

# Semantic SOA to Promote Integration of Heterogeneous B2B Services\*

Tomas Vitvar, Matthew Moran, Maciej Zaremba,  
Armin Haller  
Digital Enterprise Research Institute  
National University of Ireland, Galway  
{firstname.lastname}@deri.org

Paavo Kotinurmi  
Helsinki University of Technology  
Finland  
paavo.kotinurmi@tkk.fi

## Abstract

*Modeling the semantics of business services and their corresponding messages using ontologies enables flexible integration that is more adaptive to business-driven change. In this paper we demonstrate our approach building upon an established Semantic Web Service Framework (WSMX) to facilitate a conversation between heterogeneous services that support both the RosettaNet standard and proprietary information models.*

## 1 Introduction

Inter-enterprise integration is an essential requirement for today's successful business. With the aim of overcoming heterogeneity, various technologies and standards for the definition of languages, vocabularies and integration patterns are being developed. For example, RosettaNet defines standardised Partner Interface Processes (PIPs), which include standard inter-company choreographies (e.g. PIP3A4 Request Purchase Order (PO)), and the structure and semantics of business messages. Although such standards certainly enable B2B integration, they still suffer from several drawbacks. All partners must agree to use the same standard and often the rigid configuration of standards makes them difficult to adapt to local business needs.

With regard to new emerging trends in enterprise computing, the adoption of Service Oriented Architectures (SOA) is becoming a defacto standard approach. However, today's SOA technologies only provide a partial solution to interoperability, mainly through unified technological environments, while generic and scalable solutions are still in their infancy. In particular, message level interoperability is often hardwired in business processes using traditional XSLT approaches, and process level interoperability is often maintained through manual configuration of workflows. In order to address these drawbacks, the extension of SOA

with semantics offers a scalable integration, more adaptive to changes in business requirements.

In this paper we focus on how semantic modeling of B2B standards, allied with a rich semantic SOA execution environment, provides the basis for dynamic integration of services in a heterogeneous environment. Based on the underlying technologies of Semantic Web services and B2B standards we describe the integration process within the (1) *service creation phase* and (2) *service execution phase*.

## 2 Use Case Scenario

Our use case scenario, adopted from the SWS Challenge<sup>1</sup>, describes a situation where two companies aim to build an automated B2B integration. A trading company, called Moon, uses a Customer Relationship Management system (CRM) and an Order Management system (OMS) to manage its order processing. Moon has signed agreements to exchange PO messages with a company called Blue using the RosettaNet standard for PO exchange (PIP3A4). In this scenario, Blue sends a PIP3A4 PO message, including all items to be ordered, and expects to receive a PIP3A4 PO confirmation message. In Moon, various interactions with the CRM and OMS systems must be performed in order to process the order, e.g. get the internal ID for the customer from the CRM system, create the order in the OMS system, add line items into the order, close the order, and send back the PO confirmation.

In order for integration to be possible, both Moon and Blue must comply on three interoperability levels - **Communication**, **Message** and **Process**. We focus on the two latter assuming both companies communicate via SOAP over HTTP. For the **Message level** both partners need to understand the exchanged messages including both the message structure and the semantics of its content. In our scenario, Blue uses PIP3A4 to define the PO request and confirmation messages. However, Moon uses a proprietary XML Schema for its OMS and CRM systems. On the **Process**

\*This work is supported by the Science Foundation Ireland Grant No. SFI/02/CE1/I131, and the EU projects Knowledge Web (FP6-507482), DIP (FP6-507483) and SUPER (FP6-026850).

<sup>1</sup><http://sws-challenge.org/>

level, the exchange of messages in the right order is an essential requirement for partner integration. The Blue company conforms to the PIP3A4 process while Moon follows its own internal business process. Figure 2 provides an illustration of the architecture reflecting these requirements.

### 3 Service Creation

The core aspect of the Service Creation Phase is to model the semantics of services and publish their descriptions. A number of Semantic Web service ontologies are available including OWL-S[7], SWSO[1] and WSMO[6]. In each case, the intent is to define a conceptual model that best captures the various aspects of Web services. We choose to use WSMO as our model because of its (1) explicit support of mediators and (2) ontological separation of service requester and provider roles. The WSMO Service Model defines service semantics along with *non-functional properties*, *functional properties* and *interfaces (behavior definition)* as well as *ontologies* that define the information models on which services operate. The service creation phase involves (1) the *WSDL Service Annotation* and (2) the *WSMO Service Creation*.

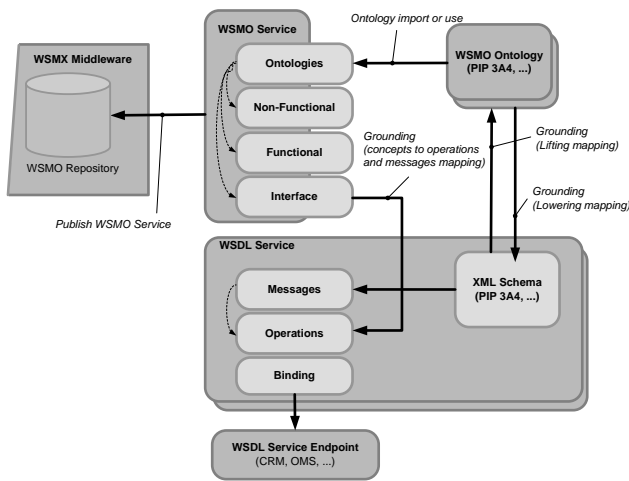


Figure 1. WSMO and WSDL Services

#### 3.1 WSDL Service Annotation

Blue's RosettaNet and Moon's CRM and OMS systems are already available as WSDL services. In order to enable semantics for these services and enable grounding from WSMO to WSDL (see figure 1), they need to be annotated with concepts from WSMO ontologies (created during WSMO Service Creation). For this purpose, we use Semantic Annotations for WSDL (SAWSDL)<sup>2</sup>. We use the *modelReference* extension attribute for annotating each type

<sup>2</sup><http://www.w3.org/2002/ws/sawsdl/>

used in WSDL with the relevant concept from the ontology. In addition we use *loweringSchemaMapping* and *liftingSchemaMapping* for the definition of transformations between ontology and Schema types and vice-versa respectively. This information is used when ontological instance data needs to be transformed to XML (and vice-versa) for the purpose of invocation using SOAP.

#### 3.2 WSMO Services Creation

In order to create WSMO services, the ontologies must be created (or reused) together with non-functional, functional and interface descriptions of services. In addition, a grounding (lowering and lifting schema mappings) must be defined. Semantic Web services are described according to WSMO Service and WSMO Goal definitions. We describe a WSMO Goal for the PIP3A4 service and a WSMO Service for the CRM/OMS service. Please note that WSMO Goal and WSMO Service have similar structural definitions but differ in what they represent. The difference is in the use of defined capability and interface – a WSMO Goal describes a capability and an interface requested by the user whereas a WSMO service describes a capability and an interface provided by a service. WSMO Goals enable goal-based service invocation which is the basis for advanced semantic discovery and mediation provided by the WSMX environment.

##### 3.2.1 Ontologies, Mappings and Grounding

Ontologies describe information models used in semantic service descriptions. In our scenario, we assume that both Blue and Moon use independent ontologies i.e. different ontologies for RosettaNet and CRM/OMS systems<sup>3</sup>. The message level interoperability must be thus reached through mappings between used ontologies which are defined during design-time and executed during runtime.

```

/* Lifting rules from XML message to WSML */
instance PurchaseOrderUID memberOf por#purchaseOrder
por#globalPurchaseOrderTypeCode hasValue "<xml:value-of select=
'dict:GlobalPurchaseOrderTypeCode'/>"
por#isDropShip hasValue
  lsDropShipPo<xml:for-each select="po:ProductLineItem">
  por#productLineItem hasValue ProductLineItem<xml:value-of
  select="position()"/>
</xml:for-each>
<xml:for-each select="core:requestedEvent">
  por#requestedEvent hasValue RequestedEventPo
</xml:for-each>
<xml:for-each select="core:shipTo">
  por#shipTo hasValue ShipToPo
</xml:for-each>
<xml:for-each select="core:totalAmount">
  por#totalAmount hasValue TotalAmountPo
</xml:for-each>

/* message in WSML after transformation */
instance PurchaseOrderUID memberOf por#purchaseOrder
por#globalPurchaseOrderTypeCode hasValue "Packaged product"

```

<sup>3</sup>Another approach would be to use one domain ontology maintained by Moon however our intention in this scenario is to demonstrate the power of WSMO mediators in service integration of services

```

por#isDropShip hasValue IsDropShipPo
por#productLineItem hasValue ProductLineItem1
por#productLineItem hasValue ProductLineItem2
por#requestedEvent hasValue RequestedEventPo
por#shipTo hasValue ShipToPo
por#totalAmount hasValue TotalAmountPo

```

### Listing 1. Lifting from XML to WSML

We assume that all ontologies are not available up-front and must be created by an ontology engineer by means of the Web Service Modeling Toolkit (WSMT)<sup>4</sup>. The engineer takes the existing standards and systems as a basis, namely RosettaNet PIP 3A4 and CRM/OMS schemas, and creates *PIP3A4* and *CRM/OMS* ontologies respectively. When creating ontologies, the engineer describes the information semantically, i.e. with richer expressivity as opposed to that of the underlying XML schema. Thus, the engineer captures the logic of “getting” from the XML schema level to the semantic level and vice-versa by lifting and lowering rules respectively. These rules are part of transformation definitions of annotated WSDL description. Listing 1, shows an example extract of lifting rules and the resulting WSML instance of a RosettaNet message in XSLT.

#### 3.2.2 Functional Description

The WSMO functional description contains the formal specification of what the service can provide. This includes the definition of conditions on service “inputs” and “outputs” which must hold before and after the service execution respectively. The functional description for our back-end systems contains conditions that the input PO data must be of a specific type and contain various information such as customer id, items to be ordered, etc. (this is modeled as preconditions of the service). In addition, the service defines its output as PO confirmation as well as the fact that the order has been dispatched. The functional description of service is used mainly for discovery and may be augmented by the specification of non-functional properties.

#### 3.2.3 Interfaces and Grounding

Interfaces describe service behavior, modeled in WSMO as a *choreography* describing how service functionality can be consumed by a service requester and, *orchestration* describing how the same functionality is aggregated out of other services. Interfaces in WSMO are described using Abstract State Machines (ASM) defining rules modeling the interactions performed by the service including grounding definitions to underlying WSDL operations.

Listing 2 shows a fragment of the choreography for the CRM/OMS service. The choreography is described from the service point of view thus the rule says that in order to send *SearchCustomerResponse* message, the *SearchCustomerRequest* message must be available. By

<sup>4</sup><http://wsmt.sourceforge.net>

executing the action of the rule (*add(...)*), the underlying operation is invoked according to the grounding definition of the *SearchCustomerResponse* concept (the grounding is determined through its *modelReference* annotation on a corresponding WSDL type) which in turn results in receiving instance data from the Web service.

```

choreography MoonChoreography
stateSignature
in moon#SearchCustomerRequest
out moon#SearchCustomerResponse

transitionRules MoonChoreographyRules
forall {?request} with (
?request memberOf moon#SearchCustomerRequest
) do
add(.# memberOf moon#SearchCustomerResponse)
endForall

```

### Listing 2. CRM/OMS Choreography

## 4 Service Execution

The solution architecture in figure 2 has two categories, namely **Existing systems** which include Moon’s back-end applications (CRM and OMS systems) as well as Blue’s RosettaNet system available as WSDL services, and the **WSMX** integration platform which facilitates the goal-driven systems integration.

Blue sends a PO request to the WSMX middleware and expects to receive a PO confirmation. In Moon, the PO request is automatically broken down to several messages and interactions: (1) obtaining internal customer ID from the CRM system, (2) opening the order in OMS system, (3) placing ordered items to the opened order in the OMS system, and (4) sending back order confirmation from the OMS system. Both Blue and Moon back-end systems have semantically rich descriptions of the information models and behavior (choreography) of both systems. This, along with additional mappings between the ontologies of the Blue and Moon systems, allows both choreographies to “connect” at run-time and resolve process interoperability issues (mediate between both choreographies). As opposed to traditional centralized solution (when a central workflow would solve this integration problem), this approach enables the automatic adaptation when changes to service *descriptions* are introduced. In contrast, solutions based on a central workflow would additionally require changes to the workflow type definition. A detailed description of the execution phase is available in [3].

## 5 Evaluation and Related Work

Our implementation has been evaluated, by peer-review, according to the criteria defined by the SWS Challenge<sup>5</sup>. The evaluation criteria targets the adaptivity of the solutions – solutions should handle introduced changes by modification

<sup>5</sup><http://sws-challenge.org>

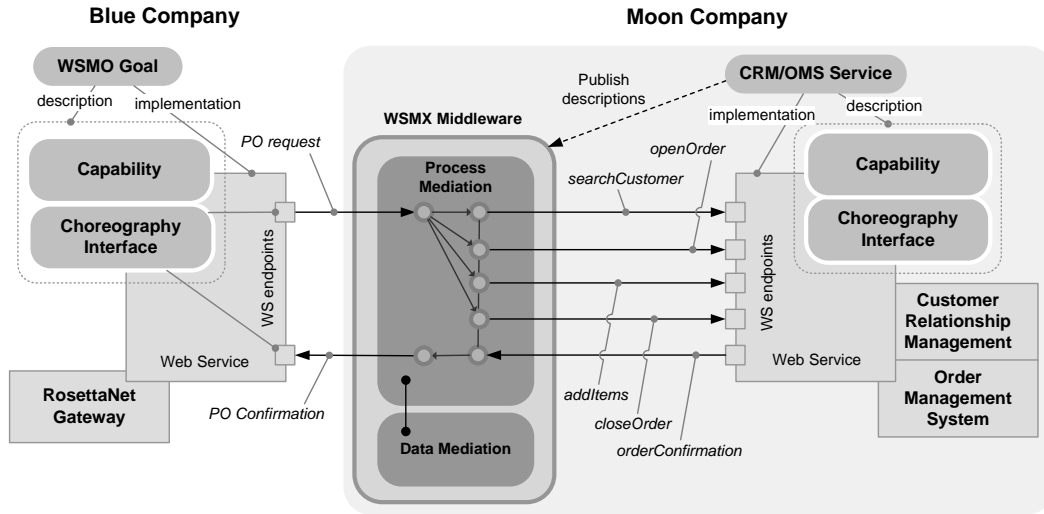


Figure 2. B2B Integration Architecture

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.

Our solution was evaluated during the SWS Challenge workshop in Budva, Montenegro<sup>6</sup>. In the data mediation scenario we had to make some changes to the code to overcome limitations of the existing data mediation tool (success level 1). For process mediation, we only needed to change the description of the service interfaces (choreographies) according to the changes in back-end systems (success level 2).

One of the main advantages of the WSMX-based integration is the strong partner de-coupling. Other solutions (e.g. WebML [2], Diane [5] or jABC [4]) require a tighter coupling which results in the limited scalability of these solutions as each new partner requires a new mapping. In our solution less integration effort is necessary. Data models expressed in terms of ontologies are built to be shared among the parties limiting the number of the data model mappings. Process mediation is performed during the runtime according to the partners' choreographies and messages exchanged between them. On the other hand, WSMO Choreography modeling is not yet as mature as the business process modeling offered by other platforms (e.g. WebML graphic process modeling).

<sup>6</sup>[http://sws-challenge.org/wiki/index.php/Workshop\\_Budva](http://sws-challenge.org/wiki/index.php/Workshop_Budva)

## 6 Conclusion

In this paper we described an approach to the integration of B2B services using Semantic Web services described in terms of WSMO and executed on a semantically enabled SOA. Taking a scenario defined by the SWS Challenge, we explained how modeling of the RosettaNet and proprietary messages ontologically allowed us to overcome heterogeneities in the data and process models used by the respective services. In particular, changes introduced to these models could be handled through modifications to the model descriptions, rather than to the system source code. We described how our approach was evaluated by peer-review and against other participants in the SWS Challenge.

## References

- [1] S. Battle et al. Semantic Web Services Framework (SWSF) Overview. Member submission, W3C, 2005.
- [2] M. Brambilla, I. Celino, S. Ceri, D. Cerizza, E. D. Valle, and F. M. Facca. A Software Engineering Approach to Design and Development of Semantic Web Service Applications. In *ISWC*, 2006.
- [3] T. Haselwanter, P. Kotinurmi, M. Moran, T. Vitvar, and M. Zaremba. WSMX: A Semantic Service Oriented Middleware for B2B Integration. In *ICSOC*, 2006.
- [4] C. Kubczak, R. Nagel, T. Margaria, and B. Steffen. The jABC Approach to Mediation and Choreography. In *Semantic Web Services Challenge, in conjunction with ESWC*, 2006.
- [5] U. Küster, B. König-Ries, M. Stern, and M. Klein. DIANE - An Integrated Approach to Automated Service Discovery, Matchmaking and Composition. In *WWW*, 2007.
- [6] D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web Service Modeling Ontology. *Applied Ontologies*, 1(1):77 – 106, 2005.
- [7] The OWL Services Coalition. OWL-S: Semantic Markup for Web Services, v1.1. Member submission, W3C, 2004.