WSMX – INFRASTRUCTURE FOR EXECUTION OF SEMANTIC WEB SERVICES

Michal Zaremba, Armin Haller, Maciej Zaremba, Matthew Moran

Digital Enterprise Research Institute, DERI, Ireland
{michal.zaremba, armin.haller, maciej.zaremba, matthew.moran}@deri.org

ABSTRACT

In this paper we present a demo implementation of an event driven system, which aims to enable execution of Semantic Web Services. Our demo infrastructure is based on the Web Service Modeling Ontology (WSMO) [Roman et al., 2004]. The implementation follows research carried out by the Web Services Modeling Execution[1] working group. WSMX is a Service Oriented Architecture (SOA) and allows an agent acting on behalf of a user to talk to WSMX enabling discovery, selection, mediation and invocation of formally described Web Services to carry out specific client tasks. Through the implementation of WSMX we aim to prove the viability of WSMO and although the first implementation of the system remains incomplete in terms of the required functionality, all components are in place and the system is capable of executing simple scenarios. Through continuous open source development[2] we aim to enhance WSMX with the intended functionality for all defined conceptual components.

KEYWORDS

Semantic Web Services, Event Oriented Architecture, Service Oriented Architecture

1. INTRODUCTION

Web Services constitute one of the hot areas in web technology supporting the remote invocation of business functionality over the Internet through exchange of well defined standardised messages. Existing Web Services' technologies already provide the basic functionality for discovering (UDDI), interface description (WSDL) and messaging (SOAP) in **heterogonous**, **autonomous** and **distributed** systems. Ongoing developments in the area of Web Services constitute the future way of providing value-added services over the Internet. Despite the promises given by inventors of Web Services standards and technologies about their automated or semi-automated execution, in reality Web Services remain almost nothing more than enhanced RPC calls over Internet. Most of the Web Services used today support one-way data retrieval and update functions (getting schedules, selling tickets, checking quotes etc.), which are hard coded into software programs. While useful, they do not provide semi-automated or automated fulfilment of customers' goals (Web Services can not be automatically bound to requestor goals at run time). Also the functionality required by B2B & EAI systems is currently not supported by Web Services standards. The complex issues surrounding system integration are well documented, but no Web Services software infrastructure currently exists that tackles this problem. We are building our system on the most recent research in the area of Semantic Web Services. The intentional purpose of Web Services Execution Environment (WSMX) is to support the functionality of B2B & EAI integration systems while fulfilling the promise of automatic (semi-automatic) service discovery, selection, mediation and execution.

2. WSMO, WSMX AND WSML

The application of ontologies to Web Services is seen as a key requirement to enable semantics-driven data access and processing [Bussler et al., 2002]. There are currently three initiatives providing semantic mark-up and the means for semi-automated discovery, composition and execution of Web Services: OWL-S (formerly DAML-S) [OWL Service Coalition, 2004], METEOR-S [Patil et al., 2004] and WSMO [Roman et al., 2004].

---

[1] WSMX – http://www.wsmx.org/
[2] Sourceforge – WSMX hosting site - http://sourceforge.net/projects/wsmx

WSMX is based on WSMO and its underlying formalized representational languages family WSML. WSMO itself is a Semantic Web Services initiative being led by DERI[3] and several EU projects (DIP [4], SEKT [5] and Knowledge Web [6]) to provide an ontology describing various aspects related to Semantic Web Services such as dynamic discovery, mediation, choreography, orchestration and invocation. The conceptual foundation for WSMO has been defined in the Web Service Modelling Framework (WSMF) [Fensel/Bussler, 2002].

WSMX is a reference implementation for WSMO defining its own conceptual model [Cimpian et al., 2004], execution semantics [Oren, 2004], architecture [Zaremba et al., 2004] and implementation [Moran et al., 2004]. There are also use cases provided by WSMX working group [Haller, 2004]. Once other aspects of execution of Semantic Web Services such as for example choreography and orchestration are conceptually defined through the WSMO initiative, they will become part of WSMX as well. Internally the execution semantics of WSMX provide the reliability, security and trust required by industrial B2B & EAI integration systems. WSMX provides a mechanism to compile (checks syntax correctness and converts to internal system data representation) semantic descriptions of Semantic Web Services that have been described using Web Services Modelling Language (WSML)[7]. The formal definition of WSML language is provided in [de Bruijn, 2004].
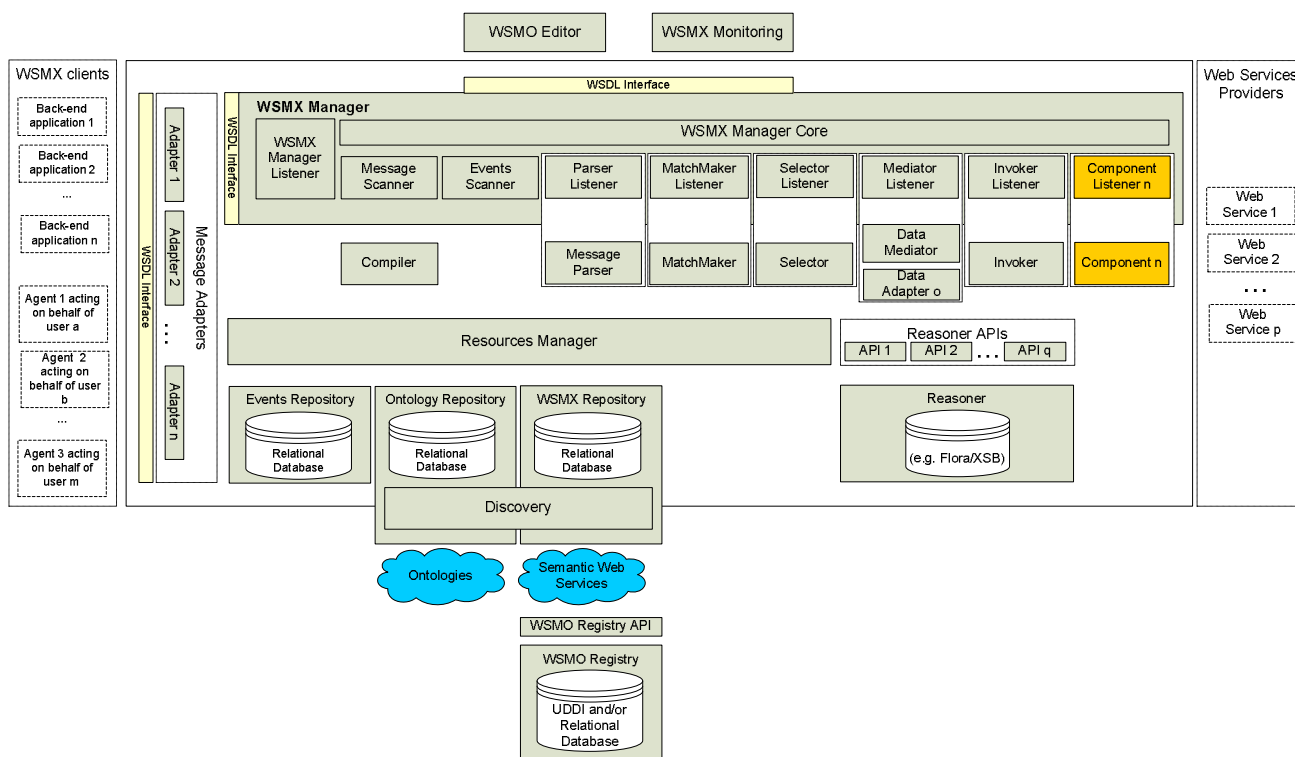
3. WSMX ARCHITECTURE

Figure 1: WSMX Architecture

---

[3] Digital Enterprise Research Institute (DERI) - http://www.deri.org/

[4] DIP – http://dip.semanticweb.org/

[5] SEKT - http://sekt.semanticweb.org/

[6] Knowledge Web – http://knowledgeweb.semanticweb.org/

[7] WSML – http://www.wsmo.org/wsml

WSMX offers complete support for interacting with Semantic Web Services. These Web Services must provide a WSMO-compliant (WSML) description of themselves specifying their capability, how to interact with them, the ontologies they use along with other properties. Figure 1, above, presents the architecture that has been partially implemented for the first version of WSMX and is going to be fully supported in subsequent implementations. The WSMO Editor can be used at design time to create semantic description for Web Service, ontologies, mediators and goals. WSMX provides a WSDL [WSDL, 2003] interface to accept these descriptions. Due to the WSDL interface the creation of WSMO descriptions (capabilities) is not bound to a specific editor. There are already several initiatives to design such an editor (within SWWS[8] and DERI). Additionally to WSMO Editor, the Monitoring Tool also connects through WSDL interface to watch functioning of the system. If requested by the Editor, any WSML file received through the WSDL interface can be compiled and permanently stored in the system. The compiler component validates the semantic descriptions and stores them persistently in the Ontology Repository.

The WSMX Manager controls the operational flow of the system. It is also responsible for event management, keeping a full record of the life cycle at each stage of processing user goal by the system. WSMX provides a Web Service Interface described in WSDL [WSDL, 2003] to accept service requesters' goals. WSMX regularly scans for new messages in a running state. Once a new message representing a requester goal is picked up, it is decomposed, validated and translated into internal date representation by the Message Parser. The Matchmaker then attempts to match the client goal to capabilities of Web Services, which are known to WSMX. The concept of Capability is used by WSMO to formally describe the functionality a Web Service offers. Any Web Service with a capability that matches the goal is returned by the Matchmaker. In this version of WSMX, the logical expressions representing the postconditions and effects of the requester goal are compared with the expressions for the postconditions and effects of capabilities for Web Services known to WSMX. The Selector component selects the Web Service that provides the best match for the goal. The initial version of the Selector in WSMX selects the first available service. The Data Mediator component finds a mediator in the Ontology repository that can mediate between the ontology used by the goal (source) and that used by the selected Web Service (target). Mediation is carried out based on a set of mapping rules between the two ontologies that are applied to the instance data contained in the goal. Once the data has been mediated, it may need to be translated from a logical language format to an XML format before it is included in the SOAP message sent by the Invoker to the target service. This translation is the job of the XML Converter. Once the data is in a format that can be used during Web Service invocation, the Invoker makes the actual Web Service invocation on the selected Web Service using the mediated and converted data.

Event-based communication is managed by the Event Manager. Components never communicate directly (one component never calls the other component) but they create and consume events (or change status of existing events), which are broadcast to all components subscribed to listen for events in the system. Such an approach enables decoupling communication from processing, which enables flexible communication patterns and facilitating flexible run-time architecture.

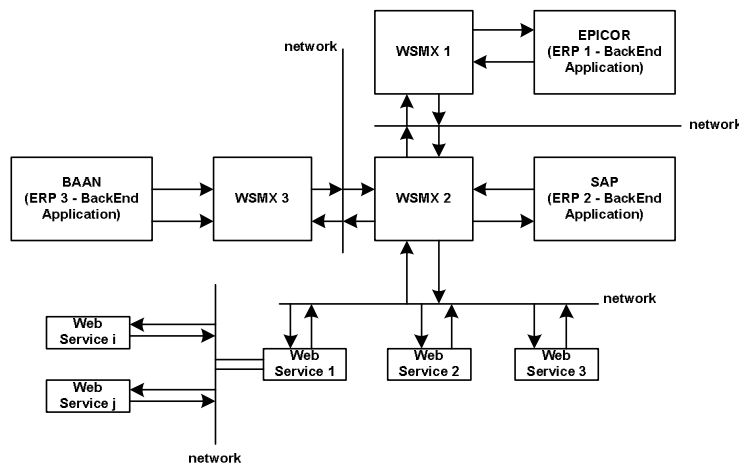WSMX can be used by enterprises that want to electronically exchange business documents across the Web (see figure 2).

Figure 2: B2B Integration with WSMX

---

[8] SWWS - http://swws.semanticweb.org/

It is assumed that partners may not know each other before carrying out business transactions and for this reason contract negotiation and contract agreement are going to be essential elements to be supported by future WSMX implementations. Each business partner exposes a set of WSMO-described Web Services with the capabilities to handle conversations using any possible known B2B protocols. The contract agreement defines the roles of enterprises in the conversation. The infrastructure provided by WSMX takes care of any necessary mediation between Web Services, ontologies, business goals and also mediates between Web Services and goals.

4. CONCLUSION

Web Services merely support the remote invocation of business functionality over the Internet. WSMX - the reference implementation for Semantic Web Services goes much further, as the system aims to enable any information systems to interact with any other information system and to preserve message, process and protocol semantics. WSMX is able to discover Web Services matching with the agent's goal as long as the service description in any variant of WSML is stored in any WSMX *service registry* within the entity. In the same process WSMX can execute them and hence reduce the time for completing the full process by avoiding any user interaction.

ACKNOWLEDGEMENT

REFERENCES

[Bussler et al., 2002] C. Bussler, D. Fensel, A. Maedche. A Conceptual Architecture for Semantic Web Enabled Web Services. SIGMOD Record, Vol. 31, No. 4, 2002.

[Cimpian et al., 2004] E. Cimpian, A. Mocan, M. Moran, E. Oren, and M. Zaremba. WSMX Conceptual Model. WSMO Working Draft v01, 2004.

[de Bruijn, 2004] J. de Bruijn, D. Foxvog, E. Oren, D. Fensel. WSML-Core, Working Draft v01, 2004.

[Fensel/Bussler, 2002] D. Fensel, C. Bussler. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, Vol. 1, Issue 2, Elsevier Science B.V.

[Haller, 2004] A. Haller. WSMX Use Cases. WSMO Working Draft v01, 2004.

[Moran et al., 2004] M. Moran, M. Zaremba, E. Cimpian, A. Mocan, E. Oren. WSMX Implementation. WSMO Working Draft v01, 2004.

[Oren, 2004] E. Oren. WSMX Execution Semantics. WSMO Working Draft v01, 2004.

[OWL Service Coalition, 2004] OWL Services Coalition (2003). OWL-S: Semantic Markup for Web Services. Available at http://www.daml.org/services/owl-s/1.0/.

[Patil et al., 2004] A. A. Patil, S. A. Oundhakar, A. P. Sheth, K. Verma. Semantic Web Services: Meteor-S Web Service annotation framework. Proceedings of the 13th WWW conference, May 2004.

[Roman et al., 2004] D. Roman, H. Lausen, and U. Keller. Web Services Modeling Ontology Standard. WSMO Working Draft v02, 2004.

[WSDL, 2003] Web Services Description Language (WSDL) 1.2, W3C Working Draft, 3 March 2003.

[Zaremba & Moran, 2004] M. Zaremba, M. Moran. WSMX Architecture. WSMO Working Draft v01, 2004.