An Approach to Manage Ontology Dynamically based on Web Service Composition Requests

Debajyoti Mukhopadhyay
Department of Information Technology
Maharashtra Institute of Technology
Pune 411038, India
+91-77091-52655
debajyoti.mukhopadhyay@ gmail.com

Archana Chougule
Department of Information Technology
Maharashtra Institute of Technology
Pune 411038, India
+91-97656-60044
chouguleab@gmail.com

ABSTRACT
Web services are playing a major role in electronic business now a day. These applications have ability to perform business activities on their own. But in some cases, one service is not able to perform certain task and it is required to compose two or more services to complete a task. In semantic web, different methods for ontology based web service composition are available which work with the help of domain ontology base. Maintenance of ontology base is an important concern to carry out the composition successfully even after long period. The ontology base need to be updated whenever new web service is generated or existing web service is updated. A new ontology should be added for newly registered web service and for updated web service, changes should also get reflected in the corresponding ontology. Updating the database every time will affect the performance of database and the database access will require more time. So, there must be some policy to decide when to update the ontology base. Considering this fact, we propose a new approach for ontology maintenance where ontology will be updated depending on web service composition request.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]**:** Information filtering, Retrieval models.

General Terms
Algorithms, Languages, Management

Keywords
Web service composition, domain ontology, owl

1. INTRODUCTION
The number of web services available to serve different purposes is increasing continuously [17, 18]. It is difficult to find out a specific service to complete the task due to lack of semantic information and complex and varying user requirements. Web service composition is carried out in such scenario, where new and more useful solutions can be achieved [15]. Different methods are suggested in literature to carry out web service composition. Composition can be carried out manually or it can be automated. Manual composition methods are highly inconvenient in case of more complex compositions [10, 14-16]. This paper focuses on automated approach where composition is done dynamically with the help of web service ontology and domain ontology. An optimum composition which can best satisfy the user's need is selected. Whenever a new web service is registered, the ontology is created for that web service and added to ontology base which is used for composition. If the ontology base contains the all the required service ontologies, the composer generates the composition and the query is answered. But there exists some cases where it is possible that the composition cannot be carried out with existing ontologies available. In such case its required to create new service ontology for completing the composition. The ontology base can be updated to fulfil composition request by analysing the previous composition requests for which the ontology is not present in the database. This paper proposes method for achieving dynamic web service composition, where the ontology base will be updated dynamically considering the composition request.

The rest of the paper is structured as follows: Section 2 describes work done by other researchers related to our approach. Section 3 details about ontology based web service composition and section 4 details about ontology management system. Then we present our approach to update ontology dynamically based on web service composition request in section 5. Next we present the implementation details of our approach in section 6. Section 7 is the impact analysis and we conclude the paper at last in section 8.

2. RELATED WORK
As web service composition is one of the important issues in semantic web, different methods to achieve web service composition are suggested in literature. F. G. Alvares and J. S. Parente[1] have proposed a dynamic web service composition method, where they consider Quality of Service parameters such as price, availability, duration and reputation. They make use of composition cache every time to check whether it satisfies the required QoS Parameters. If it does not, then composition is changed accordingly. Goal driven and ontology based approach for web service composition is explained by Jiangang Ma, Yanchun Zhang and Minglu Li[4]. In this architecture, they decompose the user's goal to sub goals. The information in the goal and web services is annotated with domain specific ontology. For carrying out composition of web services they use AI technology and theory of reasoning about action. Charlie Abela [2] proposes a web service composition engine which

CUBE 2012, 3-5 September, 2012, Pune, India.

automatically handles the integration of Web services. He uses a web service description language such as DAML-S. He considers the planning of workflow definitions, scheduling of tasks, status monitoring of the execution process, handling of faults and communication with other entities such as user agents, service registries and other composition engines to complete the composition. He explains languages available to achieve web service composition with their advantages and disadvantages. Heuristic approach for web service composition is also suggested by YU Qing-mei, XIAO Peng-yan and JIN Ting [3], where it finds similarity based on ontology. They construct a web service composition graph. They generate a serial of composition path using directed acyclic graph. They introduce a heuristic function which reduces the searching area. The explained algorithm considers the services semantic similarity and adjusts the composition plan dynamically. Yajuan Song and Lei Liu [3] present a two way composition method to process domain ontology for dynamic web service composition. Domain ontology is annotated with the input and output parameters of the web service according to the concepts in domain ontology they belong to. Yang-Seung Jeon, Eun-Ha Song and minyi Guo [6] present mediator for interoperability of web services while doing ontology based composition. To avoid data type collision, the mediators are implemented for conversion between two different data types and for extracting necessary output parameters from available output parameters. Static and dynamic service composition environments are suggested by Liquan Han and Shufen Liu [7]. They apply knowledge reasoning process to service ontology for composition. With the definition and application of ontology, they propose the concept and definition format of service-ontology, and with the construction of service composition and corresponding algorithm, they apply it in the dynamic Web service composition application. Combined with intelligent smart transcript repository and ontology knowledge repository, they describe the static and dynamic composition environments to facilitate knowledge accumulation, logic inference, binding web service and creation of process driven model. The approach considers the semantics of services along with the quality and the efficiency of the composition. Web service discovery and composition framework called AIMO is also described in literature by S. Gholam and Suhaimi Ibrahim [8]. Framework has HTN planner which produces a sequence of actions that perform some task and HTN-DL to match the task with method in OWL and description logic. B. Arpinar, R. Zhang, B. Aleman-Meza and Angela Maduko [10] suggest a web service composition method which is based on ontology. They provide service profiles in DAML-S which is used for semantic descriptions of service interfaces and functions. For development of ontolgies, ontology management framework is described by Robbert Harrison, Danial Obst and Christine W. Chan [11]. The ontology versioning system required to maintain ontology base is described in detail. Several domain independent constructs required for generic ontology representation are also described with examples. Yu Juan and Dang Yanzhong[12] also suggest one more framework for ontology management where they give overview of important components in ontology management framework such as ontology building module, ontology maintenance module and ontology query executer. They also give detail description of how ontologies can be accessed concurrently. Fouad Zablith[13] suggests a practical approach for Ontology evolution. He highlights two research approaches in the domain of ontology evolution. The first approach considers the evolution as a pure management of changes performed by the user while the second approach takes into account dynamically updating and learning ontologies.

3. ONTOLOGY BASED WEB SERVICE COMPOSITION

As large number of web services is available, we must have an efficient web service composition method to respond customer requests accurately. It is required to support business-to-business or enterprise application integration. Various methods are available for web service composition which may be static or dynamic. Examples of dynamic web service composition can be AI based composition, two way composition, composition based on acyclic graphs as mentioned in related work. Ontology based dynamic web service composition is one of the efficient and fast method in the world of semantic web. In ontology based composition, when a new service is registered, it is mapped to the concepts of the ontology under specific domain. All the services are classified by the concepts according to the input and output parameters and stored in the ontology base. The details specified in the domain ontology and service ontologies under related domain ontology makes the automated composition possible. Knowledge reasoner in semantic web which plays main role to carry out dynamic service composition works with the help of service profile of the web service. Service profile of registered web service is prepared using OWL-S. OWL-S mark-up of web services provides a declarative, computer-interpretable API that includes the semantics of the arguments to be specified when executing the composition [9]. An OWL-S Profile describes a service as a function of three basic types of information: what organization provides the service, which functions the service computes, and a host of features that specify characteristics of the service. After profile registration, registered service ontology is used to annotate the domain ontology. When request comes from the user, knowledge reasoner finds the matching web service. Service matcher which is part of knowledge reasoned is responsible to find out matching web service form available services.

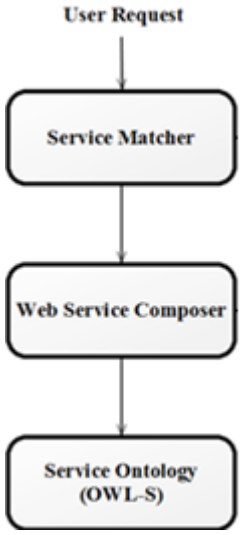

Figure 1. Web service composition using ontology

Service matcher semantically searches a single web service to satisfy the user request and sends the response to the user. If no single web service can satisfy the user request, then composer will

be called to compose a serial of web services to satisfy the request as depicted in figure 1. The composition starts from the service that needs one or more of input parameters matching to the parameters in the user request. The composition can be done with the help of directed acyclic graphs having web services as vertices and input, output of service as edges. If the precondition of user request is compatible to the precondition provided in composition and request's other properties satisfy corresponding properties in the composition, the composition can be carried out successfully. Ontology based knowledge reasoning is the basic step of dynamic web service composition. The main advantage of this approach is, when a query comes, whether to match or to compose, only the concept related services will be searched. It saves the time required for searching and matching the services. The available ontologies can also be used to carry out QoS based composition.

4. ONTOLOGY MANAGEMENT SYSTEM

The domain ontologies and service ontologies created for web services are collectively stored in the ontology base. Ontology management system is responsible to manage all these ontologies. Main parts of ontology management system are Ontology maintenance tool and Ontology query executer.

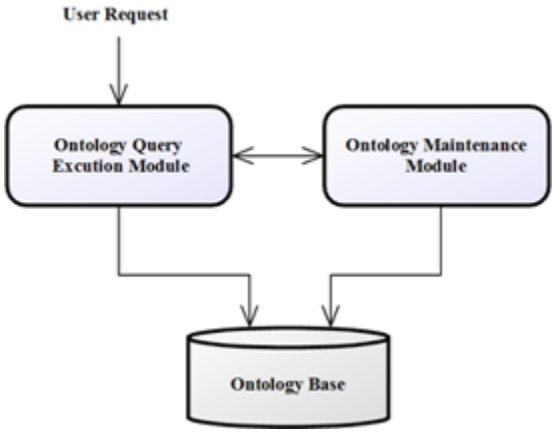

Figure 2. Ontology management system

Ontology query executer checks queries requested by ontology maintenance tool and execute them. Input for query executer is the query of altering the rule of ontology and its output is the execution information or error information. Ontology maintenance tool manages ontology versioning scheme. An ontology versioning scheme is used to maintain the ontology base. In ontology versioning scheme, there is a major version number for each domain ontology. Each service ontology under respective domain ontology is assigned with minor version number [11]. When ontology is updated the version no for the ontology will also be updated and when new ontology is added the new minor version number will be assigned to the service ontology. If there exists required domain then it will updated with matching major version number. In case where there is no matching domain ontology, the new domain ontology will be created by ontology building tool. It will be assigned with major version number, and the service ontology will be updated with that major version number. Versioning of ontology is very similar to versioning of source code of program. Version control can be implemented using a copy-modify-merge model. This model allows concurrent access to the ontology which being updated because each request will get its private copy of specific ontology. When ready, the modified ontologies will be merged again to create the next version. The version system which is part of ontology management system is also responsible for detecting possible conflicts caused by the update. The major version operation updates the ontology conceptualization information and minor version number updates to the next minor version for specific ontology. The versioned ontology will then be added to specific domain.

5. PROPOSED APPROCH

Our approach takes advantage of ontology management tools to carry out automated web service composition. As described in section 4, the domain ontology base contains service ontologies for all the registered web services in that domain. The ontology base needs to be updated periodically. Whenever some web service gets updated, input and output parameters for the web service will change. This change must be reflected in the ontology base. Also when new web service is registered, ontology for the same should be added to the database, so that each composition request will be fulfilled successfully. There must be some policy to decide when to update it.

5.1 Logging failed composition

The updating of ontology base can be carried out with the help of records in ontology log which are added while completing web service composition requests.

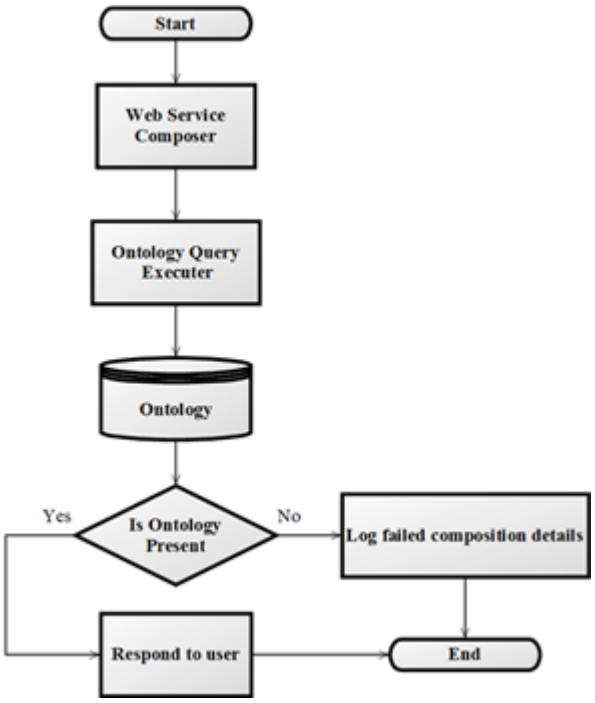

Figure 3. Logging failed web service composition

When request comes for web service composition, the composer passes the request to ontology query executer for each web service. An ontology query executor checks query requested by the web service composer and executes those executable and non-inconsistency-causing ones. If the query is executed successfully

for all the web services, the composer generates the answer and response is sent to user. If the ontology required to complete the composition is not available in the ontology base, the composition will fail. In this case we can use the details of failed composition to achieve ontology updation. Whenever the composition fails; the details of failed composition request will be logged in the separate database called as ontology log which is part of ontology generator. Ontology log will contain the details such as input and output parameters of the request for failed composition and required types of the parameters.

5.2 Updating Existing Ontology

Records stored in ontology log will be used by ontology generator also contains ontology search engine and ontology building tool. The ontology generator will check the ontology log to decide whether it is required to update existing ontology or new ontology should be created. If there exists ontology partially matching to the request, then the existing ontology will be modified with the help of ontology query executer. Required matching ontology will be selected from ontology base by matching input and output parameters in the composition request using service matcher. The number and type of input and output parameters will be updated considering the composition request. Also new functions from changed web service will be mentioned in ontology. Overall description in ontology for the requested composition can be updated by updating OWL and OWL-S using java based ontology updating tools.

5.3 Creating New Ontology

If no single web service exists for required input, output parameters, the ontology search engine which is part of ontology management system, will get the list of required input and output parameters and search in web service registry for the web services having matching input and output parameters required to complete the composition. After searching the required web services, the new service ontology will be generated for that web service by ontology building tool. The building process can be divided into two stages: conceptualization and formalization. Ontology building tool supports both stages of the ontology building process. The conceptualization process is carried out on the data collected by ontology search engine. At the stage of formalizing, the building tool automatically represents and stores the conceptualization in OWL format. Some of the available ontology building tools are Protégé, OntoBuilder and OntoLiFT. Service ontologies for matching domain ontologies also will be generated in OWL-S format. Generated service ontology will be added to matching domain ontology by ontology maintenance tool. Algorithm 1 shows the ontology steps for ontology updating by ontology generator containing steps for updating existing ontology and also for creating new ontology. The described approach gives a new and efficient way to keep the ontology base up to date dynamically. All the composition requests can be carried out successfully with this approach as the ontology base will have all the ontologies with required input and output parameters.

6. IMPLEMENTATION OF OUR APPROACH

In this section, we describe the example of online flower shop which is implemented to show all the steps in the presented approach.

Algorithm 1: Creating/Updating Ontology

Input: Failed composition details

Output: Updated ontology

1: Begin

2: Store failed composition details in ontology log

3: Start ontology search engine

4: If (Ontology exists having signature in service profile partially matching with log record) Then

5: Update existing ontology

6: Update version number

7: Else

8: Search web service having signature matching with Input

9: Create new ontology for found web service

10: Add version number to ontology

11: Store ontology in Ontology Base

12: End If

13: End

6.1 Development environment

The described example uses OWL-S to describe ontologies of web services which enables automated composition of web services. The scope for ontologies is defined to include the classes, functions and input/output parameters for functions. The web services are built using Java and run on GlassFish Server 3.1. The online portal of flower shop is developed using JSP. The ontologies are updated using APIs provided by mindswap (Maryland Information and Network Dynamics Lab Semantic Web Agents Project).

6.2 Scenario of online flower shop

As a working example, we have created virtual flower shop like book store. It provides customers with the facility to purchase flowers from anywhere in the world. We have developed online portal where customer has to select name of the flowers to purchase and the location of the customer from given lists. Customer has to enter number of flowers to purchase from selected location. Then the customer will be shown the total cost of the selected flowers in the currency of location selected by customer and will be asked to continue shopping.

6.3 Web service composition

In FlowerShop application, when name of flower, location of flower and count of flowers is given the total cost of flower should be shown to the user. The total cost is calculated with the help of web services. Available repository of web services is searched and it is found that no single web service can fulfill the requirement. So the calculation of total cost of flowers is done by sending request to web service composer. Web service composer searches for suitable web service in ontology base with the help of ontology search engine. Ontology search engine works on Service Profiles and Process Models of available web services and returns the names of three web services to composer. The web service composer then composes three web services namely

CurrencyConverter, FlowerCost and FlowerLocation. The name of flower and location of customer is given as input to composite service.

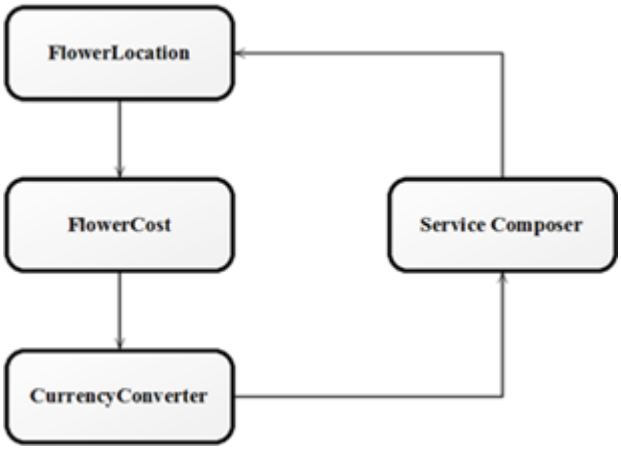

Figure 4. Web service composition for FlowerShop

The composite service then gets the location of selected flower from FlowerLocation web service. The returned location indicates the country in which flowers are available for sell. Then obtained location of the flower and name of the flower is given as input to FlowerCost web service which returns current cost per flower at the given location.

```
.....
<owl:Class rdf:ID="FlowerTypes">
 <owl:oneOf rdf:parseType="Collection">
  <CreditCardType rdf:ID="Rose"/>
  <CreditCardType rdf:ID="Lily"/>
  <CreditCardType rdf:ID="Lotus"/>
  <CreditCardType rdf:ID="Orchid"/>
 </owl:oneOf>
</owl:Class>
<process:AtomicProcess rdf:ID="getLocation">
 <process:hasInput>
  <process:Input rdf:ID="fname">
  <process:parameterType  rdf:resource="#FlowerTypes"/>
  </process:Input>
        </process:hasInput>
        <process:hasOutput>
  <process:UnConditionalOutput rdf:ID="FlowerLocationOutput">
   <process:parameterType rdf:resource="&xsd;#string"/>
  </process:UnConditionalOutput>
 </process:hasOutput>
</process:AtomicProcess>
</rdf:RDF>
```

Figure 5. ProcessModel for FlowerLocation web service

Finally CurrncyConverter web service is used to convert the total cost of flowers in the currency of customer location as customer may be ordering from the country different from that of flower location. Final output of the composition will be the total cost of the flowers in currency of customer location. This composition of three services is carried out automatically by using ontologies defined for the web services and stored in ontology base. The ontologies for web services are written in OWL-S. With OWL-S, we provide essential knowledge about each web service. For each web service, ServicePorfile, ServiceModel and ServiceGrounding is created and stored in ontology base. For completing composition of web services, matching of input, output parameters of functions and their data types in web service is carried out. Service Profile is used for web service composition as it contains details of various functions and their input, output parameters for web services. Figure 5 is an example of Process Model for FlowerLocation web service. In Process Model, getLocation is an atomic process with input as name of flower which is one of the names mentioned in ontology.

6.4 Updating Ontology

When the customer selects the name of flower, customer location and no of flowers, the composite service carries the composition of the services based on their ontologies and passes the input to composite service. The composite service successfully returns total cost of flowers which is used for further processing. But there can be case where the customer may requests for flowers which are not mentioned in FlowerLocation ontology e.g.; Blossom flower. In this case web service composition will not be completed. Web service composer fails to complete the composition due to failure of FlowerLocation web service execution. The information about failed composition will now be recorded in ontology log as described previously. Ontology log contains information such as name of the service because of which composition failed, name of the function and the values for the parameters for the functions in web services regarding failed composition. Whenever a new entry is made in ontology log, process for updating ontology is carried out so that the composition will be carried out successfully for the next request. Here, a new ontology can be added or existing ontology can be updated. For flower shop example, Service Profile and Service Model which comes under Process Model for FlowerLocation will be updated and new entry will be done for blossom flower. The changed ontology for FlowerLocation will be replaced with old one in ontology log. When the customer tries to purchase blossom flowers next time the composition process will be carried out successfully as the ontology is updated dynamically.

7. IMPACT ANALYSIS

The Online Flower Shop system without automated ontology updating facility was deployed and response for it was tested for 5000 registered customers. The system was kept working for first one month. The new flower name Blossom was then added afterwards in the list of flowers to be selected by the customer. But the ontology was kept as it is. As the provision for sell of Blossom flowers is not available in composite service, the system failed whenever user selected blossom flower for purchase. Eventually the number of customers visiting to site decreased. The same system, but with automated ontology updating facility described above was also deployed having 5000 customers registered. The system was kept working for first few days. One of customers tried to purchase the blossom flowers and the system failed to respond. Immediately the ontology for FlowerLocation web service was updated automatically as per proposed approach and a new entry for blossom flower was added in ontology. Customer requests were completed for subsequent requests for blossom flowers. Number of customers visiting the site went on increasing because of automated ontology maintenance as shown in the Figure 6.

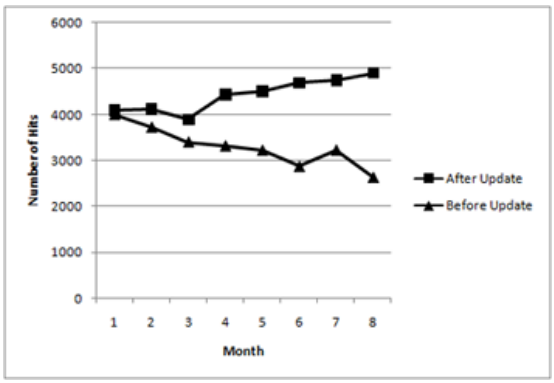

Figure 6. Effect of ontology update on customer response

Figure 7 shows the scalability of the approach in terms of ontology updating time. We also tested our approach on seven other examples. The number of ontologies required to be updated to make a useful example for customer is different for each example. The time required to carry out the update and make the example working increases as the number of ontologies to update increases.

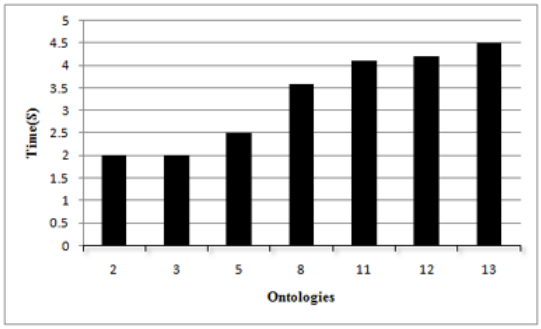

Figure 7. Scalability of the approach in terms of ontology updating time

8. CONCLUSION

Ontology updating is important part of ontology maintenance. This paper presents a new approach for updating ontology dynamically. It details how ontology can be updated with the help of web service composition request. The details of dynamic web service composition are given first. We have described how information required for ontology updating is collected as log records and then used later. We suggest an efficient approach as the ontology base will be updated on need basis. With this approach we can keep the ontology base up-to-date. The dynamic web service composition which is dependent on ontology can be carried out successfully with this approach as all required ontologies required for composition will be available eventually.

9. REFERENCES

[1] F. G. Alvares and J. S. Parente, 2010, An Infrastructure for Evolving Dynamic web Services Composition, In *International Journal of Intelligent computing research(IJICR), Volume 1, Issue1/2*.

[2] Charlie Abela, Semantic Web Services Composition, http://www.cs.um.edu.mt/~csaw/CSAW03/.../WebServiceComposition.pdf

[3] YU Qing-mei, XIAO Peng-yan, JIN Ting, HUANG Dong-mei, 2010, Semantic and Heuristic Approach to the Composition of Web Services, In *International Conference on Web Information Su=ystems and Mining, IEEEXplore.*

[4] Jiangang Ma, Yanchun Zhang and Minglu Li, 2005, OMWSC – An Ontology-Based Model for Web Service Composition, *Proceedings of Fifth International Conference on Quality Software, IEEEXplore..*

[5] Yajuan Song, Lei Liu, Ping Ren, 2009, Web Service Composition Based on the Annotated Ontology, *Fifth International workshop on education technology and computer science, IEEEXplore*

[6] Yang-Seung Jeon, Eun-Ha Song, Minyi Guo, Laurence T. Yang, Young-Sik Jeon, Sung-Kook Han, Ontology-based Composition of Web services for Ubiquitous Computing

[7] Liquan Han, Shufen Liu, Lu Han, Zhilin Yao, 2008, Service Composition Engine based on Service-ontology, *IET*.

[8] Sayed Gholam Hassan Tabatabaei, 2008, Wan Mohd Wan Kadir, Suhaimi Ibrahim, Semantic Web Service Discovery and Composition Based on AI-planning and Web Service Modeling Ontology, In *Asia-Pacific Services Computing Conference, IEEEXplore*.

[9] OWL-S: Semantic Markup for Web Services http://www.w3.org/Submission/OWL-S/#2

[10] Budak Arpinar, Boanerges Aleman-Meza, 2008, Ruoyan Zhang and Angela Maduko, Ontology-Driven Web Service Composition Platform, In *IEEE International conference on E-Commerce Technology.*

[11] Robbert Harrison, Danial Obst, Christine W. Chan, Design of an Ontology Management Framework.

[12] Yu Juan, Dang Yanzhong, 2010, A Framework of Ontology Management System, In *IEEE International Conference on E-Business and E-Government.*

[13] Fouad Zablith, 2009, Ontology Evolution: A Practical Approach, *A Poster in the Proceedings of the AISB Workshop on Meaning and Matching (WMM), Edinburgh, UK.*

[14] Han, S., G. Skinner, Potdar, V., and Chang, E., 2006. A Framework of Authentication and Authorization for e-Health Services. In: Proceedings of the 3rd ACM workshop on secure web services (ACM SWS). Alexandria, Virginia, USA, October 27-31.

[15] C. Wu, V. Potdar, and E. Chang, "A Conceptual Framework for Privacy Policy Negotiation in Web Services," in Proceedings of the 6th International Network Conference (INC2006), Plymouth, UK, 2006, July 11–14, pp. 195–202.

[16] Wu, C., Potdar, V., Chang, E., 2008. Latent Semantic Analysis – The Dynamics of Semantic Web Service Discovery. In: T. Dillon, Chang, E., R. Meersman, K. Sycara eds. 2008. Advances in Web Semantics I. LNCS. Berlin, Heidelberg: Springer. pp. 346-373. 978-3-540-89783-5.

[17] Chai, K., Hayati, P., Potdar, V., Wu, C. and Talevski, T., 2010. Assessing post usage for measuring the quality of forum posts, In: International Conference on Digital Ecosystems and Technologies (IEEE DEST 2010). Dubai, U.A.E.

[18] Chai, K., Potdar, V., and T. Dillon, 2009, Content Quality Assessment Related Frameworks for Social Media. In: Proceedings of the International Conference on Computational Science and Its Applications (ICCSA 2009). Seoul, Korea, June 29 - July 2.