D2.4.7 Web Service Invocation and Interoperation

Tomas Vitvar, Paavo Kotinurmi
(National University of Ireland, Galway)

with contributions from:

Armin Haller, Mick Kerrigan, Jana Viskova, Matthew Moran and Maciej Zaremba (National University of Ireland, Galway)

Abstract.
EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB Deliverable D2.4.7 (WP2.4) Version 2

The goal of this deliverable is to show how Semantic Web Service technology around WSMO, WSML and WSMX can be used to facilitate interoperation and invocation of services within inter-enterprise integration settings. We address this interoperation at technical, data and process levels. In particular, we show how RosettaNet e-business framework and SWS technologies can be used together and how SWS can facilitate integration of back-end information systems.

Keyword list: Web Services, Invocation, Interoperation, Interoperability, WSMX, B2B integration, RosettaNet

Document Identifier	KWEB/2004/D2.4.7/v2
Project	KWEB EU-IST-2004-507482
Version	v2.0
Date	June 30, 2006
State	final
Distribution	public

Knowledge Web Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2004-507482.

University of Innsbruck (UIBK) - Coordinator
Institute of Computer Science
Technikerstrasse 13
A-6020 Innsbruck
Austria
Contact person: Dieter Fensel
E-mail address: dieter.fensel@uibk.ac.at

École Polytechnique Fédérale de Lausanne (EPFL)
Computer Science Department
Swiss Federal Institute of Technology
IN (Ecublens), CH-1015 Lausanne
Switzerland
Contact person: Boi Faltings
E-mail address: boi.faltings@epfl.ch

France Telecom (FT)
4 Rue du Clos Courtel
35512 Cesson Sévigné
France. PO Box 91226
Contact person : Alain Leger
E-mail address: alain.leger@rd.francetelecom.com

Freie Universität Berlin (FU Berlin)
Takustrasse 9
14195 Berlin
Germany
Contact person: Robert Tolksdorf
E-mail address: tolk@inf.fu-berlin.de

Free University of Bozen-Bolzano (FUB)
Piazza Domenicani 3
39100 Bolzano
Italy
Contact person: Enrico Franconi
E-mail address: franconi@inf.unibz.it

Institut National de Recherche en Informatique et en Automatique (INRIA)
ZIRST - 655 avenue de l'Europe -
Montbonnot Saint Martin
38334 Saint-Ismier
France
Contact person: Jérôme Euzenat
E-mail address: Jerome.Euzenat@inrialpes.fr

Centre for Research and Technology Hellas / Informatics and Telematics Institute (ITI-CERTH)
1st km Thermi - Panorama road
57001 Thermi-Thessaloniki
Greece. Po Box 361
Contact person: Michael G. Strintzis
E-mail address: strintzi@iti.gr

Learning Lab Lower Saxony (L3S)
Expo Plaza 1
30539 Hannover
Germany
Contact person: Wolfgang Nejdl
E-mail address: nejdl@learninglab.de

National University of Ireland Galway (NUIG)
National University of Ireland
Science and Technology Building
University Road
Galway
Ireland
Contact person: Tomas Vitvar
E-mail address: tomas.vitvar@deri.ie

The Open University (OU)
Knowledge Media Institute
The Open University
Milton Keynes, MK7 6AA
United Kingdom
Contact person: Enrico Motta
E-mail address: e.motta@open.ac.uk

Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo sn
28660 Boadilla del Monte
Spain
Contact person: Asunción Gómez Pérez
E-mail address: asun@fi.upm.es

University of Karlsruhe (UKARL)
Institut für Angewandte Informatik und Formale
Beschreibungsverfahren - AIFB
Universität Karlsruhe
D-76128 Karlsruhe
Germany
Contact person: Rudi Studer
E-mail address: studer@aifb.uni-karlsruhe.de

University of Liverpool (UniLiv)
Chadwick Building, Peach Street
L697ZF Liverpool
United Kingdom
Contact person: Michael Wooldridge
E-mail address: M.J.Wooldridge@csc.liv.ac.uk

University of Manchester (UoM)
Room 2.32. Kilburn Building, Department of Computer
Science, University of Manchester, Oxford Road
Manchester, M13 9PL
United Kingdom
Contact person: Carole Goble
E-mail address: carole@cs.man.ac.uk

University of Sheffield (USFD)
Regent Court, 211 Portobello street
S14DP Sheffield
United Kingdom
Contact person: Hamish Cunningham
E-mail address: hamish@dcs.shef.ac.uk

University of Trento (UniTn)
Via Sommarive 14
38050 Trento
Italy
Contact person: Fausto Giunchiglia
E-mail address: fausto@dit.unitn.it

Vrije Universiteit Amsterdam (VUA)
De Boelelaan 1081a
1081HV. Amsterdam
The Netherlands
Contact person: Frank van Harmelen
E-mail address: Frank.van.Harmelen@cs.vu.nl

Vrije Universiteit Brussel (VUB)
Pleinlaan 2, Building G10
1050 Brussels
Belgium
Contact person: Robert Meersman
E-mail address: robert.meersman@vub.ac.be

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to writing parts of this document:

École Polytechnique Fédérale de Lausanne
France Telecom
Freie Universität Berlin
National University of Ireland Galway
University of Innsbruck
University of Liverpool
University of Manchester
University of Trento

Changes

Version	Date	Author	Changes
1.2	14.06.06	Paavo Kotinurmi	First full version based on version 1.0
1.7	19.06.06	Paavo Kotinurmi	Changes after reviewers comments from conferences
1.8	30.06.06	Paavo Kotinurmi	Version before internal review
1.9	15.07.06	Tomas Vitvar	Changes throughout the document
2.0	20.07.06	Paavo Kotinurmi	Reviewers comments implemented, final version

Executive Summary

Interoperation and invocation of web services can be distinguished at the levels of (1) technical interoperation, (2) data (semantic) interoperation, and (3) process interoperation. Technical level interoperation is the ability of systems to handle different communication protocols and invoke services using different languages. Data level interoperation must be ensured when different meanings (semantics) of messages are used for communication. Process level interoperation must be ensured when different communication patterns (choreographies) are used during communication.

With respect to all interoperation levels and within the context of ongoing work on interoperation and invocation in other research projects, we are focused in this deliverable on particular problem of interoperation of services in inter-enterprise integration settings taking into account B2B standards used by different business partners. In order to achieve interoperation of business partners by means of SWS, we show all necessary steps which need to be fulfilled within the integration set-up phase as well as describe in detail how SWS middleware facilitates integration of decoupled services during run-time.

This deliverable is directly based on two loosely dependent scientific publications. In the first publication described in the chapter 2, we are focused on definition of guidelines for integration set-up phase and integration run-time phase to be fulfilled between business partners that wish to interoperate. This work has been published in Lecture Notes in Computer Science at Springer Verlag and presented at the 2nd IEEE Workshop on Data Engineering Issues in E-Commerce 2006 (DEEC2006) in San Francisco, USA. In the second publication described in chapter 3, we are focused on run-time invocation and conversation aspects of semantic services and integration of standard RosettaNet B2B protocol with proprietary enterprise back-end systems. This work has been presented and demonstrated at the SWS challenge workshop in Budva, Montenegro while at the same time has been submitted to the 4th International Conference on Service Oriented Computing (ICSOC2006) which will be held in December 2006 in Chicago, USA. At the time of writing this deliverable, the result of this submission is not known.

Contents

Chapter 1

Introduction

1.1 Goal of the Deliverable

The goal of this deliverable is to define guidelines for interoperation and invocation of services in the inter-enterprise integration settings and to show how conversation between requesters and providers with heterogeneous interfaces can be achieved by enabling SWS middleware during the run-time by means of the Semantic Web Services (SWS) concepts and technologies, namely WSMO, WSML and WSMX. With this respect we address all interoperation levels (technical, data, and process) and show how this interoperation can be achieved with the use of SWS technology between different business partners using existing e-business frameworks and in particular RosettaNet. Although research into SWS is well established with an active community, there remains very few actual scenarios that showcase the benefits of this technology. The overall aim of this work is to discuss such a scenario and present how SWS can help to establish and maintain the integration between independent systems.

This is the second version of the deliverable which contributes to the work done within the WP2.4 Semantic Web Services including requirements for semantic description of web services, conceptual and formal framework for the Semantic Web Services, guidelines for the integration of agent-based services and web-based services, theoretical integration of Web Service discovery and composition, and reputation mechanism. In general, the work in this deliverable contributes and complement the work done for the SWS specifications around Web Service Modelling Ontology (WSMO) [RLK04] and Web Service Modelling Language (WSML) [dBLPF06], and in particular for the Web Service Modelling eXecution environment (WSMX) [ZMH05].

1.2 Overview of the Deliverable

Further in this chapter we present the terms used throughout the deliverable and describe Semantic Web Services concepts and technologies according to the WSMO and WSMX

specifications. We also explain what we understand by interoperation and invocation of web services.

The core of this deliverable is directly based on two loosely dependent scientific publications. In the first publication described in the chapter 2, we are focused on definition of guidelines for integration set-up phase and integration run-time phase to be fulfilled between business partners that wish to interoperate. This work has been published in Lecture Notes in Computer Science at Springer Verlag and presented at the 2nd IEEE Workshop on Data Engineering Issues in E-Commerce 2006 (DEEC2006) in San Francisco, USA [KVH+06]. In the second publication described in chapter 3, we are focused on run-time invocation and conversation aspects of semantic services and integration of standard RosettaNet B2B protocol with proprietary enterprise back-end systems. This work has been presented and demonstrated at the SWS challenge workshop in Budva, Montenegro while at the same time has been submitted to the 4th International Conference on Service Oriented Computing (ICSOC2006) which will be held in December 2006 in Chicago, USA. At the time of writing this deliverable, the result of this submission is not known. In chapter 4, we describe expected benefits from this approach and discuss related work. Finally in the chapter 5, we conclude the deliverable and discuss our future work.

1.3 Terminology

In this section, we present major terms used throughout the deliverable.

- **CRM** - Customer Relationship Management system. An information system used in enterprises to handle customer information.
- **e-business framework** - a standard currently used for B2B communication such as RosettaNet or EDI, that define how B2B integration takes place.
- **EDI** - Electronic Data Interchange. Standard for B2B integration, i.e. an e-business framework. First EDI standards are from 1970's but they are still widely used in B2B integration.
- **ERP** - Enterprise Resource Planning. An information system used in enterprises for storing financial, logistics, material etc. information.
- **Idoc** - SAP intermediate documents. A format for connecting SAP ERP system to other systems.
- **Inter-company integration** - B2B integration across companies, typically happens using standards.
- **Intra-company integration** - Enterprise internal application integration, deal with heterogeneous back-end systems.

- **OMS** - Order Management System. An information system used in enterprises to handle order-related information.
- **Organization** - refers to the buyer in the scenario of this deliverable.
- **Partner** - Refers to the potential suppliers (sellers) in the scenario of this deliverable.
- **PIP** - Partner Interface Process. RosettaNet standard for B2B message exchange. Contain both process and message guidelines.
- **PO** - Purchase Order. A process used in the scenario to get the information for purchase the needed devices.
- **RFQ** - Request For Quote. A process used in the scenario prior to PO to get the price and shipment information.
- **RNIF** - RosettaNet Implementation Framework. RosettaNet standard for secure transport of RosettaNet PIPs. Guides how messages are packed, secured and acknowledged, when exchanged over the Internet from peer-to-peer.
- **SWS** - Semantic Web Service. Semantically enabled web services are represented so that computers can understand them.
- **UDDI** - Universal Description Discovery and Integration. Repository where WSDL documents are often stored.
- **WSDL** - Web Services Description Language. Way to describe the web service information.
- **WSML** - Web Service Modelling Language. WSML is a language that formalizes the WSMO.
- **WSMO** - Web Service Modelling Ontology. WSMO adheres to the principles of loose coupling of services and strong mediation among them. WSMO defines an underlying model for WSMX.
- **WSMX** - Web Service Modelling eXecution environment. WSMX is the reference implementation of WSMO and it uses WSML as internal language. It is an execution environment for business application integration where enhanced web services are integrated for various business applications.

1.4 Semantic Web Services

Web Services are small units of functionality, which are made available by service providers for use in larger applications. The intention when developing Web Services was to reduce the overhead needed to integrate functionality from multiple providers. Communication with Web Services is usually achieved using the SOAP protocol [GHM+03]. SOAP is an XML-based protocol for communication between distributed environments. Descriptions of the interfaces of the Web Services are described using the Web Service Description Language (WSDL) [CCMW01]. WSDL documents are generally stored in a Universal Description Discovery and Integration (UDDI) [1] repository where services can be discovered by end-users.

A major issue with Web services is that their interfaces are only explained in a syntactic fashion using WSDL documents. While it is possible for computers to process these documents it is not possible for computers to 'understand' them. Inevitably a human is required to find WSDL documents in a UDDI repository, study these documents and understand them in order to integrate the Web services into a system. While Web services have indeed reduced the overhead needed to integrate functionality from multiple providers, extensive human interaction is still required in the process. Semantically-enabled web services are forming the research area known as Semantic Web Services (SWS). Semantic Web Services complement standards around WSDL, SOAP and UDDI with aim to enable total or partial automation of tasks such as discovery, selection, composition, mediation, invocation and monitoring of services. The research lies in definition and development of concepts, ontologies, languages and technologies for SWS. A number of initiatives exist looking at how to create and manage semantic description for Web Services including OWL-S [M+04], Meteor-S [POSV04], WSDL-S [WSD05] and WSMO [RLK04].

One of the major added values of the SWS according to the WSMO concepts lies in the (semi) automated interoperation and invocation of services. By interoperation and invocation of services we understand the ability of different services to communicate at different levels, namely *technical*, *data* and *process* levels.

1. Technical level – interoperation must be ensured when different protocols are used for communication as well as different languages. Interoperation at this level is achieved by adaptation of protocols and languages used, that is by their syntactical translations as well as grounding to underlying communication protocols (invocation of WS).

2. Data level – interoperation must be ensured when different meanings (semantics) of messages are used for communication. It is achieved by data mediation with use of ontology integration techniques, such as ontology mapping/aligning.

3. Process level –interoperation must be ensured when different communication patterns (choreographies) are used during communication. It is achieved by process

[1] http://www.uddi.org/

mediators providing functionality for a runtime analysis of two given patterns, and compensates for the possible mismatches which may appear.

1.4.1 WSMO, WSML, and WSMX

Web Services Modeling Ontology (WSMO) has its conceptual basis in the Web Service Modeling Framework (WSMF) [FB02], which adheres to the principles of loose coupling of services and strong mediation among them. WSMO defines an underlying model for the WSMX SWS execution environment [ZMH05] as well as draws up requirements for a WSML ontology language [dBLPF06] used for formal description of WSMO elements. Thus, WSMO, WSML and WSMX form a complete framework to deal with all aspects of the Semantic Web Services.

WSMO top-level conceptual model is composed of *Ontologies*, *Goals*, *Web Services* and *Mediators*.

Ontologies provide formal explicit specification of shared conceptualization that is formal semantics of information used by other components (goals, web services, and mediators). WSMO specifies the following constituents as part of the description of an ontology: *concepts*, *relations*, *functions*, *axioms*, and *instances* of concepts and relations, as well as *non-functional properties*, *imported ontologies*, and *used mediators*. The latter allows the interconnection of different ontologies by using mediators that solve terminology mismatches.

Goals provide description of objectives of a service requester (user) that he or she wants to achieve. WSMO goals are described in terms of desired information as well as "state of the world" which must result from execution of a given service. In WSMO, a goal is characterized by a set of *non-functional properties*, *imported ontologies*, *used mediators*, a *requested capability* and a *requested interface* (these definitions are the same as for web services).

Web Services provide a functionality for a certain purpose, which must be semantically described. Such description includes *non-functional properties*, *imported ontologies*, *used mediators*, *capability* and *interfaces*. *Capability* of a web service is modeled by *preconditions* and *assumptions* for the correct execution of the web service as well as *postconditions* and *effects* resulting from this execution. The interface for every web service is modeled as *choreography* describing communication pattern (interactions) with this web service and *orchestration* describing partial functionality required from other web services.

Mediators describe elements that aim to overcome structural, semantic or conceptual mismatches that appear between the different components that build up a WSMO description. WSMO specification currently covers four types of mediators: (1) *OOMediators* import the target ontology into the source ontology by resolving all the representation mismatches between the source and the target, (2) *GGMediators* connect goals that are

in a relation of refinement and resolve mismatches between those, (3) *WGMediators* link Web services to goals and resolve mismatches, and (4) *WWMediators* connect several Web services for collaboration.

1.4.2 SWS Execution Environment (WSMX)

Based on WSMO concepts, the Web Services Execution Environment (WSMX) is the execution environment for discovery, composition, engagement, selection, mediation and invocation of Semantic Web Services.

The WSMX Manager and the Execution Engine facilitate a SWS execution process (execution semantics) by triggering discovery, composition, engagement, selection, mediation and invocation components upon receiving users' request (goal) and within the whole interaction process between service requester and service providers. The WSMX execution semantics is the core of the WSMX intelligence providing value-added services to traditional communication. Different execution semantics can be used according to the domain-specific requirements, e.g. mediation components are only required for heterogeneous environments, a selection component is used when (semi) automated decisions to select the best services are required based on requesters' preferences, or specific execution semantics exists for registering ontology or service in WSMX repositories. Typically, execution semantics is triggered based on the WSMX system entry point invoked by external application (adapter, front-end application, etc.).

The Resource Manager is responsible for the management of repositories to store definitions of web services, goals, ontologies and mediators.

Discovery and Composition of web services is one of the key processes of the SWS technology. A number of services could be returned from this step, services which satisfy the goal composed to a predefined process (*once-for-all* composition) as well as dynamically created process (*on-the-fly* composition). Such composition of services could also include "duplicate" services with the same capabilities however with different characteristics (non-functional properties). For duplicate ones, selection of services will be performed.

Engagement is composed of two phases, namely *contracting* and *negotiation*. Usually, discovery and composition operate on more general (abstract) goals. They result with a set of web services that can potentially fulfill a requester goal. However, for a complete guarantee that discovered web services will be able to provide requested concrete service, communication between a requester and a provider is necessary. This phase is called contracting. In addition, negotiation with each perspective web service to reach agreement on terms of services can also be performed.

Selection of the best or optimal service is performed when a number of duplicate services is returned from the discovery and composition process. To find an optimal service, different techniques can be applied, ranging from simple selection criteria ("always

the first") to more sophisticated techniques, such as multi-criteria selection of variants also involving interactions with a service requester. Different variants of services could be described by different values of parameters (non-functional properties specific to web services), such as financial properties, reliability, security, etc.

Data and Process Mediation facilitate interactions between two entities when different ontologies or different choreographies are used by these entities. Data Mediation is ensured by mappings between concepts from one ontology to another one. It is based on paradigms of ontology engineering, i.e. ontology mapping/aligning. Process mediation provides the necessary functionality for a runtime analysis of two given choreography instances and compensate possible mismatches that may appear, for instance, grouping several messages into a single one, changing their order or even removing some of the messages.

Invoker and Receiver implement an entry point of the WSMX responsible for receiving of incoming requests and invoking web services respectively. Invoker and receiver also handle grounding of WSMO services to underlying WSDL and SOAP protocol.

Adapters lie outside of WSMX and facilitate the interoperability between a requester and a provider at the technical level. WSMX is designed to internally handle WSML messages encapsulated in WSDL and sent or received using SOAP protocol. Therefore, adapters must ensure that interactions between a service requester and provider can be performed using different communication protocols (e.g. FTP) as well as in different languages (such as XML).

Chapter 2

Set-up and Runtime in the Semantic B2B Integration

In this chapter we present all aspects of the B2B integration built on the SWS technologies defining two integration phases, namely integration set-up phase and integration runtime phase. We illustrate these phases on the typical B2B scenario where one organization faces the problem of multiple B2B protocols used by its counterparts. With use of SWS technologies we present how integration of heterogeneous organizations using different B2B protocols can be managed in more flexible way in regards to changes that might occur over a software systems lifetime.

2.1 Use Case Description

We consider an *organisation A* that manufactures electronic devices. For a particular device, organisation A needs specific components that can be delivered by approved suppliers, referred here as *partners B* and *C*. In the current situation, the B2B integration only covers purchasing activities as shown in figure 2.1 and there is no competition for purchasing per delivery basis. In this proposed scenario, organisation A first submits *Requests For Quotes* (RFQ) to all its suppliers for the components. After the responses, it selects the best quote and initiates the *Purchase Order* (PO) process with the selected partner.

We will show from the picture 2.1, how SWS technologies can be used from the organisation A point of view concentrating in the integration setup phase and overall the **inter-company** integration heterogeneity.

Considering the integrations, the following heterogeneities exist with partners according to general B2B integration levels [MBB+03, NK04]:

Transport level interoperation is needed to understand different languages used to

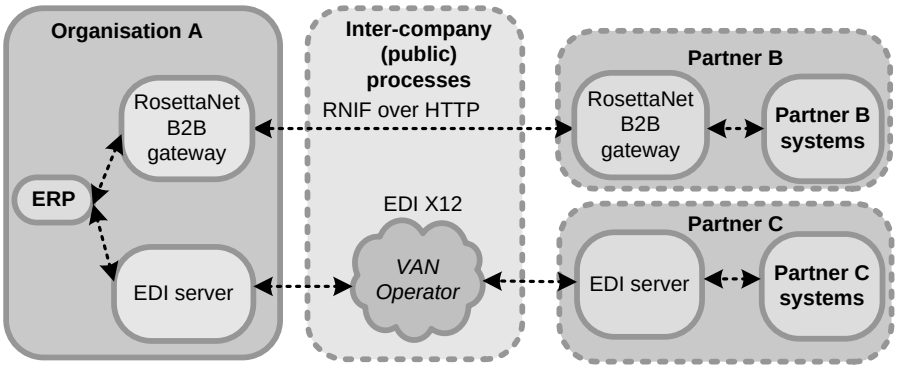

Figure 2.1: Overall scenario

describe the messages exchanged and how the message exchange happens. In the inter-company integration there are following heterogeneities. RosettaNet partners use Implementation Framework (RNIF) 2.0 over HTTP(S) for secure communication and the message contents are in XML. RNIF guides how the messages are sent and acknowledged and how digital signatures are used. With EDI partners the communication is achieved via a Value Added Network (VAN) operator, which takes care of the communication between the partners. EDI X12 format messages are put to a file system folder, where the VAN operator collects the messages and ensures the secure delivery of the messages to the partners.

Data level interoperation is the ability to understand exchanged messages (sometimes referred as business documents or payload). In the inter-company integration, RosettaNet defines *Partner Interface Processes* (PIPs) that define standard inter-company process choreographies and the related schemas for the XML messages exchanged. *Partner B* uses the PIP *3A1 Request for Quote* and *3A4 Request Purchase order* messages [1] according to the message guidelines provided by RosettaNet. Both PIPs contain request and response messages. *Partner C* uses EDI X12 messages and expects the *840 Request for Quotation* for quotes and *850 Purchase Order* for orders [2]. The quotes are responded with *879 Price Information* message and the purchase orders by *855 Purchase Order Acknowledgment* message. These PIP and EDI X12 messages use different terms and identifiers in referring to the same concepts. In addition, for example the product identifiers used differ among the companies.

Business Process level interoperation is the ability of companies to exchange messages in the right sequence and timing. In the inter-company integration *Partner B* complies with PIP 3A1 and 3A4 standard choreographies. That means the partner's response arrives within 24 hours of sending the requests. For every PIP message sent, there is a receipt acknowledgment for delivery. *Partner C* with EDI X12 has not such fixed response times between different messages as it is not dictated by EDI X12. In this case the partner

[1] http://www.rosettanet.org/pipdirectory
[2] http://www.disa.org/x12workbook/ts/

C has agreed to answer the quotes and purchase orders in the same 24 hours. Hence, the choreography differs since the receipt acknowledgment message is not always used with EDI.

2.2 B2B Integration with SWS Technologies

This section introduces a WSMX enabled architecture to address the requirements of the scenario from buyers point of view. We show how SWS technologies can help to mediate the technical, data and process level heterogeneities in inter-company integration. We outline some prerequisites for a SWS enabled solution, describe the integration set-up phase using the WSMX and then describe the run-time behaviour. For brevity, we concentrate on presenting the scenario with *Partner B*, who utilises RosettaNet e-business framework as depicted in figure 2.2.

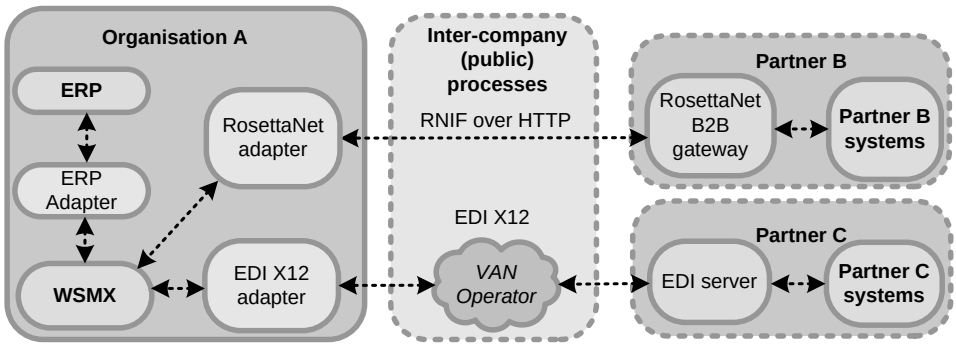

Figure 2.2: Use Case Scenario

2.2.1 Prerequisites for SWS infrastructure for Organisation A

In this section we analyse the prerequisites organisation A has to address when it sets up the WSMX environment for the B2B integration described above.

Message Ontologies Based on its requirements, organisation A has to create or ideally reuse *domain ontologies*. In our example these ontologies are used for a formal description of the RFQ and PO process messages. Creating these domain ontologies requires an expert who first understands specific e-business scenarios and second has knowledge about ontology languages to be able to capture information in messages *semantically*. However, since we are still far from an industry wide recognised formal ontology, organisation A in our example needs to define the ontology itself. We assume that organisation A is not in a position to dictate its proprietary ontology to its partners. It is further realistic to assume that it bases its ontology on an existing e-business framework, in our case RosettaNet. This for two reasons, first that organisation A minimises the effort of

lifting the RosettaNet PIP XML messages most of its partners still use to the ontological level and second to further minimise the mapping requirements to its internal ERP system, which still operates on syntactic messages. Figure 2.3 outlines how the ontology is applied by organisation A and where the lifting/lowering of messages is performed.

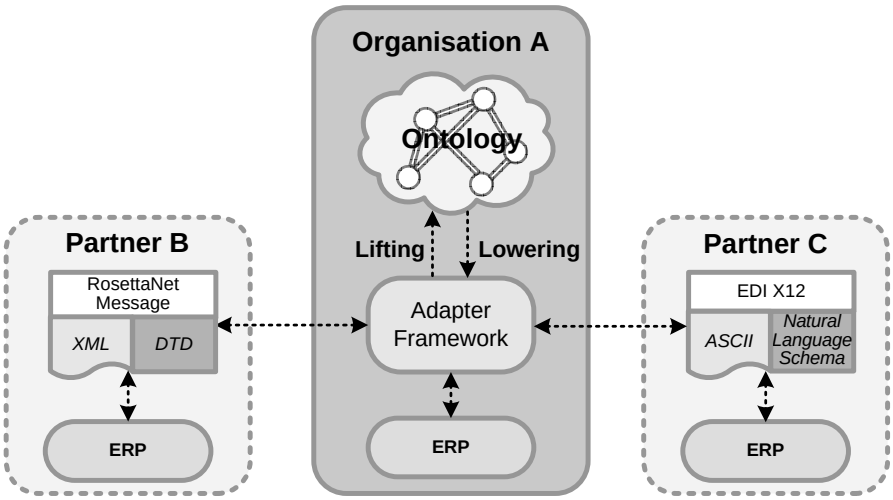

Figure 2.3: Lifting/Lowering to/from Domain Ontology

The ontology in our scenario ultimately represents simply a different serialisation of the information in the RosettaNet framework with the advantages that it explicitly states logical relationships between elements which can not be expressed in the RosettaNet PIP XML Schemas and DTDs. In RosettaNet this information is included in a natural language document explaining the respective PIP. One of these advantages of the higher expressivity in the WSML ontology language is depicted in the following example.

The DTD versions of PIP 3A1 and PIP 3A4 support two different kind of product identifiers; the Global Trade Identification Number (GTIN), which is recommended by RosettaNet, and company-specific identifiers. The extract in listing 1 shows the definition of product identifiers in the PIP 3A1 (and 3A4 DTD version). The PIP3A1 DTD is very long so only the relevant lines (291-304) are shown.

```
291  <!ELEMENT ProductIdentification
292    (GlobalProductIdentifier?,
293     PartnerProductIdentification*)>
294
295  <!ELEMENT GlobalProductIdentifier
296    (#PCDATA)>
297
298  <!ELEMENT PartnerProductIdentification
299    (GlobalPartnerClassificationCode,
300     ProprietaryProductIdentifier,
301     revisionIdentifier?)>
302
303  <!ELEMENT ProprietaryProductIdentifier
304    (#PCDATA)>
```

Listing 1: PIP 3A1 DTD extract

RosettaNet message guidelines for PIP 3A1 add a *natural language constraint* for ProductIdentification that the DTD's expressive power does not capture: *Constraint: One instance of either "GlobalProductIdentifier" or "PartnerProductIdentification" is mandatory.* Without this constraint, a valid ProductIdentification could be without any identifiers as both identifications are optional.

Some of the RosettaNet PIPs have also an XML Schema definitions that can present cardinality constraints for the elements. Listing 2 shows an extract of the PIP 3A4 XML schema, where namespaces and annotations are dropped for brevity. The XML Schema has different element names than the ones in DTDs. It also allows arbitrary authorities to specify the identification schemes, which introduces another mapping challenge.

```
<xs:element name="ProductIdentification" type="ProductIdentificationType" />
<xs:complexType name="ProductIdentificationType">
 <xs:complexContent><xs:sequence>
   <xs:element name="ProductName" type="xs:string" minOccurs="0" />
   <xs:element name="Revision" type="xs:string" minOccurs="0" />
   <xs:choice><xs:element ref="AlternativeIdentifier" maxOccurs="unbounded" />
   <xs:element ref="GTIN" /></xs:choice>
 </xs:sequence></xs:complexContent>
</xs:complexType>
<xs:element name="AlternativeIdentifier" type="AlternativeIdentifierType" />
<xs:complexType name="AlternativeIdentifierType">
 <xs:sequence><xs:element name="Authority" type="xs:string" />
 <xs:element name="Identifier" type="xs:string" /></xs:sequence>
</xs:complexType>
```

Listing 2: PIP 3A4 XML Schema extract

The product identifier information in the WSML domain ontology is presented in listing 3. In this, the GTIN is handled as any other identification authority/qualifier (*qualificationAgency*) and the RosettaNet DTD, XML Schema, and EDI X12 product identification information can be presented in this ontology including the natural language constraints. The value of qualification agency can be for example the *buyer, seller or manufacturer* or any other identification scheme provider. The axiom in listing 3 makes sure that the value of *qualificationAgency* is among those supported for organisation A. Thus, the benefit from applying a more expressive language such as WSML is that it allows the description of logical relationships between the elements. This information can subsequently be applied for better validation of the message contents.

```
244 concept productIdentification
245     nonFunctionalProperties
246         dc#description hasValue "Collection of business properties describing identifiers ."
247     endNonFunctionalProperties
248     productIdentifier ofType (1 1) _string
249     qualificationAgency ofType (1 1) _string
250     revision ofType (0 1) _string
251 axiom qualificationAgencyConstraint
252     nonFunctionalProperties
253         dc#description hasValue "The valid list of agencies who have defined product identifiers."
254     endNonFunctionalProperties
```

```
255     definedBy !– ?x[qualificationAgency hasValue ?type]
256         and (?type = ”GTIN” or ?type = ”Manufacturer” or ?type = ”Buyer”
257             or ?type = ”Seller” or ?type = ”EN” or ?type = ”BP”).
```

Listing 3: Product ontology extract in WSML

Adapters to the Back-end applications

For the integration of back-end systems, the messages used within the ERP system have to be mapped to/from the domain ontology. The ERP adapter is required to perform the lifting/lowering of the internally used messages in the ERP system (e.g. Intermediate Documents (IDocs) in the case of SAP) to the logical framework, i.e. WSML, required by WSMX.

Registration

In current e-business frameworks, a prior agreement between business partners determines with whom the partners do business. For RosettaNet, this includes the specification on the set of PIPs used by the partner in the communication and the *role* for the partner in a certain PIP (e.g. *seller* or *buyer*). In addition, the partners need to provide the *endpoint* information of the *IP address* and the *port*, as well as the public *certificate* used by the partner to sign the messages.

In a SWS enabled integration process a *registration interface* allows a partner to register this information to the SWS environment of organisation A. The registration interface can be accessed by a partner through a web portal or an API of organisation A. By invoking this registration interface, a service description based on a set of PIPs and roles is created and described in WSML. The semantic service description provides all information including the endpoint information necessary to invoke the service.

Adapter framework to external partners

The adapter framework is required to provide a communication interface to partners, who are not able to directly provide WSML compliant messages to WSMX. The adapter framework receives every non-WSML message and acts for WSMX as the actual service. Thus, essentially the adapter functionality is registered as a service with the system. Further, WSMX only operates on the choreography of the adapter (c.f. left part of figure 2.4), which maps between the choreography of the partner's (c.f. right part of figure 2.4) e-business environment and the choreography registered with WSMX. The choreography definition is part of the WSMO description of a service and specifies the input and output operations as well as transition rules and constraints on the states in the communication.

Figure 2.4 shows the RosettaNet adapter execution flow in a PIP process of RFQ request and response messages:

- WSMX first sends the WSML message to the RosettaNet adapter as a WSML RFQ message.
- The adapter receives the RFQ and translates it to a RosettaNet RFQ XML message.

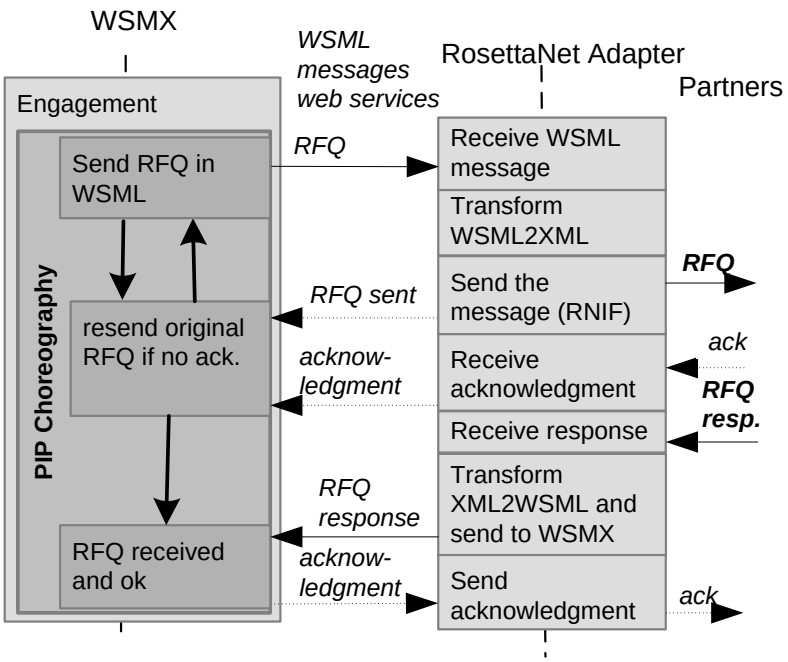

Figure 2.4: RosettaNet Adapter for Organisation A

- The adapter creates a RNIF envelope for this message and signs the message using certificates and sends it to the endpoint of partner B (certificate as well as endpoint are implemented in the adapter). As a result, a confirmation that the message has been received is sent back to WSMX.
- WSMX subsequently expects an acknowledgment message by partner B as an RNIF 2.0 signal message.
- After receiving the acknowledgment, WSMX waits for the RFQ response from partner B.
- The adapter receives the RFQ response and translates it using an XSLT script to WSML and sends it to WSMX to check that the response does not violate the axioms of the ontology. This response is processed in WSMX and sent to the back-end applications. WSMX also forwards an acknowledgment signal indicating that their RFQ response was received to the adapter which translates it RosettaNet message acknowledgment.

2.2.2 Integration Set-up Phase

In the integration set-up phase, the B2B integration for a specific partner is built. The integration set-up phase also includes the registration of the partner's service description with the SWS system of organisation A. Hence, in case a new RosettaNet partner wants to

register a service, he needs to provide his endpoint information and choreography details. Partners need to provide this information using the *Registration Interface* described in the previous section. In addition, the information what components they supply is required for discovery.

In case that the new partner uses some other standard or a proprietary format for his messages appropriate adapters and mapping rules might need to be defined.

Creating the adapter to the partners e-business frameworks

The integration of the organisation A and the e-business frameworks of each partner needs specific *RosettaNet* and *EDI X12 adapters*. The role of the adapters is to translate the RosettaNet XML and EDI X12 data formats to WSML and taking care of e.g. RosettaNet-specific RNIF protocol details. For example, the E-business framework message translation based on the mapping rules happens in the WSMX adapter.

- The **communication interface with a partner** is used to send and receive e-business framework-specific messages. It acts as a wrapper for RosettaNet communication with the partner.

  Other basic functionality of the RosettaNet adapter involves the functionality related to enveloping, encrypting and decrypting and validation of RosettaNet messages. Here, the existing B2B gateway functionality can be used if the organisation already has a product for RNIF communication. The RosettaNet adapter needs to have roughly similar functionality to the system presented in [TKS05] with the additional step of XML2WSML and WSML2XML translations. Similarly the ERP and EDI X12 adapters need analogical functionality.

- **Creating data mapping rules from RosettaNet messages to domain ontologies**

  The mapping rules need to be defined for the run-time phase to lift RosettaNet instance messages to the ontology applied by organisation A and lower it back to the XML level respectively. In the scenario mapping rules for PIPs 3A1 and 3A4 are required. There are two options to do that, either to lift the messages from XML Schemas to WSML and then use a data mediation tool such as the one included in the Web Services Modeling Toolkit [3] to perform the mappings on the ontological level or to directly lift the messages to the domain ontology and essentially implement the mediation in the adapter. In this case, organisation A has chosen the latter option and we perform the using XSLT stylesheets. Listing 4 contains such an example mapping from a PIP DTD to WSML. The mapping lifts the GTIN number to the uniform identification scheme in the ontology. Similarly with EDI X12 the information is lifted to our domain ontology. In the lowering of messages, by knowing that a GTIN identifier and company-specific identifiers point to the same product, the mapping can provide an identifier needed by the given partner. The mapping rules need to be registered in the WSMX ontology

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

repository for run-time mappings. As the product information definitions in all DTD and XML Schema based PIPs are similar, these mapping templates can be reused with all the PIPs. With small modification it is easy to create templates for other e-business frameworks as well.

```
<xsl:for−each select="ProductIdentification/GlobalPartnerClassificationCode">
  instance localUID memberOf productIdentification
    productIdentifier hasValue <xsl:value−of select="."/>
    qualificationAgency hasValue GTIN
</xsl:for−each>

<xsl:for−each select="ProductIdentification/PartnerProductIdentification/">
  instance localUID memberOf productIdentification
    <xsl:for−each select="ProprietaryProductIdentifier">
      productIdentifier hasValue <xsl:value−of select="."/>
    </xsl:for−each>
    <xsl:for−each select="GlobalPartnerClassificationCode">
      qualificationAgency hasValue <xsl:value−of select="."/>
    </xsl:for−each>
</xsl:for−each>
```

Listing 4: DTD-based PIP instance mapping extract

2.2.3 Integration Run-time Phase

After the set-up phase is completed, WSMX is ready for running the processes. We describe here the whole execution process and interactions in WSMX according to the scenario: (1) *Converting back-end message to a WSMX goal*, (2) *Discovery* of the possible suppliers capable of fulfilling this request, (3) *Engagement* to negotiate and contract with the discovered suppliers to get the price and condition information, (4) *Selection* of the best supplier, (5) *Invocation* of the PO process with the selected supplier and finally (6) *Returning the answer* to the ERP. The sequence diagram for the run-time behaviour is depicted in figure 2.5. In the figure 2.5, the acknowledgement messages shown in figure 2.4 are abstracted for clarity.

- **Converting back-end message to WSMX goal**. Organisation A's ERP system sends out a request in its proprietary format to the back-end adapter. The request is *to get 10 display units X delivered to the plant in Galway, Ireland within 8 days*. The adapter translates this to WSML and converts it to a *goal* in WSML and sends it to WSMX.

- **Discovery**. The execution process starts by invoking the WSMX discovery component. All services in the repository matching the request are found. In our case the services of partners B and C are discovered as potential suppliers. During the discovery, data mediation rules could be executed to resolve differences in the ontologies used for the goal and the service descriptions. However, in our example we only deal with one ontology and all the mediation is done in the lifting and lowering of the XML messages.

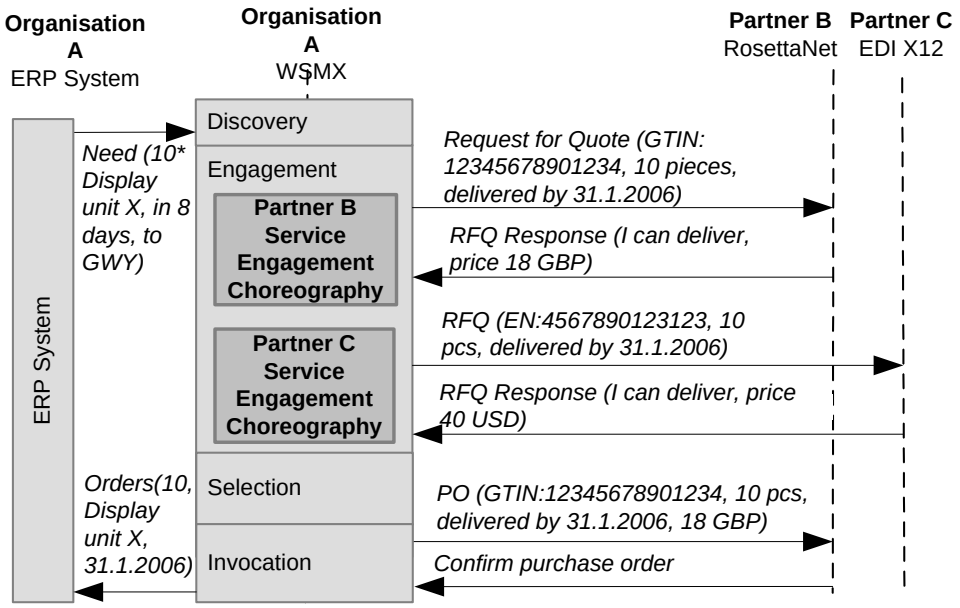

Figure 2.5: WSMX Process interactions

- **Engagement**. As discovery operates on abstract description of services, the next step is to find out whether each discovered service can deliver the required product within the given time and give a price for that. In our example, engagement is performed for partners B and C by sending RFQ documents and the partners answer those with the RFQ responses. Data and process mapping rules are implemented in the adapters, which handle differences between the RosettaNet and the X12 message exchange patterns (choreographies). Responses coming in RosettaNet PIP 3A1 and EDI X12 879 messages are translated to WSML and sent to WSMX.

- **Selection**. Based on the information provided from engagement, the best service is selected. In this scenario, this is done simply according to the cheapest price. To do this a conversion of different currencies used in quotes is done in WSMX by invoking an appropriate currency transformation service. In our scenario, the partner B has a cheaper quote and is selected.

- **Invocation**. The PO process starts with the partner B using PIP3A4. The concrete interactions between WSMX and partner B happens analogically to the case of the engagement choreography, just the messages exchanged are different. This interaction is similar to the one handled in more detail on receiver part in the chapter 3.

- **Returning answer**. After the invocation returns the PO response, the necessary data mediation for the product identifiers and currencies expected by the organisation A's ERP is done. Then the result is sent back to ERP adapter as expected by the ERP system.

Chapter 3

Conversation of Decoupled Services in the Semantic B2B Integration

In this chapter we present the conversation between a requester and a provider in a B2B integration scenario. Services used in the conversation have heterogeneous interfaces and the conversation is achieved by SWS middleware platform performing data and process mediation during runtime. Our solution is based on the WSMX-based implementation from the supplier point of view in a RosettaNet Purchase Order exchange. This view is also used to tackle the first two phases of the SWS Challenge [1] considering data and process mediation. The SWS Challenge initiative is in more detail described in the deliverable D2.4.13 Report on SWS Challenge. In particular we show how SWS technologies can be applied to a real-world B2B scenario involving (1) semantic representation of XML schema for RosettaNet as well as a proprietary purchase order using WSML ontology language, (2) semantic representation of services provided by partners using WSMO ontology, (3) executing a conversation between partner services using the WSMX integration middleware, and (4) application of data and process mediation between heterogeneous services where necessary.

The work described in this chapter has been implemented on the WSMX technology and demonstrated at the SWS Challenge workshop in Budva, Montenegro in June 2006. The source codes and running environment for this solution is available through the SWS Challenge web site[2].

3.1 Use Case Description

An overview of the architecture for our solution to the SWS Challenge use case is depicted in figure 3.1. In the use case, a fictitious trading company called Moon uses two back-end

[1] http://sws-challenge.org/

[2] http://sws-challenge.org/2006/submission/deri-submisson-mediation v.1/

systems to manage its order processing, namely, a Customer Relationship Management system (CRM) and an Order Management system (OMS). The challenge provides access to both systems through public Web services described using WSDL. Moon has signed agreements to exchange purchase order messages with a partner company called Blue using the RosettaNet PIP 3A4. In order to address the integration of Blue and Moon, we use SWS technology to facilitate conversation between all systems, to mediate between the PIP 3A4 and the XML schema used by Moon, and to ensure that the message exchange between both parties is correctly choreographed. Data mediation is required to map the Blue RosettaNet PIP 3A4 message to the messages of the Moon back-end systems. Process mediation is required to map the message exchange defined by the RosettaNet PIP 3A4 process to that defined in the WSDL of the Moon back-end systems. The data and process mediation operates on semantic descriptions of messages, thus lifting and lowering of messages from the syntactic to the ontological level is also necessary.

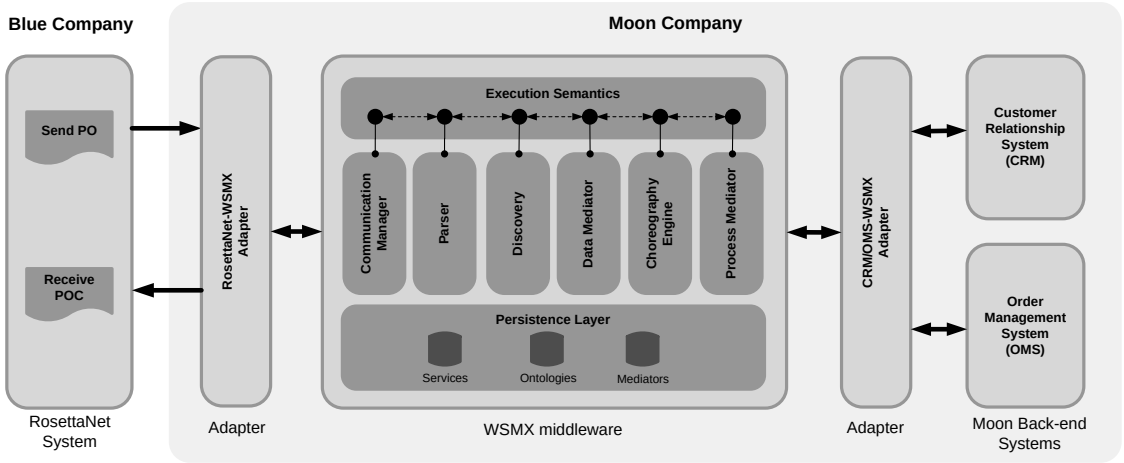

Figure 3.1: Architecture Overview

Figure 3.1 shows that theWSMX platform used to integrate the Blue and Moon partners is placed within Moons infrastructure allowing the seamless integration of its back-end systems. Following is a description of the basic blocks of the architecture.

- **Existing Systems.** The existing systems are Moon's back-end applications including CRM and OMS systems as well as the RosettaNet. Each system communicates using different formats. RosettaNet interaction is according to the RosettaNet PIP 3A4 (Purchase Order (PO)), whereas communication with the CRM and OMS systems is proprietary, specified in their WSDL descriptions (available at SWS Challenge web site: http://sws-challenge.org).
- **Adapters.** In order to connect existing systems with WSMX, adapters in general are used to mediate between different communication protocols and languages. Since WSMX internally operates on the semantic level (WSML), adapters facilitate *lifting* and *lowering* operations to transform between XML and WSML. In

addition, WSMX adopts a principle of a Goal-based invocation as the basis for advanced semantic discovery and mediation, thus the RosettaNet-WSMX adapter is also responsible for identifying the WSMO Goal that corresponds to a PO request and sending it to WSMX. The same adapter also subsequently sends the lifted form (WSML) of the PO message.

- **WSMX.** WSMX is the integration platform which facilitates the integration process between different systems. The integration process is defined by the execution semantics of WSMX, which defines the interactions of middleware services including discovery, mediation, invocation, choreography, repository services, etc. A detailed description of the execution semantics and middleware services are provided later in this section.

3.2 Integration Phases

There are two phases to the integration of the Blue and Moon partners: (1) integration setup phase and (2) integration runtime phase. During the setup phase, the integration ontologies are designed including those used for the RosettaNet PIP3A4 and those used by the CRM and OMS systems. The design and implementation of adapters, creation of WSMO ontologies and services, rules for lifting/lowering, mapping rules between used ontologies and registration of ontologies, services and mapping rules with WSMX are all carried out. During the runtime phase, the interactions between Blue and Moon are executed via WSMX.

The integration setup phase is described in a detail in previous chapter. In this chapter we concentrate on integration runtime phase with respect to the B2B use case, showing fragments of ontologies, service descriptions, and rules for lifting and lowering.

3.3 WSMX Runtime Interactions

In this section, we describe in detail all interactions between the RosettaNet and Moon systems facilitated by WSMX through its middleware services. The activity diagram is depicted in figure 3.2. In the subsequent text we describe various parts of the figure 3.2 referencing numbers from the figure which are included before title of each subsection.

1 – Sending Request.

A PIP3A4 purchase order is sent from the RosettaNet system to the entry point of the RosettaNet-WSMX adapter. On successful reception of the message by the adapter, an acknowledgment is sent back to the RosettaNet system.

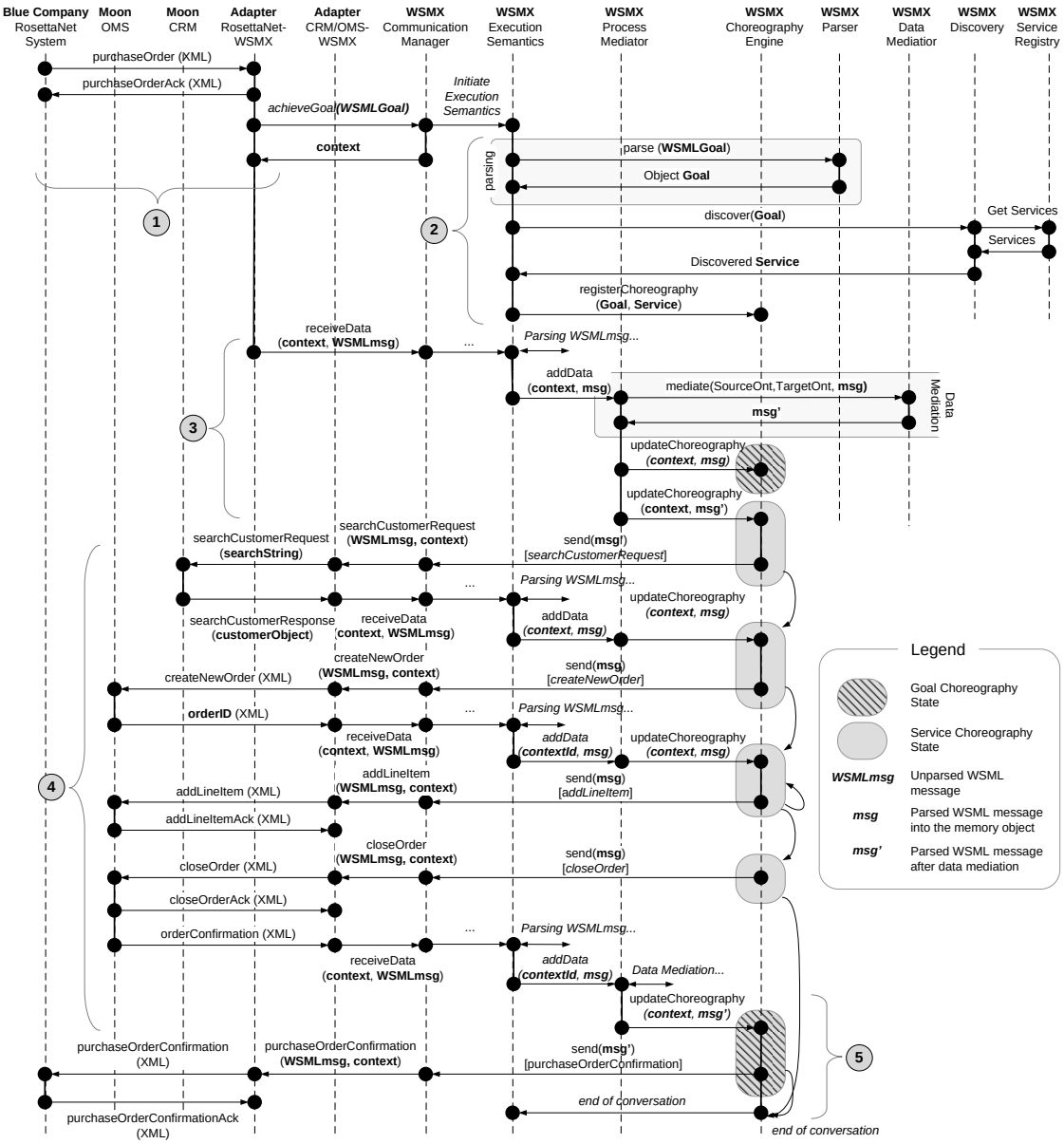

Figure 3.2: Activity Diagram

In our implementation, sending all acknowledgments from WSMX to the Customers RosettaNet and Moon systems and vice versa is implemented at the adapter level. In principle this would better be achieved using a choreography description executed by the WSMX environment, which could be used for e.g. handling exceptions as described in section 2.2.1. This is however a forced limitation of our implementation as there is currently no support for handling exceptions in choreography within conversations. This will be the subject of the future work of the WSMO and the WSMX working groups.

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

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

In the RosettaNet-WSMX adapter, the message captured in XML conforming to PIP3A4 XML Schema is lifted to WSML according to the PIP3A4 ontology and rules for lifting. A fragment of the lifting-rules in XSLT and the resulting WSML message are shown in listing 1. Finally, a WSMO Goal is created including the definition of the desired capability and a choreography. The capability of the requester (Customer company) is used during the discovery process whereas the Goal choreography describes how the requester wishes to interact with the environment. Since the RosettaNet system is, from the WSMX point of view, represented by an adapter (the adapter can be understood as a wrapper around an existing application – in our case Customer's RosettaNet system), the choreography here reflects the communication pattern of the adapter (hence it does not include interactions regarding acknowledgments of messages). After the goal is created, it is sent to WSMX through the *AchieveGoal* entrypoint. In return, a *context* is received containing the identification of the *conversation* as well as an identification of the *role* of the sender (i.e. *requester* or *provider*). This information is used in subsequent asynchronous calls

from the requester.

2 – Discovery and Conversation Setup.

The *AchieveGoal* entrypoint is implemented by the WSMX Communication Manager – the WSMX middleware service, which facilitates the inbound and outbound communication with the WSMX environment. After receipt of the goal, the Communication Manager initiates the *execution semantics* which manages the whole integration process. The Communication Manager sends the WSML goal to the instance of the execution semantics, which in turn invokes the WSMX Parser returning the Goal parsed into an internal system object.

The next step is to invoke the discovery middleware service in order to match the requested capability of the Goal with the capabilities of services registered in the WSMX repository. Services matching the requested capability are returned from to the execution semantics. A very simplified implementation of discovery is used for this use case since only one service in the repository (CRM/OMS service) can be matched with the goal, thus we do not deal with selection of services. This simplification is based on the fact that discovery is not of interest to us at this stage; we only concentrate on mediation and conversation aspects of the integration process.

Once a service is discovered, the execution semantics registers both the requester's as well as the provider's choreography with the Choreography Engine (these choreographies are part of the goal and service descriptions respectively). Both choreographies are set to a state where they wait for incoming messages that could fire a transition rule. This completes the conversation setup.

3 – Conversation with Requester.

The instance data for the goal is sent from the RosettaNet-WSMX adapter to the WSMX asynchronously by invoking the *receiveData* entrypoint. Along with the instance data, the context is also sent to WSMX in order to identify the sender and the conversation (the context has been previously obtained as a result of invocation of the *achieveGoal* entrypoint).

The data in WSML (WSMLmsg) is passed through the Communication Manager to the execution semantics which again first parses the data into internal system objects using the WSMX Parser. In general, multiple independent conversations can be running inside WSMX, thus information carried by the context is used to identify the specific conversation to which the message belongs. The execution semantics then passes obtained data to the WSMX Process Mediator.

The first task of the WSMX Process Mediator is to decide, which data will be added to

which choreography, i.e. requester's or provider's choreography[3]. This decision is based on analysis of both choreographies and concepts used by these choreographies and is in detail described in [MC05a]. In our scenario, Process Mediator first updates the memory of the requester's choreography with the information that the Purchase Order request has been received. The Process Mediator then evaluates how that data should be added to the memory of the provider's choreography. In the use case, data mediation must be first performed to the ontology used by the provider (service ontology). For this purpose, the source ontology of the requester, target ontology of the provider and the instance data are passed to the WSMX Data Mediator. Data mediation is performed by execution of mapping rules between both ontologies (these mapping rules are stored within WSMX and have been created and registered during the integration setup phase). More information on the WSMX Data Mediator can be found in [MC05b]. Once mediation is complete, the mediated data is added to the provider's choreography.

4 – Conversation with Provider (opening order, add line items, closing order).

Once the requester's and provider's choreographies have been updated, the Choreography Engine processes each to evaluate if any transition rules have been fired. In our scenario, the requester's choreography remains in the waiting state as no rule can be evaluated at this stage. For the provider's choreography, the Choreography Engine finds the rule shown in the listing 2 (lines 14-21). Here, the Choreography Engine matches the data in the memory with the the antecedent of the rule and performs the action of the rule's consequent (i.e. update/delete of the memory). The rule says that the message *SearchCustomerRequest* with data, *searchString*, should be sent to the service provider (this data has been previously added to the choreography memory after the mediation - here, *searchString* corresponds to the *customerId* from the requester's ontology). The Choreography Engine then waits for the *SearchCustomerResponse* message to be sent as a response from the provider. Sending the message to the service provider is carried out by Choreography Engine passing the message to the Communication Manager which, according to the grounding defined in the choreography, further passes the message to the *searchCustomer* entrypoint of the CRM/OMS-WSMX Adapter.

The listing 2 shows the fragment of the provider's choreography and selected rule described above. The choreography is described from the service point of view thus the rule says that the service expects to receive the *SearchCustomerRequest* message and send the reply *SearchCustomerResponse* message. In the *StateSignature* section (lines 3-11), concepts for the *input*, *output* and *controlled* vocabulary are defined. Input concepts corresponds to messages sent to the service, output concepts corresponds to messages

[3]Note that choreographies of WSMO services are modeled as Abstract State Machines [BS03] and are processed using standard algorithms during runtime. Each choreography has its own *memory* containing instance data of ontological concepts. A choreography rule whose antecedent matches available data in the memory is selected from the rule base and by execution of the rule's consequent, the memory is modified (data in the memory is updated, deleted of removed)

sent out of the service, and controlled concepts are used for controlling the states, and transition between states during processing of the choreography. Each concept used is prefixed with the namespace definition (e.g. moon, oasm) corresponding to the imported ontologies (lines 4, 5). The choreography is part of the service definition which in addition also contains definition of *non-functional properties* and *capability*. For brevity, these elements are not included in the listing.

```
1  ...
2  choreography MoonWSChoreography
3      stateSignature _"http :// www.example.org/ontologies/sws-challenge/MoonWS#statesignature"
4        importsOntology {_"http://www.example.org/ontologies/sws-challenge/Moon",
5                                          _" http :// www.example.org/ontologies/choreographyOnto" }
6
7        in moon#SearchCustomerRequest withGrounding { _"http://intranet.moon.local/wsmx/services/
             CRMOMSAdapter?WSDL#wsdl.interfaceMessageReference(CRMOMSAdapter/CRMsearch/in0)"}
8              ...
9        out moon#SearchCustomerResponse
10       ...
11       controlled oasm#ControlState
12
13 transitionRules _"http :// www.example.org/ontologies/sws-challenge/MoonWS#transitionRules"
14         forall {?controlstate, ?request} with (
15                         ?controlstate[oasm#value hasValue oasm#InitialState] memberOf oasm#ControlState
                               and
16                         ?request memberOf moon#SearchCustomerRequest
17                 ) do
18                         add(?controlstate[oasm#value hasValue moonc#SearchCustomer])
19                         delete(?controlstate[oasm#value hasValue oasm#InitialState])
20                         add(_# memberOf moon#SearchCustomerResponse)
21         endForall
22          ...
```

Listing 2: Requester's Service Choreography

In the adapter, lowering the WSML message to XML is performed using the lowering rules for the CRM/OMS ontology and the CRM XML Schema. After that, the actual service of the CRM system behind the adapter is invoked, passing the parameter of the *searchString*. The CRM system searches for the customer and returns back to the CRM/OMS Adapter a *customerObject* captured in XML. Analogously to sending data from the RosettaNet-WSMX adapter, the XML data is lifted to the CRM/OMS ontology, passed to the WSMX, parsed by the WSMX Parser and after the evaluation of the WSMX Process Mediator, the data is added to the provider's choreography memory.

Once the ontology of the provider's choreography is updated, the next rule is evaluated resulting in sending a *createNewOrder* message to the Moon OMS system. This process is analogous to sending the *searchCustomerRequest* described in the previous paragraph. As a result, the *orderID* sent out from the OMS system is again added to the memory of the provider's choreography.

After the order is created (opened) in the OMS system, the individual items to be ordered need to be added to that order. These items were previously sent in one message as part of order request from Blue's RosettaNet system (i.e. a collection of *ProductLineItem*) which must be now sent to the OMS system individually. As part of the data mediation in the step 3, the collection of items from the RosettaNet PO request have been split into

individual items which format is described by the provider's ontology. At that stage, the Process Mediator also added these items into the provider's choreography. The next rule to be evaluated now is the rule of sending *addLineItem* message with data of one *lineItem* from the choreography memory. Since there is more then one line item in the memory, this rule will be evaluated until all line items from the ontology have been sent to the OMS system.

The next rule to be evaluated in the provider's choreography is to close the order in the OMS system. The *closeOrder* message is sent out from the WSMX to the OMS system. Since no additional rules from the provider's choreography can be further evaluated, the choreography gets to the end of conversation state.

5 – Conversation with Requester (order confirmation, end of conversation).

After the order in OMS system is closed, the OMS system replies with *orderConfirmation* (apart from the acknowledgment which is stopped at the CRM/OMS adapter as previously noted). After lifting and parsing of the message, the Process Mediator first invokes the mediation of the data to the requester's ontology and then evaluates that the data needs to be added to the memory of the requester's choreography. At this stage, the next rule of the requester's choreography can be evaluated saying that *purchaseOrderConfirmation* message needs to be sent to the RosettaNet system.

After the message is sent, no additional rules can be evaluated from the requester's choreography, thus the choreography gets to the end of conversation state. Since both requester's and provider's choreography are in the state of end of conversation, the Choreography Engine notifies the execution semantics and the conversation is closed.

Chapter 4

Expected Benefits and Related Work

4.1 Benefits of SWS in B2B Integration

As WSML is a more expressive language than the schema languages used currently, the lifting of PIPs to ontologies can represent more information and help easier to manage changes in the messages. As a simple example, we provided the mapping of product information to ontologies that captured the natural language constraints and made the "GlobalProductIdentifier" RosettaNet meaning of GTIN number more explicit. The use of formal ontologies enables using common conversion functions to mediate some differences with logical dependencies. RosettaNet currently defines more than 300 *GlobalProductUnitOfMeasureCode*s as a list without any relations to each other. With help of using axioms, automatic transformations between e.g. "*25 Kilogram Bulk Bag*" and "*50 Pound Bag*" can be done. Currently matching all the details related to PIP messages takes time and only small differences can cause additional system development and testing work. SWS techniques can be applied to describe how companies use the PIPs and automate message compatibility matching, thus making the B2B integration process faster. The resulting integration is more flexible to slightly varying use of messages. For example, different measurement units can be supported easily if they are specified. Adding new partners should be a lot quicker, as a new partner using a domain ontology in describing his services would only need to register the needed details to WSMX before participating in the RFQ processes. Furthermore all this does not require changes to the back-end interfaces. As a result, organisation A would get more quotes to select from.

The benefits for the Moon company lie in automatically driven conversation between back-end systems and RosettaNet partner when all services are decoupled having heterogeneous description of interfaces and capabilities. While data mediation is performed in semi-automatic way partially done during design-time, process mediation is done automatically during runtime. This allows the requestors' and provider's services to be decoupled and automatically integrated during runtime as needed according to the particular integration scenario.

4.2 Related Work

There are a number of papers discussing the use of SWS to enhance current e-business frameworks. Some concentrate on ontologising e-business frameworks [FB05, AIJ06]. Foxvog and Bussler describe how EDI X12 can be presented using WSML, OWL and CycL ontology languages [FB05]. The paper focuses on the issues encountered when building a general purpose B2B ontology, but does not provide an architectural framework and implementation. Anicic et al. present how two XML Schema-based automotive B2B standards are lifted using XSLT to OWL-based ontology. They use a two-phase design and run-time approach similar to the one presented in this deliverable. The paper is based on different e-business frameworks and focus only on the lifting and lowering to the ontology level.

Other papers apply SWS technologies to B2B integrations [PCB+05, TPC03, PLS+05, JIG01]. Some concentrate on the use of SWS technologies to service discovery and selection. Preist et al. [PCB+05] presented a solution covering all phases of a B2B integration life-cycle, starting from discovering potential partners. The paper also addresses the lifting and lowering of RosettaNet XML messages to ontologies on a high level, but no mediation on the ontological level is provided. Trastour et al. [TPC03, TBP03] augment RosettaNet PIPs with partner-specific DAML+OIL constraints and use agent technologies to automatically propose modifications if partners use messages differently. However, the paper also focuses on the discovery and contracting phases. Paolucci et al. [PLS+05] describe an architecture by using OWL-S to extend SOA over organisational boundaries. Again, discovery is the main focus of the paper.

Many papers present B2B integration solutions that do not use any SWS technologies, but still have more similarities to our work. *Dogaz et al.* [DTP+02] present an implementation where an ebXML infrastructure is developed by exploiting the UDDI registries and RosettaNet PIPs. The UDDI registry is used to store ebXML documents and process descriptions that correspond to WSMX registries described here. They also provide a solution for secure communication. *Sundaram and Shim* [SS01] present an infrastructure for B2B exchanges with RosettaNet. They have a three-tier client-server prototype that allows customers to send PIP messages using a browser. *Sayal et al.* [SCDS02] present a tool, that supports RosettaNet PIPs and allows generating complete processes from PIPs by taking internal integration needs into account. These approaches are more static and lack both the mediation capabilities enabled by SWS and secure communication. Common to all these related works is a lack of the concept for integrating with existing back-end systems.

A common property of previous work using SWS seems to be that the solutions are still very conceptual and evaluations of the solutions in terms of performance or usage is still missing. The implementation related to SWS challenge presented here has been peer-evaluated against the requirements of the SWS Challenge with the results available on the challenge website. [1]

[1] http://sws-challenge.org/wiki/index.php/Workshop_Budva

Chapter 5

Conclusions and Future Work

This is the second version of the deliverable describing invocation and interoperation of web services. We discussed in particular the problem of the interoperation taking into account different e-business frameworks used by different parties involved in inter-enterprise integration settings. With this respect, we presented a inter-enterprise integration scenario, where a buyer organization communicates with multiple partners, which all behave a bit differently. The partners support different e-business frameworks for this communication. In the scenario, requests for quotes (RFQ) and purchase order (PO) processes are used. We showed how the interoperation aspects are handled from the buyer organization point of view, where the use of Semantic Web Service (SWS) technologies enables communication with all its business partners. We demonstrated parts of the ontologising process of existing messages and how they can be used to mediate the differences in B2B integrations and how the functionality of a RosettaNet adapter. Furthermore, in this deliverable we presented a primarily intra-enterprise B2B scenario implemented using Semantic Web Services on WSMX addressing in particular the challenges of conversations between heterogeneous systems requiring data and process mediation. This was based on SWS Challenge scenario presenting the suppliers point of view of RosettaNet PO exchange.

This work addresses the fact that although research into the area of Semantic Web Services is well established, there is a scarcity of implemented use cases demonstrating the potential benefits. Showcasing realistic scenarios and their evaluation is an essential step for transfer of this emerging technology to the industry. In order to enable this transfer, it is essential to show how semantic technologies can co-exist within existing enterprise infrastructures with existing integration standards such as RosettaNet.

Future work is planned to expand our approach to cover more e-business frameworks and to integrate key enterprise infrastructure systems such as policy management, service quality assurance, etc. We also plan to layer our solution on top of an existing integration platform, such as SAP NetWeaver. This work is in the context of our ongoing work tackling the increasingly complex scenario stages defined by the SWS Challenge including discovery and composition.

Bibliography

[AIJ06] Nenad Anicic, Nenad Ivezic, and Albert Jones. An Architecture for Semantic Enterprise Application Integration Standards. In *Interoperability of Enterprise Software and Applications*, pages 25–34. Springer, 2006.

[BS03] Emil Brger and Rudi Strk. *Abstract State Machines: A Method for High-Level System Design and Analysis*. Springer-Verlag, 2003.

[CCMW01] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web service description language (wsdl) 1.1. Note, W3C, March 2001.

[dBLPF06] Jos de Bruijn, Holger Lausen, Axel Polleres, and Dieter Fensel. The web service modeling language: An overview. In *Proc. of the European Semantic Web Conference*, 2006.

[DTP+02] Asuman Dogac, Yusuf Tambag, Pinar Pembecioglu, Sait Pektas, Gokce Laleci, Gokhan Kurt, Serkan Toprak, and Yildiray Kabak. An ebXML infrastructure implementation through UDDI registries and RosettaNet PIPs. In *Proc. of the 2002 ACM SIGMOD International Conference on Management of Data*, pages 512–523, 2002.

[FB02] Dieter Fensel and Christoph Bussler. The web service modeling framework wsmf. *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.

[FB05] Doug Foxvog and Chris Bussler. Ontologizing edi: First steps and experiences. In *Proceedings of the International Workshop on Data Engineering Issues in E-Commerce*, 2005.

[GHM+03] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen. SOAP version 1.2 part 1: Messaging framework. Recommendation, W3C, June 2003.

[JIG01] Albert Jones, Nenad Ivezic, and Michael Grüninger. Toward self-integrating software applications for supply chain management. *Inf. Systems Frontiers*, 3(4):403–412, 2001.

[KVH+06] Paavo Kotinurmi, Tomas Vitvar, Armin Haller, Ray Boran, and Aidan Richardson. Semantic web services enabled b2b integration. In Juhnyoung Lee, Junho Shim, Sang goo Lee, Christoph Bussler, and Simon Shim, editors, *Proc. of the 2nd International Workshop on Data Engineering Issues in E-Commerce and Services*, LNCS 4055, pages 209–223. Springer, June 2006.

[M+04] David Martin et al. Owl-s: Semantic markup for web services. Member submission, W3C, 2004. Available from: `http://www.w3.org/Submission/OWL-S/`.

[MBB+03] Brahim Medjahed, Boualem Benatallah, Athman Bouguettaya, Anne H. H. Ngu, and Ahmed K. Elmagarmid. Business-to-business interactions: issues and enabling technologies. *VLDB Journal*, 12(1):59–85, 2003.

[MC05a] Adrian Mocan and Emilia Cimpian. D13.7 process mediation in wsmx. Wsmo working draft draft v0.1, DERI, 2005. Available at: `http://www.wsmo.org/TR/d13/d13.7/v0.1/`.

[MC05b] Adrian Mocan and Emilia Cimpian. Wsmx data mediation. Wsmo working draft draft v0.2, DERI, 2005. Available at: `http://www.wsmo.org/TR/d13/d13.3/v0.2/`.

[NK04] Juha-Miikka Nurmilaakso and Paavo Kotinurmi. A review of xml-based supply-chain integration. *Production Planning and Control*, 15(6):608–621, 2004.

[PCB+05] Chris Preist, Javier Esplugas Cuadrado, Steve Battle, Stuart Williams, and Stephan Grimm. Automated Business-to-Business Integration of a Logistics Supply Chain using Semantic Web Services Technology. In *Proc. of 4th Int. Semantic Web Conference*, 2005.

[PLS+05] Massimo Paolucci, Xiong Liu, Naveen Srinivasan, Katia P. Sycara, and Paul A. Kogut. Discovery of information sources across organizational boundaries. In *Proc. of the Int. Conference on Services Computing*, pages 95 – 102. IEEE Computer Society, 2005.

[POSV04] A. Patil, S. Oundhakar, A. Sheth, and K. Verma. Semantic Web Services: Meteor-S Web Service Annotation Framework. In *13th International Conference on World Wide Web*, pages 553–562, 2004.

[RLK04] D. Roman, H. Lausen, and U. Keller. Web Service Modeling Ontology (WSMO). WSMO Working Draft, 2004.

[SCDS02] Mehmet Sayal, Fabio Casati, Umeshwar Dayal, and Ming-Chien Shan. Integrating Workflow Management Systems with Business-to-Business Interaction Standard. In *Proc. of the 18th International Conference on Data Engineering*, pages 287–296. IEEE Computer Society, 2002.

[SS01] Meera Sundaram and Simon S. Y. Shim. Infrastructure for B2B Exchanges with RosettaNet. In *Proc. of the Third International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems*, pages 110–119, 2001.

[TBP03] David Trastour, Claudio Bartolini, and Chris Preist. Semantic Web support for the business-to-business e-commerce pre-contractual lifecycle. *Computer Networks*, 42(5):661–673, 2003.

[TKS05] Juho Tikkala, Paavo Kotinurmi, and Timo Soininen. Implementing a RosettaNet Business-to-Business Integration Platform Using J2EE and Web Services. In *7th IEEE International Conference on E-Commerce Technology*, pages 553–558. IEEE Computer Society, 2005.

[TPC03] David Trastour, Chris Preist, and Derek Coleman. Using Semantic Web Technology to Enhance Current Business-to-Business Integration Approaches. In *Proc. of the Int. Enterprise Distributed Object Computing Conference*, pages 222–231, 2003.

[WSD05] Web service semantics - wsdl-s w3c member submission, 2005.

[ZMH05] Michal Zaremba, Matthew Moran, and Thomas Haselwanter. Wsmx architecture. available from http://www.wsmo.org/tr/d13/d13.4/v0.2/. WSMX Final Draft, 2005.