Web Service Composition Approaches to Support Dynamic E-Business Systems

Sayed Gholam Hassan Tabatabaei*, gtsayed2@siswa.utm.my
Wan Mohd Nasir Wan Kadir*, wnasir@utm.my
Suhaimi Ibrahim*, suhaimiibrahim@utm.my
*Department of Software Engineering, Faculty of Computer Science and Information Systems,
University of Technology Malaysia (UTM), 81310 Skudai, Johor, Malaysia

Abstract

Nowadays, since many companies decide to implement and publish their core business and outsource other application services over Internet, the number of Web services has dramatically increased. Thus, Web service based dynamic E-Business, as the next evolutionary step of E-Business, becomes an important task in the industry. In many cases, a single service is not sufficient to fulfil the user's request and services should be combined together. Thus, dynamic composition of Web services to provide considerable flexibility for modifying and extending the operations of E-Business systems during runtime, is one of the recent critical issues. A number of approaches have been presented, to solve this problem. In this paper, we classify the competing approaches to three categories (Workflow-based, XML-based, and Ontology-based), which can be complementary, and describe them. Then, we compare these approaches based on some benchmarks (like QoS, scalability, and correctness).

1. Introduction

The term "Web services" has been used very often nowadays. According to W3C, "A Web service is a software system identified by a URI [1], whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols "[2]. Another definition of Web service is provided by IBM [3], A Web service is a software interface that describes a collection of operations that can be accessed over the network through standardized XML messaging. It uses protocols based on the XML language to describe an operation to execute or data to exchange with another Web service.

Basically, Web service operation can be described as follows. First of all, a *client* program via a yellow page (UDDI) [3] finds a *Web services server* that can fulfil certain requirements from, and acquire a detailed specification from WSDL [4] about the service. Then, the client sends a *request* to the server through a standard message protocol (SOAP) [5], and in return receives a *response* from the server. With interpreting XML tags, applications can interpret the operations and data much easier than conventional programming interface.

Currently, an increasing amount of companies and organizations implement their applications over Internet. Thus, the ability to efficiently and effectively select and integrate inter-organizational and heterogeneous services on the Web at runtime is an important step towards the development of the Web service applications. Recent research studies how to specify them (in a formal and expressive enough language), how to (automatically) compose them, how to discover them (on the Internet) and how to ensure their correctness. We focus on Web Service composition (WSC).

When no atomic Web service (WS) can satisfy the user's requirements, there should be a possibility to combine existing services together in order to accomplish the request. For example, if a user wants to participate on one international conference, it is not sufficient to register only, but he should also take care of booking a flight, reserving a hotel, renting a car, and so on. Therefore, the notion of composite services is starting to be used as a collection of services combined to achieve a user's request. Indeed, this composition is considered as a single service from a user's point of view, even though it is composed of several web services.This trend has inaugurated a considerable number of research efforts on the WSC both in academia and in industry.

A composite service, in many ways, is similar to a workflow [6]. The definition of a composite service includes a set of atomic services together with the control and data flow among the services. Similarly, a workflow has to specify the flow of work items. The dynamic workflow approaches provide the means to bind the abstract nodes with the concrete resources or services automatically.

In the research related to Web services, several initiatives have been conducted with the intention to provide platforms and languages for WSC such as Business Process Execution Language for Web Services (BPEL4WS) [7]. Nowadays some languages have ability to support semantic representations of the WSs available on the Internet such as the Web Ontology Language for Web Services OWL-S [8] and the Web Service Modeling Ontology WSMO [9].Although all of these efforts, the WSC still is a highly complex task. The complexity, in general, comes from the following sources [10]. First, the number of services available over the Web increases dramatically during the recent years, and one can expect to have a huge Web service repository to be searched. Second, Web services can be created and updated on the fly, thus the composition system needs to detect the updating at runtime and the decision should be made based on

the up to date information. Third, Web services can be developed by different organizations, which use different concept models to describe the services, however, there does not exist a unique language to define and evaluate the Web services in an identical means.

In this paper, we focus on the WSC problem and offer a survey of recent approaches that provide automation to Web service composition. The automation means that either the approach can generate the process model automatically, or the method can locate the correct services if an abstract process model is given [11]. We then compare them with respect to the set of benchmarks. By offering this overview and classification of existing proposals for Web service composition, as well as a constructive review of them, we hope to help service-composition designers and developers focus their efforts and deliver more usable solutions, while also addressing the technology's critical requirements.

2. Classification of the WSC Approaches

We can classify the WSC approaches using the following three aspects which can be complementary:

2.1. Workflow-based WSC Approaches

Workflow-based composition methods have been distinguished to the static and dynamic workflow generation [11].

The *Static Composition* means that the requester before starting the composition planning should build an abstract process model. The abstract process model includes a set of tasks and their data dependency. Each task contains a query clause that is used to search the single WS to fulfil the task. Thus, just the selection and binding of single WS is done automatically by software.

However, in *Dynamic Composition*, creating process model and selecting single WSs are done automatically. The requester has to specify several constraints, such as the user's preference.

In this section we describe two principal approaches, namely:

2.1.1. EFlow

EFlow [12] is a platform for the specification, enactment and management of WSC which uses a static workflow generation method. In that case, WSC is modelled by a graph that defines the order of execution among the nodes in the process. The graph is created manually but it can be updated dynamically. The graph may include service (represent the invocation of WS), decision (specify the alternatives and rules controlling the execution flow) and event nodes (enable service processes to send and receive several types of events). Arcs in the graph denote the execution dependency among the nodes. The definition of a service node contains a search recipe that can be used to query actual service. As the service node is started, the search recipe is executed, returning a reference to a specific service.

2.1.2. Polymorphic Process Model

Polymorphic Process Model (PPM) [13] uses a method that synthesizes the static and dynamic WSC. The static setting is supported by reference process-based multi-enterprise processes. These processes include abstract sub processes that have functionality description but lack implementation. The abstract subprocesses are implemented by service and bined at runtime.

The dynamic part of PPM is supported by service-based processes. Here, a service is modeled by a state machine that specifies that possible states of a service and their transitions. Transitions are caused by service operation invocations or internal service transitions. In the setting, the dynamic service composition is enabled by the reasoning based on state machine.

2.2. XML-based WSC Approaches

Currently there are two main approaches in the field of XML-based WSC [14]:

Web Service Orchestration: combines available WSs by adding a central coordinator (the orchestrator) that is responsible for invoking and combining the single subactivities. An orchestration also describes how other WSs are composed in order to achieve the required functionality of the WS.

Web Service Choreography, instead does not assume a central coordinator but rather defines complex tasks via the definition of the conversation that should be undertaken by each participant; the overall activity is then achieved as the composition of peer-to-peer interactions among the collaborating WSs. A Choreography also describes the external visible behavior of the WS. Choreography languages are still in an introductory phase of definition .WS-CDL [15] is example of this approach.

One of the most important orchestration languages namely BPEL4WS is defined as follows.

2.2.1. BPEL4WS

This XML-based language was designed to enable the coordination and composition of a set of WSs. Also this language is based on WSDL [4], which is essentially an interface description language for WS providers. In fact, BPEL4WS is a merge between XLang and WSFL, but all of them are considered as a web service flow language [16]. WSC using BPEL4WS enables the definition of a new web service by composing a set of existing ones. The interface of the composite service is described as a collection of WSDL *PortTypes*. A BPEL4WS process defines the roles involved in a composition as abstract processes. A buyer and a seller are examples of two roles. They are expressed using partner link definitions. We can have a role for each web service that is composed and does some activity. In order to integrate services, they are treated as partners that fill roles [17]. BPEL4WS depends directly on the WSDL of the service. A business

process defines how to coordinate the interactions between a process instance and its partners. Thus, a BPEL4WS process provides one or more WSDL services. The BPEL4WS process is defined only in an abstract manner, allowing only references to service *portTypes* in the *partnerLink* [7]. Each partner is characterized by a partner link and a role name. In sum, business process is used to create an organizer that point to each service endpoint that will be actually executed.

2.3. Ontology-based WSC Approaches

The Semantic Web [18] allows the representation and exchange of information in a meaningful way, facilitating automated processing of descriptions on the Web. Annotations on the Semantic Web express links between information resources on the Web and connect information resources to formal terminologies. These connective structures are called ontologies.

Ontologies are used as data models throughout these types of approaches, meaning that all resource descriptions and all data interchanged during service usage are based on ontologies. Ontologies are a widely accepted state-of-the-art knowledge representation, and have thus been identified as the central enabling technology for the Semantic Web. The extensive usage of ontologies allows semantically enhanced information processing and support for interoperability.

In this section we consider two principal approaches, namely:

2.3.1. OWL-S:

OWL-S is an OWL service ontology for describing various aspects of Web services [19]. OWL-S has tried to adopt existing Semantic Web recommendations yet still maintain bindings to the world of Web services by linking OWL-S descriptions to existing WSDL descriptions [20]. In the following, we describe the four top-level concepts of the OWL-S ontology which are illustrated in Figure 1.

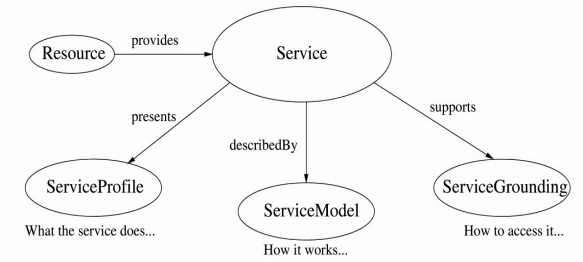

Figure 1. OWL-S Conceptual Model

- *SERVICE:* The SERVICE concept serves as an organizational point of reference for declaring WSs. Every WS is declared by creating a SERVICE instance. It links the remaining three elements of a WS through properties like PRESENTS, DESCRIBEDBY and SUPPORTS.

- *SERVICE PROFILE:* declares what a SERVICE does in order to advertise and serves as a template for service requests at a high level, therefore enabling discovery and matchmaking. The profile includes nonfunctional aspects such as references to existing categorization schemes or ontologies, provider information, and the quality rating of the service. The most essential information presented in the profile, however, is the specification of what functionality the service provides. Information transformation is represented by *inputs* and *outputs*; the change in the state of the real world caused by the execution of the service is represented by *preconditions* and *effects*. Inputs and outputs refer to OWL classes describing the types of instances to be sent to the service and the respective responses to be expected. A feasible problem is that the semantics of these conditions is not covered by the (description logics) expressivity of the OWL-S ontology itself, but by reference to these languages. So, parties need to consent on the language for expressing conditions and also the notions of a "match" which is not addressed in the standard.

- *SERVICE MODEL:* SERVICE could be described by a SERVICE MODEL which describes how a service works to enable invocation, enactment, composition, monitoring, and recovery. The service model views the interactions of the service as a process. A process is not necessarily a program to be executed, but rather a specification of ways in which a client may interact with a service. OWL-S distinguishes between single processes, simple processes which are simple operations, and composite processes which are built up from single processes by standard workflow constructs such as sequence, split, or join to determine the control flow, plus additional dataflow information (which outputs are routed to which inputs within the workflow).But, a feasible problem is that the semantics of the workflow constructs is not expressible in the description logics underlying OWL, for which reason this semantics has been externally defined [21].

- *SERVICE GROUNDING:* In order to map to the Web service world, an OWL service can *support* a grounding which maps the constructs of the PROCESS MODEL to detailed specifications of message formats, protocols, and others. In fact, SERVICE GROUNDING describes how to use a WS (i.e. how clients can actually invoke it). In detail, OWL-S allows one to map single processes to WSDL operations and their inputs and outputs to WSDL messages. However, OWL-S does not restrict itself to WSDL as the only underlying service technology, but should be understood as a general service description ontology, and is extensible to other grounding mechanisms.

2.3.2. WSMO:

WSMO defines a model to describe semantic WSs, based on the conceptual design set up in the WS Modelling Framework WSMF [22]. Following the key aspects identified in the Web Service Modeling

Framework, WSMO identifies four top-level elements as the main concepts: *Ontologies*, *Web services*, *Goals*, and *Mediators* (Figure 2). These have to be described in order to describe Semantic Web services [21]:

- *Ontologies:* provide the (domain specific) terminologies used and is the key element for the success of Semantic Web services. Furthermore, they use formal semantics to connect machine and human terminologies. WSMO describes an epistemology of ontologies, i.e. the conceptual building blocks such as concepts and relations.
- *Web services:* are computational entities that provide some value in a certain domain. They are described from three different aspects: non-functional properties, functionality and behavior.
- *Goals:* describe aspects related to user desires with respect to the requested functionality, i.e. they specify the objectives of a client when consulting a WS. Thus they are an individual top-level entity in WSMO.
- *Mediators:* describe elements that handle interoperability problems between different elements, for example two different ontologies or services. Mediators can be used to resolve incompatibilities appearing between different terminologies (data level), to communicate between services (protocol level), and to combine Web services and goals (process level). Their existence allows one to link possibly heterogeneous resources.

Besides these main elements, *Non-Functional* properties such as *accuracy, network-related QoS, performance, scalability,* and *reliability* are used in the definition of WSMO elements that can be used by all its modeling elements. Furthermore, there is a formal language to describe ontologies and Semantic Web services called WSML (Web Service Modeling Language) which contain all aspects of Web service descriptions identified by WSMO. To introduce aspects of Semantic Web services in WSMO, the Meta-Object Facility (MOF) [23] specification is used, which defines an abstract language and framework for specifying, constructing, and managing technology-neutral metamodels. MOF defines a metadata architecture consisting of four layers, namely *information, model, metamodel, and metametamodel.* In addition, WSMX (Web Service Modeling eXecution environment) is the reference implementation of WSMO, which is an execution environment for business application integration. [24].The goal is to increase business processes automation while providing scalable integration solutions.

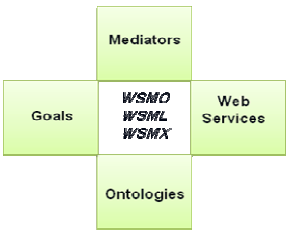

Figure 2. Four top-level elements of WSMO

2.3.3. Differences in the conceptual models of OWL-S and WSMO:

- OWL-S is specified using the Web Ontology Language, while WSMO uses an abstract MOF model. On the other hand, OWL-S defines its Meta model in the same language that it uses for concrete service descriptions. However, WSMO's basis in the abstract MOF model successfully avoids this problem.
- OWL-S does not separate what the user wants from what the WS provides. The service profile of a WS (such as its name, a human-readable description and contact information) is not explicitly based on standard metadata specification. WSMO recommends the use of widely-accepted vocabularies (like the Dublin Core [25]).
- OWL-S has need to "retrofit" more expressive languages into the OWL framework which then opens up new research questions on how they should interact. WSMO has overcome this problem by using WSML.
- Non-functional properties in OWL-S are restricted to the service profile. However, this can be expressed in any WSMO element.
- OWL-S intends by combining various notations and semantics with OWL for the description of service conditions and effects, whereas WSML provides one uniform language for capability descriptions.

2.3.4. Similarities in the conceptual models of OWL-S and WSMO:

- A service profile in OWL-S is close to a capability of a service or goal in WSMO. But, WSMO makes a conceptual distinction between the provider's and the requester's view, which is not made in OWL-S.
- The process model of OWL-S is conceptually similar to the WSMO service and goal interfaces. However, the distinction between the description of external behavior (choreography interface) and of the internal behavior (orchestration interface) is not made explicit in OWL-S.
- As for the grounding, WSMO and OWL-S adopt similar ideas with respect to binding to WSDL. However, the grounding is not a top-level concept in WSMO, but is instead integrated into the WSMO interfaces.

3. Comparative Evaluation

In this section we compare the above WSC approaches with respect to the following benchmarks. We claim that, any approach to WSC should satisfy these set of benchmarks. We consider an approach as a "*good"* quality approach, if the approach can provide all aspects of the benchmark. If an approach can provide part of what the benchmark expects, is considered as an *"average"* quality approach based on that benchmark. If an approach

does not provide what the benchmark expects, is considered as a "*low"* quality approach regarding that benchmark. The result can be seen in Table 1.

3.1. QoS

Currently, considering *quality of service* (QoS) to describe as nonfunctional properties is one of the critical issues in the WSC. When referring to QoS, nonfunctional properties such as performance, cost, or reliability are intended. Since a composed service uses other services to form itself, its quality depends on the WSs it uses. To be accepted by its customers, a business should try to provide good quality regarding the customers' requirements to a composed WS.

QoS aspects are considered when selecting WS candidates for a composition. The web services' directory service UDDI only considers functional aspects when services are searched for, thus a *QoS broker* is developed which complements UDDI by non-functional aspects. By defining aggregation formulas for several QoS aspects which are applied to simple composition patterns, the whole workflow pattern of a composed service can be collapsed stepwise, and each time the most suitable collection of simple services is selected.

As QoS information assigned with each basic service, *performance*, *financial*, *reliability*, and *availability* were chosen.

- *Performance:* This represents how fast a Web service request can be completed. According to [26], performance can be measured in terms of throughput, latency, execution time, and transaction time. The response time of a Web service can also be a measure of the performance. High-quality Web services should provide higher throughput, lower latency, lower execution time, faster transaction time and faster response time.
- *Financial:* This represents the cost-related and charging-related properties of a Web service [27]. This property is a complex property, which includes charging styles (e.g. per request or delivery, per unit of measure, granularity, etc.), aspects of settlement such as the settlement model (transactional vs. rental) and a settlement contract, payment obligations, and payment instruments.
- *Reliability:* This represents the ability of a Web service to perform its functions (that is, to maintain its Web service quality). It can be measured by the number of failures of the Web Service in a certain time interval.
- *Availability:* the probability that a WS is available at any given time, measured as the percentage of time a WS is available over an extended period of time.

The management of QoS when composing WSs requires a careful consideration of the QoS benchmarks of the constituent WSs. To enable the specification and monitoring of QoS aspects like performance, financial, reliability, and availability, various approaches have been developed. An excellent research for considering QoS aspects in WSC can be found in [28].

Most of workflow based approaches like EFlow neglect specification of nonfunctional QoS properties such as *security*, *dependability*, or *performance*. Thus these approaches can be considered as *low* quality approaches regarding the QoS. Also, BPEL4WS does not directly support the specification of most QoS measures which can be considered as an *average* quality approach in this criterion. However, in OWL-S, QoS measures such as availability are specified as service parameters in the WS description definition, but the specification of metrics and guarantees is missing. Moreover, there is no way to specify functional relations between metrics and therefore quality-aware WS discovery is not feasible [14].Therefore we can consider this approach with *average* quality regarding the QoS. Finally, QoS (Nonfunctional properties) are applicable to all the definitions of WSMO elements such as *Ontologies*, *Web services*, *Goals*, and *Mediators*. Which QoS properties apply to which WSMO element is specified in the description of each WSMO elements. Thus this approach can be measured with *good* quality with respect to the QoS.

3.2. Automatic Composition

Many composition approaches aim to automate composition, which promises faster application development and safer reuse, and facilitates user interaction with complex service sets. With automated composition, the end user or application developer specifies a goal (a business goal expressed in a description language or mathematical notation) and an "intelligent" composition engine selects adequate services and offers the composition transparently to the user. The main problems are in how to identify candidate services, compose them, and verify how closely they match a request [29]. Generally, we cannot assign any of the above approaches as an automated approach. Although, most of these approaches like OWL-S and WSMO can be assigned as a semi automated approach which can be considered as an *average* quality approach regarding the automatic composition.

3.3. Composition Scalability

This represents the ability of the WS to process multiple requests in a certain time interval. It can be measured by the number of requests resolved in a certain time interval.

Composing two WSs is not the same as composing ten or more WSs. In a real-world scenario, end users will typically want to interact with many WSs while enterprise applications will invoke chains of possibly several hundred services. Thus, one of the important issues is how the proposed approaches scale with the number of WSs involved.

In BPEL4WS, since XML files have increased a lot, WSC is a bit tiresome. BPEL4WS composition can be modularized, because this approach is recursive. But, BPEL4WS has no standard graphical notation. Some orchestration servers offer graphical representation for descriptions, such as UML, but they don't map one-to-one to complex constructs of

BPEL4WS. Thus, this approach achieves *average* scalability.
Finally, OWL-S and WSMO have similar issues. The Web component approach achieves *good* scalability with class definitions, but requires additional time for mapping and synchronization between class definitions and XML.

3.4. Correctness
Verifying correctness depends on the WS and composition specifications. The composition of WSs may lead to large and complex systems of parallel executing WSs. An important aspect of such systems is the correctness of their behavior. All of these approaches offer no direct support for the verification of WSC at design time, to evaluate in this way its correctness. For example, BPEL4WS is a Turing complete language dealing more with implementation than specification, and thus it's difficult to provide a formalism to verify the correctness of BPEL4WS flows [30]. Consequently, all of these approaches can be considered as *low* quality approaches regarding the correctness.

Table 1. Comparing WSC approaches

	Benchmarks			
Approaches	QoS	(Semi) Automatic	Scalability	Correctness
EFlow	**Low**	**Low**	**Low**	**Low**
PPM	**Low**	**Low**	**Low**	**Low**
BPEL4WS	**Average**	**Low**	**Average**	**Low**
OWL-S	**Average**	**Average**	**Good**	**Low**
WSMO	**Good**	**Average**	**Good**	**Low**

4. Conclusion
This paper has aimed to provide a general overview and compare recent progress in dynamic Web service composition to support E-Business systems. We classify these approaches to three categories which can be complementary. But we cannot claim that this classification is very exhaustive. In each category, we give the introduction and comparison of selected approaches. The workflow-based approaches are usually used in the situation where the request has already defined the process model, but automatic program is required to find the atomic services to complete the requirement. XML-based approaches concentrate on two main approaches, namely: orchestration and choreography. Choreography languages are still in an introductory phase of definition. In ontology-based approaches, ontologies are used as data models throughout these types of approaches, meaning that all resource descriptions and all data interchanged during service usage are based on ontologies. The main problems with most of these approaches are the verification of correctness of WSC and the analysis of QoS aspects. Consequently, these approaches help increase the flexibility for modifying and extending the operations of E-Business systems during runtime.

5. Acknowledgement
This research is supported by the Ministry of Science & Technology and Innovation (MOSTI), Malaysia and University of Technology Malaysia under the Vot. 79277.

6. References
[1] Berners-Lee, T., Fielding, R., and Masinter, L. 1998. *Uniform Resource Identifiers (URI): Generic Syntax*, IETF RFC 2396. Retrieved from http://www.ietf.org/rfc/rfc2396.txt

[2] Web Services Architecture Requirements.2004. Retrieved from http://www.w3.org/TR/wsa-reqs/

[3] New to SOA and Web services. Retrieved from http://www.ibm.com/developerworks/webservices/

[4] WSDL v1.1.2001. Retrieved from http://www.w3.org/TR/wsdl

[5] SOAP v 1.2.2007. Retrieved from http://www.w3.org/TR/soap12-part1/

[6] Casati, F., Sayal, M., and Shan, M.-C.2001.Developing e-services for composing eservices. In *Proceedings of 13th International Conference on Advanced Information Systems Engineering(CAiSE)*, Interlaken, Switzerland, (June 2001). Springer Verlag.

[7] Andrews T., Curbera F., Dholakia H., Goland Y., Klein J., Leymann F., Liu K., Roller D., Smith D., Thatte S., Trickovic I. and Weerawarana S. 2007.BPEL v 1.1. Retrieved from http://www.ibm.com/developerworks/library/specification/ws-bpel/

[8] Ankolekar, A.2002. DAML-S: Web Service Description for the Semantic Web. in *Proceedings of* ISWC'02, ser. LNCS, vol. 2342. Springer, 348–363.

[9] WSMO working group. Retrieved from http://www.wsmo.org

[10] Rao, J. 2004. Semantic Web Service Composition via Logic-based Program Synthesis. Doctoral Thesis. Norwegian University of Science and Technology. Norway.

[11] Rao, J. and SU, X. 2004. A survey of automated web service composition methods. In *Proceedings of SWSWPC.*

[12] Casati, F., Ilnicki, S., and Jin, L.2000. Adaptive and dynamic service composition in EFlow. In *Proceedings of 12th International Conference on Advanced Information Systems Engineering(CAiSE)*, Stockholm, Sweden, June 2000. Springer Verlag.

[13] H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker. Modeling and composing service-based and reference process-based multi-enterprise processes. In *Proceeding of 12th International Conference on Advanced Information Systems Engineering (CAiSE)*, Stockholm, Sweden, June 2000. Springer Verlag.

[14] Beek, M., Bucchiarone, A., and Gnesi, S. 2007. Web Service Composition Approaches : From Industrial Standards to Formal Methods. In *Proceedings of Int'l Conf. On Internet and Web Application and Services (ICIW'07) ,IEEE.*

[15] Kavantzas, N., Burdett, D., and Ritzinger, G. 2004.WSCDL v1.0. Retrieved from http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/

[16] Van der Aalst W.M.P.2003. Don't Go with the Flow: Web Services Composition Standards Exposed. IEEE Intelligent Systems, 18(l):72-76.

[17] Mandel, D.J., Mcllraith, S.A.2003. Adapting BPEL4WS for the Semantic Web Bottom-up Approach to Web Services Interoperation. In Second International Semantic Web Conference (ISWC), Sanibel Island, Florida.

[18] Berners-Lee, T., Hendler, J., and Lassila, O.2001. The Semantic Web. *ScientificAmerican*, 284(5):34–43.

[19] Fensel,D. , Lausen,H. , Polleres,A. , Bruijn , J. , Stollberg , M. , Roman D. , AND Domingue , J. 2007. Enabling Semantic Web Services . Springer

[20] *OWL-S: Semantic Markup for Web Services.* W3C Member Submission, (22 November 2004). Retrieved from http://www.w3.org/Submission/OWL-S/.

[21] Narayanan, S. and McIlraith, S. 2002. Simulation, verification and automated composition of web services. In *Proceedings of the 11th International World Wide Web Conference (WWW2002)*, Honolulu, Hawaii.

[22] Battle, S. Semantic Web Services Framework (SWSF).

[23] Object Management Group Inc. (OMG). Meta Object Facility (MOF) Specification v1.4, 2002.

[24] WSMX working group. Retrieved from http://www.wsmx.org

[25] Weibel S., Kunze, J., Lagoze, C., and Wolf, M. 1998. Dublin Core Metadata for Resource Discovery.IETF 2413. Retrieved from http://www.ietf.org/rfc/rfc2413.txt

[26] Rajesh, S. and Arulazi, D. Quality of service for Web services – demystification, limitations, and best practices.

[27] O'Sullivan, J., Edmond D., and Hofstede, A. T.2002. What is a service? Towards accurate description of non-functional properties. *Distributed and Parallel-Databases*, 12(2–3):117–133.

[28] Dirk ThiBen, D., Wesnarat, P. 2006. Considering QoS Aspects in Web Service Composition. *In Proceedings of* the 11th IEEE Symposium on Computers and Communications (ISCC'06).

[29] Milanovic, N. and Malek, M.2004.Current solution for Web service composition. IEEE.

[30] Fu, X., Bultan, T., and Su, J.2004. Analysis of Interacting BPEL Web Services. *In Proceedings of WWW'04*. ACM Press, 621–630.