D13.0v0.3 Overview and Scope of WSMX

WSMX Draft 1st of December 2005

This version:
http://www.wsmo.org/TR/d13/d13.0/v0.3/

Latest version:
http://www.wsmo.org/TR/d13/d13.0/v0.3/

Previous version:
http://www.wsmo.org/TR/d13/d13.0/v0.2/

Editor:
Laurentiu Vasiliu

Authors:
WSMX working group

Table of Contents

---

1. Introduction of deliverable

This deliverable presents the ongoing work within Web Services Execution Environment (WSMX) working group. WSMX [2] is an execution environment which addresses two primary research issues for Semantic Web Services:

- How to design and implement a goal-driven service execution environment where enough semantics have been added to allow for meaningful service discovery, composition, data and process mediation
- How to design an architecture where the invocation order can be configured dynamically rather than be hard coded in the individual components

As case studies, there are envisioned several WSMX applications to be implemented: e-banking, e-Government and Telecom in DIP project, geospatial applications in SWING project and business process management in SUPER project.

The work on WSMX is divided as follows: the conceptual work on architecture, execution semantics and API is done within OASIS Semantic Execution Environment Technical Committee while within WSMX working group the reference implementation is performed.

2. Conceptual overview of WSMX – main concepts and components

WSMX (Web Service Execution Environment) is being designed as a flexible platform able to carry in/out and interpret semantic messages towards and from various web services that have their capabilities described in a semantic manner. It is an execution environment for SWS and provides a platform technology for a distributed system over the internet (figure 1).

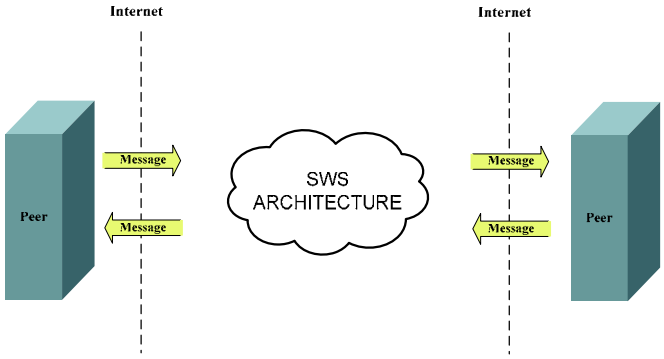

Figure 1. WSMX SWS architecture for distributed systems.

There are significant obstacles to automation resulting from the lack of semantics. Environments such as Biztalk, implementations for OASIS WS-BPEL, and WebSphere are examples of environments for Web Services but require the invocation order of Web Services to be established at design time. WSMX tackles this problem with the use of Semantics.

WSMX main concepts (most being reflected in WSMX components) can be listed as follows:

Syntactic adaptation (adapter framework component)
Syntactic adaptation is concerned with syntactic mapping from different formats (UBL, EDI etc) to WSML (and back from WSML to UBL, EDI, etc).

Protocol adaptation (adapter framework component)
Protocol adaptation takes care of the protocol mapping between various protocols that may be used by different back-end applications and web services and WSMX.

Discovery (discovery component) [3]
The aim of Web service Discovery is to locate Web services that (totally or partially) fulfil a requestor's goal. If a specific service is required by a user, it is possible that this service may be found and used without the user having any prior knowledge of where the service is. Web service Discovery is based on matching abstracted descriptions of formalized goals with semantic annotations of formalized Web services. At this point a Web service can be selected which would potentially be used to realise the desired service. Discovery can be based on simple keyword-based matching, simple-semantic matching or rich semantic-description matching.

Selection (selection component)
From within discovered web services, the one that is able to satisfy the requester goal it is selected to be invoked.

Invocation (invocation component)
Once the desired web service is discovered, it needs to be invoked and established a message exchange based on its declared choreography.

Choreography (choreography component)
The choreography of a Web Service defines its communication pattern, that is, the way a requester can interact with it. The requestor of the service has its own communication pattern and only if the two of them match, a direct communication between the requestor and the provider of a service can take place. If they don't match, the process mediator will ensure proper matching.

Orchestration (orchestration component)
The orchestration concept refers to the complex composition of several web services, composition that can be in parallel, or chained or made of combinations of these two. Choreography is used by orchestration to communicate with individual web services.

Mediation (mediation components) [1], [5]
Mediation is needed when two entities that cannot communicate directly need to interact due to different syntax or semantics of data they are dealing with and/or processes for the same operation might mismatch:

Data mediation (data mediation component)
Data Mediators deals with the heterogeneity problems that may appear at the data level of the communication between various entities.

Process mediation (process mediation component)
Process Mediator Processes mediation is concerned with determining how two public processes can be matched in order to provide certain functionality.

Execution Semantics [10]
Execution Semantics defines the operational behavior of WSMX. This is a formal specification, describing exactly how the system behaves. The execution semantics ensures a common understanding among developers of what the system should do and how it should behave in all possible situations.

3. WSMX - Service Orientated Architecture, support components and tools

WSMX architecture [11] is build around the concepts defined in the previous section 2. In this respect, almost each concept defined there has a correspondent architectural component (see next figure 2):

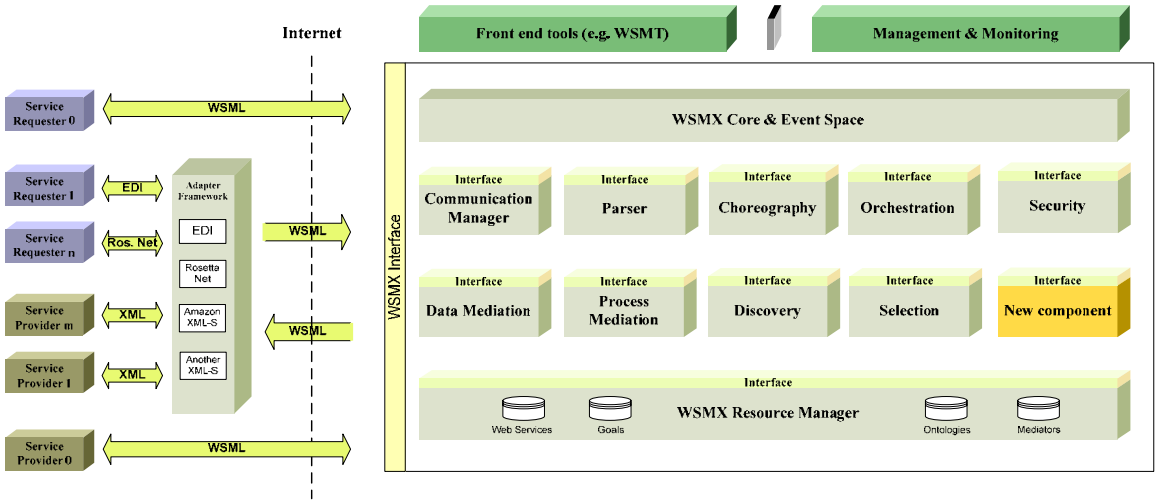

Figure 2 WSMX Architecture

Furthermore, several other components are needed to provide functional support for the main components. These new components are presented next.

Tool:

Web Service Modeling Toolkit (WSMT)[4]. The Web Services Modeling Toolkit (WSMT) is a collection of tools for Semantic Web Services developed for the Eclipse framework. An initial set of tools exist including the WSMO Visualizer for viewing and

editing WSML documents using directed graphs, a WSMX Management tool for managing and monitoring the WSMX environment and a WSMX Data Mediation tool for creating mappings between WSMO ontologies.

Support components:

WSMX Core. The Core Component is the central component of the system, where all the other components are connected to it.

Resource Manager The Resource Manager is responsible for management of repositories to store definitions of any WSMO (Web Services, goals, ontologies and mediators) and non-WSMO related objects for WSMX.

Web Services Repository. deals with semantic description of Web Services, such as their capabilities (pre-conditions, post-conditions, assumptions and effects), interfaces (choreography and orchestration) and non-functional properties, both general and Web Service specific.

Ontology repository deals with ontologies to be stored in the registry describing the semantics of particular domains. Any components might wish to consult an ontology, but in most of the cases the ontologies will be used by mediator related components to overcome data and process heterogeneity problems.

Mediator repository. deals with mediators to be stored in the registry. WSMO mediators have become a standard proposal covering various approaches for data and process mediation.

Data repository. During run time of the system, system specific data is produced. Also, components might generate data, which should be persistent. Data repository allows to store any system data and any component-specific data required for correct system execution.

Communication Manager WSMX will offer invocation of selected services, obeying their interfaces that are choreography and orchestration patterns. Communication Manager consists of two subcomponents: invoker and receiver.

Reasoner Although development of a reasoner is not part of the WSMX development process, a WSML-compliant reasoner will be an important element of the whole WSMX framework.

As well as developing the system itself, the research group will also address all the key aspects of enterprise software. Thus the updated releases of the system will take account of aspects like reliability, scalability, security, etc. These will become an integral part of the WSMX server.

4. Related systems

Implementations based on SWS concepts are nowadays the subject of intensive research. Apart from WSMX as the reference implementation of WSMO [9], other implementations based on SWS concepts also exist such as IRS III [6] virtual machine, OWL [7] and METEOR-S [8]. It is the intention of WSMX to be interoperable with such systems where possible.
The main components of IRS III (Internet Reasoning Server) are the IRS Server, the IRS Publisher and the IRS Client. The IRS server holds descriptions of SWS. IRS Publisher links Web services to their semantic descriptions within the IRS server and generates a wrapper which allows a web service to be invoked. IRSIII provides infrastructure for creating WSMO-based SWS, building upon the previous implementation of IRS.
A set of OWL-S tools exists rather than a complete execution environment based on OWL-S concepts. For example, OWL-S Editors allow the maintaining of OWL-S service descriptions, OWL-S Matcher provides an algorithm for different degrees of matching for individual elements of OWL-S, OWL-S Axis plug-in advices service providers on how to provide service description using OWL Services, etc.
The main components of METEOR-S are Abstract Process Designer, Semantic Publication and Discovery Engine, Constraint Analyzer, and Execution Environment. Abstract Process Designer allows a user to create the control flow of the process using BPEL constructs. Semantic Publication and Discovery Engine provide semantic matching based on subsumption and property matching. Constraint Analyzer dynamically selects services from candidate services, which are returned by the discovery engine. Execution environment performs actual late binding of services returned by the constraint analyzer and converts abstract BPEL to executable BPEL.

Projects where WSMX stays as technology platform:

- DIP Project http://dip.semanticweb.org/index.html DIP (Data Information and Process Integration with Semantic Web Services
- SWING Project (to start March 2006)
- SUPER Project http://super.semanticweb.org/ SUPER (Semantics Utilized for Process management within and between Enterprises) (to start April 2006)

5. Summary and future work

This document presents the main concepts needed for a SWS platform to operate with semantically described web services. Also there are described the components that reflect the concepts needed in such platform together with the auxiliary components needed for the platform to operate.

Additional functionalities : Some of additional functionalities, which are going to be available with the later WSMX system implementations include

(1) a plug-in mechanism to handle any distributed components that have been developed since the earlier versions of WSMX were designed,

(2) an internal workflow engine capable of executing formal descriptions of the behavior of system components and

(3) a resource manager enabling persistence of any arbitrary data (WSMO and non-WSMO related) produced by components during run-time of the WSMX server.

The future work will consider two directions with high potential: GRID and Business Process Management.

Acknowledgements

The work is funded by the European Commission under the projects DIP, Knowledge Web, Ontoweb, SEKT, and SWWS; by Science Foundation Ireland under the DERI-Lion project; and by the Austrian government under the CoOperate programme.

The authors would like to thank to all the members of the WSMO working group for their advises and inputs to this document.

References

[1] Emilia Cimpian and Adrian Mocan. D13.7 v0.1 Processes Mediation in WSMX. Technical report, Jan 2005. http://www.wsmo.org/2005/d13/d13.7/v0.1/.

[2] Emilia Cimpian, Adrian Mocan, Matthew Moran, Eyal Oren, and Michal Zaremba. D13.1v0.3 Web Service Execution Environment – Conceptual Model. Technical report, Dec 2004. http://www.wsmo.org/2004/d13/d13.1/v0.3/.

[3] Uwe Keller, Rubn Lara, Axel Polleres, Ioan Toma, Michel Kifer, and Dieter Fensel. D5.1v0.1 WSMO Web Service Discovery. Technical report, Nov2004. http://www.wsmo.org/2004/d5/d5.1/v0.1/20041112/.

[4] Mick Kerrigan. D9.1v0.1 Web Service Modeling Toolkit (WSMT). Technical report.

[5] Adrian Mocan and Emilia Cimpian. D13.3v0.2 WSMX Data Mediation.Technical report, Jan 2005. http://www.wsmo.org/2005/d13/d13.3/v0.2/.

[6] E. Motta, J. Domingue, L. Cabral, and M. Gaspari. IRS: A Framework and Infrastructure for Semantic Web Services. In 2nd International Semantic Web Conference, 2003.

[7] OWL Services Coalition. OWL-S: Semantic Markup for Web Services. Available from http://www.daml.org/services/owl-s/1.1/, 2004.

[8] Prof. Amit Sheth, Prof. John Miller, Kunal Verma. METEOR-S: Semantic Web Services and Processes. Available from http://lsdis.cs.uga.edu/Projects/METEOR-S/, 2004.

[9] D. Roman, H. Lausen, and U. Keller. Web Service Modeling Ontology Standard. Technical report, WSMO Working Draft, 2004. Available from http://www.wsmo.org/2004/d2/v02/20040306/.

[10] Maciej Zaremba and Eyal Oren. D13.2v0.2 WSMX Execution Semantics. Technical report, Feb 2005. http://www.wsmo.org/2005/d13/d13.2/v0.2/.

[11] Michal Zaremba and Matthew Moran. D13.4v0.2 WSMX Architecture. Technical report, Jan 2005. http://www.wsmo.org/2005/d13/d13.4/v0.2/.