An Integration Life Cycle for Semantic Web Services Composition

Muhammad Ahtisham Aslam[1], Jun Shen[2], Sören Auer [1], Michael Herrmann[3]
[1]*Betriebliche Informationsysteme, Universität Leipzig, Germany*
{aslam,auer}@informatik.uni-leipzig.de
[2]*School of Information Systems and Technology*
Faculty of Informatics, University of Wollongong, Australia
jshen@uow.edu.au
[3]*DaimlerChrysler AG, Sindelfingen Germany*
michael.hm.herrmann@daimlerchrysler.com

Abstract

Business applications are more and more often developed on the basis of Web services. The aim is to provide platform independence and loose coupling between business applications to facilitate distributed and grid computing scenarios. However, most efforts to deploy and publish Web services are manual. Manual discovery, invocation and composition of Web services in a distributed computing environment significantly hamper the automatic process of enterprise application integration. Semantic enhancements in Web services aim at making the process of Web services discovery, invocation and composition dynamic by exposing the machine understandable description of Web service capabilities and Web service requests. In this paper we compare recent dynamic Web service composition approaches. We highlight some dynamic composition issues and compare existing approaches with respect to these issues. Based on these findings we present a new and generic semantic Web services integration and composition lifecycle to facilitate the semantic based integration and composition of Grid services. The proposed semantic Web services integration and composition life cycle explains the necessary integration phases beginning with the modeling and developing of processes as Web service composition and ending with their execution. With this lifecycle, integration hurdles among different service composition approaches will be diminished.

Keywords: Semantic Grid, Service Composition, E-service Framework, System Design

1. Introduction

Grid technology provides an information infrastructure for sharing and coordinating between different scientific and engineering design resources (such as services) in the emergent and supposedly ubiquitous Web services environment. Semantic enhancements in Web services make it more attractive for grid and distributed computing to leverage existing work from business and scientific environments. The successful adoption of semantic Web services (SWSs) in Grid scenarios depends on how efficiently Web services address issues like dynamic discovery, invocation and composition. To tackle these issues, semantic enhancements in Web services are proposed. Three major efforts (i.e. OWL-S [11], WSMO [15] and WSDL-S [13]) are currently going on to add semantics to Web services. The SWS community has presented different solutions (as discussed in Section 2) for dynamic and automated composition and integration of Web services by using these SWS languages. Unfortunately, none of these approaches fully addresses dynamic composition issues. Some major challenges in semantic based integration and composition of Web services are:

- With a growing number of services, manual discovery and composition is an inefficient and non-flexible approach.
- Design time composition is not able to handle services, which change on the fly.
- Static binding of Web services result in the failure of a composition task, already when a single service within the composition is not accessible on the network.
- Syntax based composition prevents to dynamically discover and compose alternate services, which perform the same task.

Traditional Web services (WSDL services) provide syntactical interfaces and UDDI registries support only index word based searching of required services. Web services and SOA need to be semantically enhanced to support the Web service integration process in a machine understandable way. Also, a generic methodology is needed to integrate business applications (rendered as services) and express business rules and logic in a more flexible way that is understandable for computer agents.

Business logic can be modeled by using different process modeling techniques. For example, a value generating business system can be defined as a composition of value-generating activities in a *Value Chain Diagram* [2]. The *Event Driven Process Chain* [10] [7] is another way to compose methods. We can structure the control flow of a business process as a chain of events and functions by using EPC diagrams. At the same time *Activity Diagrams* [7] can also be used to model business processes and to implement the logic of a business process. But in the rapidly growing service oriented world an executable orchestration lan-

guage like BPEL or the OWL-S *Process Model* ontology is more useful to model business processes as composition of Web services. Such an orchestration language can define control and data flow between Web services and implement business rules, logic and technical details of a business process.

As an example from automobile industry we consider a scheduled business process, called, "planning and execution" which, calculates the daily production of gears. Gears and engines are a part of the assembly process in order to build a car. The "planning and execution" process optimizes the production of gears restricted (e.g. by capacity of production). Gears and cars are not linked by type series. In fact, a class of gears matches to a class of cars. Enriching process steps with semantic annotations promises a better-optimized production plan.

Academia and industry have made large efforts on SWS and related issues. A drawback of the ongoing work is that all these efforts are being done on individual platforms. It is needed to synchronize these efforts to make SWS functional. For this purpose a life cycle for SWS integration is needed. Its goal should be to combine the SWS integration efforts, starting with the design and, stepping through the phases publishing, discovery, invocation, composition, and finally resulting in the service execution.

The remaining paper is organized as follows: Section 2 provides recent research efforts and existing approaches for semantic based discovery and composition of Web services. In Section 3 we highlight some challenges and dynamic composition issues and compare existing approaches with respect to these issues. In Section 4 we provide a SWS integration and composition life cycle to bring these SWS efforts in one circle. Section 5 concludes our work with a short discussion and a perspective on future work.

2. Existing SWS Efforts and Composition Approaches

2.1. Existing SWS Efforts

Business Process Execution Language for Web Services (BPEL4WS (BPEL)) [1] is the language that can be used to define the composition and orchestration of multiple Web services. It provides a rich vocabulary in the shape of *primitive* and *structured* activities for expressing the behaviour of business processes. WSDL services can be used to expose operations of applications but they don't handle integration aspects. Integration of Web services within and across enterprises needs definition and collaboration activities and data exchange between Web services. Such collaboration can be modelled as a process by composing different Web services with defined control flow and data flow. Interaction between Web services within a BPEL process model can be synchronous or asynchronous.

The Web Ontology Language for Services (OWL-S) [11] is a set of markup language constructs that can be used to define properties and capabilities of Web services in computer understandable way. It aims at providing an ontological description of Web services to facilitate dynamic and automated discovery, invocation and composition of Web services. OWL-S provides Web service semantics by ontologically annotating: (1) the input required by a service (as shown in the sample code below), (2) the output generated by a service, (3) pre-conditions that need to hold to perform a service and (4) effects that the service will produce after its execution. OWL-S is a suite of OWL ontologies (*Profile*, *Process Model* and *Grounding* ontologies). The *Process Model* ontology can be used to model the composition of SWSs by defining the control flow and data flow on the basis of matching semantics of sub processes.

```
<process:Input rdf:ID="CarRequest">
  <process:parameterType rdf:datatype="&xsd;#anyURI">
    &bibtex;#Roadster</process:parameterType>
  <rdfs:label>Roadster is a type of car.</rdfs:label>
</process:Output>
```

Meanwhile, WSDL-S [13] is also a candidate language for SWS. Instead of defining separate ontologies to provide service semantics, the WSDL-S approach extends tags of the existing Web services description language (WSDL). In addition with annotating input/output messages, the WSDL-S extensions enable the description of preconditions and effects of a Web service operation. The sample below gives an example of WSDL-S annotation of WSDL message tag.

```
<wsdl:message name="CarRequest">
  <wsdl:part name="in0" type="tns1:TypesOfAvailableCars"
    LSDISExt:onto-concept="LSDISOnt:Roadster"/>
</wsdl:message>
```

WSMO [15] is another initiative to develop specifications for SWSs. It has three approaches to model Web services composition (i.e. state machine, structured and data flow models). State-based model is some how related to WSFL in which each state defines control flow to control activities. Structured model is based on structured design methodology and is used in workflow languages (e.g. BPEL). Third model (i.e. Data flow model) is based on parallel programming languages and is based on concurrent control components of structured model.

All above efforts involve planning of Web services composition. Algorithmically, a planning problem has as input a set of possible courses of actions, a predictive model for underlying dynamics, and a performance measure for evaluating courses of action.

2.2. Existing Composition Approaches

2.2.1. A Bottom-Up Approach. The work discussed in [3] presents a bottom-up approach by integrating the semantic Web technology into Web service technology while considering BPEL as a composition of Web services. Idea behind this approach is to add semantics in BPEL that provide machine understandable descriptions of required services within process and extending workflow execution engine (BPWS4J) to realize these semantic descriptions. With these semantic descriptions the bottom-up approach uses Semantic Discovery Service (SDS) to dynamically

discover a required service on the basis of matching semantics and bind it within composition. In case, if a single service does not meet a service requirements, the SDS uses a recursive back-chaining algorithm to determine a sequence of service invocations or service chain, which takes input provided by the BPWS4J and returns the output required by the BPWS4J. However, the system efficiency goes down as the number of service *Profiles* increases in service chain. Another limitation of this approach is that it doesn't consider pre and post conditions for discovery and composition purposes.

2.2.2. METEOR-S Approach. In the METEOR-S project [12], the working group has developed a tool for dynamic composition of Web services. The METEOR-S tool (METEOR-S process designer) allows process designers to design processes on the basis of business and process constraints. Idea behind Web Services Composition Tool is to write required service specifications as an abstract process within BPEL process and to discover services whose *Profile* matches to defined abstract process. Once required services are discovered, candidate services are selected on the basis of process and business constraints. The process designer uses BPEL for process modeling. A service template is created by using functional as well as QoS specifications of all operations of a Web service in a process [12]. Major drawback of this approach is that end user has to manually select a service for composition among bundle of dynamically discovered matching services.

2.2.3. Template Based Composition: An AI approach. In [5], Evren Sirin uses workflow templates to write abstract activities. These abstract activities can be used to describe required services. On the basis of these activities specifications required services can be discovered to create executable workflows. This approach focuses on value of adding preferences in templates so that services can be ranked to find most suitable one among a bundle of discovered services. Evren Sirin proposes the use of semantic Web technology (OWL) for writing such templates, which allow reasoning for flexible and more consistent match making of required services. This approach focuses on extending the OWL-S process ontology by proposing the addition of abstract process. Evren Sirin proposed that process ontology should have an abstract process that can be used to refer to the *Profile* ontology of an OWL-S service with other specifications that can be used to rank and find best suitable service. The proposed abstract process, unlike to *atomic process* is not connected to specific *Profile* or *Grounding* and unlike to *simple process* is not connected to any existing process. This approach implements use of AI planning approach (i.e. Hierarchical Task Network (HTN) planning) with its extended formalism as HTN-Description Logic (HTN-DL).

2.2.4. WSMO Composition Approach. WSMO community has also developed a tool [4] for dynamic composition of Web services and has integrated it with IRS-III [9]. The composition tool allows users to select goals, mediators and control flow operators to define control flow between components. The composition process starts by selecting a composition goal from the list of available goals defined in the IRS-III server. Data flow between these goals can be defined by specifying the data source as input of goal and the data destination as an output of the goal. Type mismatch between inputs and outputs of goals can be managed by using mediators. Mediators map and perform transformation between goals. Defining XSL Transformations can support such a data mapping between messages of different types in OWL-S.

2.2.5. Automated Composition by Using SHOP2. The work discussed in [6] describes how an AI planning system (SHOP2) can be used with the DAML-S (OWL-S) Web service description to automatically compose Web services. This approach gives partial support for composing services on the basis of their matching functional and non-functional semantics. [6] Does not support the creation of a *composite process* with all OWL-S supported control constructs (e.g. this approach does not support synchronization between process components by implementing support for OWL-S *Split-Join* control construct).

2.2.6. SWORD. The method reported in [14] provides a set of tools for composition of a class of Web services. The SWORD implements use of rule-based expert system that determines possibility of automatic creation of composite service from existing services. In case of such possibility a plan is created. Execution of such a plan generates composite service. This approach is limited with respect to selecting Web services for composition just on the basis of input and output and does not handle services that have certain pre-conditions or effects.

2.2.7. Plængine. Plængine [8] is a software system that supports planning for service composition and service enactment. The Plængine uses integrated meta-model approach to plan for Web services composition. The Plængine consists of two components: a composer and an enactor. The composer is responsible to generate composition with the help of its sub-component ComposerThread that uses search-planning algorithm to perform composition. The enactor is responsible for scheduling and execution of individual services within a composition. This work focuses on overcoming limitations (e.g. handling exceptions, sophisticated support for control flows and extending architecture of meta-models).

3. Limitations of Existing Approaches

On the basis of major challenges and existing composition approaches (discussed in Sections 1 and 2 respectively) we would summarize above approaches by compiling them with their level of support for issues that need to be addressed for dynamic Web services composition. Some major dynamic composition issues are:

Service Discovery and Selection on the basis of matching Functional and Non-Functional Semantics: This

issue addresses the discovery of a service on the basis of matching functional semantics (e.g. input, output, pre- and post-conditions) and non-functional semantics (e.g. service response time, geographical location etc.). It is also concerned with selection of a single service from a bundle of semantically discovered services.

Service Binding & Referencing: In case of a workflow language as Web services composition, *Service Binding & Referencing* describes that how a selected service is bound in final composition. In case of an AI planning approach, it describes how a service is referred in final composition generated by an AI plan.

Composition Strategy: This employs the composition approach used for SWS composition. For example in case of a workflow language as Web services composition, *composition strategy* describes that either composition is *dynamic* or not. Or, in case of an AI planning approach *composition strategy* describes that either the final composition is generated automatically (*automatic*) or semi-automatically (*semi-automatic*).

Execution: This issue focuses on execution support for the execution of final composition.

Table 1 summarizes capabilities and limitations of above discussed approaches with respect to these SWS composition issues. It shows that none of the above approaches address all of these composition issues. For example in bottom-up approach (discussed in Section 2.2.1) QoS semantics, pre and post conditions of services play no role in discovery and composition mechanism. In this approach process designer handles pre and post conditions at design time. Similarly, the approach discussed in Section 2.2.2 also defines basic workflow in BPEL and dynamically discovered services are bound in the final process at design time.

To discover and compose Web services in dynamic and automated fashion, a composition approach should successfully address all these issues. With such an approach we can avoid problems that arise due to syntax based static composition of Web services. For example, selecting and composing required services dynamically and at run time on the basis of both matching functional and non-functional semantics can help to avoid problems that occur when a single service within composition is not accessible, or when its functional and non-functional semantics no longer match to required service semantics.

Composition Approach	Service Discovery		Service Selection		Service Binding & Referencing	Composition Strategy	Execution	Semantic Web Technology
	Functional Semantics	Non-Functional Semantics	Functional Semantics	Non-Functional Semantics				
Bottom-up Approach	*Partial*	*No*	*Partial*	*Yes*	*Run-time*	*Dynamic*	*Yes*	*OWL-S*
METEOR-S	*Yes*	*Partial*	*Yes*	*Partial*	*Deployment/Design time*	*Dynamic*	*Yes*	*WSDL-S*
Template Based Composition	*Partial*	*Partial*	*Partial*	*Partial*	*Dynamic*	*Automatic*	*Yes*	*OWL-S*
WSMO Approach	*Yes*	*Partial*	*Yes*	*Partial*	*Dynamic*	*Semi-Automatic*	*Yes*	*WSMO*
HTN Planning using SHOP2	*Partial*	*Partial*	*Partial*	*Partial*	*Dynamic*	*Automatic*	*Yes*	*OWL-S*
SWORD	*Partial*	*No*	*Partial*	*No*	*Off-line/ Composition time*	*Semi-automatic*	*Yes*	*Independent of Standards*
Plængine	*Yes*	*No*	*Yes*	*No*	*Dynamic*	*Automatic*	*Yes*	*Integrated Meta-Model*

Table 1: Comparison of existing dynamic and automated Web services composition approaches.

Semantic Web Technology: It concerns with approach used to add semantics to Web service technology (e.g. OWL-S, WSDL-S or WSMO etc.).

Successful integration and composition of Web services is needed to bring these Web services discovery, invocation and composition efforts in one circle. For this purpose we present a semantic based Web services integration and

composition life cycle that addresses dynamic composition issues and challenges.

4. Semantic Web Services Integration Life Cycle

The semantic Web services integration and composition life cycle (Figure 1) describes an engineering and development cycle to fully harness and sharpen the power of SWS. The proposed life cycle is based on a top down approach starting from modeling business processes as Web services composition and ending with their execution. It consist of multiple modules including developing business processes, adding technical and business constraints to processes, annotating the composition workflow with domain ontologies to prepare semantic base service requests in workflow and deploying and executing the final process. Each phase of the SWS integration life cycle is responsible to perform a specific task. We herein discuss characteristics of these phases individually.

Business Process Modeling. Business departments define how a single process steps are combined with each other and control and data flow between these process steps – business logic. In fact, they do not know the technical aspects and implementation of these processes and how Web services work, but they are able to design and model the business logic. Different methods like Value Chain Diagram, Event Driven Process Chains and UML Activity Diagrams can be used to model a business process. These methodologies are more useful for business experts to describe business logic as business processes that are annotated with management requirements. Deliverables of such business analysis and design processes are not readable for computers. They need some technical descrip scriptions to become readable and executable by machines.

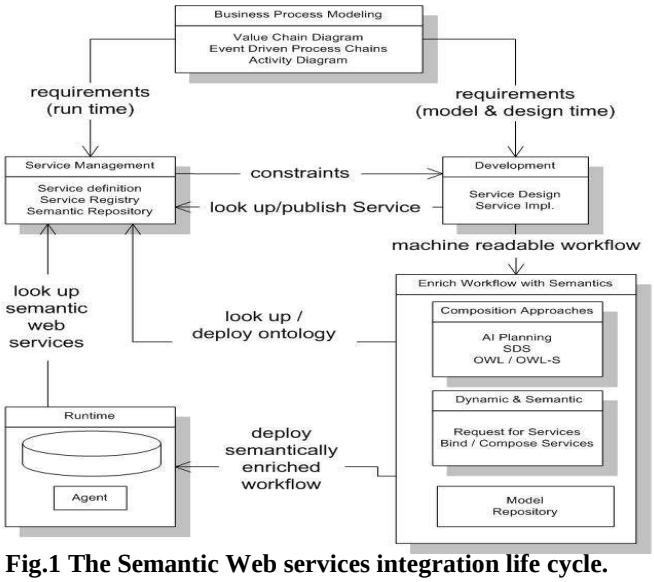

Fig.1 The Semantic Web services integration life cycle.

Development. Once defined, business processes are developed as a composition of Web services with their technical implementation. Technical descriptions of business processes make them machine readable for the purpose of deployment and execution. Machine-readable descriptions of processes can also refer to some existing services available in the service registry to perform a specific part of the total business goal. Business constraints (e.g. business rules, data exchange format, communication protocols etc.) are applied to the business process to meet the management aspects of integration process. Even though technical descriptions of processes have been implemented to make them executable for machines at this phase, but implementation of semantic descriptions of required services is still needed for the purpose of dynamic discovery and composition.

Semantics Enrichment of Workflow. Instead of binding required services within composition at design-time (development phase), required services can be described in the process semantically. These semantic service requests can be annotated with domain ontologies. Domain ontologies are managed in the service management scope. The final business process is a process defined in some workflow language (e.g. OWL-S composite process or BPEL4WS enriched with process semantics). The process of preparing and sending a request for SWS, discovering a service on the basis of matching semantics and getting its response is dependent on semantic enhancements in the participants (service provider, requester and registry) of SOA.

Runtime Phase. Semantically enriched workflows can be deployed on semantic enabled execution engines (i.e. execution engines capable of understanding workflow semantics). Execution engine is capable of invoking Web services that are statically bond in the process during the development phase of life cycle. Also, services define semantically in the workflow are searched in the semantic services registry. Services discovered on the basis of matching semantics are bound in the workflow at run time.

Discovering a service just on the basis of matching functional semantics (input, output) may not always acquire right service, therefore a semantic service request with in process should be defined on the basis of both functional and non-functional semantics. At the end, the final process as a composition of services is executed with defined control flow and data flow.

Service Management. As described before, the service management phase is the always-on and helping phase within the life cycle. Managers and developers can manage the service publishing and serving requests for semantic and syntax based Web services. Web services registries are enhanced to SWSs registries for publishing and querying SWSs. Domain ontologies are also managed in this phase. These domain ontologies can be used to annotate Web services and business processes to provide data semantics. Business processes can be managed for the deployment and execution in this phase as well. Service Management phase helps to provide business constraints for modeling business and technical perspectives of a Web services integration scenario.

5. Conclusions and Future Work

In this paper we described a comparative study of recent approaches for semantic based discovery and composition of SWSs and highlighted limitations of these approaches with respect to dynamic composition issues. We provided an integration and composition life cycle that addresses SWSs discovery and integration issues and attempts to bring these efforts together. The proposed life cycle starts with adding semantics to Web services and modeling business goals as business processes. The paper discusses how these processes could be annotated with business logic, rules and constraints in some machine-readable workflow language. We discussed the annotation of these processes with domain ontologies to provide semantics of required services in a defined workflow. Such a semantically annotated workflow can be deployed and executed by an execution engine capable of understanding the process semantics. The Web Services Description Language (WSDL) does not support the specification of various constraints, management statements, classes of service, Service Level Agreement (SLAs) and other contracts and protocols between Web services.

We are exploring these upcoming SWS languages and composition approaches in our lab by concentrating on their semantic capabilities and by implementing and updating our business processes with semantics. The goal is to annotate business processes and services that are already hosted in our infrastructure in order to reuse them in a dynamic and automated way.

Acknowledgment

This work is partially supported by the **Higher Education Commission (HEC) of Pakistan** under the scheme *"Partial Support Scholarship for PhD Studies Abroad"*.

References

[1] Business Process Execution Language for Web Services Version 1.1. 5th May 2003. [online] Available ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf

[2] C. Gray: Entrepreneurship, resistance to change and growth in small firms. Journal of Small Business and Enterprise Development, March 2002, ISSN 1462-6004, Volume 9, Issue 1, pp.61-72.

[3] D. J. Mandell and S. A. McIlraith: Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperations. In proceedings of the Second International Semantic Conference, volume 2870 of LNCS, pages 227-247. Springer, 2003.

[4] D. Sell, F. Hakimpour, J. Domingue, E. Motta and R. C. S. Pacheco: Interactive Composition of WSMO-based Semantic Web Services in IRS-III. Proceedings of the First AKT Workshop on Semantic Web Services (AKT-SWS04) KMi, The Open University, Milton Keynes, UK, December 8, 2004.

[5] E. Sirin, B. Parsia, and J. Hendler: Template-based Composition of Semantic Web Services. In AAAI Fall Symposium on Agents and the Semantic Web, Virginia, USA, November 2005.

[6] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau: HTN Planning for Web Service Composition using SHOP. Journal of Web Semantics, 1(4): 377-396,2004.

[7] Ferdian: A Comparison of Event-driven Process Chains and UML Activity Diagram for Denoting Business Processes (Master Thesis).

[8] H. Meyer, H. Overdick, and M. Weske: Plængine: A System for Automated Service Composition and Process Enactment. Proceedings of WWW Service Composition with Semantic Web Services, p. 3 - 12. University of Technology of Compiègne, 20.

[9] J. Domingue, L. Cabral, F. Hakimpour, D. Sell and E. Motta: IRS-III. A platform and Infrastructure for Creating WSMO-based Semantic Web Services. Proceedings of the Workshop on WSMO Implementations (WIW 2004) Frankfurt, Germany, September 29-30, 2004, ISSN 113-0073.

[10] J. Mendling, G. Neumann and M. Nüttgens: Yet Another Event-Driven Process Chain. Proceedings of 3rd International Conference on Business Process Management (BPM 2005), Nacy, France, September 5-8, 2005, LNCS 3649, pp. 428-433.

[11] OWL-S: Semantic Markup for Web Services. [online] Available http://www.daml.org/services/owl-s/1.2/overview/

[12] R. Aggarwal, K. Verma, J. Miller and Wilnor: Dynamic Web Service Composition in METEOR-S. Technical Report, LSDIS Lab, Computer Science Dep., UGA, May 2004.

[13] R. Akkiraju, J. Farell, J. A. Miller, M. Nagarajan, A. Sheth and K. Verma : Web Service Semantics – WSDL-S [online] Available http://www.w3.org/2005/04/FSWS/Submissions/17/WSDL-S.htm.

[14] S. R. Ponnekanti and A. Fox: SWORD: A Developer Toolkit for Web Service Composition. In Proceedings of the 11th International World Wide Web Conference, WWW 2002, Honolulu, Hawaii, May 7-11, 2002. ACM Press, 2002.

[15] Web Services Modeling Ontology [online] Available http://www.wsmo.org