Simplifying Web Service Discovery & Validating Service Composition

Shrabani Mallick,Rajender Pandey,Sanjeev Neupane, Shakti Mishra, D.S.Kushwaha
CSED, Motilal Nehru National Institute of Technology, Allahabad, India
{shrabani, cs0920, cs0923, shaktimishra, dsk}@mnnit.ac.in

Abstract— **Web services are software components developed to simplify machine-to-machine interaction over the Web. Many researches are targeted towards Web service standardization, and these efforts have significantly contributed towards improving functionality of Service Oriented Architecture (SOA). However, there are number of issues yet to be resolved. Among them, one of the major challenges is the standardization of Web service composition. When a single web service cannot satisfy the given request, composition of web services need to be incorporated. In this paper, we address Web service composition problem with the signature-based service discovery and composition approach[30]. In the proposed approach, each web service is described by WSDL. Our design eliminates the need of complicated discovery agents like UDDI and also facilitates validation of the service before actually accessing it for integration. The composition problem has been modelled as a finite state machine, which means if the all the intermediate states are rightly composed then the final composition is successful. We propose a simple yet efficient algorithm DISCOMP for the discovery and composition. This paper analyses build time and runtime issues related to signature-based approach. We support our design decision with implementation and performance results obtained on a decentralized setup.**

Keywords- Web Service, machine*-to-machine interaction, Web service discovery and composition, parameter matching, design decision.*

I. INTRODUCTION

Web service composition is required when a request cannot be fulfilled by a single web service. In such cases, it is preferable to integrate existing web services to satisfy user's request. This value-added service and the process to integrate various pre-existing service is referred as composite Web service and Web service composition respectively. One of the challenging problems in Web service composition is to identify one or a set of Web service suitable for a given request. How to locate a Web service is an inseparable issue in web service composition. Locating or discovering, one or more related documents that describe a particular Web service is known as web service discovery. It is through the discovery process, web service clients learn that a web service exists and where to find the web service description document. Many researchers have focused on a centralized UDDI registry, as an effective method to solve a web service discovery problem. UDDI, centralized repositories, stores information describing web services developed and is published by service providers, and is used to query by service requesters. As an alternative of UDDI, specialized portals, such as *XMethods [1], RemoteMethods [2]*, etc exist. They assemble Web services via the manual registration and support a keyword-based service search by focused on crawlers. When number of web services grows, such a centralized approach becomes impractical. As a solution to this problem, systems based on ontology and/or using Peer-to-Peer (P2P) technologies have been introduced. Many researchers tackle the composition problem with their own flavors. The vocabulary representing the design issues in the service composition is the mixed in many literatures [22, 23, 24, 25, 26].

The web service standards lay down means to deliver the inter-operable heterogeneous distributed systems. The inter-operability is based on separating the definition of a web service from, its implementation details. Web services assume a mostly stateless service-oriented programming model whereby an application may, occasionally, requests a service from a remote application [30]. The UDDI standard addresses some aspects related to the publication and querying of enterprise business services. However, the description of services that is used in UDDI registries relies on meta-data (manually assigned property-value pairs) to convey the semantics of the services. Further, the querying mechanisms can only search for single services that address a particular business need; no provision is made for cases where a business need may be accommodated by a composition of services [20]. In this paper, we propose a method for discovering such compositions by focusing on the messages exchanged between services.

The remainder of the paper is organized as follows. Section 2 presents the related work. Web service composition strategy is proposed in section 3. Section 4 discuss about Web Service Composition architecture. The implementation aspects are highlighted in section 6 followed by conclusion & future work in section 7.

II. RELATED WORK

As web services become ubiquitous, a lot of effort is being spent in studying different ways of composing them to create more useful and complex services [3]. Four models of service composition have been proposed [4, 5] based on centralized and distributed flow of data and control messages between the services. The only two models are relevant for composition of third party web services are the centralised control centralized data flow model (centralized orchestration) and distributed control flow and distributed data flow model (decentralized orchestration).

978-0-7695-4461-8/11 $26.00 © 2011 IEEE
DOI 10.1109/SERVICES.2011.60

There have been many research works on Web services and a lot of issues of web service composition have been studied by many researchers [6, 7, 8, 9]. To study the potential of web services, it is required not to implement applications but also required to improve current theories of web service composition. Large numbers of Web service composition languages have been developed by researchers to study and compose Web services. However, the current Web service composition languages like WSDL-S [27, 28, 29], OWL-S [10, 11, 21], WSMO [12, 13], METEOR-S [14, 15] still have many limitations such as lack of elements to describe dynamic transformations among components of web service compositions. Above mentioned technologies handle only part of the issues and not deal with dynamic transformations. What is required to facilitate Web service composition is an ontology-based Web service semantic description language. Semantic language is capable to describe dynamic transformations among components of web services. Further, the semantic language ease the process of web service composition, thereby reducing end-users' demands such as time, integration efforts and composition expanses. Thus web service semantic description language should be able to describe the dynamic transformations among the components of a composite service in a precise manner.

[30] discusses the problem of querying a UDDI registry with a functional specification of a service, and getting in return a single service, or a composition of services that address the functional need. The approach relies on service signatures for discovering composed web services. Our approach is inspired by the strategy but eliminates the need of complicated discovery agents like UDDI by implementing a UDDI like registry that also facilitates validation of the service before actually accessing it for integration.

An ideal web semantic language should enable greater access not only to content but also to services on the Web. Users and software agents should be able to discover, invoke, compose, and monitor Web resources offering particular services and having particular properties, and should be able to do so with a high degree of automation if desired. Effective theories and powerful tools should be enabled by service descriptions, across the Web service lifecycle.

WSDL operates at the syntactic level and lacks the semantics needed to represent the requirements and capabilities of Web services. Semantics improves software reuse and discovery and significantly facilitates composition of Web services. WSDL-S [27, 28, 29] developed by LSDIS laboratory at the University of Georgia, is a Web service semantic model, and defines a mechanism to associate semantic annotation with Web services that are described using WSDL. This model assumes that the semantics relevant to the services already exist and maintained outside of the WSDL documents are refined from the WSDL document via WSDL extensibility elements. The type of semantics information would be useful in describing a web service encompasses the concepts defined by the semantic web community in OWL-S and other efforts [METEOR-S, WSMO]. The semantic information specified in this document includes definitions of the precondition, input, output and effects of web services operations. This approach offers multiple advantages over OWL-S. First, user can describe in an upwardly compatible way, both the semantics and operations level details in WSDL- a language with which the developer community is familiar. Secondly, by categorizing the semantic domain models, this allows Web service developers to annotate the web services with their choice of ontology language (such as UML or OWL).

WSMO [12, 13] is a standard of Web service fundamental framework, based on the Web Service Model Framework (WSMF), developed by European Semantic Systems Initiative (ESSI). The aim of WSMO, which is a concept model for describing semantic web services, is to define ontologies to describe web services from different points of view including web service discovery, invocation and composition. Compared with these issues, ESSI focus more on solving the problems about web service interoperability. WSMO is usually used to describe web services by utilizing four top level elements namely ontologies, which provide the terminology used by other WSMO elements. Web service descriptions; describe the functional and behavioural aspects of a Web service. Goals; represent user desires, and mediators. The mapping relationship between different element of WSMO and their interoperability are able to be illustrated by "Mediator" which is the kernel element to fix the web service incompatible problems on data and protocol levels.

METEOR-S [14, 15] is another implementation of web service semantic description language developed by Large Scale Distributed Information Systems (LSDIS). The aim of METEOR-S is to create a web semantic language that is capable to define and support the complete lifecycle of semantic web processes. The key features of this project are to build a new semantic web standard-WSDL-S that combines the semantic web techniques with the current industry web service standard-WSDL. The main idea is to map the message types (input, output) and operations of WSDL to the terms defined in the ontologies so that it may achieve the highly automatic web service discovery, validation, composition, etc. by utilizing the existing semantic reasoning techniques during the whole web service lifecycle.

OWL-S [10, 11, 21] (formally DAML-S, built upon DAML+OIL the predecessor of OWL) is a recently proposed semantic web language that builds upon the Ontology Web Language (OWL).

OWL-S is an ontology of services that makes possible to discover, compose and monitor web resource with a high degree of automation. To realize these aims, OWL-S implements three main parts: the service profile for advertising and discovering services; the process model,

which gives a detailed description of a service's operations; and the grounding, which provides details on how to interoperate with a service, via messages.

However, OWL-S still has some limitations such as the lack of elements to describe dynamic transformations of web service compositions which is an important feature. OWL-ES [15] (extended OWL-S) proposed by Chao Ma et al tries to capture this feature. They proposed a Fast Web Service Composition Approach (FWSCA) that adopts the OWL-ES. According to Chao Ma et al, OWL-ES is able to describe dynamic transformations among components of web service compositions by extending the extending the existing process model of OWL-S.

Work in [26], for web service discovery architecture based on x-SOA, organizes the method of web service discovery in an efficient and structured manner using an intermediate, Request Analyser (RA), between service provider and service consumer. The RA facilitates the processing of a plain text request query to resolve to a most appropriate web service. Our proposed algorithm, Layered Web Service Delivery Mechanism (LWSDM), works for a complete cycle of web-service discovery. The technologies discussed above are not yet fully developed and matured; much more effort is needed to use these technologies for the actual implementation.

III. WEB SERVICE COMPOSITION

In this section, we discuss our strategy of Web service composition.

Proposed Strategy

The various web service composition techniques proposed in the past not suitable for real time development environment. The factors that contribute to the overhead in the existing techniques are:

1) The number of web services involved in the composition increases the overhead.
2) Most of the existing approaches use semantic composition and thus the cost of building and maintaining the reference ontologies.

This to some extent can be overcome if service provider can register web services and service requester can analyse, test and contact service provider.

The composition problem can be represented as a Finite State Machine (FSM) or acceptor comprising of five tuple $(\Sigma, S, s_0, \delta, F)$ where

Σ – Set of input and output parameters.

S- Set of states (the set of web services in the business domain)

s_0 - Start state (the first web service of the composition problem)

F- the final state or the accept state (the final web services that will give the desired output)

δ – the state transition function $\Sigma \times S \rightarrow S$, *we assume the FSM to be deterministic in nature.*

The deterministic acceptor states that failure in any of the intermediate states would result in the halt of the machine in a non-accepting state and hence result in a unsuccessful composition.

Moreover testing of the web services in the UDDI like registry will reduce the chances of failure in composition.

IV. PROPOSED ARCHITECTURE

As shown in the Fig. 1, there are number of web services such as Web service$_1$, Web Service$_2$, Web Service$_3$. Each of the web service serves one specific problem. In our proposal, the user's requirement cannot fulfill just by using single web service. Hence, we propose a Controller Service that can compose various services in the pool and fulfills user's requirement based on the proposed algorithm DISCOMP. The controller service contacts registry service and reads the service endpoints. The service endpoints give the operation signature. Based on the users input and output, the controller service composes services in the list, solves the user requirement and defines the flow and sequence of request and response. Using this controller service, a web application can be developed where the user can directly invoke the composed service without knowing how this is being done in the background. A two-way communication occurs between various components of the proposed architecture.

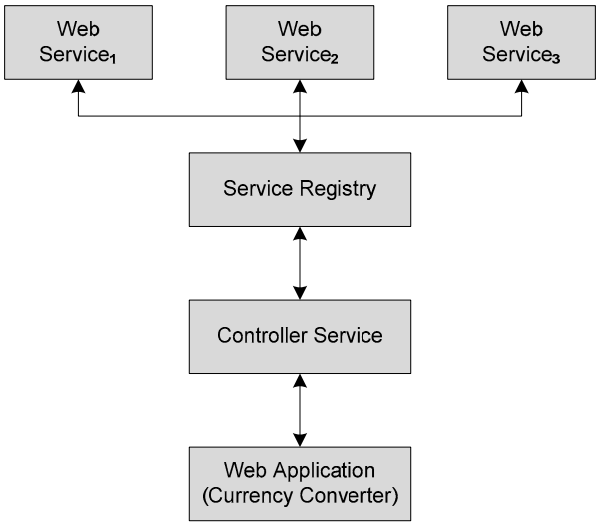

Fig. 1. The proposed architecture

The proposed algorithm for discovery and composition named as DISCOMP can be decomposed into following steps based the used FSM model:

a. Requirement analysis that will give the Σ
b. Finding the set of web services *S,* that fulfil the requirement.

c. Once S is determined s_0 and F which are subsets of S can be calculated depending on the requirement analysis.
d. Composition given by δ

Algorithm: DISCOMP
Requirement is defined as set of input parameters mapped onto a set of output parameters.
(Input : Requirement R: $R_{in} \rightarrow R_{out}$
Output: Web Service WS : $\langle in, out \rangle$ where *out*=R_{out}

1. FOR each R in input
   $\Sigma \leftarrow R_{in} \cup R_{out}$
   ENDFOR
2. Select the set of web services that match our requirement type
   $S \leftarrow \{WS_i\}$
3. FOR each $\{WS_i\}$ in S
   if R_{in} matches $WS_i \langle in \rangle$ then
   $s_0 \leftarrow \{WS_i\}$
   ENDFOR
4. Defining δ //the composition step
   Starting with s_0 <in,out>
   REPEAT
   FOR each $\{WS_i\}$ in S
   if s_0 <out> = $WS_i \langle in \rangle$ then
   else
   composition unsuccessful
   ENDFOR
   UNTIL s_0 <out> = R <out>
   new $s0 \leftarrow WS_i$
5. F= s_0 is the accepting state

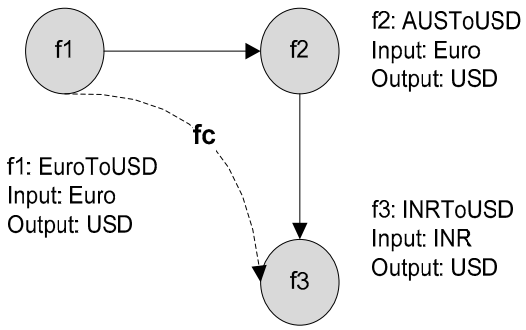

Fig. 2. The composition graph

If more than two web services match our-input or any intermediate inputs or outputs, we can choose any of the web service. This issue can be solved with the QoS of service.
To understand the proposed model, we develop a web application "Currency Conversion" as shown in Fig. 8.The web services available, with their input and output are:

- EuroToUSD(input Euro, output USD),
- AUSToUSD(input AUS, output USD),
- INRToUSD(input INR, output USD),

Using composition model in Fig.2, we can convert different currencies to and from other currency and vice-versa; without re-writing the web services. We can match the input, output of web services and accordingly, we can compose web services as our requirement. This is a prototype implementation to show our model works perfectly. We have developed three web services with two operations in each and composed to a web service with twelve operations. To ease the composition, we have developed a specialized portal which acts like UDDI registry, where web services can be looked up, analysed and tested.

V. IMPLEMENTATION

The following are the software and hardware requirements for the implementation.

Software Requirements

Operating Systems- Windows Server 2003 or Windows server 2008 x 86 and\or 64 bit.
Development IDE- Visual Studio 2008 Professional Edition.
Database Engine - Microsoft SQL Server 2005 or 2008.
Database Editor - Microsoft SQL Server Management Studio Express.
Application Server- Internet Information Services (IIS) 5.1Package/Library Requirement - .Net framework 3.5 or more.

Hardware Requirements

1. A system to develop web services and web application with the following configuration.
   Minimum: 1.6 GHz CPU, 384 MB RAM, 1024 X 768 display, 5400 RPM hard disk.
2. A system with Windows Server 2003 or Windows Server 2008 x86 and\or 64 bit editions
3. 3+ systems to deploy web services with IIS 5.1+ with the following configuration.
   Minimum: 1.6 GHz CPU, 384 MB RAM, 1024 x 768 display, 5400 RPM hard disk with 10 GB or higher Hard Disk.

Initially, we developed three independent web services with two operations in each. These web services can convert currency from one domain to other domain and vice versa. As shown in the Fig. 5, the web services are EuroToUSD with operations EuroToUSD() and USDToEuro(), AUSToUSD with operations AUSToUSD() and USDToAUS() and third web service INRToUSD with operations INRToUSD() and USDToINR(). These are independent web services and can serve only the two operations as mentioned above. If we need to convert currency from other format, either we have to develop web service for that format or we may compose already available

web services. We are proposing the later approach. With our approach, we can save lot of development time, cost. This is the concept provided by Service Oriented Architecture (SOA) and known as reusability.

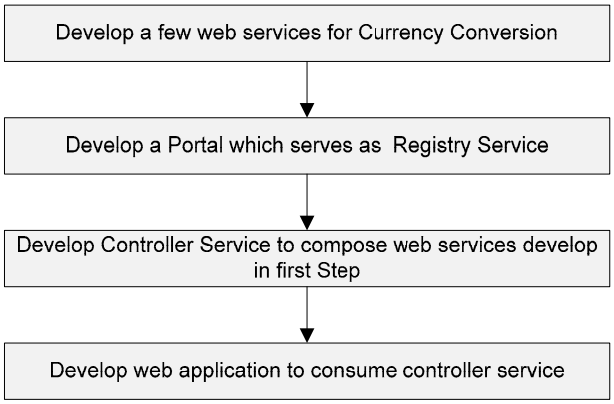

Fig. 3. The implementation steps

We have divided the implementation part into four phases: (as shown in the Fig. 3). In the first phase, we develop Web services for three currency formats. If such Web services are already available, then we can also use these services. The second phase is to develop a specialized portal like *XMethods [1], RemoteMethods [2], etc*, as an alternative to UDDI registry service, to publish Web services developed in the first phase. This portal can be used by Service Providers (SP) to register Web services and by Service Consumers (SC) to search required Web services. SC can analyse WSDL, test Web services. If his/her requirement is met, SC can contact SP. Agreed upon the certain service policy they can fulfill their business requirements. The portal has administrator pages, where the administrators can monitor the Web services. They can edit, modify, and delete Web services, Service Providers, and other users (low power admin users) in the system. The low - power admin users can monitor Web services, Service Provider. The low-power admin user and SP can register in the system. With the limitation of paper length, the detail design and implementation of this phase is limited just to introduction. The third phase in our implementation part is to design a controller service to compose Web services developed in the first phase. This is the most important part in the entire implementation part. There are different Web Service composition model such as Signature-based, Semantic Annotation-based, QoS description based, Process Behaviour-based, etc.

To our knowledge, the current technology and research has minimal support to other than the signature-based composition model. The developers in the industry are provided with the fully developed web services. Generally, signature of such web services remains unchanged. Since the signature remains unchanged, the signature-based model gives the optimal performance. If still there are any changes in the web services signature, the only requirement is to update the web references and change the web application accordingly. The overhead in updating web references and change in web application is minimal compared to the automatic composition of Web services.

The basic idea behind signature-based composition model is to freeze the design of Web services, list the Web services interface that are involved in the composition model and match the interfaces of Web services i.e. match the output of a Web service to the input of other and vice versa. The composition model gives the best result if Web services are designed properly and there is no change in the web service interface and in the composition model.

The fourth phase is to develop a Web application that can consume the controller service developed in the phase third. The Web application is for a company (travel agent) which will use the composite Web services consisting of Web services from airlines, hotels and payment services. The controller service automates the whole process of hotel reservation, ticket booking and payment system. Though there involve different Web services, the user has no knowledge of different Web services acting in the automation process.

The basic web services with two primitive operations are shown in the Fig. 4. These web services are developed in different machine with the same hardware and software requirements. But we can develop and register web services developed in different platform in different development such as Java, gSoap, etc.

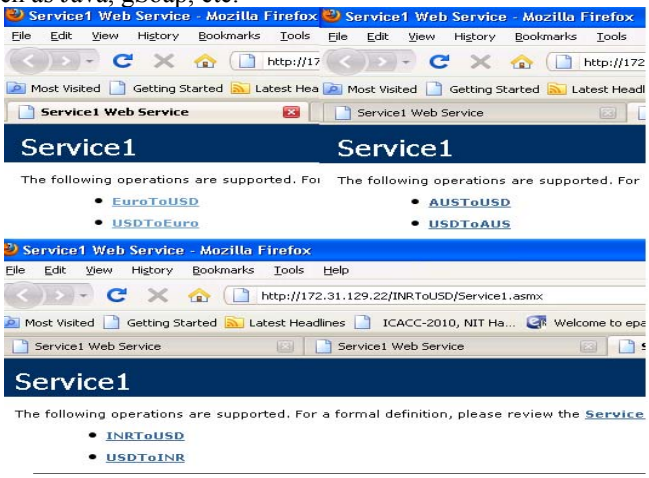

Fig. 4. The independent web services with operations

To serve the user requirements that cannot be solved by available web services, we propose a signature-based web service composition as discussed in section 4. To discover web services and study their operations, inputs and outputs; we need registry service; where service publisher can publish their services and service consumer can analyse, test and contact the service provider. We have developed a

specialized portal where service provider can register web services and service requester can analyse, test and contact service provider. Fig. 5 shows the various web services registered by various publishers. We have developed the portal in .Net technology using C# as a development language and MS-SQL as database engine. Though our implementation is in MS.Net with Windows Operating System, the same design can be implemented in other platform also.

Name of Service	Publisher	OneLineDescription	Implementation
CalcService	CalcService	CalcServiceDescription	MS.Net
HelloService	HelloService	HelloServiceDescription	MS.Net
TestService	TestService	this is a test service description	gSOAP
Add	Add	addserviceonelinedescription	MS.Net
AddnHelloWorld	AddnHelloWorld	addandhelloworldwebservice	Java
EuroToUSD	EuroToUSD	Euro to USD currency conversion.	MS.Net
AUS To USD Currency Converter	AUS To USD Currency Converter	Description for AUS to USD currency converter	MS.Net
INR to USD Currency Converter	INR to USD Currency Converter	Indian rupees to US dollor currency converter.	MS.net

Fig. 5. The Portal with various web services

Our portal ensures portability over other existing UDDI based Registry service. In our portal, heterogeneous web services can be registered such as web services implemented in C#, Java, gSoap, etc.

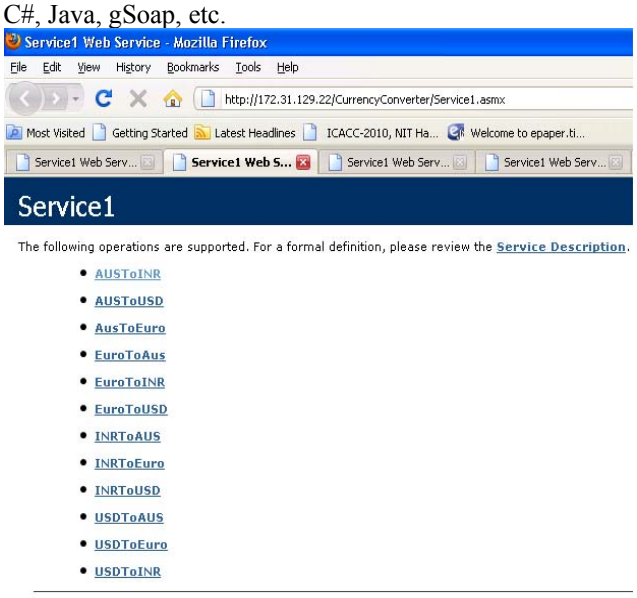

Fig. 6. The Composed web service

Using the independent web services shown in Fig. 4, and the composition model discussed in section 4, Fig. 2; we are able to generate composed web service as shown in the Fig. 6. This web service has the previously defined operations and composed operations. The number of operations in the composed web services increases with the properly designed primitive operations in the independent web services and user requirement. Just with three web service and six primitive operations, we are able to obtain twelve operations.

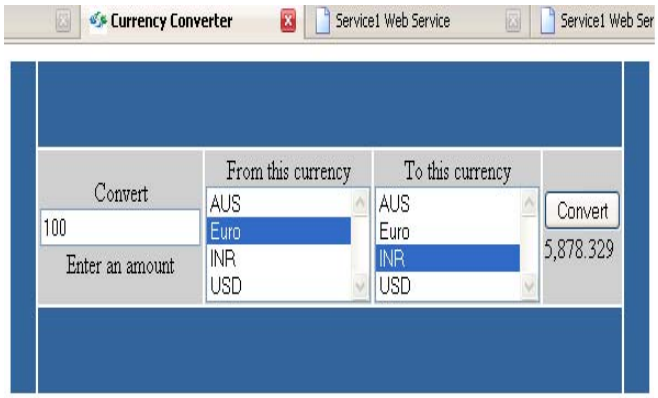

Fig. 7. Currency Converter

We have developed a web application that consumes the controller service. This application has both the primitive operations as well as the composed operations. We don't need to code for each of the intermediate operations. In our case, we have developed primitive operations, but if web services are already available we can use them also. Only requirement is one or compositions of various web services that can fulfill our requirement. Our implementation shows the reusability of components.

VI. CONCLUSIONS AND FUTURE WORK

To properly compose Web services, we need a description of their semantics. As discussed the availability of ample semantic information is a major issue. Cost wise also exploiting semantic and ontological information for web service composition is intensive. We argued that by narrowing the search space to services that are within particular business domain (or industry), we reduce the chances of accidental matches and overwhelming set of web services. Hence, the service composition problem is framed as the problem of composing functions in order to producing a desired set of output messages, given a set of input messages. We have implemented an algorithm using signature based strategy with a FSM approach which returns a successful output only when the intermediate compositions are successfully done. Moreover instead of using the centralized UDDI, we have developed a customized UDDI like registry that allows registering and composing heterogeneous web services. The portal also offers validating the web services before composing them. Such validations would reduce the chances of unsuccessful compositions.

In this paper, we tried to explain the problems with the current web service composition languages and proposed the straightforward method which has more benefit then the semantic-based Web service composition model. The only issue with signature-based model is to update the Web references in the Web applications and necessary change that is going to occur in the Web application.

This paper is a model to our future work. We will apply the proposed model to a traveller who needs to plan a trip. This can do this in one of two ways. One could either look for travel services that combine all three services, such as *expedia.com* [17], *orbitz.com* [18] or *travelocity.com* [19]. Alternatively, a traveller would need to look for the three services separately (flights, hotels, and car), and compose them manually, by invoking them in the proper sequence, using the appropriate constraints. Our next phase is to implement the later option.

REFERENCES

[1] xMETHOD,[Online].Available. ttp://xmethods.net/ve2/index.po

[2] REMOTEMETHODS,[Online]. Available.http://www.remotemethods.com/

[3] Benatallah, M.Dumas, M.C. Fauvet, F.Rabin, and Q.Z. Sheng. Overview of Some Patterns for Architecture and Managing Composite Web Services. In ACM SiGecam Exchanges, Volume 3.3, pages 9 -16, 2002

[4] Liu, K.H. Law, and G Wiederhold. Analysis of Integration Models for Service Composition. In Proceedings of the third international workshop on Software performance, Rome, Italy, July, 2002.

[5] Yew, A. Strand, A. Liotta, and G. Pavlou. Aggreagation of Composite Location-Aware Services for Mobile Cellular Networks. In Proceedings of 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, Germany, October, 2003.

[6] Patil, et al. Meteor-s Web service Annotation Framework. In Proceeding of the 13th International Conference on World Wide Web, 2004.

[7] Wu., et al. Automating daml-s web services composition using shop-2. In proceeding of International Semantic Web Conferences, 2003.

[8] S. Ponnekanti and A. Fox. SWORD: A developer Toolkit for Web Services Composition. In Proceeding of the 11th International Conference on World Wide Web, 2002.

[9] S. Dustdar and W. Schreiner. A Survey on Web Services Composition. Journal of Web and Grid Services, Volume 1, Issue 1, pages: 1-30, 2005.

[10] S. Mcllraith, T. Son. Adapting Golog for composition of semantic web services. In Proceeding of the Eighth International Conference on Knowledge Representation and Reasoning, France, April 2002.

[11] Shmueli. Architecture for internal Web services deployment. In Proceeding of the 27th International Conference on Very Large DataBases. Morgan Kaufmann Publishers, pages: 641- 644, Roma, 2001.

[12] D. Roman, et al. Web Service Modelling Ontology, WSMO Final Draft, April 2005. [Online]. Available. http://www.wsmo.org/TR/d2/v1.2/

[13] Keller U, et al. WSMO: Web service discovery. [Online]. Available: http://www.wsmo.org/2004/d5/d5.1/v0.1/20041112, 2004

[14] S. Staab, et al. Web Services: Been there, done that? IEEE Intelligent Systems, IEEE Computer Society, Volume 18, Issue 1, pp. 72-85, 2003.

[15] Chao Ma; Yanxiang He, "An Approach for Visualization and Formalization of Web Service Composition," International Conference on Web Information Systems and Mingin, Shanghai, pp. 342–346, Dec. 2009.

[16] Hugo Haas, "Web service use case: Travel reservation", [Online]. Available. http://www.w3.ogr/2002/06/ws-example, May 2002.

[17] Expedia, [Online]. Available. http://www.expedia.com/

[18] ORBITZ, [Online]. Available. http://www.orbitz.com/

[19] Travelocity, [Online]. Available. http://www.travelocity.com/

[20] Hafedh Mili, Radhouane Ben Tamrout, and Abdel Obaid, "JRegistry: an extensible UDDI Registry" in Reports of NOTERE'2005 (High Tech distribution), Gatineau, Quebec, 29 August-1 September 2005, pp . 115-128.

[21] D. Martin, et al, OWL-S: Semantic Markup for Web Services, W3C Member Submission, 22 November, 2004. [Online]. Available. http://www.w3.org/Submission/OWL-S/

[22] Faisal Mustafa, T.L. McCluskey, "Dynamic Web Service Composition", International conference on Computer Engineering and Technology, ICCET '09, Jan 2009.

[23] Xuanzhe Lin, et al. "Discovering Homogeneous Web Service Community in the User-Centric Web Environment", IEEE Transaction on Service Computing, Vol. 2 – No. 2, April 2009.

[24] Aviv Segev and Eran Toch, "Context-Based Matching and Ranking of Web Services for Composition", IEEE Transaction on Service Computing, Vol. 2 – No. 3, Sep. 2009.

[25] Howard Foster, et al., "An Integrated Workbench for Model-Based Engineering of Service Compositions", IEEE Transactions on Service Computing, Vol.3 – No.2, June 2010.

[26] Shrabani Mallick, D. S Kushwaha " An Efficient Web service Discovery Architecture", International Journal of Computer Applications, Volume 3- No 12 , July 2010

[27] R. Akkiraju, J. Farrell, J.Miller, M. Nagarajan, M. Schmidt, A. Sheth, K. Verma, " Web Service Semantics – WSDL-S.", A joint UGA-IBM Technical Note, Version 1.0, April 18, 2005. [Online]. Available: http://lsdis.cs.uga.edu/library/download/WSDL-S-V1.pdf

[28] A. Sheth, K. Verma, J. Miller and P. Rajasekaran (University of Georgia), " Enhancing Web Service Description using WSDL-S", Research Industry Technology Exchange at EclipseCon 2005, Burlingame, CA, Feb. 28 – Mar. 3 2005. [Online]. Available: http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/METEOR-S-eclipsecon.pdf

[29] K. Sivashanmugam, K. Verma, A. Sheth, J. Miller, Adding Semantics to Web Services Standards, Proceedings of the 1st International Conference on Web Services (ICWS'03), Las Vegas, Nevada (June 2003) pp. 395 - 401. [Online]. Available: http://lsdis.cs.uga.edu/lib/download/SVSM03-ICWS-final.pdf

[30] Alkamari, A.; Mili, H.; Obaid, A.; e-Technologies, 2008 International MCETECH Conference Digital Object Identifier: 10.1109/MCETECH.2008.34 Publication Year: 2008 , Page(s): 104 - 115