WSMX: a Solution for B2B Mediation and Discovery Scenarios (demo)

WSMX: a Solution for B2B Mediation and Discovery Scenarios *

Maciej Zaremba and Tomas Vitvar

Digital Enterprise Research Institute (DERI),
National University of Ireland, Galway
{firstname.lastname}@deri.org

Abstract We demonstrate Web Service Execution Environment (WSMX), a semantic middleware platform for runtime service discovery, mediation and execution, applied to SWS-Challenge scenarios. We show the modelling principles as well as execution aspects of the WSMX semantic technology addressing the real-world requirements.

1 Introduction

Semantic Web service technologies offer promising potential to enable integration and discovery that is more flexible and adaptive to changes that might occur over a software system's lifetime. However, there remains very few publicly available, realistic, implemented scenarios that showcase the benefits of semantic technology. In this respect we develop the WSMX[1] – a middleware system that operates on semantic description of services and facilitates automation in service integration. We demonstrate the value of the WSMX in real-world scenarios by participating in the community-driven effort called SWS Challenge[2]. SWS Challenge is an initiative that provides a set of scenarios with real Web services and a methodology for evaluation of different semantic technologies. We contribute to the SWS Challenge by implementing solutions based on the WSMX showing how this technology can be used to facilitate dynamic discovery and mediation in B2B integration. In particular we show, how existing Web services can be semantically modeled using the Web Service Modeling Ontology (WSMO) and how mediation and semantic service discovery is implemented. Users of our demonstration are able to learn how existing, non-semantic Web services can be semantically enabled and what are the benefits of semantics in the context of B2B integration and service discovery.

2 SWS-Challenge Scenarios and WSMX

SWS Challenge defines scenarios for service discovery and data mediation imposing requirements on SWS challenge entrants to demonstrate the value of semantics for improved B2B integration and dynamic service binding. We base our solution on the SWS framework developed in DERI including conceptual

* *This work is supported by the Science Foundation Ireland Grant No. SFI/02/CE1/I131, and the EU projects SUPER (FP6-026850), and SemanticGov (FP-027517)*

[1] WSMX is an open-source project, see http://sourceforge.net/projects/wsmx.

[2] http://www.sws-challenge.org

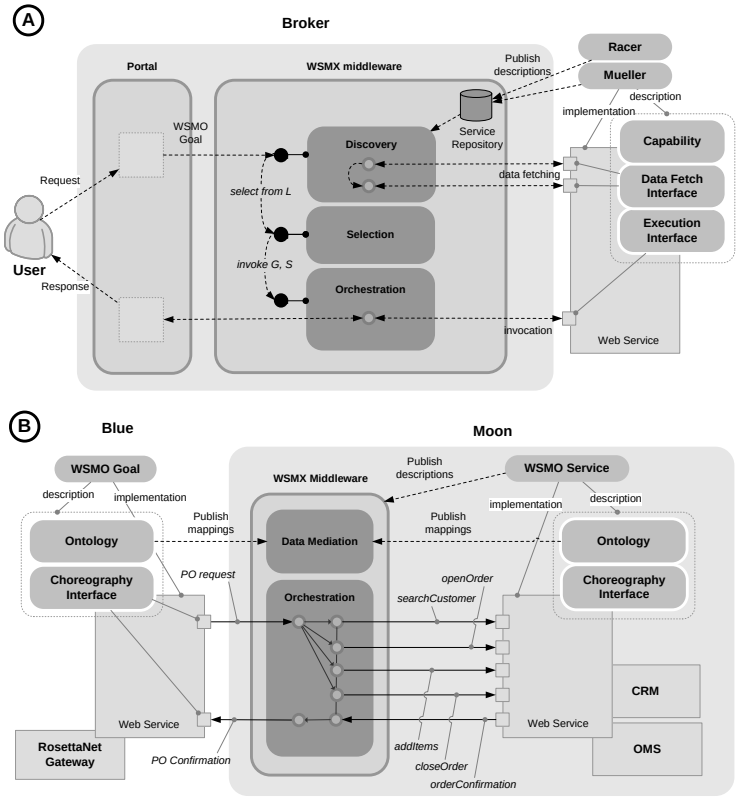

Figure 1. Solution Architectures for SWS-Challenge Scenarios

model for SWS (Web Service Modeling Ontology, WSMO[1]), language for service modeling (Web Service Modeling Language, WSML[1]), middleware system (Web Service Execution Environment, WSMX[3]), and modelling framework (Web Service Modelling Toolkit, WSMT[3]). In order to model the scenario, we use WSMO for modeling of services and goals (i.e. required and offered capabilities) as well as ontologies (i.e. information models on which services and goals are defined) all expressed in the WSML ontology language.

Figure 1 depicts solution architectures for SWS-Challenge discovery (part A) and mediation (part B) scenarios. The core to both solutions is the WSMX middleware located between service requestors and service providers. As shown in Figure 1, WSMX functionality can be customized to conform to particular integration needs through choosing appropriate components and their configuration. The following are the major components we use in the scenarios[4]:

[3] `http://sourceforge.net/projects/wsmt`

[4] Please note, that WSMX in addition contains other components which deal with communication, persistence, etc. For brevity, these are not described here.

– *Discovery* (A) defines tasks for identifying and locating Business Services,
– *Selection* (A) selects most appropriate service according to user's preferences,
– *Orchestration* (A and B) executes the composite business process,
– *Mediation* (B) resolves the heterogeneity issues at the data and process levels
– *Reasoning* (A, B) performs logical reasoning over semantic descriptions of services;

2.1 Mediation Scenario

The mediation scenario (Figure 1, part B) describes a data and process mediation of a trading company called Moon. Moon uses two back-end systems to manage its order processing, namely a Customer Relationship Management system (CRM) and an Order Management System (OMS). The SWS-Challenge provides access to both of these systems through public Web services. The scenario describes how Moon interacts with its partner company called Blue using RosettaNet PIP 3A4 purchase order specification[5]. Using the WSMT data mapping tool we map the Blue RosettaNet PIP 3A4 message to messages of the Moon back-end systems. We then apply the WSMX data and process mediation components to resolve incompatibilities of message exchanges defined by the RosettaNet PIP 3A4 process and those defined in the Moon back-end systems.

Our major contributions to the mediation scenario shows:

– how flat XML schema of RosettaNet purchase-order and other proprietary messaging schema used by different partners could be semantically enriched using the WSML ontology language as Listing 1.1 shows,
– how services provided by partners could be semantically described as WSMO services and built on top of existing systems,
– how conversation between partners and their services can be facilitated by the WSMX integration middleware enabling semantic integration, and
– how data and process mediation can be applied between heterogeneous services within the integration process.

```
1  /* XSLT Extract of lifting rules from XML message to WSML */
2  ...
3  instance PurchaseOrderUID memberOf por#purchaseOrder
4    por#globalPurchaseOrderTypeCode hasValue "<xsl:value-of select="dict:
         GlobalPurchaseOrderTypeCode" />"
5    por#isDropShip hasValue
6      IsDropShipPo<xsl:for-each select="po:ProductLineItem">
7          por#productLineItem hasValue ProductLineItem<xsl:value-of select="position()" />
8      </xsl:for-each>
9  ...
10 /* message in WSML after transformation */
11 ...
12 instance PurchaseOrderUID memberOf por#purchaseOrder
13   por#globalPurchaseOrderTypeCode hasValue "Packaged product"
14   por#isDropShip hasValue IsDropShipPo
15 ...
```

Listing 1.1. Lifting in XSLT and resulting WSML message

[5] RosettaNet is the B2B integration standard and PIP (Partner Interface Process) define various interactions patterns and vocabularies for business integration.

Since the core WSMX functionality operates on semantic descriptions of messages, WSMX needs to also facilitate transformations between semantic and non-semantic messages through so called grounding descriptions (i.e. lifting and lowering). We demonstrate two phases of the scenarios, namely the modelling phase and the execution phase. For the modelling phase we show how we apply WSMT toolkit to modeling of both semantic and grounding definitions for the Moon and Blue companies, that is, how we model service orchestrations, domain ontologies, lifting/lowering groundings, and how we define mappings between non-semantic XML and semantic WSML messages. For the execution phase, we present details of semantic B2B integration focusing on types of data and process heterogeneities that WSMX is able to handle. In this respect, we present a complete system run-through of the mediation scenario with involved middleware components operating on previously defined semantic description of Blue and Moon.

2.2 Discovery Scenario

The discovery scenario (Figure 1, part A) describes a user who uses a third-party company (broker or e-hub) in order to buy certain products with shipment to certain location. A number of shippers allow to ship products with different shipment conditions (places of shipment, price, etc.). Our approach to discovery is to match a WSMO Goal with a WSMO Web service through their semantic descriptions as well as to use additional data not available in the semantic descriptions (e.g., shipment price). The WSMX fetches this information during runtime through a specific Web service data-fetching interface. In our previous work[2], we described a conceptual framework supporting integration of dynamically fetched data into discovery context.

```
1  /* general abstract definition of the axiom in the common ontology */
2  relation isShipped(ofType sop#ShipmentOrderReq)
3
4  /* specification of the axiom in the Mueller ontology */
5  axiom isShippedDef
6      definedBy
7        ?shipmentOrderReq[sop#to hasValue ?temp, sop#package hasValue ?package] memberOf sop#
               ShipmentOrderReq and
8        ?temp[so#address hasValue ?to] and
9        ?to[so#city hasValue ?city] and
10       isShippedContinents(?city, so#Europe, so#Asia, so#NorthAmerica, so#Africa) and
11       ( (?package [so#weight hasValue ?weight] memberOf so#Package) and (?weight =< 50) )
12     implies
13       sop#isShipped(?shipmentOrderReq).
```

Listing 1.2. isShipped relation declared in the common and Mueller ontologies

The discovery scenario indicates problems associated with making service discovery an automated process. WSDL provides an insufficient language for matching client requests with Web Service descriptions. We thus semantically describe shipment capabilities offered by different companies using common shipment ontology. We take the advantage of the shared ontology when defining "abstract" axioms and their specialization in the concrete shipment service ontology (e.g., *isShipped* axiom as Listing 1.2 shows). The axiom is shared by both the shipping

services and the goals (representing service requester) and provides an interface-like mechanism[6] to define a common evaluation criteria for service discovery. Requestor does not need to know how *isShipped* is specified by the service, but it can use it in its request to check whether given service is able to ship for a specified input (i.e., source and target location, package weight, dimension, etc.).

Our solution demonstrates how domain ontologies, shipment goals and services are semantically described (concepts, instances, relationships, rules) as well as how service discovery works. We present Web services and Goals modelling principles as well as how extra information (e.g., shipping price) can be dynamically provided into the discovery context by utilizing data-fetching service interface.

3 Conclusion

With our contribution to the SWS Challenge we proved the value of the WSMX semantic technology in the context of B2B integration. Our solutions have been evaluated, by peer-review, according to the evaluation methodology of the SWS Challenge[7]. The evaluation criteria targets the adaptivity of the solutions, that is, solutions should handle introduced changes by modification of declarative descriptions rather than code-changes. Success level 0 indicates a minimal satisfiability level, where messages between middleware and back-end systems are properly exchanged. Success level 1 is assigned when changes introduced in the scenario require code changes and recompilation. Success level 2 indicates that introduced changes did not entail any code modifications and only declarative parts had to be changed. Finally, success level 3 is assigned when the system is able to automatically adapt to the new conditions. WSMX proved to deliver a generic solution scoring level 2 as there were no changes required in WSMX code when addressing new scenarios but it sufficed to adapt or provide a new semantic descriptions of involved services and service requestors. More details about our evaluation with respect to other solutions can be found at the SWS-Challenge web site[8].

References

1. D. Roman, *et al.* Web Service Modeling Ontology. *Applied Ontologies*, 1(1):77 – 106, 2005.
2. T. Vitvar, M. Zaremba, and M. Moran. Dynamic Service Discovery through Meta-Interactions with Service Providers. In *Proceedings of the 4th European Semantic Web Conference (ESWC 2007)*. 2007.
3. T. Vitvar, *et al.* Semantically-enabled service oriented architecture: Concepts, technology and application. *In Service Oriented Computing and Applications, Springer London*, 1(2), 2007.

[6] An analogy are interfaces in programming labguages like Java. The interface declares some functionality but does not say how this should be implemented.

[7] http://sws-challenge.org/wiki/index.php/SWS_Challenge_Levels

[8] http://sws-challenge.org/wiki/index.php/Workshop_Innsbruck