services, manage new information and knowledge and accomplish new service [2].

2.1 Knowledge-based Dynamic Semantic Web Services Framework

Knowledge-based Dynamic Semantic Web Services (KDSWS) Framework instructs in an integrated mode. The life cycle of activities involved in preparing, creating, requesting, exploring, selecting, changing, and delivering Semantic Web Services. Figure2 clarifies the life cycle of KDSWS framework [7].

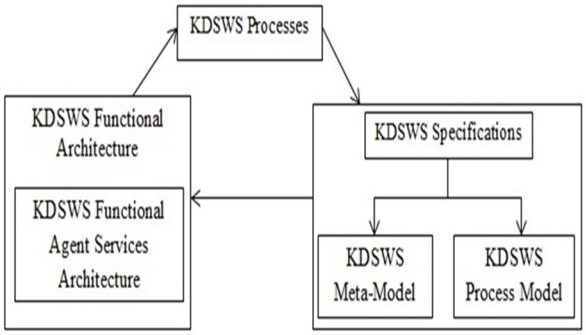


Fig. 2: The KDSWS framework life cycle

The KDSWS Processes illustrate the steps to deliver functionality by web services and threads as global layer of functionality. KDSWS Specifications are built on two models the KDSWS Meta-Model, and the KDSWS Process Model which are based on the Knowledge/Data Model. Features Specification to increasing the semantic web services is the responsibility of this section. The KDSWS Functional Architecture presents the execution components to affirm the Framework. A central component of the KDSWS Functional Architecture is the KDSWS Functional Agent Services Architecture to manage services into specialized liabilities [7].

2.2 The importance of Dynamic Semantic Web

The most necessities to creating a system for dynamic semantic web services are:

- In wireless environments, minimizing resource such as bandwidth to prevent receiving too many responses to queries. For resolving these barriers we can have a completely decentralized topology that have the capacity to quick update without need to republish and reproduce any new services [7].
- Minimizing manual configuration to automatic registry discovery on LANs and WANs [7].

- Explore the best correlated services for task accessibility, and selection of the best services among many others based on semantic descriptions [7].

Need to create Content Management System (CMS) for evaluating the dynamic data in dynamic environment is a new controversial issue to decline the recent problems. CMS procedures could be manually or computer based. A CMS allows creating, revising, and altering content as site maintenance from a central page. It supplies a set of procedures to manage workflow in collaborative environments. Furthermore CMS regards to resource management, transaction control, evolution of the virtual organization, and management of workflow [6].

3 Semantic Web and Knowledge Management systems

For being successful in KM we require to be equipped to several techniques which are related to KM such as Groupware, expert systems, decision support systems and various forms of collaborative systems, Because this field is combined of difference professional sciences. The efficient and effective management of knowledge resources need to dynamic communication among departments and members due to quick respond to change. Capturing, sharing, supplying and managing are the factors of effective knowledge-based organization.

There are so many technologies that prepare environment for people to share information and knowledge, use them for enhancing their skills and abilities, and enrich experiences in all over the word. Hence, by Increasing advances in technologies time by time and reusing knowledge for getting high performance, people require to have up-to-date and dynamic environment in order to managing knowledge in knowledge environment. As information and knowledge needed and used continually, searching for way to supply dynamic knowledge environment required [12].

Data accuracy and up-to-date information and knowledge are unavoidable elements in each organization with dynamic infrastructure. Combining KM with SWS would supply dynamic environment to corroborate immediate situation. According to recent studies the semantic web is a web of data that is directly or indirectly adapted by agent systems. HTML technology supplies static environment in WWW, but by interfering Semantic Web technology we will be able to overwhelm these barriers. SWS is constructed on the environment with software agent to affirm fast decision making. SW technologies exploit taxonomies and ontologies to prepare web content. With SW tools such as Protégé and knowledge representation models, the development has covered sharply. SW is formed on distributed and collaborative environment that ontologies will engage. Evolution of shared inter organizational ontology is a new area in this field that has attempted to produce integrated collaborative environment [12].

4 Using CMS for KMS due to create dynamic environment

Knowledge management system is the collection of information technologies used to expedite the collection, organization, and distribution of knowledge among users and individuals. An information management system massages data to create information and knowledge. A knowledge management system is an information management system with all the tools required to help individuals turn information into knowledge which could be useful for decision making [11].

One of the main issues in some of the knowledge bases in knowledge management system is to be up-to-date. Nowadays knowledge base systems are based on the static knowledge and they should examine for updating of knowledge bases time by time. If the knowledge management system provides one collaborative environment for data and knowledge in order to distribute data dynamically, some of the barriers in up-to-dating data will be resolved. The environment which prepares collaboration between knowledge bases is called collaborative environment. We propose to apply CMS for creating dynamic knowledge bases [12].

A CMS is a computer application utilized to create, edit, supervise, and share content in a website. CMSs are frequently exploited for sharing industry-specific documentation such as operators' manuals, blogs, articles, sales guides, technical manuals, news, and marketing brochures. The content managed may contain computer files, image media, audio files, video files, electronic documents, and web content. Most of tasks that they do exist in the following [13]:

- Allow for the large number of people to contribute and share knowledge
- Control access to data, information and knowledge according to user accessibilities (defining which information and knowledge users and user groups can view, edit, publish, etc.)
- Aid in easy storage and retrieval of information
- Control of knowledge validity and compliance
- Reduce repetitive duplicate input
- Improve the ease of report writing
- Improve communication between users

Content is necessary, any type or 'unit' of digital information. It can be: text, images, graphics, video, sound, documents, records and etc. In other words anything that is probably to be managed in an electronic format. Content Management is efficient management of the

content depicted above, through combining rules, process, procedures and/or workflows in a way that its electronic storage is supposed to be 'managed' rather than 'un-managed' [11].

5 Discussion

Staying on up-to-date and dynamic situation for knowledge bases is one of the critical issues in recent KMSs. DKMS need dynamic data, dynamic information and dynamic knowledge. Todays, KMSs are based on the static knowledge bases and they should check for updating version of knowledge bases time by time. If the knowledge management system prepares one dynamic knowledge environment or on the other meaning collaborative environment in order to share data dynamically some of the problems in up-to-dating and dynamic environment have been solved. It means when KMS create collaborating environment for share and collaborate knowledge between knowledge bases, some knowledge base collaboration produce, then dynamic environment can occur. For solving this issue, using CMS and create connection between it and KMS is offered. CMS can control version of knowledge bases and keep versions up to date dynamically rather than statically. It can prepare accessibility for each of user in different areas by producing CMS user interface.

CMS is the system which is defined as a tool or combination of tools that promote the efficient and effective production of the desired 'output' using the managed content. Additionally CMS is a tool that enables an assortment of centralized (technical) and de-centralized (non-technical) user accessibility to create, edits, manages and finally publish number of formats in different variety of content such as text, graphics, video, documents, and etc. In addition being constrained by a centralized set of rules, process and workflows that ensure coherent, validated electronic content is required for any management system.

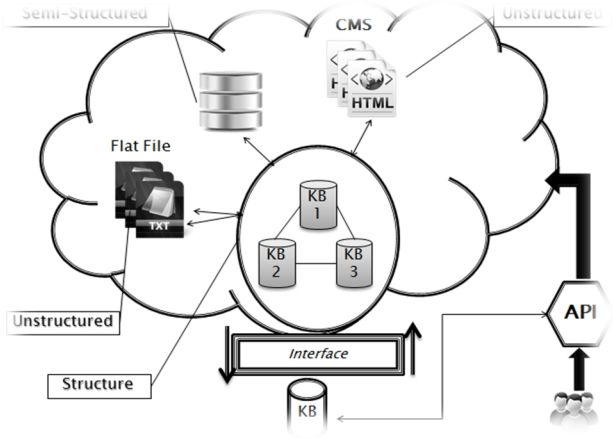


Fig. 3: CMS for KMS

As it is determined in figure 3, there are different files in CMS for managing and embedding in knowledge bases. In this figure, variety of sections can be structured, un-structured and semi-structured in whole of the system. For accessing to this system and managing files in dynamic environment we need to implement interface and create Application Programming Interface (API) due to help users to extract new knowledge. CMS supplies versioning and prepares up-to-date knowledge and information and also it provides the ability to its users due to use new knowledge anywhere and anytime. In addition, extra new knowledge can be added to this dynamic architecture.

Joomla (with php programming language) and Plone (with Python programming language) are two types of content management system that works with different programming languages. Plone has higher security in compare of Joomla, because Plone utilizes Python. Python is management programming language with high security, while Joomla is more popular content management system which is written by php that is web base language. By creating connection between CMS and KMS, dynamic knowledge management system can appear.

6 Conclusion

Rapid growth in web size and quick change in its application has been appeared in different layers of web. In complicated and dynamic web environment, SWS of information becomes critical issue due to search, share, manage knowledge, and also automatically communicate among software agents, web services and human. The semantic web and automatic processing of semantic information has defined as a controversial issue. In this paper we mention to CMS method as a way to improve dynamic environment in semantic web technology.

7 References

[1] Ankolekar, A., Burstein, M., Hobbs, J., Lassila, O., Martin, D., McDermott, D., . . . Payne, T. (2002). DAML-S: Web service description for the semantic web. *The Semantic Web—ISWC 2002*, 348-363.

[2] Benatallah, B., Sheng, Q. Z., & Dumas, M. (2003). The self-serv environment for web services composition. *Internet Computing, IEEE, 7*(1), 40-48.

[3] Berners-Lee, T., & Hendler, J. (2001). Scientific publishing on the semantic web. *Nature, 410*, 1023-1024.

[4] Corby, O., & Faron-Zucker, C. (2002). *Corese: A corporate semantic web engine*.

[5] De Lathauwer, L., De Moor, B., & Vandewalle, J. (2000). A multilinear singular value

decomposition. *SIAM Journal on Matrix Analysis and Applications, 21*(4), 1253-1278.

[6] Han, Y. (2004). Digital content management: the search for a content management system. *Library Hi Tech, 22*(4), 355-365.

[7] Howard, R., & Kerschberg, L. (2004). A Framework for Dynamic Semantic Web Services Management. *International Journal of Cooperative Information Systems, 13*(4), 441-485.

[8] Klyne, G., Carroll, J. J., & McBride, B. (2004). Resource description framework (RDF): Concepts and abstract syntax. *W3C recommendation, 10*.

[9] Korfhage, R. R. (2008). Information storage and retrieval.

[10] Martín-Recuerda, F. (2006). Application integration using conceptual spaces (CSpaces). *The Semantic Web–ASWC 2006*, 234-248.

[11] McKenna, F. (2008). A Knowledge Management System: A Discourse. *Online im Internet unter:http://www.knowledgeonecorp.com/news/pdfs/A% 20Knowledge% 20Management, 20*.

[12] Muthaiyah, S., Raman, M., & Dorasamy, M. (2010). *Knowledge Evolution System for Dynamic Emergency Planning and Response*.

[13] Quadri, S. A. (2011). Developing, Managing and Maintaining Web Applications with Content Management Systems: Drupal and Joomla as case study.

[14] Tempich, C., Simperl, E., Luczak, M., Studer, R., & Pinto, H. S. (2007). Argumentation-based ontology engineering. *IEEE Intelligent Systems*, 52-59.

[15] Wiemer-Hastings, P., Wiemer-Hastings, K., & Graesser, A. (2004). *Latent semantic analysis*.

SESSION

ONTOLOGIES

Chair(s)

TBA

Towards a Hybrid Ontology Design and Development Life Cycle

Rishi Kanth Saripalle and Steven A. Demurjian
Department of Computer Science & Engineering, University of Connecticut, Storrs, CT, USA

Abstract: *Ontologies are artifacts employed for defining and sharing domain and semantic conceptualization which may involve stakeholders with disparate backgrounds. These ontologies are created, imported, exported, and reused utilizing different frameworks, tools, and techniques. In order to build a robust shared ontology conceptualization, the ontology development process needs to consider an approach that mirrors the software development process (SDP), allowing ontologies to be constructed in a more structured and organized process. Towards this goal, this paper proposes a Hybrid Ontology Design and Development Life Cycle or HOD^{2}LC, where each phase is characterized to find the best fit design technique for it role in the life cycle as compared to typical SDP phases. Next, we explore, compare, and contrast the existing ontology design and development alternatives with respect their phases as related to varied SDP models, and use this as basis to compare and contrast those alternatives with HOD^{2}LC.*

1 Introduction

An *ontology* can be defined as "*formal specification of conceptualization*" [1], a composition of *concepts* which represents a class or entity identified by their unique *attributes*, *relationships,* or *associations* including the interaction between the concepts and *axioms* which are rules or constraints on classes or attributes or relationships (and in that sense properties of relations are kinds of axioms). The axioms guide users (both human and agents) to definitely interpret the semantic meaning of the concepts. Ontologies are being extensively used in research communities such as knowledge engineering [1], domain modeling [2], database analysis [3], natural language processing [4, 5], bioinformatics [6] etc. Further, ontologies serve as the basis for the Semantic Web in order to allow semantics to be attached to data that augments the static information with domain meanings.

Ontologies are conceptual models intended to capture both the *structure* and *semantics* of the domain and in general they can be categorized based on the *formalness* of the knowledge captured. A *Top-Level Ontology* [1] describes generic concepts such as space, time, events, etc. which are *independent* of any *domain*. A *Domain Ontology* [1] describes the vocabulary related to a domain such as medicine, automobiles, people, education, etc. or a task or activity, by specializing the terms introduced in the top-level ontology. Lastly, an *Application Ontology* [3] describes the concepts of a focused domain, which are often specializations of both the above categorized ontologies such as International Classification of Diseases (ICD) codes [28] for the medical domain and *friend of a friend* (FOAF) ontology [29] to describe individuals and their social network connections. To demonstrate, in Figure 1, *Time* (top-level ontology) can be used by *Medicine* (domain ontology) for time-stamping medical entries, which is later utilized for building *ICD Codes* (application ontology). Based on the requirements of the information system, the developers can modularize and hierarchically organize their ontologies.

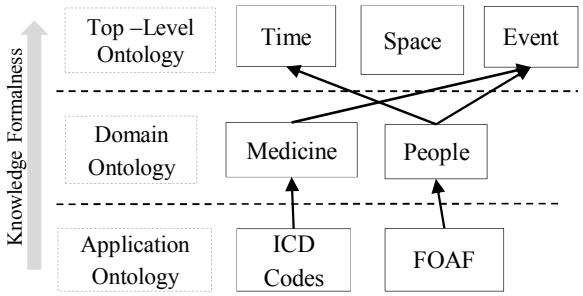


Figure 1: Dependency between ontologies based on their knowledge formalness

The process of creating a set of ontologies for an application is predominately, with specific ontology details defined to conceptualize the semantics of the domain by stakeholders from varied backgrounds (ranging from ontology engineers to end uses) that work together towards a consensus. During this conceptualization, a *general conceptual theory* is developed which is later converted into a *formal theory* to yield a set of *schemas* or *domain modes* that represents the intended ontologies. The issue is that this process is often conducted in an ad-hoc fashion, either without or with at most a partial utilization of anything akin to a software development process (SDP) that would be able to work towards this conceptualization of multiple ontologies for an application with a more structured approach. Our main question in this paper is to ascertain whether a SDP can be applied to ontology design and development to result in a solution is thought-out, consistent, and well-structured in terms of modularity, reusability, and efficiency.

Historically, SDPs can be leveraged via varied systematic methodologies to take an application from its

inception to deployment. SDP models include waterfall, spiral, iterative, incremental, and agile development [7-10]. Many of these SDP models share a well accepted set of phases: r*equirements* to capture the capabilities the software; *specification* to provide a description components and their inter-connections; *design* to propose varied conceptual models; *analysis* to verify if the proposed design meets requirements; *implementation* of components and the entire integrated solution; *testing* of correctness and comparison against requirements and specification; and *maintenance* to track system performance, identify bugs, updates etc.

Our first objective in this paper is to propose a Hybrid Ontology Design and Development Life Cycle (HOD^{2}LC) borrowing concepts from SDP models while simultaneously clarifying the differences and additions, in order to narrow the gap between ontology development and software development process. To assist in this discussion, we utilize our work on model extensions to OWL from varied perspectives in terms of *attributes*, *schemas*, and *profile* that leverage the UML metamodel [2]. Using HOD^{2}LC as a basis, the second objective in this paper is to explore, compare, and contrast the existing approaches to ontology design and development [11, 12, 13, 14, 15, 16, 17] against HOD^{2}LC. This results in a further justification of the phases of HOD^{2}LC by understanding the way that our proposed approach measures against the competition.

The rest of the paper is organized as follows: Section 3 details the proposed Hybrid Ontology Design and Development Life Cycle (HOD^{2}LC) with a set of 9 phases as compared to the SDP process. Section 3 introduces and reviews six alternatives to ontology development processes. Section 4 compares and contrasts HOD^{2}LC with the ontology methodologies of Section 3. Finally, Section 5 concludes the paper.

2 Proposed HOD^{2}LC Model

This section details our work on a *Hybrid Ontology Design and Development Life Cycle* (HOD^{2}LC) model that leverages various software engineering methodologies and capabilities of various ontology approaches to arrive at a process approach that is more structured and comprehensive than the current traditional ontology development approaches (as we will discuss in Section 3). The HOD^{2}LC model has a number of phases that represent different aspects of the ontology design and development process. The phases involved in the proposed model are as follows:

Phase 1: The *Problem Analysis* phase identifies and analyzes the problem faced in the information system leading to the development of new ontology and/or extending an existing ontology. Problem analysis is similar to SDP's requirement phase, but utilizes different methodologies such as *abstraction techniques* from heuristic classifications [31] to identify domain or domain concepts or instances. Generally the problems faced are related to instance data of the domain ontology or a state of the ontology which are commonly represented using *competency questions* or UML usecase diagrams from which an abstract domain problem has to be formulated. For example, List all the *symptoms* of *Radiation Chemotherapy* on *Breast Cancer*; Find *symptoms* of *acetaminophen* on patient suffering from *bronchitis* with pre-condition of *Asthma*; *Injuries* which can cause *Internal Bleeding* etc. When the abstraction techniques is applied the domains *symptoms*, *treatment* and *diseases* are identified along with concepts *Asthma, Chemotherapy, Cancer, Breast Cancer* etc. The problem analysis is to identify the domains involved and may also identify abstract meta-concepts (detailed in Design Phase), domain model concepts of the domain from the instance data.

Phase 2: The *Integration Phase* allows designers to search for existing ontologies meeting the problem criteria identified in the Problem Analysis Phase (Phase 1). For example, we can reuse *RxNorm* [19] vocabulary which provides normalized names for clinical drugs and UMLS semantic network which is composition of semantic types and relationships [13] to support various medical concepts. For the domain of Symptoms, Injuries, Procedure we can use ICD vocabulary [13] and LONIC for laboratory codes.

Phase 3: In *Knowledge Acquisition*, designers interact with domain experts (providers, researchers etc.) searching multiple resources (medical records, data, ontologies etc.) to identify the concepts and domain vocabulary required to develop the complete ontology. For the domain vocabulary a *Glossary of Terms* (GT) can be defined to encompass *instances* of various *classes* defined in the conceptual model, *associations* and values of the classes *attributes*. The GT identifies and gathers *all* the useful and potentially usable domain knowledge and its meanings. The GT can be built by reusing methods proposed by Mariano Fernandez et al.[11] or Asunción Gómez-Pérez et al.[12]. This phase can be performed in parallel with *Specification*, *Design* and/or *Analysis* phases.

Phase 4: In the *Specification phase,* the designer defines the scope of various domains and functionality of the ontology and its concepts. For instance, Phase 1 identified *disease* and *symptoms* as one of the domains, but this phase defines boundaries on what kinds of diseases are to be conceptualized such as: The diseases under consideration are Immune Disease, Respiratory Disorders and Digestive Disorders; All the diseases must be associated with a symptom(s), any injury causing it, Procedure, and Test; A Procedure must be associated with a Disease and Symptoms for capturing side effects; A Test must be associated to a Disease or a Symptom; All the entities must have a *unique ID*, *name* and *scientificName* Etc.

Phase 5: In the *Design Phase*, the concepts in the domain(s) are identified, which can be classified into *Meta concepts* and *Domain Model concepts* as shown in Figure 4. The term *concept* encompasses classes, associations and attributes. To illustrate Meta-Concepts, consider the UML Class diagram in Figure 2 which acts as data model for domain information. The *Circulatory System Diseases*

class is a parent for *Acute Rheumatic Fever*, and *Hypertensive Disease* classes*; General Symptoms* is a parent class for *Cardiovascular Symptoms* and *Skin Symptoms*. The two parent classes are associated using a *causes* association while inheritance connects the subclasses to the parent class. Similarly other diseases and symptoms such as *Respiratory Disease, Nervous Diseases, Reproductive Diseases, Respiratory Symptoms,* and *Reproductive Symptoms* etc. can be defined. From the class diagram, further abstraction can be achieved to define *domain specific* yet *generic* concepts. For example, the *Circulatory System Diseases* (and its subclasses), *Respiratory Diseases, Cardiac Diseases, Reproductive Diseases* etc. are all of type *Disease* and *General Symptoms* (and its subclasses), *Respiratory Symptoms, Reproductive Symptoms* etc. are all of type *Symptoms*. Similarly, we can consider *Medication, Procedure, Treatment, Injury* etc. These abstracted concepts which are related to a given domain (medicine) yet generic in nature for that domain of disclosure are called meta-concepts. These meta-concepts can be identified and inter-connected to form a conceptual abstract theory upon which a domain model can be built. As shown in the Figure 3, extracted meta-classes (Disease, Symptom, Treatment, Procedure and Medication) can be connected to each other using meta-association (hasSymptom, hasProcedure, hasMedication and hasTreatment). The meta-classes can be associated with meta-attributes (such as Id, name, scientificName etc.). Once, the meta-concept are captured, we can use these concepts to build the domain model and impose the meta-concepts onto the domain model[27]. For example, *Circulatory System Diseases* a domain class *is-of-type Disease* meta-class. In software engineering, the process of defining concepts at various levels and using the concepts from the top-level to develop the bottom-concepts is called Meta Process Modeling. As show in Figure 4, meta-concepts (MC_1, MC_2. MC_3) can be used to develop domain model concepts (DC_1, DC_2. DC_3) which can later be instantiated (is-a) to build instance data (ID_1, ID_2. ID_3).

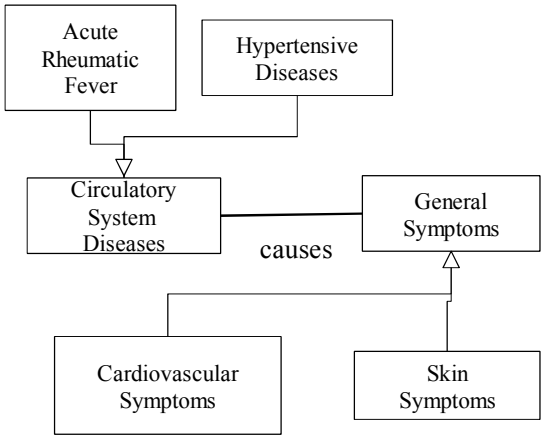


Figure 2: Sample UML Class Diagram

Phase 6: *Analysis* is another important step that has to be executed before *implementation* of the design model(s). In this phase, the developers and the end users revisit the *specification phase* to validate the design models developed in the *Design Phase*. The end users will validate to see if all the required system requirements can be solved using the design models and developers will look back into to the models to check for *modularity*, *reusability, efficiency* and any other software metrics according the development environment. A feedback loop involving *specification, design* and *analysis* phases will bolster incremental learning process, where the developers and end users can learn from the previous cycle. This loop will also allow flexibility in adding/modifying any user-defined specifications. For instance, for querying symptoms of *pneumonia*, we would involve say *General Symptoms* and *Respiratory Symptoms* class with a *select* query.

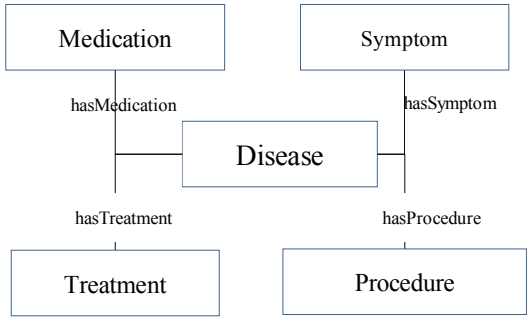


Figure 3: Sample *Meta-Concepts*

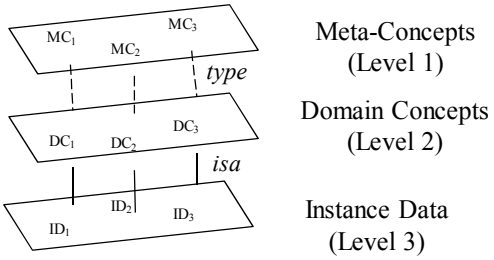


Figure 4: Hierarchical organization of concepts

Phase 7: The *Implementation phase* provides the transition from conceptual model (UML Class Diagram or ERD diagram) to concrete implementation using *KIF, OWL DL, OWL Lite, Frames* etc. This phase requires decisions to be made regarding the particular ontology language, tool and API framework for the implementation. For example, the ontology can be realized in a formal language such as *KIF, frames*, *OWL*, *RDF* or even as a simple database schema for *is-a* type vocabulary. Alternatively, via our layered approach (Figure 4), frameworks such as UML profile [26] or *OWL Domain Profile* (ODP) [2] or DOGMA [22] can be utilized. The choice of the language will also depend on the *usability, performance* and *availability* of the language.

Phase 8: The *Testing phase* carries out a technical judgment of the ontologies, their software environment and

documentation with respect to a frame of reference (in our case, the requirements, specification document) during each phase and between phases of the life cycle, using techniques such as Ontolingua [24] or a framework for evaluating knowledge sharing technology (software, ontologies, documentation) [25]. If the ontology is realized in OWL, the OWL ontology debugger [30] can be used for circling on inconsistent concepts and axioms that are causing them. Other frameworks such as UML, frames, and the entity relationship diagram model can be mapped to an OWL framework to utilize the OWL debugger for testing ontology consistency.

Phase 9: The *Maintenance and Documentation* phase is where the developed ontology has to be monitored for smooth and efficient performance of the system (maintenance) backed by a detailed narrative report of the ontology concepts, its axioms and usage (documentation). This phase can start with knowledge acquisition and run in parallel with subsequent phases.

Figure 5 illustrates the Hybrid Ontology Design and Development Life Cycle (HOD^{2}LC) model involving the aforementioned phases and its iterative process through Phase 2 to Phase 7.

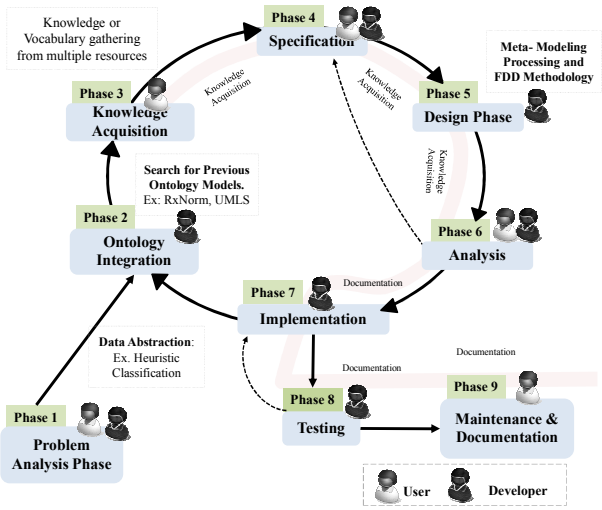


Figure 5: HOD^{2}LC Model

The *iterative* approach assists developers to take advantage of what was learned during previous phases. There is also a need of two feedback loops: one, between *Analysis* phase and *Specification* phase, helping developers have to validate the ontology model to check if the specifications have been fulfilled and flexibility in editing specifications so as to make necessary changes to the ontology model without go through the whole cycle; and another between *Implementation* phase and *Testing* phase helping developers resolve any implementation errors. These loops are represented using dotted line differentiating it from the life cycle's solid line. The *Knowledge Acquisition phase* can be executed in parallel with other phase until the *Implementation* phase which is responsible for developing the vocabulary of the domain from the information gathered from the *Knowledge Acquisition* phase. The *Documentation* of the ontology can also be executed in parallel starting from *Analysis* phase.

To support the iterative process employed in the *Design phase*, we utilize Feature Driven Development methodology (FDD) [23], a model driven agile software development process. FDD is a model driven agile software development process comprising of five activities: *overall model* where a high-level walkthrough of the domain scope of the system and its context is performed; *Build Feature List* (feature can either be a whole class or a method call on a class) where a detailed domain walkthroughs are performed to decompose the domains into small groups which are presented for discussions; *Plan By Feature* where the generated features are prioritized for further the development plan; *Design* and *Build By Feature* where a programmer selects a small group of features that are to be developed within two weeks. Abstracting out the

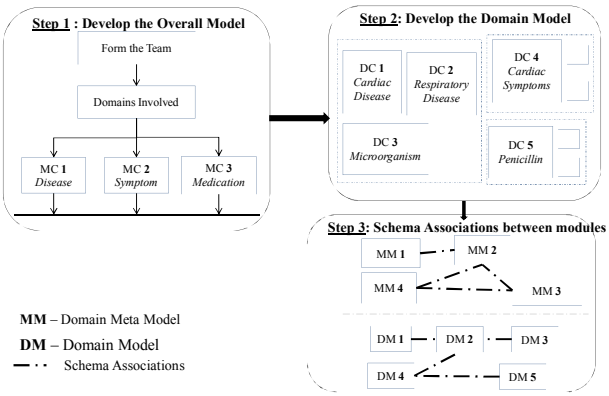


Figure 6: Feature Driven Development of Meta-Concepts and Domain Concepts.

steps from FDD and applying it to our approach (Figure 6), we have the following steps. *Step 1*, a higher-level walkthrough of the domains involved in the domain problem should be performed to identify meta-concepts. For example, meta-classes such as *disease*, *treatment*, *symptom*, *medication*, and *treatment* etc.; meta-attributes such as *uid*, *name*, etc.; and meta-associations such as *hasSymptom*, *hasTreatment, hasMedication, hasParent*, *isa* etc. This step is equivalent to identifying profile concepts at the metamodel level. In *Step 2*, once there is agreement on the meta-concepts, they are decomposed into smaller domain concepts by multiple groups. For example, classes such as *Respiratory Diseases, Cardiac Diseases* can be defined which are of type *Disease*; *Cardiac Symptoms* and *Mental Disorder Symptoms* etc. of type *Symptom*. The respective attributes and associations are also identified. Finally, in *Step 3*, as the concepts that have been identified the attributes can be assigned to classes and associations can relate classes forming the meta-conceptual model and domain model of the ontology at different layers.

Once the respective models have been built, the modular models can be interconnected for network of

ontology models (Figure 6, Step 3). The iterative nature of the cycle will help developers learn from the previous phase and incremental nature shows the sign of progress and partial output to the end users. The cycle is stopped once an agreement has been reached on structural and semantic aspects of the ontology. The work presented in this section is our initial effort to quantify an ontology design and development process; our work is ongoing in this area to fine tune the process and apply to more complex, realistic examples.

3 Ontology-Based Alternatives

In this section, we will discuss six prominent ontology development processes [11-17] which employ alternative design and development approaches. Our intent is to understand the way that each of these alternatives uses similar and/or different concepts than HOD^{2}LC and/or SDP. In terms of SDP, we consider some of the more dominant models: *waterfall* which flows from phase to phase with each with each phase being completed finished before moving to the next phase [7]; *spiral* [7, 8] that focuses on a continuous refinement of determining objectives, identifying and resolving risks, development and testing, and planning the next cycle; and an *iterative* [7] that allows waterfall phases to be revisited; and *Agile development* [10] for self-organized, cross disciplinary teams that follow an iterative and incremental process. By reviewing these six alternatives in this context, we can then effectively compare them in Section 4 to HOD^{2}LC.

The first alternative, the *Methontology* [11] model composed of the following phases: *specifications (similar to Phase 4 in HOD^{2}LC); knowledge acquisition (similar to Phase 3 in HOD^{2}LC)*; *conceptualization* where designers define the domain vocabulary and a conceptual model for the ontology, *integration (included in Phase 3 of HOD^{2}LC);* and *implementation* and *evaluation (similar to Phases 7 and 6, respectively)*. Methontology opts for an *evolutionary SDP* instead of *waterfall* and *Spiral* since they believe it will more easily facilitate expanding the ontology as needed. Methontology lacks HOD^{2}LC's Phases 2, 5 and 8 with respect to integration of conceptual models, design for defining associations, and testing that exceeds its evaluation.

A second alternative, by Fernandaz et al. [12] proposed a *method* ***not*** *a complete methodology* or *development cycle* for an ontology with the following steps: develop a requirement document (*similar to phases 1 HOD^{2}LC*); conceptualize the domain terminology via a *Data Dictionary* which identifies domain classes and instances; create a *Concept Classification Tree* which groups the identified concepts; and, create *tables of Constants* of the domain, *Instance and Class Attributes* of the domain, and *Formulas* which are used to infer numerical values of attributes. The dictionary, trees, tables, and formulas will all occur, in Phases 3 and 5 of HOD^{2}LC. Note that the steps enumerated in the method are similar to waterfall model where the methodology phases are executed sequentially without feedback loop thus inherits the drawbacks of the waterfall model.

A third alternative, the Enterprise Ontology (EO) project [13, 14] consists of four phases: *purpose* of the ontology, *building* the ontology via coding, capture, and integration, *evaluation* to check the ontology against requirements, and *documentation* to clearly relate all ontology concepts *(similar, respectively to HOD^{2}LC's Phases 1, 2, 6, 7 and 9)*. The *purpose* phase is equivalent to SDP's *requirements* phase. The *building* phase is equivalent to combination of SDP's *implementation* phase. The *evaluation* phase is similar but not as extensive as SDP's *Analysis* phase. One drawback of EO project is that there is no model as indicates the way that the phases are connected and no guidelines provided to achieve the building phase.

A fourth alternative, the TOVE project [15], has a primary goal is to answer enterprise queries to existing or future usecase scenarios. Based on these scenarios, a set of questions named *informal competency questions* are raised that for which ontology has to be developed to answer. The motivating scenario and competency questions provide the designer with the information necessary to decide on whether to develop a new or extend and existing ontology, i.e., the set of questions form the *requirement* phase (*similar to HOD^{2}LC's Phase 1)*. The next step is to specify the terminology of the ontology by using first-order logic forming the *conceptualization* and *implementation* phase (*similar to Phases 3 and 7 of HOD^{2}LC)*. TOVE defines the various phases an ontology design has to address for building ontology, but doesn't provide a life cycle or a model connecting them with one another.

The fifth alternative by Uschold [16] presents a unified methodology by combining methodologies EO and TOVE projects. The first step is to define the purpose of the ontology which can be done in several ways, e.g., to identify the intended users, or as in the TOVE project with motivating scenarios and competency questions, to form the user *requirements* document *(similar to Phases 1 and 3 in HOD^{2}LC*). In the *conceptualization* phase, the developer should decide what level of formality the ontology must have and identify the concepts and the relations among them *(similar to Phase 5 in HOD^{2}LC)*. The work describes four different approaches for constructing the ontology: use an ontology editor to define terms and axioms; perform the previous steps and then begin a formal encoding; produce an intermediate document that consists of the terms and definitions that appeared in the second step which may result in a specification of the formal code or its documentation; and identify and differentiate the formal terms from the set of informal terms. The work also includes an evaluation or revision cycle, where the developed ontology is compared to the competency questions or the user requirements. Similar to the EO Project [13, 14], this work also doesn't provide any model interconnecting the phases.

The sixth alternative by Noy and McGuinness [17] describes a way to develop an ontology *iterative* methodology starting with a rough concept and then revising and filling in the details; this could correspond to both an iterative or spiral SDP. The first step in their methodology is to determine the domain and the scope of the ontology fulfilling the *requirements* phase *(similar to portions of HOD²LC's Phases 1 and 4).* Their next step is to explore the use of existing ontologies, akin to Phase 2 of HOD²LC. A list of all the terms that could be needed or used is then produced, mirroring some of the activities in Phase 3 of HOD²LC. The next step involves the design of a class hierarchy that represents an "is-a" relation where siblings should have the same level of generality, and also guidelines regarding when to introduce new classes or instances. This also includes the classes to be defined, the terms and the relations, the properties of the classes (attributes), relations among classes and attributes (including if the relations are inverses), default attribute values, and class properties such as cardinality, domain and range *(similar to Phase 5 in HOD²LC).* These steps satisfy the *Design* and *Conceptualization* phase of SDP. The final step in the process is the creation of individual instances which corresponds to portions of the latter phases 7 of HOD²LC.

4 HOD²LC vs. Ontology Methodologies

This section presents our proposed HOD²LC and its nine phases (see Section 2) compared and contrasted with the six ontology alternatives (see Section 3). We acknowledge that this comparison is somewhat biased towards HOD²LC which has all of the phases as compared to the six alternatives. While all of these alternatives are all sufficient to their own degree, they are primarily focused on the ontology development. HOD²LC seeks to expand this to design and development to provide a broader process that is more comprehensive and considers a larger range of requirements in order to effectively define ontologies from model to schema to instance levels. To compare, we define three qualitative criteria for each phase: **None** – the alternative does not support the phase; **Partial** – the alternative may have partial implemented the phase; and **Full** – the alternative has the phase in its life cycle. Table 1 summarizes alternatives vs. HOD²LC phases.

From Table 1, all alternatives fulfill the *problem analysis phase* either through *requirements* generation, *usecase* scenarios, formulating *competency questions* [13, 14, 15] or through instance *data abstraction.* The *ontology integration phase* is not given prominence in any alternatives except TOVE which is primarily responsible for reuse of existing ontologies. The *knowledge Acquisition phase* is the most focused phase in all alternatives; they all maximize the ontology domain vocabulary. The alternatives either combine the *specifications* with other phases or assume they are given to build the ontology, with only Methonotology with dedicated phase for defining concrete specifications. HOD²LC also has a focused *specification phase* primarily for defining boundaries and additional for analyzing the developed ontology domain model. The *Design phase* has varied importance across alternatives, achieved in Methonotology, EO Project, and TOVE, primarily centered on developing domain models only by using metamodels such as UML, RDF/RDFS, OWL, etc., HOD²LC uses a layered architecture (Figure 4) and FDD (Figure 6) for designing domain models. *Analysis phase*, a high priority in HOD²LC to validate the domain model with specification phase, has limited consideration by other the alternatives. The *implementation* phase is not taken into consideration in all but Uschold and Noy with the ontology realized at some point, without a consideration of timing. All alternatives have no formal *testing*, while HOD²LC uses various frameworks for evaluating and testing the realized ontology. Various alternatives take different approaches for maintaining and documenting the developed ontology, as there are no set standards for executing them.

Table 1: Comparison of Ontology Alternatives vs. Phases.

Phases	Six Ontology Process Alternatives						
	Methontology	Fernandaz	EO Project	TOVE	Uschold	Noy	HODLC
Problem Analysis	Partial	Full	Full	Full	Full	Full	Full
Ontology Integration	None	None	Partial	Full	None	Partial	Full
Knowledge Acquisition	Full	Full	Full	Full	Full	Full	Full
Specification	Full	None	Partial	None	None	Partial	Full
Design	Partial	Partial	Full	Full	Full	Full	Full
Analysis	None	None	Partial	None	None	None	Full
Implementation	Full	None	Full	Partial	Partial	Full	Full
Testing	None	None	None	None	None	None	Full
Maintenance/ Documentation	Partial	None	Partial	None	None	None	Full

5 Conclusion

In this paper, we have outlined a robust software engineering development life cycle model to build ontology models and its vocabulary by studying the various phases of Software Development Process (SDP) models and assessing them to find the best fit methodology. Using this work, we have proposed Hybrid Ontology Design and Development Life Cycle (HOD²LC) model in Section 2 that contains nine phases in an iterative and incremental development process, leveraging concepts from SDP models and other techniques, e.g., *Heuristic Classification* in Phase 1, *MPM* model and *FDD Methodology* in Phase 4 and *analysis phase* as Phase 5 To more fully understand HOD²LC, in Section 3, we presented five alternative research efforts [11-17] on ontology design and development and studied and evaluated them against both SDP models and phases, and our own HOD²LC. The end result is a detailed qualitative comparison of the five alternatives vs. HOD²LC which is summarized in Table 1 of Section 4. We believe that HOD²LC as presented

herein represents an encompassing ontology design and development process that transcends existing alternatives through a leveraging of SDP process concepts that results in an approach that creates ontology solutions in a more structured and rigorous manner.

6 References

[1] F. Baade, D. Calvanese, D. L. McGuinness D. Nardi and P. F. Patel-Schneider, "The Description Logic Handbook: Theory, Implementation, and Applications", 1st Ed., 2003.

[2] R. Saripalle, S. Demurjian, and S. Berhe, "Towards Software Design Process for Ontologies," *Intl. Conf. on Software and Intelligent Information (ICSII 11)*, Puerto Rico, October, 2011.

[3] L. Pazzi, "Three Points of View in the Characterization of Complex Entities", *Formal Ontology in Information Systems*. IOS Press, 1998.

[4] K. Knight and S. Luk, "Building a Large Knowledge Base for Machine Translation". *Proc. of American Association of Artificial Intelligence Conference (AAAI-94),* Seattle, 1994.

[5] K. Mahesh, "Ontology Development for Machine Translation: Ideology and Methodology", 1996.

[6] S. Demurjian, R. Saripalle and S. Berhe, "An Integrated Ontology Framework for Health Information Exchange," *Intl. Conf. on Software Engineering and Knowledge Engineering (SEKE 09)*, August, 2009.

[7] M. Docherty, "Object-oriented Analysis and Design: Understanding System Development with UML 2.0", 1st Ed., 2005.

[8] B. Boehm, "A Spiral Model of Software Development and Enhancement", *ACM SIGSOFT Software Engineering Notes*, Vol. 11(4), pp. 14-24, 1986.

[9] C. Larman, V. R. Basili, "Iterative and Incremental Development: A Brief History", *IEEE Computer Society*, Vol. 36 (6), pp. 47–56, 2003.

[10] L. Craig, "Agile and Iterative Development: A Manager's Guide", Addison-Wesley, 1st Ed., 2003.

[11] M. Fernández-Lopez, A. Gomez-Perez and N. Juristo, "METHONTOLOGY: from Ontological Art towards Ontological Engineering", *Proc. of AAAI Spring Symposium*, pp. 33-40, 1997.

[12] A. Gómez-Pérez, M. Fernández and A. J. de Vicente, "Towards a Method to Conceptualize Domain Ontologies".

[13] M. Uschold and M. King, "Towards a Methodology for Building Ontologies", *In Workshop on Basic Ontological Issues in Knowledge Sharing. International Joint Conference on Artificial Intelligence*, 1995.

[14] Uschold M, "The Enterprise Ontology", *Knowledge Engineering Review*, Vol. 13(1), 1998.

[15] M. Grüninger, M. Fox, "Methodology for the Design and Evaluation of Ontologies", *Proc. of IJCAI95's Workshop on Basic Ontological Issues in Knowledge Sharing*. 1995.

[16] M. Uschold, "Building Ontologies: Towards a Unified Methodology". *16th Annual Conf. of the British Computer Society Specialist Group on Expert Systems,* pp. 16-18, 1996.

[17] N. Noy and L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology". *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05*, 2001.

[18] W. J. Clancey, "Heuristic classification", *Journal in Artificial Intelligence,* Vol. 27(3) pp. 289-350, 1985.

[19] S. Liu W. Ma, R. Moore, V. Ganesan and S. Nelson, "RxNorm: prescription for electronic drug information exchange", *IT Professional*, Vol.7(5), pp. 17-23, 2005.

[20] V. Kashyap and A. Borgida, "Representing the UMLS Semantic Network Using OWL: (Or "What's in a Semantic Web Link?")", *Second Intl. Semantic Web Conf.*, 2003.

[21] C. Rolland, and N. Prakash and A. Benjamen "A Multi-Model View of Process Modeling". *Journal of Requirements Engineering,* Vol. 4 (4), pp. 169-187, 1999.

[22] M. Jarrar, J. Demy and R. Meersman, "On Using Conceptual Data Modeling for Ontology Engineering." *Journal on Data Semantics Special issue on Best papers from the ER/ODBASE/COOPIS 2002 Conferences*, Vol. 1(1), pp. 185-207, 2003.

[23] S. R. Palmer and J. M. Felsing, "A Practical Guide to Feature-Driven Development", Prentice Hall, 1st Ed., 2002.

[24] A. Gomez-Porez, N. Juristo and J. Pazos, "Evaluation and Assessment of Knowledge Sharing Technology", *Towards Very Large Knowledge Bases*, pp. 289-296, 1995.

[25] A. Gomez-Porez, "A Framework to Verify Knowledge Sharing Technology", *Expert Systems with Application*, Vol. 11(4), pp. 519-529, 1996.

[26] UML Profile, http://www.omg.org/technology/documents/profile_catalog.htm

[27] P. Clark, J. Thompson, and B. Porter, "Knowledge Patterns", Handbook of Ontologies, pp. 191-207, Springer, 2003.

[28] ICD Codes, http://icd9cm.chrisendres.com/index.php

[29] FOAF Ontology, http://www.foaf-project.org/

[30] Aditya Kalyanpur, "Debugging and Repair of OWL ontologies", PhD Thesis, University of Maryland, 2006.

[31] W. J. Clancey, "Heuristic classification", *Journal in Artificial Intelligence,* Vol. 27(3), pp. 289-350, 1985.

Two-step Role-Based Access Control method for Ontology Storage

Sukhoon Lee[1], **Jangwon Kim**[1], **and Doo-Kwon Baik**[1]
[1] Dept. of Computer Science and Radio Communications, Korea University, Seoul, Republic of Korea
leha82@korea.ac.kr, ikaros1223@korea.ac.kr, baikdk@korea.ac.kr

Abstract - *Because of the advance of ontology technology and the explosion of web ontology, ontology repository has become a necessity. As the information security issue, research for adapting access control to the ontology storage has been studied, and there are two approaches which are model-driven approach and query rewriting approach. To solve the problems which these approaches have, this paper proposes two-step Role-Based Access Control method and describes a system architecture applying proposed method. The proposed method is efficient and reliable compared by the typical approaches.*

Keywords: Access Control, Ontology Storage, Role-based Access Control

1 Introduction

the concept of Semantic Web was appeared by Tim Berners-Lee, ontology technology has been advanced to represent some knowledge or information [1]. Especially by W3C, the web ontology has been developed to represent ontology for web pages such as OWL/RDF [2, 3], and an amount of web ontology is increased explosively like web pages does. Due to processing and managing more ontology fast, the ontology storage which uses database system has been developed such as Jena [4], KAON [5].

Meanwhile, the web ontology has security issues like, is it validate that some information could access for a user? Because access control could not only restrict some systems, but also some information, it is suitable technology for the ontology storage of security issues that some information has to restrict to some users[6].

The researches for adapting access control techniques to the ontology storage have two approaches such as Model-driven approach and Query rewriting approach.

First, Model-driven approach executes SPARQL query in the ontology storage. As a result of query execution, ontology model is created. After inference of this model, the access control policies are adapted [7, 8]. This approach builds ontology model in memory as the result of query. It cause much cost and time because some ontologies are useless so that the ontologies are filtered in access control process. If a huge amount of ontology is loaded in memory as a query result and most of them are filtered in access control process, it takes much cost and time.

Query rewriting approach is a method of rewriting the SPARQL query before query execution in the ontology storage. The query rewrites to allow the information to user by authority [9-11]. This approach just rewrites query, so it is easy to adapt to any type of ontology storage which is able to SPARQL query. But after the query execution, the access control process is not supported about inferred ontology. Although this approach does not load useless ontology, it does not completely guarantee the privacy after ontology inference.

There are several the access control techniques. As one of them, Role-based Access Control (RBAC) divides users by roles, pairs objects with operations (access or deny) as permission, and define policies to pair of roles and permissions [12]. Because RBAC identifies users in web ontology and makes available access not only systems but also information, it is suitable technique to define policy for access control.

To solve pre-mentioned two problems, this paper proposes two-step role-based access control for ontology storage, implements and adapts to the ontology storage.

After this section, Section 2 describes two-step role-based access control method, and Section 3 describes the system architecture. Finally, Section 4 presents conclusion with evaluation.

2 Two-step Role-Based Access Control Method

The proposed method uses two-step access control process to deal with limitation of typical access control methods.

In the first step, it uses query rewriting method adapted policies by roles before query is executed in ontology storage. This step prevents loading the useless ontology in memory.

In the second step, it processes second access control after ontology model is expanded by inference. Without the second step, it does not guarantee the privacy about expanded ontology by inference, so the second step is necessary for reliability.

Figure 1 shows two-step Role-Based Access Control process as a proposed method.

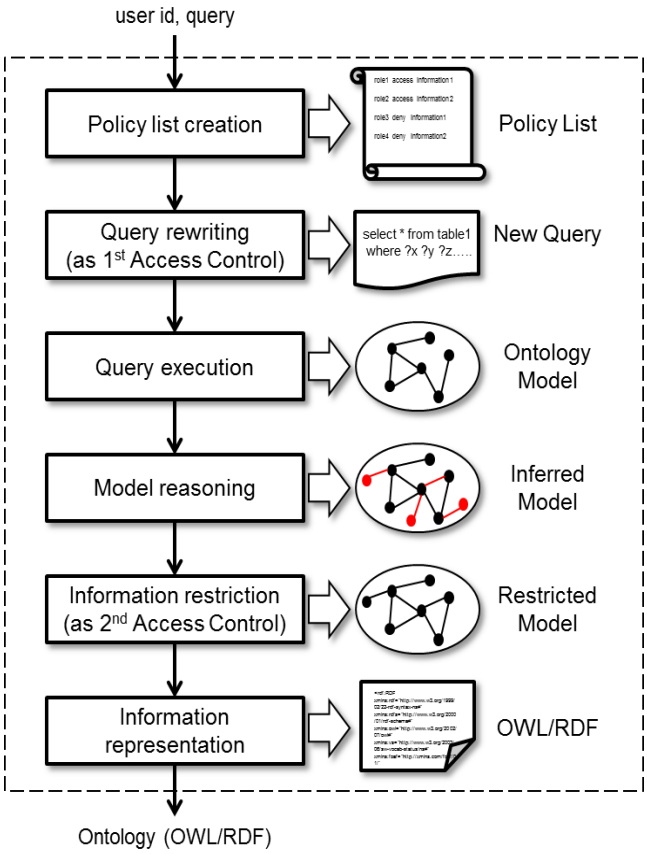


Figure 1 – 2-step Role-Based Access Control process

2.1 Policy list creation

This process validates the policies which role of user has using user ID, and makes policy list consist of the policies which role of user has.

2.2 Query rewriting

This process rewrites given SPARQL query for adapting policy list which is made by previous process. As the first access control process, the inaccessible information from user is filtered in query level. So it is able to use resource efficiently because the useless ontology is filtered before model creation.

2.3 Query execution

This process creates ontology model by execution rewritten query. The ontology model is created in memory.

2.4 Model reasoning

This process expands the ontology model using inference. At this time, the inferred ontology may include inaccessible information from user.

2.5 Information restriction

This process does the second access control work using policy list which is made in first process. Because this process checks the policies about an expanded part of information, it guarantees the privacy about the inferred ontology.

2.6 Information representation

This process represents the ontology model which is formed by graph type in memory as ontology language like OWL/RDF due to supporting.

3 System Architecture

Figure 2 shows proposed system architecture. The proposed system consists of a proposed system and a database system. The proposed system inputs a SPARQL query from user, and communicates with database system to process the information.

3.1 Database System

Database system includes RBAC which works access control based on role and Ontology Storage which administer and store ontologies.

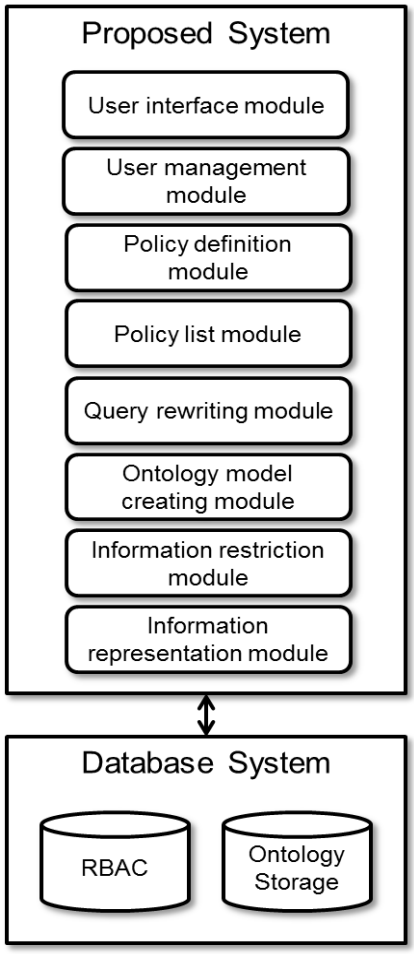


Figure 2 – Proposed System Architecture

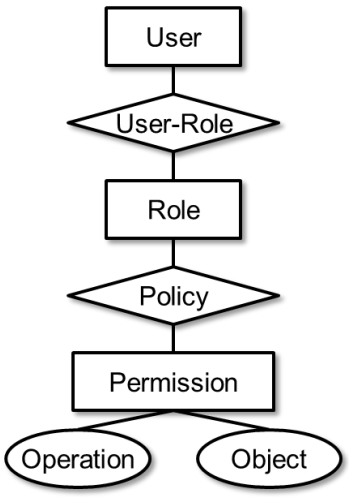


Figure 3 – RBAC model

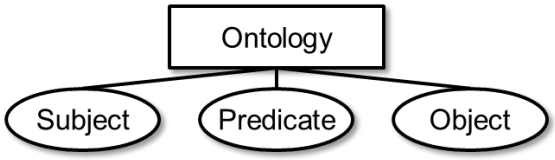


Figure 4 – Ontology Storage model

Figure 3 shows database model of RBAC. The policy defined in the RBAC consists of a pair of role and permission. Each user has a role and a permission defines whether some object is able to be accessed (operation) or not by user.

Figure 4 shows database model of Ontology Storage. The Ontology Storage is constructed to store triple statement. The triple statement consists of Subject, Predicate, and Object.

3.2 Proposed System

The proposed system consists of every modules in Figure 2 for input queries from user and two-step RBAC method.

- **User Interface module:** a module for user input/output. It receives user ID and SPARQL query from users, and return ontology in OWL/RDF type.
- **User management module:** a module for administration of user login and user information. It defines relation associated by User_Role in RBAC.
- **Policy definition module:** a module for defining a policy about a role. It defines relation associated by Permission in RBAC.
- **Policy list module:** a module for building policy list as searching policies which have role and permission pair by role of user.
- **Query rewriting module:** a module for rewriting SPARQL query. It rewrites query using policy list built in pre-process.
- **Ontology module creation module:** a module for executing query, creating ontology model, and reasoning the model. The ontology model loads in memory.
- **Information restriction module:** a module for the second access control of the ontology model using policy list in memory

- **Information representation module:** a module for representing ontology as ontology language like OWL/RDF to support ontology model to user.

4 Conclusions

This paper proposed and implemented the two-step RBAC method for overcoming the limitation of typical access control methods.

Table 1 shows a qualified evaluation of the two-step RBAC method compared with typical access control methods. The proposed method can faster and more efficient access control work than Model-driven access control method. And the proposed method have additional process as the second access control work for guarantee reliability about inferred ontology after query rewriting method.

Table 1 – Qualified evaluation of Two-step RBAC mehtod

	Model-driven	Query Rewriting	Two-step RBAC
query performance	slow	fast	fast
resource usage	high	low	Low
reliability	reliable	unreliable	reliable

As the result, in an environment that administers and represents web ontology explosively increasing, the proposed method efficiently and reliably supports information when access control works.

In further works, we experiment useful dataset for implementing the proposed method, and determines performance and reliability for a qualified evaluation between the proposed method and the typical methods.

5 Acknowledgement

This study was supported by Second Brain Korea 21 Project and by the National IT Industry Promotion Agency (NIPA) under the program of Software Engineering Technologies Development. The corresponding author is Doo-Kwon Baik.

6 References

[1] Tim Berners-Lee, James Hendler, and Ora Lassila. “The Semantic Web”; Scientific American Magazine, March 26, 2008.

[2] http://www.w3.org/2004/OWL/. “Web Ontology Language (OWL)”. W3C, 2004.

[3] http://www.w3.org/RDF/. “Resource Description Framework (RDF)”. W3C, 2004.

[4] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. “Jena: Implementing the Semantic Web Recommendations”; Technical Report HPL-2003-146, Hewlett-Packard Labs, 2003.

[5] E. Bozak, M. Ehrig, S. Handschuh, A. Hotho, A. Maedche, B. Motik, D. Oberle, C. Schmitz, R. Studer, G. Stumme, Y. Sure, S. Staab, L. Stojanovic, N. Stojanovic, J. Tane, R. Volz, and V. Zacharias. "KAON - Towards a large scale Semantic Web”; in Proc. of EC-Web 2002, LNCS, 2002.

[6] Amit Jain and Csilla Farkas. “Secure Resource Description Framework: an Access Control Model”; in Proc. of the eleventh ACM symposium on Access control models and technologies (SACMAT '06), 2006.

[7] Wei-Tek Tsai, and Qihong Shao. “Role-Based Access-Control Using Reference Ontology in Clouds”; 2011 Tenth International Symposium on Autonomous Decentralized Systems, pp121-128, 2011.

[8] Lorenzo Cirio, Isabel F. Cruz, and Roberto Tamassia. “A Role and Attribute Based Access Control System Using Semantic Web Technologies”; in Proc.of the 2007 OTM Confederated international conference on On the move to meaningful internet systems, Vol.2, 2007

[9] Fabian Abel, Juri Luca De Coi, Nicola Henze, Arne Wolf Koesling, Daniel Krause, and Daniel Olmedilla. “Enabling Advanced and Context-Dependent Access Control in RDF Stores”; in Proc. of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference (ISWC'07/ASWC'07), 2007.

[10] Giorgos Flouris, Irini Fundulaki, Maria Michou, and Grigoris Antoniou. “Controlling Access to RDF Graphs”; Third Future Internet Symposium (FIS 2010), LNCS 6369, pp. 107–117, 2010.

[11] Alban Gabillon and Léo Letouzey. “A View Based Access Control Model for SPARQL”; 2010 Fourth International Conference on Network and System Security, pp105-112, 2010.

[12] David F. Ferraiolo. "Proposed NIST Standard for Role-Based Access Control"; ACM Transactions on Information and System Security, Vol. 4, No. 3, pp.224–274, 2001.

SESSION

WEB SERVICES AND APPLICATIONS

Chair(s)

TBA

A Platform for Discovery and Execution of Semantic Web Services Compositions

Guilherme C. Hobold
Department of Informatics and Statistics
Federal University of Santa Catarina
Florianópolis, SC – Brazil
guich@inf.ufsc.br

Frank Siqueira
Department of Informatics and Statistics
Federal University of Santa Catarina
Florianópolis, SC – Brazil
frank@inf.ufsc.br

***Abstract*— Semantic descriptions provide more accurate information related to operations supported by Web services, enabling their dynamic discovery and execution without human intervention. Furthermore, semantic descriptions allow Web services to be automatically combined by using discovery mechanisms able to identify composed services. These compositions can also be described and published as if they were a single service, allowing service consumers to discover and invoke a composition transparently. This paper presents a platform for automatic discovery and execution of semantic Web services compositions. A composer mechanism identifies semantic Web services compositions based on the information annotated on service descriptions using SAWSDL (Semantic Annotations for WSDL). The identified service compositions are described and published as a single service. The interaction among services is described using WS-BPEL (Web Services Business Process Execution Language) and executed by a BPEL engine.**

Keywords- Semantic Web Services, Web Service Compostion, Web Service Discovery, SAWSDL.

I. INTRODUCTION

The current hype around SOA (Service-Oriented Architecture) and SaaS (Software as a Service), combined with the ubiquity of the World Wide Web, have contributed for the widespread availability of Web Services. This technology, despite enabling the integration of computing systems in heterogeneous environments, still requires human intervention during the integration process, given that only humans are able to infer the semantics behind data and operations provided by different services.

In this context, combining two or more services is often necessary in order to accomplish a required task. However, the assembly of service compositions is not favored by the syntactic nature of the description language employed for service description – i.e., WSDL (Web Services Description Language) [1] – which limits itself to associate keywords with operations and messages exchanged by services.

The Semantic Web Services (SWS) technology has been advocated as a solution for enabling system integration without human intervention. Semantic languages such as OWL-S (Web Ontology Language for Web Services) [2], WSML (Web Service Modeling Language) [3] and SAWSDL (Semantic Annotations for WSDL) [4] provide resources for describing the semantic meaning of messages and operations provided by Web Services. Furthermore, such information may be explored to automatically discover, compose and execute services.

This paper describes a platform for discovery and execution of Semantic Web Service compositions. This platform comprises a mechanism for dynamically identifying compositions based on semantic descriptions of services, and an engine for execution of compositions, which are invoked transparently as a single service.

The remainder of this paper is organized as follows. Section II discusses the use of semantic languages for Web Services description, the processes of web service discovery and composition, and the related work in this field of research. Section III presents the platform for automatic semantic web services discovery, composition and execution. Section IV presents the prototype of the platform and analyzes its performance. Section V concludes the paper, summarizing the contribution brought by this work, comparing the proposed platform with other research works and describing potential improvements to this work.

II. STATE-OF-THE-ART

This section presents the available technology to describe Semantic Web Services and to discover and execute compositions, as well as recent research efforts in this area.

A. Semantic Description

Semantic description languages, such as OWL-S, WSML and SAWDSL provide means for describing Web Services based on ontologies. Both OWL-S and WSML create semantic descriptions dissociated from the syntactic description given by WSDL. On the other hand, SAWSDL, which is the most recent W3C recommendation for SWS description, allows annotations to be added to WSDL elements in order to associate them with concepts defined in a domain ontology.

SAWSDL is language-neutral, since it allows any semantic-capable language to be employed for defining domain ontologies. Besides, it is more flexible and easier to use than the other languages, because it requires only the addition of semantic annotations to WSDL descriptions [5].

SAWSDL also lets behavioral constraints to be specified in ontologies and referenced by inputs and outputs through annotations, in order to associate pre- and postconditions with operations provided by a web service.

B. Discovery of Semantic Web Services

Semantic descriptions allow discovery mechanisms to interpret the meaning of functionalities exposed by web services. This process requires inferences to be performed based on ontologies that describe elements contained in

service descriptions. These mechanisms are able to compare the functionalities exposed by the existing semantic services with the requirements specified in a discovery procedure, selecting services that semantically match the specified requirements.

The matching process requires each operation under analysis, as well as its inputs and outputs, to be compared with the desired operation, including the available inputs and the required outputs, which were specified in the discovery procedure. This comparison is based on the concepts defined in ontologies associated with operations, input and output messages, in the aim of identifying semantic matches [6].

The similarity between two concepts may be evaluated using the criteria proposed in [7], which specifies an accuracy degree based on how similar concepts are. The similarity analysis takes into account any existing hierarchical relationship, as well as the existing properties and other characteristics defined in the domain ontology [6]. During this evaluation, the concepts under analysis are compared and classified into one of the similarity degrees defined in table I.

TABLE I. SIMILARITY DEGREES

Class	Description	Value
Exact	Concepts being compared are identical	0
Plugin	A supertype of the concept is available	1
Subsume	A subtype of the concept is available	2
Fail	Concepts are not hierarchically related	3

C. Semantic Web Services Composition

A composition can be defined as a group of services that, by working together, are able to cooperatively accomplish a task, and therefore can be seen as a new service. A composition is necessary when a single service that executes the whole task is not available [8].

The assembly of compositions requires understanding messages and operation semantics. Dealing with behavioral constraints, requirements and results provided by each service involved is also mandatory. [9]. The composition process may rely on user interaction, resulting in non-automated approaches, or on semantic technologies, which allow the process to be partially or fully automated. During this process, new services may be dynamically discovered and added to the composition. This process is repeated as many times as needed, until the result obtained by the composition matches the required outputs and behavior. It is also possible to give up searching based on a stop criterion (e.g., reaching the maximum number of services allowed in a composition) or when no more services able to take part in the composition are found.

D. Execution of Composed Services

The execution of service compositions, either semantically described or not, may be accomplished by adopting two different strategies: orchestration or choreography.

The first strategy relies on an orchestrator, which is responsible for invoking each service in a composition and for handling inputs and outputs. The WS-BPEL (Web Services Business Process Execution Language) standard allows compositions to be described and executed by orchestration engines [10].

The second strategy, on the other hand, relies only on the services themselves to collaboratively execute the composed service. The WS-CDL (Web Services Choreography Description Language) standard defines how choreographies can be described and executed [11].

E. Related Work

In the past few years, the research on semantic web technologies has grown constantly. Different approaches for semantic web service discovery and execution of compositions have been proposed. A representative share of these research works is described in this section.

Puntheeranurak and Tsuji [12] propose a discovery mechanism that executes a matching algorithm which relies on semantic information obtained through SAWSDL. The idea behind the proposed mechanism is to compare the functional requirements of the requested service with the available services and identify levels of similarity among them. The semantics associated with the description of the services through SAWSDL is employed to make inferences. Inputs and outputs are compared based on subsumption reasoning. The algorithm proposed in [12] does not discover compositions, but just individual services.

Prazeres et al [6] propose a solution based on OWL-S and on an algorithm that implements a minimum cost policy for discovery and composition of semantic web services. Services published in a UDDI registry are associated with a cost, which is calculated based on the amount of inputs and outputs of the corresponding service. A graph is created in order to allow compositions to be identified. The algorithm does not take into account the inputs available at request time. When a request is made, the graph is updated to take into account the available inputs, and the cost is adjusted. Discounts are given on the cost of a service if it has inputs that are currently available. Finally, compositions that have the minimum cost are selected. However, even with the discounts, the selected compositions may require inputs that the client does not have. Besides, the algorithm uses the available inputs to select only the first service of the composition, and does not combine them with outputs produced by other services to be used as inputs to new ones. This is done because the graph is not created at request time, aiming to reduce the search time needed to find a composition. Services with the same inputs and outputs but different semantics are not distinguished by the algorithm.

Mehandjiev et al. [13] present a semi-automatic approach based on templates for assisted web service composition. Through a tool that abstracts the technical details and the data flow between services, users select a template that conforms to their needs and start building the composition. The tool provides options for selecting services and highlights those that are compatible through the analysis of the semantics associated with the inputs, outputs, pre- and post-conditions of each service. The discovery process requires user assistance and occurs at design time.

Finally, the work of Belouadha et al. [14] proposes an approach for describing and composing semantic web services based on UML (Unified Modeling Language). The semantic description of services is given by a UML language-independent semantic metamodel. Later, this metamodel is transformed in a WSDL annotated with SAWSDL using transformation rules. At last, the authors use BPMN (Business Process Modeling Notation) for modeling web services flows and for defining the execution plan of the service composition. The authors create compositions statically, and the process is not automatic with respect to the creation of compositions.

III. Composition and Execution Platform

This work targets the design and development of a platform that allows automatic composition and execution of semantic web services. In this platform, SAWSDL annotations are extracted from WSDL files and employed to build composition graphs, in which compositions are discovered based on service semantics. The composition can be invoked by clients through a single request, ignoring the fact that the request will be fulfilled by multiple services.

A. Architecture

The platform was designed to be both programming language and platform-independent. Its architecture has 9 components, which are shown in Fig. 1.

The *Web Service* component exposes 3 operations that are invoked by clients to request a specific service, to publish a new web service or to execute a service composition. As a result, the first operation returns the discovered composition, the second stores the service into the repository and the third returns the response produced by executing the composition.

The *Semantic Annotations Extractor Module* is responsible for extracting semantic annotations from WSDL descriptions during the publishing process and for storing them in a relational database model, represented by the *SAWSDL Repository*. This strategy is adopted in order to optimize the discovery process, since it is faster than parsing WSDL descriptions every time a request is made. This module is also responsible for retrieving annotations from the SAWSDL Repository during the discovery process.

The *Discovery and Composition Module* is responsible for actually discovering the web services compatible with the request requirements. Based on the annotations extracted previously and on the parameters specified in the request, the semantic matching is performed to verify its similarity with the requested service. This module builds a composition graph based on the relationships among concepts associated with inputs, outputs and operations. At the end of this process, the discovered compositions are stored into the *Compositions Repository* and returned to the client through the Web Service.

The result of a request may contain none, one or multiple paths leading to the desired outputs. Multiple paths will be returned when each one meets part of the request. If two paths that lead to the same output are found, just one will be selected and returned based on a proposed criterion, which will be described in section III.E.

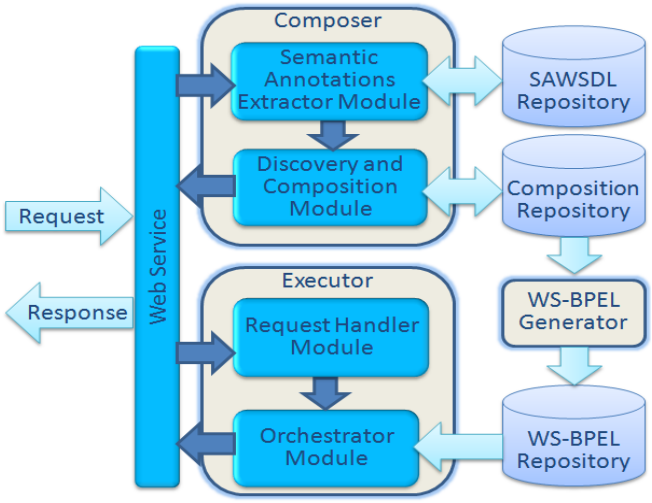


Figure 1. Platform architecture.

As soon as they are discovered, compositions are processed by a *WS-BPEL Generator*. This component retrieves compositions stored in the Composition Repository, generates WS-BPEL files that describe these compositions and stores them in a *WS-BPEL Repository*.

Finally, the client can invoke a composition through a single request directed to the Web Service. The request will be dispatched by a *Request Handler Module*, which manages the ongoing requests. This module activates the *Orchestrator Module*, which executes the corresponding WS-BPEL file. The outputs produced by the composition are then returned to the client.

B. Discovery Request

To locate a service composition, the client must invoke an operation exposed by the Web Service component. This operation requires six parameters: *availableInputs*, *desiredOutputs*, *desiredOperations*, *maximumDepth*, *timeout* and *allowRebuild*. The *availableInputs*, *desiredOutputs* and *desiredOperations* are lists of URIs (Uniform Resource Identifiers) that refer to concepts defined in domain ontologies. The *maximumDepth* determines how many services that, sequentially, may be present in each path of the compositions graph. A composition with a sequential execution flow of three services has three depth levels. The *timeout* parameter corresponds to the time interval the client is willing to wait for the result of the request. Finally, the parameter *allowRebuild* specifies whether previously found compositions can be rebuilt or not (this will be further explained in section III.F).

C. Discovery Algorithm

Fig. 4 shows the pseudo-code of the discovery algorithm, which takes as input three lists of URIs: *availableInputs*, *desiredOutputs* and *desiredOperations*. The services stored into the repository are recovered and used with the available inputs to match the startup services (lines 4-5). Next, the target services are selected based on the desired outputs and operations (lines 6-7).

```
1  Function discoverComposition(availableInputs,
                    desiredOutputs, desiredOperations)
2    Var startupServices, targetServices,
         singleServices, pathList, bestPaths := ∅
3    Foreach service ∈ serviceRepository Do
4      If semanticMatch(availableInputs, getInputs(service))
5        startupServices ← service
6      If semanticMatch(desiredOutputs, getOutputs(service))
         And semanticMatch(service, desiredOperations)
7        targetServices ← service
8    End Foreach
9    singleServices ← startupServices ∩ targetServices
10   Foreach output ∈ desiredOutputs Do
11     pathList := getBestService (singleServices, output)
12     pathList ←findPaths(startupServices, targetServices, output)
13     bestPaths ← getBestPath (pathList, output)
14   End Foreach
15   Return bestPaths
16 End.
```

Figure 4. Algorithm for Composition Discovery

Subsequently, the algorithm identifies single services that are both startup and target services, i.e., are able to produce desired outputs (line 9). Then, for each desired output, the algorithm selects single services (line 11) and paths linking startup and target services that produce the given output (line 12), and selects the best path or service leading to the output (line 13). Finally, the best paths are returned as result of the request (line 15).

D. Startup and Target Service Selection

The discovery algorithm described in the previous section takes into account the semantic annotations associated with inputs, outputs and operations of a WSDL. It calculates the similarity between these requirements and the available services via subsumption reasoning.

Annotations added to the operation element of a WSDL description allow expressing the semantic meaning of each operation provided by the corresponding service. Thus, it is possible to distinguish web services that have the same inputs and outputs, but have different semantic meaning.

Suppose, for example, an operation that informs the wine brands manufactured in a given country, and another one that returns the most sold wines in a country. Despite the fact that their input and output messages match semantically, these operations are functionally different. For this reason, annotations associated with operations must be taken into account during the matching process. In the given example, each operation will be associated with a different ontological concept that specifies its purpose, allowing the matching process to distinguish each other.

Based on the classification proposed in [7], the semantic match is given by:

$$match(A, B) = \{x \mid x \in (exact, plugin, subsume, fail)\}$$

From the result of the comparison between two concepts A and B, the algorithm assigns a value given by Table I.

The semantic matching of startup services selects services from the SAWSDL repository that require inputs which semantically match the available inputs specified in the discovery request. For being selected, all inputs of the web service must be classified as exact or plugin. The condition for service selection is given by:

$$\forall x \in In : M_x <= 1$$

where:

- In: set of service inputs;
- M_x: value resulting from the match of input x of the service, according to Table I.

The semantic matching of target services, on the other hand, takes into account not only if the service produces desired outputs, but also if it has at least one operation that semantically matches one of the desired operations. First, the outputs of the selected operations are compared with the desired outputs. If in the list of requested outputs there is a match with value lower or equal to 1, the web service is then selected. Then, during the semantic matching of operations, the algorithm selects services that have at least one operation similar to one of the desired operations. Thus, the matching is performed for each operation provided by services with the desired operations. If the list of desired operations has one with a similarity degree lower than or equal to 1, the web service is then selected.

At the end of this step, only services that have operations classified as exact or plugin, which also have outputs classified as exact or plugin, will be selected as target services. Therefore, the output is strictly related to the operation semantics, since the web service must be selected in the operation matching and in in the output matching.

After performing these steps, the algorithm checks if any service was selected in both inputs and outputs matching. If this occurs, a service that provides at least one of the desired outputs based on the available inputs was found. In the sequence, the algorithm tries to find paths linking startup services to target services, building a composition graph.

E. Composition Graph

A composition is characterized by paths in the graph linking a startup service to a target service, producing at least one desired output. Thus, we know from where to start and where to stop. Each node of the graph is represented by a web service and the edges are the semantic links between them.

Before starting to build the composition graph, the outputs of the startup services, now used as inputs, are added to the list of available inputs, initially composed by the inputs sent in the request.

The algorithm uses layers of services, as shown in Fig. 2, to represent the composition graph. The first layer is composed by the startup services. The other layers emerge as the algorithm iterates over the other services. Each iteration compares semantically the inputs required by the available services with the available inputs. The selected services form a new layer and their outputs are added to the list of available inputs for use in subsequent iterations. Thus, new paths may be created in the graph at each iteration.

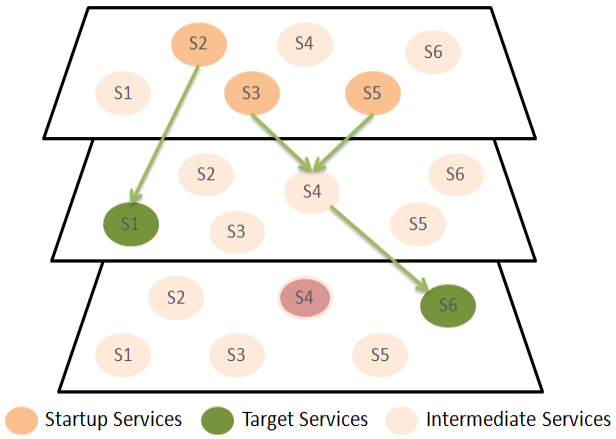


Figure 2. Composition graph

Services of a layer are linked to services of the top layer through the origin of their inputs. As illustrated in Fig. 3, the inputs of service Z may come from service X or from inputs specified in the request. Eventually, the inputs of a service may come from one or more services, such as occurs with service Z, depicted in Fig. 3.

As the algorithm progresses, there are several paths in the graph that can result in different desired outputs. Each path may or may not find an operation that meets the request. Once services are selected to compose a layer, the algorithm checks if any of them offer a desired operation and output. If so, it means that a path to an output was found. Nevertheless, the path of the graph continues being built looking for new outputs. A path is interrupted only when it is not possible to establish a semantic link with other services or when a service appears repeatedly in the same path, featuring a loop, or when the path reaches the depth specified in the request.

For the algorithm, each path in the graph leads to a single output, even if multiple requested outputs are in the same path in the graph. Fig. 3 shows such a situation, where services Z and T provide requested outputs and, despite being on the same path in the graph, the algorithm treats them as two different paths – one composed by services Y → Z and the other by services Y→ Z → T.

At the end of the graph construction, either by a timeout or because all the requested outputs were found, the algorithm examines whether two or more paths lead to the same output. In this case, the following function is applied to each path in order to select the one with the lowest semantic mismatch degree:

$$SMD(P) = \sum_{i=1}^{Ns} Min_s * \alpha + Mout_c * \beta + \sum_{i=1}^{Ns} Mop_s * \omega$$

where:

- SMD(P): Semantic Mismatch Degree of Path C;
- Ns: Number of services in the path.
- Min_s: Value resulting from the semantic matching of inputs of each operation in the path, according to Table I;
- $Mout_c$: Value resulting from the semantic matching of the output of the path, according to Table I;
- Mop_s: Value resulting from the semantic matching of each operation in the path, according to Table I.

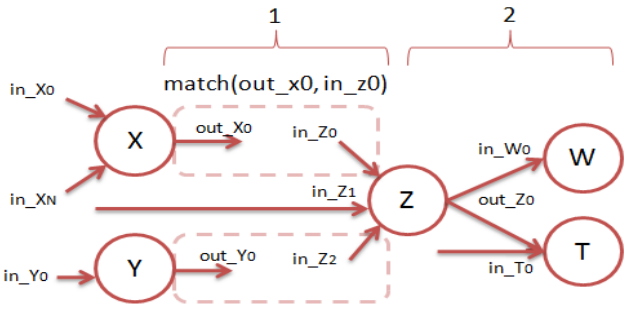


Figure 3. Semantic link between web services.

- α: Weight associated to the inputs;
- β: Weight associated to the outputs;
- ω: Weight associated to the operations;

The SMD(P) function takes into account the result of the matching inputs and operations of all services belonging to the path, but only considers the matching of the output of the last service of the path. This is because the outputs that serve as inputs to others are already considered in the Min value. Moreover, the function applies different weights to the evaluated criteria. The weight distribution occurs according to the importance of each criterion.

If after applying the function still exist two or more paths with the same value, a fourth criterion is applied:

$$Cost(P) = SMD(P) + depth$$

where:

- Cost(P): Cost of path P;
- SMD(P): Semantic Mismatch Degree of path P;
- depth: Depth of the path.

In order to prioritize paths with better semantics, the depth criterion is considered only in a second stage. The depth is an important criterion of composition quality since the larger the number of services that integrate the composition, the greater is the probability of one of them becoming unavailable and impeding the execution of the composition.

Once calculated the cost of each path, the algorithm selects the path with the lowest SMD leading to each output. This means that the path that provides the best similarity degree of their inputs, output and operations, along with the lower depth, will be selected. This calculation is performed only when there are two or more paths that provide similar outputs. Otherwise, calculating these metrics is unnecessary and the single path obtained is selected directly.

At the end, the SMD(C) function is applied to the final composition, resulting from the union of all paths, in order to calculate the global SMD according to the following function:

$$SMD(C) = \sum_{i=1}^{N} SMD(P_i)$$

where:

- SMD(C): Semantic Mismatch Degree of composition C;
- SMD(P): Semantic Mismatch Degree of path P_i;
- N: Number of paths in composition C.

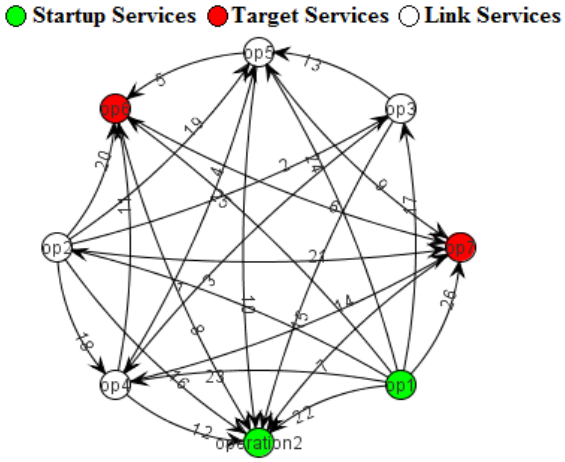


Figure 5. Composition resulting from the experiment.

F. *Composition Repository*

All compositions discovered as result of a request are stored in the Compositions Repository for future requests. Before making the whole discovery process when a request for a new service is received, the platform verifies the existence of any composition that meets the request needs. If it exists, this composition is returned to the requester just after a validation.

Validation of compositions consists in checking the availability of services involved. This process is necessary because web services can be removed from the repository. If all services remain available, the composition is returned; otherwise, a new discovery process begins.

There is also the case in which new web services become available and, in some way, improve the existing compositions. In such cases, when requesting a service that already has a composition available, the algorithm checks if any new service was published after the creation of the composition and reexecutes the discovery process, in order to identify any eventual improvement. However, this reconstruction can be prevented through the *allowRebuild* parameter sent in the request. If the value specified in this parameter is true, the reconstruction of the composition is performed; otherwise, the existing composition is returned.

IV. Implementation and Evaluation

The discovery algorithm was implemented in Java, using JDK 1.6.0. Semantic annotations are extracted using the EasySAWSDL API [15] and stored using the MySQL 5.5 database management system The Jena [16] and Pellet [17] APIs are used to load and infer meanings of concepts defined in the ontologies that are associated with WSDL elements. The JUNG API [18] is employed to build the composition graph based on the relationships among concepts.

To evaluate the performance, measure the response time and validate the compositions selected by the discovery algorithm, some experiments were executed and the obtained results are presented in this section. The environment in which the experiments were performed was composed of a Intel® Core 2 Duo 2.10 GHz CPU, with 3 GB of RAM, running the Windows 7 operating system.

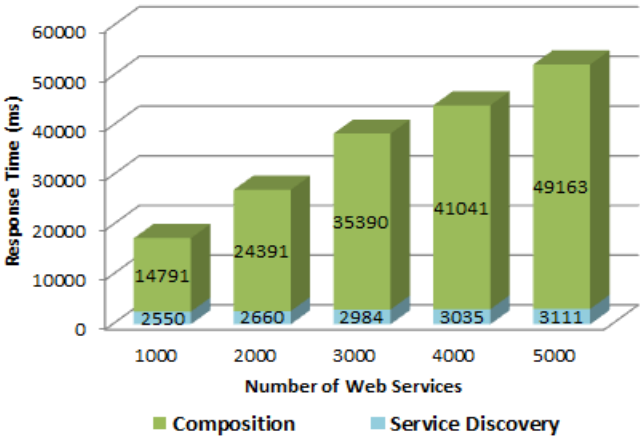


Figure 6. Response time of the discovery algorithm

For the experiment, 1,000 semantic web services were obtained from the www.semwebcentral.org repository, and these were replicated until reaching 5,000 web services. On average, services had one operation with two inputs and one output. Each input, output and operation had a single semantic annotation. Services were published and their annotations were extracted and stored in the SAWSDL Repository.

Different numbers of services were used during the tests in order to compare the time needed to find the composition as the amount of services increased. Several factors have direct influence on the performance of the composition algorithm, such as: the number of services involved; the number of operations of a service; the number of inputs and outputs of an operation; the number of semantic annotations in operations, inputs and outputs; the number of ontologies involved; the number of concepts and relations in an ontology; and the length of the shortest path from a startup service to a target service.

The SMD function was evaluated with different weights in order to select the best combination of values. After executing the composition process numerous times, the best compositions in terms of semantic quality were obtained according Table II.

TABLE II. Weights associated with evaluated annotations.

	Input	*Output*	*Operation*
Weight	0,3	0,35	0,35

Defined the weights, the web services were grouped into sets of 1000, 2000, 3000, 4000 and 5000 services. The request that had its response time evaluated was intended at finding a composition that returned 3 given outputs based on 2 available inputs.

The graph obtained with the experiment is shown in Fig. 5 and the time resulting of each evaluation is shown in Fig. 6. As the amount of services grows, the response time grows linearly, i.e., the response time is directly proportional to the amount of available services. Furthermore, the results presented confirm that the proposed approach is effective to discovery and composition of web services, since the time associated with the execution of the algorithm is polynomial.

The discovery process comprises two different stages, and the response time taken by each stage is shown in Fig. 6. The first stage, labeled *Service Discovery*, identifies startup and target services. Startup service selection requires the analysis of each input of each operation of a service in order to select it as a startup service. On the other hand, selecting the target services requires just one compatible output for service selection, resulting in a faster procedure. The second stage, named *Composition* in Fig. 6, is responsible for consuming the largest share of the time required to execute the algorithm. This result was largely anticipated, due to the successive matching operations to identify relationships between services needed for building the composition graph. In addition, this stage requires the calculation of the semantic mismatch degree of each path identified in the graph for selecting the composition that best fulfills the request.

The algorithm proves to be able to identify compositions and to obtain all the desired outputs through services available in the repository based on a list of inputs. Since the quality of the obtained results depends solely and exclusively on the matching between the parameters sent in the request and the annotations associated with services, the more detailed were the parameters and the services annotations, the better is the quality of the obtained compositions.

V. CONCLUSIONS AND FUTURE WORK

Assembling web service compositions is not a trivial task, specially with the large amount of services available nowadays. Composition is needed when a single service is not able to execute the required task. Semantic technologies can play an important role in this process, allowing the automation of the composition process.

In this context, this paper presented a platform to automatically build compositions of Web Services at request time based on the SAWSDL annotations. This platform comprises also an execution engine, which allows clients to invoke a composed service through a single web service request. This is achieved by executing a WS-BPEL description of the composed service.

Compared with related projects found in the literature that have similar goals, which were described in section II.E, compositions are obtained by the proposed algorithm at request time without user assistance, and will never require inputs that the client does not have. The results of the experiments demonstrated that the algorithm is able to combine single services to build compositions with a response time directly proportional to the amount of services available in the repository.

As a future improvement, the discovery mechanism will be capable of identifying the same concepts represented in different ontologies via ontology mediators. This is desired because people and organizations often develop different ontologies to represent the same domain, due to their different points of view of the same scenario.

Improvements are also being made to the developed implementation in order to allow the use of control structures and parallelism during the execution of the discovered compositions.

REFERENCES

[1] D. Booth and C. K. Liu, eds. *Web Services Description Language (WSDL) 2.0*, 2006. Available at. http://w3.org/TR/2006/CR-wsdl20-primer-20060327/

[2] A. Ankolekar. *OWL-S: Semantic Markup for Web Services*, 2003. Available at http://www.daml.org/services/owl-s/1.0/

[3] J. Bruijn, H. Lausen, A. Polleres and D. Fensel. *The web service modeling language wsml: An overview*. The Semantic Web: Research and Applications, 4011/2006:590-604, 2006.

[4] J. Kopecký, T. Vitvar, C. Bournez and J. Farrell. *SAWSDL: Semantic Annotations for WSDL and XML Schema*. IEEE Internet Computing, vol. 11, no. 6, pp. 60-67, 2007.

[5] D. Oliveira, C. Menegazzo and D. Claro. *Uma Análise Conceitual das Linguagens Semânticas de Serviços Web focando nas composições: comparação entre OWL-S, WSMO e SAWSDL*, In IADIS Conferência IberoAmericana WWW/Internet (CIAWI 2009). Alcalá - Madrid, Spain, October 2009.

[6] C. V. S. Prazeres, C.A.C. Teixeira and M.G.C. Pimentel. *Semantic Web Services discovery and composition: paths along workflows*. In ECOWS'09: Proceedings of the 7th IEEE European Conference on Web Services, pp. 58-65, Eindhoven, Netherlands, 2009.

[7] M. Paolucci, T. Kawamura, T. Payne and K. Sycara, *Semantic matching of web services capabilities*, in: Proceedings of 1st International Semantic Web Conference (ISWC), 2002.

[8] D. Martin, M. Burstein, D. McDermott, S. McIlraith, M. Paolucci, K. Sycara, D. L. McGuinness, E. Sirin and N. Srinivasan. *Bringing Semantics to Web Services with OWL-S*. World Wide Web 2007, pp. 243-277, 2007.

[9] Y. Charif and N. Sabouret. *An Overview of Semantic Web Services Composition Approaches*. Electrical Notes Theory Computing Science, Vol. 146(1), pp. 33–41, 2006.

[10] OASIS. *Web Services Business Process Execution Language Version 2.0*, May 2007. Available at http://www.oasis-open.org/committees/ download.php/23964/ wsbpel-v2.0-primer.htm

[11] W3C. Web Services Choreography Description Language Version 1.0, November 2005. Available at http://www.w3.org/TR/ws-cdl-10/

[12] V. Tran, S. Puntheeranurak and H. Tsuji. *A new service matching definition and algorithm with SAWSDL*, 3rd IEEE International Conference on Digital Ecosystems and Technologies (DEST'09), pp.371-376, June 2009.

[13] N. Mehandjiev, F. Lecue, U. Wajid, A. Namoune. *Assisted Service Composition for End Users*. In Proceedings of IEEE European Conference on Web Services (ECOWS 2010), Ayia Napa, Cyprus, pp. 131-138, December 2010.

[14] F. Belouadha, H. Omrana and O. Roudiès. *A model driven approach for Composing SAWSDL semantic Web Services*, IJCSI International Journal of Computer Science Issues, March 2010.

[15] N. Boissel-Dallier, J. Lorré and F. Benaben. *Management Tool for Semantic Annotations in WSDL*. On the Move to Meaningful Internet Systems: OTM 2009 Workshops, Springer, 978-3-642-05289-7, vol. 5872, pp. 898–-906, 2009.

[16] Hewlett Packard. *Jena Semantic Web Framework*, 2006. Available at http://jena.sourceforge.net/

[17] Clark & Parsia, LLC. *Pellet OWL Reasoner*, 2006. Available at http://pellet.owldl.com/

[18] J. O'Madadhain, D. Fisher, S. White, and Y. Boey, *The JUNG (Java Universal Network/Graph) Framework*, UCI-ICS Tech Report 03-17, October 2003.

Towards the Automation of the Semantic Annotation Process for Web Services

Leandro Ordóñez-Ante, Luis Antonio Rojas-Potosi, Luis Javier Suarez-Meza and Juan Carlos Corrales
Telematics Department, Telematics Engineering Group (GIT), University of Cauca, Popayan, Cauca, Colombia

Abstract- *The large and growing amount of services available in the current Web has placed the need for developing efficient mechanisms for service discovery, in order to meet either a particular user request or the requirements of software agents. In this regard, a lot of work has been addressed on adding semantics over service descriptions to improve the accuracy of search engines. Nevertheless, the adoption of semantic annotation models, nowadays, has been restricted by service designers and providers, since they require certain specialized knowledge –related to formal knowledge representations–, and given that their actual implementation is very resource intensive. In this paper, we present an approach to overcome this latter issue by automating the semantic annotation process. The approach we propose, builds automatically and incrementally formal representations of knowledge from a corpus of service descriptors, by using text mining techniques and an unsupervised learning approach.*

Keywords: web services, automatic semantic annotation, machine learning, LDA, FCA

1 Introduction

The Service Oriented Architecture (SOA) emerged as a means for heterogeneous, distributed and component-based applications to work seamlessly, through the definition of well-known standard interfaces, such as WSDL descriptors for REST and SOAP Web Services. This way, whenever a user or software agent requires consuming a service, it just has to know the content of the service interface to bind and implement its capabilities. However, due to the steady growth in the number of current Web resources, the search of suitable services for meeting some particular needs is an increasingly challenging task. In this regard, the scientific community behind this matter has proposed a way out to this problem, aligned with Semantic Web Technologies, conceiving what has been called Semantic Services Oriented Architecture (SSOA) [1].

The foundations of SSOA are laid on three key concepts [2]: SOA –from which its features of separation of concerns, standard interface provision, and capabilities discovery and reuse are taken–, Semantic-Based Computing –that provides sense to content, services and resources, using a formal and machine readable specification–, and Standard Based Design –that enable the integration of currently available applications with novel or future technologies–. Thus, SSOA would allow automatic service detection and selection (and consequently the automatic service discovery and composition).

Deploying SSOA requires, as stated, for resources and services to be formally specified, in such a way that a software agent can interpret and capture its functionalities in a semantic level. Nonetheless, the poor adoption of mechanisms for semantic description of services, by developers and providers (given their high cost in terms of time and resources) has inhibited the development and effective implementation of such architectures.

Researches documented in [3, 4, 5, 6] introduce proposals aimed at the integration of Artificial Intelligence technologies –specially, multiagent systems and planning- and semantic web services technologies, in order to enable automated service discovery and composition. Those works however, demand for each service the existence of two descriptors: the traditional (syntactical, e.g. WSDL) one, and one that defines its semantics (OWL-S/WSML). Given the complexity of such semantic descriptors, a large number of existing services don't meet the requirement of these works, thus limiting their actual implementation.

In order to overcome this limitation, currently some approaches are considered to tackle the problem of semantic service annotation, by applying knowledge discovery and emergent semantics techniques over huge corpus of service descriptors, which in some cases already contains annotations made by consumers in a collaborative way. Those approaches however, have failed in leave aside human intervention and also lack of precision in search and selection processes. Therefore it's considered necessary developing mechanisms that enable the automation of semantic service annotation tasks.

In this paper we present a research work in progress, which aims to address the stated problem by applying un-supervised machine learning techniques over a corpus of web service descriptors. This work seeks to answer the question: *How to automate the semantic annotation of web services?*

The remainder of this paper is organized as follows: we first outline the context into which our work is developed. Then, we describe a review of the current approaches regarding the stated problem. Next, it is depicted and defined

the architecture of a platform for enabling automatic semantic annotation of web services. Finally the conclusion and open issues of our work are addressed.

2 Background and Motivation

The work we are developing is framed around the transition between the dominant paradigm of the Web, the so-called *social Web*, and the establishment of the *semantic Web* or Web 3.0, specifically in regards to semantic annotation of services, which in turn is related to the subject of ontologies. Currently there is no generalized notion of the ontology concept; however in [7] it is formulated a conception that is widely accepted, according to which ontology "*is a formal, explicit specification of a shared conceptualization*".

The semantic annotation, core concept for the current proposal, is the result of a procedure that aims to make explicit for machines, the meaning (the semantics) of content and resources available in large repositories of information. This latter constitutes one of the requirements to meet to finally materializing the Semantic Web. The semantic annotation procedure is commonly supported in formal representation of knowledge, as the aforementioned ontologies, and for services, consists in associating ontological entities to the terms defining the attributes of the service in its descriptor document [8], allowing for instance, for service search engines to effectively comprehend (on a semantic level) both the services functionality as the service's clients requests, enabling them to accurately respond to service inquiries.

Traditionally, this semantic annotation procedure must be performed by hand by service designers and developers or in a collaborative way by service users (conceiving a sort of folksonomy of services). In both cases, the large and growing amount of services, along with the lack of knowledge regarding semantic description methods for services and the scarceness of suitable domain ontologies, has overwhelmed the human ability for performing this semantic annotation task. Additionally, the human intervention in marking up the services descriptors with ontological entities involves a very expensive process in terms of time, effort and resources.

In this regard, the focus of the present approach is on leveraging current techniques taken from the fields of machine learning, information retrieval and knowledge discovery, for automating the semantic annotation of web services. The next section will deal the revision of some previous works regarding the problem being tackled herein.

3 Related Work

This section explores some approaches that deal with semantic annotation, not only for web services, but also for content and other kinds of web resources.

In [9, 10, 11] the authors explore alternative approaches for semantic annotation of available services and resources in the Web. Such an approach consists of recognizing the information constructs from collaborative tagging systems (folksonomies) as specifications of shared knowledge, which can be suitable for semantically annotating service interfaces, dispensing with the use of ontologies. The main goal of these proposals, however, is to assist the process of semantic enrichment, still requiring human intervention (developers, users, providers, etcetera) for fulfilling the complete process.

The authors of [12] and [13] address two works regarding to semantic annotation of folksonomies, for various kinds of online available resources. In contrast to aforementioned works, the proposals of Angeletou in [12] and the one described by Siorpaes in [13] argue that it is required to formalize the knowledge generated within folksonomies, by using ontologies, in order to overcome their limitations in terms of organizing, searching and retrieving resources based on tags.

The work of Angeletou differs from the current proposal, as long as the former is focused on an image folksonomy. In turn, the project addressed in [13], although it takes into account the services as part of its working resources, its scope is limited to promote collaborative tagging thereof. Furthermore, the development of that project is still in an early stage, so the results from its implementation are not yet conclusive.

The approaches outlined in [14, 15, 16, 17] pose the use of techniques of machine learning such as Formal Concept Analysis (FCA) and most recently Relational Concept Analysis (RCA), for extracting and representing the knowledge covered by documental corpus, as conceptual hierarchies or taxonomies. This way, the approaches described in these works are suitable for composing formal models of knowledge, such as core ontologies, avoiding the intervention of domain experts. However, none of the aforesaid proposals had considered the automation of such a process.

From observations made on related proposals, the present work aims to automate the process of semantic annotation of web services descriptors, through an approach that combines techniques of text mining, unsupervised machine learning (FCA) and others taken from the Information Retrieval field (Latent Dirichlet Allocation–LDA and Nearest/Normalized Similarity Score–NSS) for enabling automatic and incremental generation of formal models of knowledge from service descriptors. Such models are meant to be used in annotating and categorizing services, through a platform that implements the above techniques.

Next section will address the description of our proposal, by outlining the architecture of the platform for automatic semantic annotation of service descriptors.

4 Overview of Our Approach

According to [8] there exist four types of semantics associated with web services: *data semantics* –formal definition of data in input and output messages–; *functional semantics* –formal definition of the capabilities of a Web service–; *non-functional semantics* –formal definition of quantitative or non-quantitative constraints–; and *execution semantics* –formal definition of the execution flow of services in a Process, or of operations within a service. Our proposal is

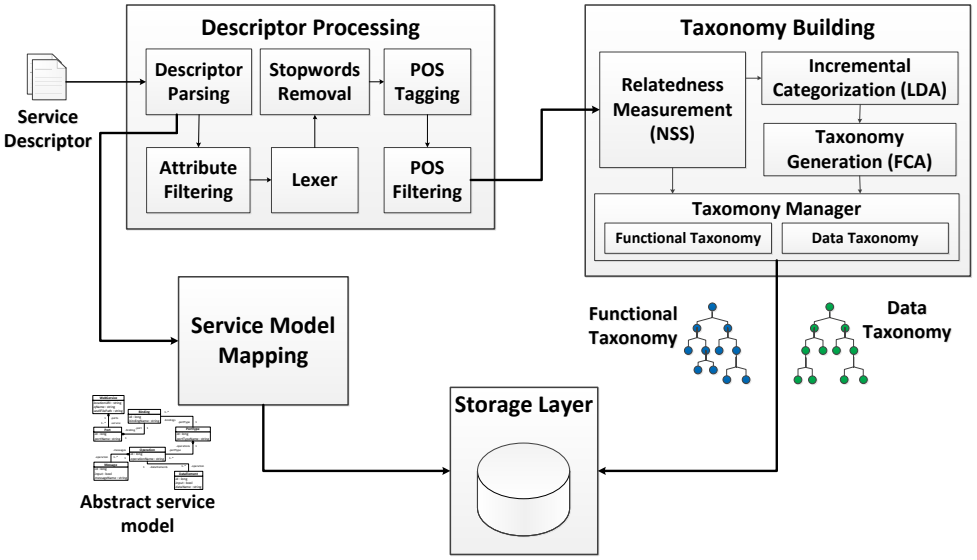


Figure 1. Platform Architecture

focused on the former two types of semantics, this way, our platform enable composing two formal models of knowledge for semantic annotating both capabilities and input/output data from web services, by harvesting the information on their descriptors. Figure 1 illustrates the components that make up the annotation platform, which will be described below.

The platform we propose, receive as input a set of wsdl service descriptors. For each of these descriptors a procedure is performed in order to map its content into an abstract entity model (which is then stored in a service registry), and to extract their relevant attributes (i.e. service, operations and input/output types). Such attributes are then categorized in two arrangements: a functional taxonomy and a data (input/output) taxonomy, which are built in an incremental way as long as new services are entered to the platform or new categories are detected. The annotation consists then in automatically associating service attributes to corresponding taxonomy categories, while these latter are made up. All the outcomes of the above procedure are in turn saved in the platform storage layer.

The main components of the proposed platform are explained next:

4.1 Descriptor Processing

The purpose of this module is to extract the relevant information from incoming service descriptors. This component of the platform provides a procedure that allows abstracting the information regarding the attributes that define the service functionality (operation and input/output types) by applying a set of text mining techniques (i.e. filtering, tokenization, POS tagging) which are described and used in [17] serving a similar purpose. The output of this module is twofold: first it loads into memory the complete service descriptor, outcome that is taken by another component of the platform, in charge of mapping the content of the descriptor into an abstract service model intended for further storage and retrieval. The second outcome has to do with applying the above text mining techniques on each of the service descriptors. This output comprises the specification of tree attributes of the service: *service*, *operation* and *types*, along with its corresponding POS tagged values. As an example, consider a service defined by[1]:

service: "*CurrencyService*"
operation: "*GetExchangeRate*"
(input)Type: "*CurrencyISO*"
(output)Type:"*GetExchangeRateResult*"

For the previous example, the outcome of the Descriptor Processing module will be:

(service, {"*Currency:NN*"})
(operation, {"*Get:VBI*", "*Exchange:NN*", "*Rate:NN*"})
(input,{"*Currency:NN*", "*ISO:SYM*"})
(output, {"*Exchange:NN*", "*Rate:NN*"})

Where, *NN* (noun), *VBI* (verb), and *SYM* (symbol) denote lexical categories (part-of-speech) for each of the words defining the service attributes. Notice that, for the service attribute, as well as for the output type, the descriptor processing module has ruled out three words: *Service* (for service), *Get* and *Result* (for output). This is due to these words does not have any valuable information for these particular attributes.

4.2 Service Model Mapping

This module is responsible for processing the information in service descriptors, to map it into the abstract

[1] Example from *seekda*, an online registry of web services. (http://webservices.seekda.com).

service model shown in Figure 2. Such a model allows capturing the services attributes –which are specified as the UML entities *WebService*, *Port*, *Binding*, *PortType*, *Operation*, *Message* and *DataElement*– along with the relationships between such attributes, in order to ease its storage, search and retrieval. This platform component, takes the descriptor information loaded into memory by the previous module, and instantiates the entities, in correspondence to the referred model.

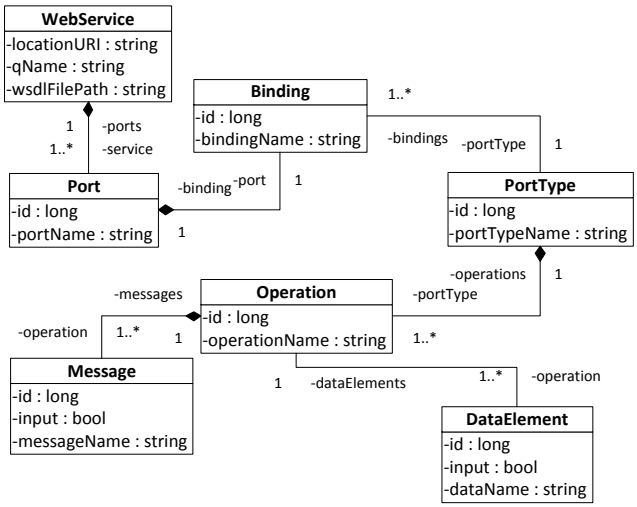


Figure 2. Abstract Service Model

4.3 Storage Layer

This component of the architecture consists, first in a service registry, whose relational model matches the service abstract model. Thus, all of the incoming services to the platform are stored and its information is maintained for further retrieval. Secondly, this storage layer implements a mechanism for storing the artifacts obtained as outcomes from the *taxonomy building* module. Such mechanism allow storing both taxonomies (functional and data) as RDF graphs, as well as querying and manipulating such knowledge structures by employing the SESAME Framework [18] along with SPARQL, the widely used RDF query language.

4.4 Taxonomy Building

This component takes as input the information generated by the *descriptor processing* module, for incrementally composing two taxonomies: one classifying the knowledge related to the terms that define the service operations (functional taxonomy), and one that is intended to arrange the knowledge about its input/output types (data taxonomy). This is a quite complex component as it is shown in the platform architecture (see Figure 1). It comprises a set of subsystems that are involved in the generation of the abovementioned taxonomies. The first of such subsystems, receives the terms supplied by the *descriptor processing* module and estimates their semantic relatedness relative to the concepts/categories previously classified in both taxonomies, by employing a measure of semantic relatedness (MSR) referred to as Nearest/Normalized Similarity Score (NSS), which is explained in [19]. As long as the measure of semantic relatedness between one of the terms of the service descriptor and one of the taxonomy concepts exceeds a predefined threshold, an association between these two elements (the descriptor term and the taxonomy concept/category) is placed and saved through the *storage layer*.

Eventually, it may not have significant similarities between the service attributes and taxonomy concepts, suggesting the income of new, uncategorized content. In that case, subsystems based on an online/incremental variant of Latent Dirichlet Allocation (LDA) [20, 21], and Formal Concept Analysis (FCA) [22] are involved. Jointly, these subsystems allow identifying additional concepts/categories, as well as their location within the taxonomy structure.

The subsystems depicted above operate over a taxonomy management component, which enables reading and updating the taxonomies and serves as mediator between both the *taxonomy building* and *storage layer* modules.

The use of FCA in approaches regarding categorization and knowledge representation of documental corpuses has spread recently. However, as argued in [23], applying this technique is constrained by the size of the corpus, due to the computational complexity that involves building the conceptual lattices, through which the information is structured in FCA. In this regard, the architecture proposed herein, applies a Latent Dirichlet Allocation (LDA) model – which is considered as an extension of the widely used PLSA in Information Retrieval applications [24] – as a mechanism for reducing the complexity linked to the generation of FCA formal contexts.

The purpose of using LDA is to understand and uncover the underlying semantic structure of a corpus of service descriptors. Through the probabilistic model proposed by [25] in LDA (which assumes each of the descriptors documents as a bag-of-words), it is possible to find out a set of categories (known as *topics*) covered by the descriptor documents, while associating for each document a probability distribution over all the categories/topics, i.e. a descriptor is conceived as a mixture of various topics, so that it can belong to more than one of them.

As stated before our proposal seeks to incrementally build representations of the information in the service descriptions. Thus, the traditional LDA model isn't suitable since it requires a training step, which is performed over a whole batch of documents. That is why the abovementioned online/incremental variant of LDA is applied in this component of the architecture. Such variant of LDA enables the incremental identification of categories/topics, as new services descriptors enter to the platform.

This incremental model is applied on both the functional (service, operations) and data (input/output types) service attributes. Such a procedure generates as outcome, several ranked sets of words (corresponding to the service attributes), each of which defining a different category. The word order in each of the ranked sets is determined by the probability of occurrence of every single word in documents regarding the

Table 1. Words by topic distribution obtained by applying LDA.

	Category/Topic 1	Category/Topic 2	Category/Topic 3
1	Currency	Weather	Coordinate
2	Convert	Climate	Locate
3	Rate	Forecast	Distance
4	Exchange	Temperature	Zip

category each set defines. Table 1 presents an example of the output obtained by applying the incremental LDA model, for the functional attributes of the services.

From the example it is possible distinguishing three ranked sets of attribute values, each related to one particular topic: *Currency Conversion*, *Weather Forecast*, and *Geo Positioning*. For the sake of space, in the example it is only considered three categories, each one with four associated service attribute values, however, there may be actually much more categories than that, then a service could belong to multiple categories, as long as its descriptor contains terms from various categories.

One of the key benefits of applying LDA over the service descriptors entering the platform is the grouping of service attributes in the aforementioned categories, which leds to a dimensionality reduction of the space over which the next subsystem in the *taxonomy building* component operates: the *lattice generator*. Such subsystem applies FCA, a well-known lattice-based technique for knowledge representation and unsupervised machine learning, which allows identifying groups of objects sharing common attributes. The formalism posed by this technique is founded on the relation between (*formal*) objects and its (*formal*) attributes or properties, from which the triplet $\mathrm{K} = (G, M, I)$ is composed, referred to as *formal context*. In this notation G represents the set of *formal objects*; M denotes the set of *formal attributes* and I states an incidence relation between an object $g \in G$ and an attribute $m \in M$ (gIm stands for "g has m").

The *lattice generator* subsystem applies this FCA technique by configuring a formal context where the formal objects are the set of services entered to the platform, and the formal attributes are the categories extracted by LDA. This way, consider $S = \{s_1, s_2, s_3, s_4\}$ a set of services and $C = \{c_1, c_2, c_3, c_4, c_5\}$ the set of categories to which the services of S belong. The formal context $\mathrm{K} = (S, C, I)$ is built from making explicit the membership relation (I) between services and categories, which can be represented by a cross table, as shown in Table 2.

In the above formal context, the relationship between services and categories are specified by the crosses, so for example service s_1 belongs to categories c_1 and c_5, service s_2 belongs to categories c_1 and c_4, and so on.

FCA defines a derivation operation ($'$) that links services and categories:
Thus, having a set $A \subseteq S$,

$$A' = \{c \in C \mid \forall\, s \in A : (s, c) \in I\} \qquad (1)$$

is the set of common categories shared between the services in A. Similarly, having the set $B \subseteq C$,

Table 2. *Services* × *Category* Formal Context (K = (*S*, *C*, *I*)).

	c_1	c_2	c_3	c_4	c_5
s_1	×				×
s_2	×			×	
s_3		×			×
s_4		×	×		×

$$B' = \{s \in S \mid \forall\, c \in B : (s, c) \in I\} \qquad (2)$$

denotes the set of common services for the categories in B.
As an example, consider the formal context depicted above, being $A = \{s_3, s_4\}$ and $B = \{c_1, c_4\}$; then, $A' = \{c_2, c_5\}$ and $B' = \{s_2\}$.

From this derivation operation stems the definition of the FCA *formal concept*: a pair of sets (A, B) is called a formal concept of the context K iff $A' = B$ and $B' = A$, (being A and B the *extent* and the *intent* of the formal concept respectively). So for instance, in our example, the pair $(\{s_3, s_4\}, \{c_2, c_5\})$ is a formal concept of the Services × Category formal context, while the pair $(\{s_1, s_2\}, \{c_5\})$ is not.

The whole set of formal concepts of K denoted by $\mathfrak{B}_\mathrm{K}$ are partially ordered by a *specialization* relation between concepts. Thus, it is said that a concept (A_1, B_1) is subconcept of (A_2, B_2) if $A_1 \subseteq A_2$. Having identified the set of *formal concepts*, it is possible to arrange them into a so-called *concept lattice*, which allows uncovering a hierarchical structure (based in the above specialization relation) of categories and the services they comprise. The concept lattice for the given formal context is depicted in Figure 3.

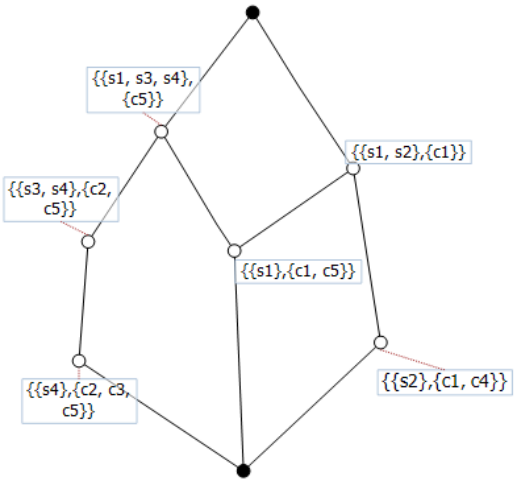


Figure 3. Concept Lattice for the *Services* × *Category* formal context.

A process similar as the explained above is performed for both, operations and input/output service attributes. This way, the taxonomy building module enables incrementally building of both functional and data taxonomies, while stating corresponding associations between the incoming web service attributes (operations and input/output types) and the categorized elements in such knowledge models.

5 Conclusions

Nowadays there is a huge amount of both resources and services available online, so much so that it has overwhelmed the search engines capability for meeting efficiently the queries from users and software agents.

Up to now there exists several research efforts aimed to tackle this problem, but it is largely an open question. Part of such efforts are focused on developing mechanisms for indexing and categorizing service through collaborative tagging applications, by using technologies of Web 2.0; nevertheless, the lack of ordering and limited reliability of the annotations on these resources, have hampered the benefits of such approaches. Other studies seek to leverage formal models of knowledge, such as ontologies, in order to attach semantic components on service descriptors, and thus enable automatic service discovery and composition. However, the costs involved in building and maintaining these formal models, as well as in the semantic annotation process had constrained, not only the actual implementation of these approaches, but also the deployment of the Semantic Web.

In this paper, we have introduced a novel and suitable approach to overcome the problem of semantic annotating web services. Our work focuses on the extraction of relevant information about services attributes, by applying some text mining techniques on their descriptors, in order to compose – in an automatic and incremental way – formal models of knowledge regarding those attributes. Such models are intended to specify two types of semantics of web services: the functional (operations) and the data (input/output types) semantics. The referred formal models are generated as conceptual taxonomies by defining hierarchical relationships between concepts, applying an online LDA model along with a FCA technique. To the best of our knowledge, the joint use of these two latter on tackling the problem we have stated, has not been addressed in previous works, thus it is one of the main contributions of our research. LDA allows uncovering the underlying semantic structure of a corpus of service descriptors as a distribution of relevant categories/topics, while FCA enables identifying a hierarchical structure of categories and the services.

This way the proposed approach merges both formal knowledge models generation, and a fully automatic semantic annotation method for web services.

The platform that implements this proposal also includes a registry where processed services are stored (once they have been mapped into an abstract model), as well as the above formal knowledge models as RDF documents.

The proposed approach is under development and as complementary work it is considered taking into account already annotated services, (i.e. services holding a semantic descriptor, such as OWL-S or WSML), in conjunction with web ontologies in order to enhance both the concepts and relationships of formal knowledge models, as the semantic annotations on services, which involves the use of semi-supervised machine learning methods.

6 Acknowledgments

The authors would like to thank University of Cauca, Colciencias and TelComp2.0 project for supporting this paper.

7 References

[1] Chance, S. 2009, "Semantic Service Oriented Architecture: An Overview," Semantic Web Site, [online]. (http://semanticweb.com/semantic-service-oriented-architecture-an-overview_b10678).

[2] Scientific Research Corporation (SRC). 2006, "Semantic Service Oriented Architecture - Case Study for OMG SOA/MDA/WS Workshop," SRC, Atlanta GA., EUA.

[3] Carman, M.; Serafini, L.; Traverso, P. 2003, "Web Service Composition as Planning," In ICAPS 2003 Workshop on Planning for Web Services.

[4] Hahn, C. et al. 2008, "Integration of Multiagent Systems and Semantic Web Services on a Platform Independent Level," In Proc. Of 2008 IEEE/WIC/ACM - WI-IAT '08 - Volume 02, Washington, DC, USA.

[5] García-Sánchez, F.; Valencia-García, R; Martínez-Béjar, R.; Fernández-Breis J. 2009, "An ontology, intelligent agent-based framework for the provision of semantic web services," Expert Syst. Appl. 36, 2, 3167-3187.

[6] Sbodio, M.; Martin, D.; Moulin, C. 2010, "Discovering Semantic Web services using SPARQL and intelligent agents," Web Semantics Science Services and Agents on the World Wide Web. Vol. 8, Issue 4 (November 2010), 310-328.

[7] Gruber, T. 1993, "A translation approach to portable ontology specifications," Knowledge Acquisition - Special issue: Current issues in knowledge modeling, vol. 5, no. 2.

[8] Nagarajan, M. 2006, "Semantic Annotations in Web Services," in Semantic Web Services, Processes and Applications, Jose Cardoso and Amit P. Sheth (Eds.), Springer.

[9] Bouillet, E. et al. 2008, "A Folksonomy-Based Model of Web Service Discovery an Automatic Composition," In Proc. f 2008 IEEE SCC - Volume 1.

[10] Meyer, H. and Weske, M. 2006, "Light-Weight Semantic Service Annotations through Tagging," in ICSOC 2006, Lecture Notes in Computer Science.

[11] Loutas, N.; Peristeras, V.; Tarabanis, K. 2009, "Rethinking the Semantic Annotation of Services," in

ICSOC/ServiceWave 2009 Workshops, Lecture Notes in Computer Science.

[12] Angeletou, S. 2008, "Semantic Enrichment of Folksonomy Tagspaces," International Semantic Web Conference, Doctoral Consortium, Proc. ISWC'08, Karlsruhe, Germany.

[13] Siorpaes, K. and Simperl, E. 2010, "Incentives, Motivation, Participation, Games: Human Computation for Linked Data," CEUR Proceedings of the Workshop on Linked Data in the Future Internet at the Future Internet Assembly, Ghent, Belgium.

[14] Ma, J.; Zhang, Y.; He, J. 2008, "Efficiently finding web services using a clustering semantic approach," In Proc. of CSSSIA.

[15] Yang, K. M; Hwang, S.-H.; Kang, Y.-K.; Yang, and H.-S. 2009, "Folksonomy Analyzer: a FCA-based Tool for Conceptual Knowledge Discovery in Social Tagging Systems," 1st International Workshop On Mining Social Media Programme.

[16] Azmeh, Z.; Huchard, M.; Tibermacine, C.; Urtado, C.; Vauttier, S. 2010, "Using Concept Lattices to Support Web Service Compositions with Backup Services," In Proc. of (ICIW '10), IEEE Computer Society, Washington, DC, USA, 363-368.

[17] Falleri, J. R.; Azmeh, Z.; Huchard, M; Tibermacine, C. 2010 "Automatic Tag Identification in Web Service Descriptions," Valencia, Spain, April.

[18] Broekstra, J.; Kampman, A.; Van Harmelen, F. 2002, "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema," in Proc. of ISWC 2002, Number 2342 in Lecture Notes in Computer Science (LNCS), Springer-Verlag, 54-68.

[19] Batra, S. and Bawa, S. 2009, "Semantic Categorization of Web Services," International Journal of Recent Trends in Engineering, Volume 2, No. 3.

[20] Hoffman, M.; Blei, D.; Bach. F. 2010, "Online learning for latent Dirichlet allocation" In: Neural Information Processing Systems.

[21] AlSumait, L.; Barbara, D.; Domeniconi, C. 2008, "On-line LDA: Adaptive Topic Models for Mining Text Streams with Applications to Topic Detection and Tracking" In ICDM 2008, pp. 3–12.

[22] Priss, U. 2006, "Formal concept analysis in information science," in ARIST 40(1): 521-54.

[23] Codocedo, V.; Taramasco, C.; Astudillo, H. 2011, "Cheating to achieve Formal Concept Analysis over a large formal context," in CLA 2011, Nancy, France, pp. 349-362.

[24] Hoffman, T. 1999, "Probabilistic Latent Semantic Analysis," in Uncertainity in Artificial Intelligence, Stockholm.

[25] Blei, D.; Ng, A.; Jordan, M.; January, 2003, "Latent Dirichlet Allocation". Journal of Machine Learning Research, 3 (4–5): pp. 993–1022. doi:10.1162/jmlr.2003.3.4-5.993.

Web Spider Performance and Data Structure Analysis

Sadi Evren SEKER

Department of Computer Engineering, Istanbul University, Istanbul, Turkey
academic@sadievrenseker.com

Abstract. *The aim of this study is performance evaluation of a web spider which almost all search engines utilize during the web crawling. A data structure is required to keep record of pages visited and the keywords extracted from the web site during the web crawling. The paper first goes into the detail of possible data structures for a web spider and critics all possibilities depending on their time and memory efficiencies. Furthermore the possibilities are narrowed into tree variations only and a tree is selected from each tree data structure family. Finally, a search engine is implemented and all the tree alternatives from each of the tree data structure family are also implemented and the performance of each alternative is benchmarked.*

Keywords: Web Spider, Web Crawling, Web Indexing, Benchmarking, Data Structures

1 Introduction

In the date of this study, there are 2 kind of possible web sources for the Internet surfers. A user trying to access information on the Internet can either use the directories or the search engines. Directories are hierarchical index lists of sites; they list sites by topic. They are widely used and in many cases offer an extremely great source of information. However, they have few problems: [1]

- Hierarchies are very vulnerable. Data and its classifications change constantly. This also leads to changes in hierarchy. A good example of this is DMOZ [2], world's largest directory. Several subcategories are created, removed or deleted each day.
- Most directories rely on human intelligence and are manually edited. They can never compete with search engines in amount of information. However, quantity is never as important as quality.

This paper concentrates on the search engine architecture rather than the hierarchical indexing. Anatomy of a search engine can be demonstrated as Fig.1.

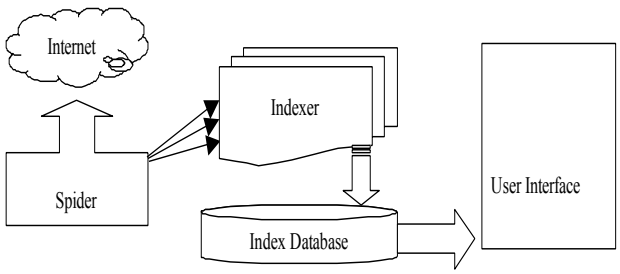


Fig. 1. A sample view of a web spider and its components

From the Fig.1 a spider gets connects to the Internet and supplies information for indexer which is responsible to keep the information for queries. This information can be kept in a database or can stay in memory for faster results. Finally, a user gets connect to the search engine through a user interface and queries the data in the indexer.

One of the most crucial points of a search engine is the indexer and the data arrangement during the data storage and querying.

The indexer implemented during this study can do:

- Web Parsing: is extracting the pure text from HTML tags,
- Extracting Keywords: is creating a set of keywords and removing duplicates and also cleaning stop words (which are the words does not effect the search like "and, or, a, an, etc."),
- Inverted Indexing: creating an index from the keywords to the web site links instead of keeping keywords in each web site.

The number of queries in an indexer is greatly higher than the number of insertions or updates. This requires a data structure with better query performance required in the indexer. This paper critiques the data structures just after discussing the alternative data structures. In the first chapter this discussion will be ignited and the narrowing alternatives and implementation and benchmarking will go on to the next chapters.

2 Data Structure of Indexer

Indexer is the core data structure in the whole project. The most complex and the most critical point is the implementation of the Indexer. There are several implementation possibilities. It is possible to implement a hash indexer or a tree as an indexer. The problem can be separated into three parts the performance of lookup in the data structure, the performance of update and the performance of memory management. Besides the memory issues, since it is the hardware update as a second choice, we have to concentrate on the time performance. The discussion gets the performance of lookup or update of the tree.

In a real living search engine, the probability of lookup queries would be much more than the queries of the insert or update. Besides the number of queries, the users are directly affected from the search queries, the worst update or worst insert query is not felt by the search engine users. So we have following assumptions in the indexer data structure design phase:

- Memory effectiveness can be sacrificed to time performance

Search queries are much more important than the update or insert queries

So according to the above criteria, we have listed all possible trees in the data structures world in the analysis phase. This section covers the possible tree implementations.

By the definitions on analysis phase, the trees can be grouped into 3 categories.

- B-tree family
- Spatial Access family (a special form of tree Access)
- Binary tree family

Besides the above tree families, in this study we have also concentrated suffix trees because of their importance and reputation on the search engines.

So this study will mainly cover these 4 type of tree implementations. Also the special case of the tree structures gives better results. For example, the AVL tree implementation yields better result than the most of the binary tree implementations. The reason of better results from AVL is the balancing of the tree. For example holding n nodes in an ordinary binary tree and AVL tree yields same worst cases O(log n) in time complexity of algorithm or the O(n) in memory complexity of the algorithm. But the AVL tree uses memory more efficient since the tree is kept in balance. So in the comparison of the AVL tree and an unbalanced binary tree, AVL yields always better results.

The same results can be applied to the k-d tree versus b-tree relation. The k-d tree implementation gives a great variety of indexing over the classical b-tree implementation. The complexity of k-d tree in the search is $O(n^{1-1/d} + k)$, where d is the number of dimensions and the k is the number of reported points. On the other hand the complexity of a classical b-tree query is only O(log n). So most of the cases the performance of k-d tree yields better results.

On the other hand the suffix tree implementations are built over several tree implementations. Most of the cases suffix tree can be built over a balanced search tree. The balanced search tree implementation gives the best result on the most of the cases. The complexity of suffix tree implementation over a balanced search tree implementation is O(log x) for the insertion and lookup where x is the number of alphabets in the language. On the other hand the complexity of traversal is O(1) which is a great speed up for the indexer of the search engine. So the next step would be an implementation of suffix tree over k-d tree or AVL tree structures.

2.1 Data structure of index database

Index database is responsible of managing huge indexes in the memory. Since the amount of ram is limited and the uptime of computers is not reliable, the index database is responsible of keeping the index data into the secondary storage (because of the limitations on

the project). The primary focusing data unit is the indexer in the search engine. The indexer database and other data structures are classified as the secondary targets. For the time being, the simplest solution for the index databases it the implementation of a simple file database holding objects in it.

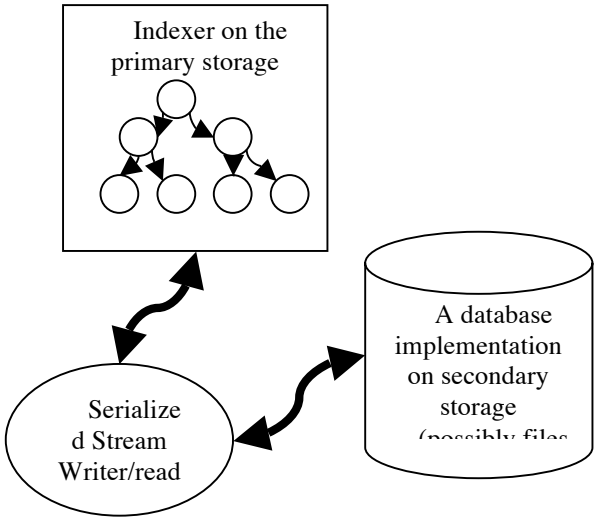


Fig. 2. Deployment of indexer database

The stream structure gives the ability of keeping serializable objects in the files. So the search engine will dump the index file in the memory to the hard disk in given time periods. The biggest problem about this implementation is the difficulties in the dividing tree into sub parts. This operation is extremely important while the index size is greater than the primary memory. Besides of dividing into sub parts the index database should keep a track of the priorities in the memory and keep the higher priority in the memory always while selecting between low hit and high hit accesses.

Also another type of implementation is the division of whole tree in to parallel or distributed computers. This approach has many benefits besides the increasing primary memory size. The computation and indexing can be divided between the computers as well.

3 Test and Performance Evaluations

This section will cover the tests, debugging and also the benchmarking and appropriate of several indexer implementations.

The basic time measurement tool in JAVA is taking the current system time by using the system library. Unfortunately in my testing environment the results of currentTimeMilis() function from the system library did not yield good results for the time measurement. There were lots of 0 results between the starting and ending time of tree accesses.

Because of these unstable results I have switched to the getting nano second function from the system library again. This function is nanoTime() from the java.lang package.

This second try resulted a valuable numbers and I have added these outputs in 3 different global cumulative variables. Each of these variables holds one of the tree operations. The results are also displayed into the screen when the print times button is clicked.

```
          long
temp=System.nanoTime();

          trie.addString(key,address)
;
          temp-=System.nanoTime();
          trieTime += temp;
          temp=System.nanoTime();
          avl.insert(key,address);
          temp-=System.nanoTime();
          avlTime += temp;
          keyURL a[]= new keyURL[4];
          temp=System.nanoTime();
          bpt.add(new
keyURL(key,address));
          temp-=System.nanoTime();
          bptTime+=temp;
```

Fig. 3. Coding of benchmarking

In Fig.3 code piece demonstrates the calculation of running time of each of the tree operations. The variable “temp” is created and filled up with the system time in the first line. After the creation of this variable the add function of the “trie“ tree is called. The return of the function is also the calculation of the next system time and getting difference from the temp variable. The same operation is repeated for the “bplustree” and the “avl” tree implementations.

Please note that the above code is in a function and called every time when an insertion operation is needed in the tree. So the variables in the above code will keep the cumulative time of each of the tree insert operations.

Also similar to the above insertion operations, the time for each search operation is calculated again. The time measurement of the search operations is same and the value of the search time is added to the cumulative variables holding the time for each data structure.

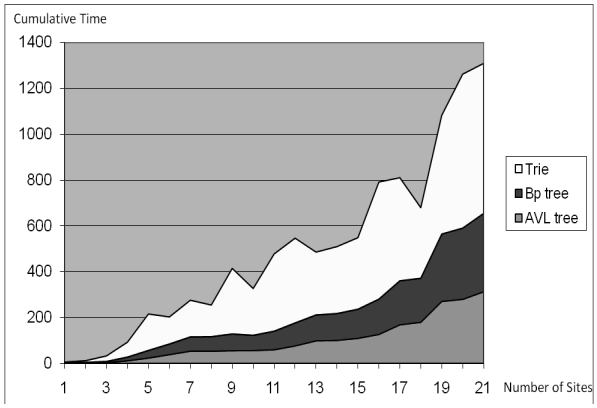


Fig. 4. Time Efficiency of the data structures

Fig.4 holds the tests run over 21 sites with 5 keyword search from each site. The sites are tested by time manner and the cumulative time value is displayed on the y axis of the Fig.4. The dataset of the above graph can be demonstrated as Table 1.

Table 1. A sample view of cse.yeditepe.edu.tr domain search with 5 keywords.

SiteName / Keyword	Time of Trie	Time of AVL	Time of BPT
Site: cse.yeditepe.edu.tr	539082176	552533245	483070868
Keyword: Faculty	+536991	+523530	+51961
Keyword: Exchange	+5565055	+7971125	+28216
Keyword: Studying	+7286401	+7262935	+31009
Keyword: Application	+1673956	+1623670	+84926

The graph is built over the above tables for each of the 21 web sites. So the web site is first indexed with three different tree data structures and than the keywords are tested as the above sample.

4. Conclusion

This project covers a basic web spider implementation with various indexer possibilities. The test results have shown us the best possible tree implementation for the search engines is the Trie implementation. Its nature also gives the signal of such a result and I have tested this case via this project. Also the bplus tree and AVL has yielded worse results than the Trie but they are very close to each other.

Acknowledgement

This study was supported by Scientific Research Projects Coordination Unit of Istanbul University. Project number YADOP-16728.

References

[1] Koulutus- and Konsultointipalvelu KK Mediat, from SEOGuy.com 2004
[2] Open Directory Project (Directory of Mozilla), 2007
[3] Main source for information on the robots.txt Robots Exclusion Standard and other articles about writing well-behaved Web robots. www.robotstxt.org , 2007
[4] Metasearch.com , a search engine working over the currently implemented search engines. 2007
[5] Sergey Brin and Lawrence Page, The Anatomy of a Large-Scale Hypertextual
Web Search Engine, Computer Science Department, Stanford University, 1999
[6] National Institute of Standards and Technology nist.gov, 2007
[7] TUSSE (Turkish Speaking Search Engine) , http://www.shedai.net/tusse, 2008[8] G.M.
[8] Adelson-Velsky and E.M. Landis, An algorithm for the organization of information, Soviet Mathematics 3 (1962), pp. 1259–1263.