

## Project IST 026850 SUPER

Semantics Utilized for Process management within and between Enterprises

### Deliverable 1.3

Process Ontology Language and Operational Semantics for Semantic Business Processes

Leading Partner: NUIG

Contributing Partner: NUIG, OU, PUE, USTUTT

Security Classification: Public (PU)

May 07

Version 0.14

Project	SUPER	SUPER-Project-No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes		
Document	Deliverable 1.3	Date	13.05.07

## Project Details

IST Project Number	026850
Acronym	SUPER
Project Title	Semantics Utilised for Process management within and between EnteRprises
Project URL	<a href="http://www.ip-super.org">http://www.ip-super.org</a>
EU Project Officer	Werner Janusch

Authors (Partner)	Agata Filipowska (PUE), Armin Haller (NUIG), Monika Kaczmarek (PUE), Tammo van Lessen (USTUTT), Jörg Nitzsche (USTUTT), Barry Norton (OU)		
Deliverable Owner (Partner)	Armin Haller (NUIG)	E-mail	armin.haller@deri.org
		Phone	+353 91 495139

Project	SUPER	SUPER-Project-No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes		
Document	Deliverable 1.3	Date	13.05.07

## Versioning and Contribution History

Version	Description	Comments
0.01	Proposed structure, TOC, first initial ideas	AH
0.02	Section 4.5 added	JN
0.03	Structure changed, section 2 changed as provided by PUE section 3.1.1 added as provided by USTUTT; additional changes to both sections	AH,JN
0.04	Introduction added	BN
0.05	Rewrote Section 2, especially making changes to requirements, added diagram as overview to Section 3	BN
0.06	Added content to Section 4 and 5	JN,TL
0.07	Providing WSMX description in Section 3.2.1.1	MZ
0.08	Finalising Section 3	BN
0.09	WSMX description changed, changes accepted for final consolidation	AH
0.10	Dimka's comments partly addressed, comments inserted from the previous documents	AH
0.11	USTUTT, PUE, NUIG contributions covering requirements from phone conference included, citations changed to Endnote	AH
0.12	OU contributions covering requirements from phone conference included	BN
0.13	Earlier internal comments included and feedback from the final reviewing addressed	AH
0.14	New activity types listing corrected	JN,DK

Project	SUPER	SUPER-Project-No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes		
Document	Deliverable 1.3	Date	13.05.07

## Table of Contents

<b>Project Details</b>	<b>II</b>
<b>Versioning and Contribution History</b>	<b>III</b>
<b>Table of Contents</b>	<b>IV</b>
<b>Table of Figures</b>	<b>V</b>
<b>1 Executive Summary</b>	<b>1</b>
1.1 Deliverable Alignment	1
1.1.1 Modelling Stack Alignment	1
1.1.2 Use case Alignment	2
<b>2 Introduction</b>	<b>3</b>
2.1 Problem definition	3
2.2 Objectives	3
2.3 Document Structure	3
<b>3 Requirements for a Process Ontology Language</b>	<b>5</b>
3.1 Scenario	5
3.2 Requirements	5
<b>4 Preliminaries</b>	<b>7</b>
4.1 BPEL	7
4.2 WSMO	8
<b>5 BPEL for Semantic Web Services</b>	<b>9</b>
5.1 Extension of BPEL Data Handling	9
5.2 Extension of BPEL Partner Links	10
5.3 Extension of BPEL Interaction Activities	11
5.4 Extension of BPEL Business Partners	14
5.5 Relationship between the BPEL4SWS processes and their Ontological Representation	15
<b>6 Conclusions</b>	<b>16</b>
<b>7 References</b>	<b>17</b>

Project	<b>SUPER</b>	SUPER-Project-No	<b>026850</b>
	Process Ontology Language and Operational Semantics for Semantic Business Processes		
Document	Deliverable 1.3	Date	<b>13.05.07</b>

## Table of Figures

Figure 1: SUPER Modelling Stack

2

IP- Project / Programme	SUPER	Project - No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	13.05.07

# 1 Executive Summary

The major goal of this deliverable is to define a language extension to BPEL [1] used to execute business processes in the SUPER architecture. The language extension, BPEL4SWS, serves a two-fold purpose. First, it is a non-obtrusive extension, not violating the schema of BPEL. Second, it introduces elements required to enable goal based invocation of Semantic Web Services. This includes the handling of data on an ontological level and its transformation to the syntactic XML Schema level.

The intended audience of this deliverable within the SUPER project are especially all the partners involved in the implementation of the semantic Business Process Execution Engine (sBPELEE). However, all partners involved in the development of ontologies have to be aware of the constraints imposed by the expressivity of BPEL4SWS. Outside of the project, the extensions introduced can be regarded as a model reference, how to enable the inclusion of Semantic Web Services in the BPEL model.

Disclaimer: The SUPER Consortium is proprietary. There is no warranty for the accuracy of completeness of the information, text, graphics, links or other items contained within this material. The document represents the common view of the consortium and does not necessarily reflect the view of the individual partners.

## 1.1 Deliverable Alignment

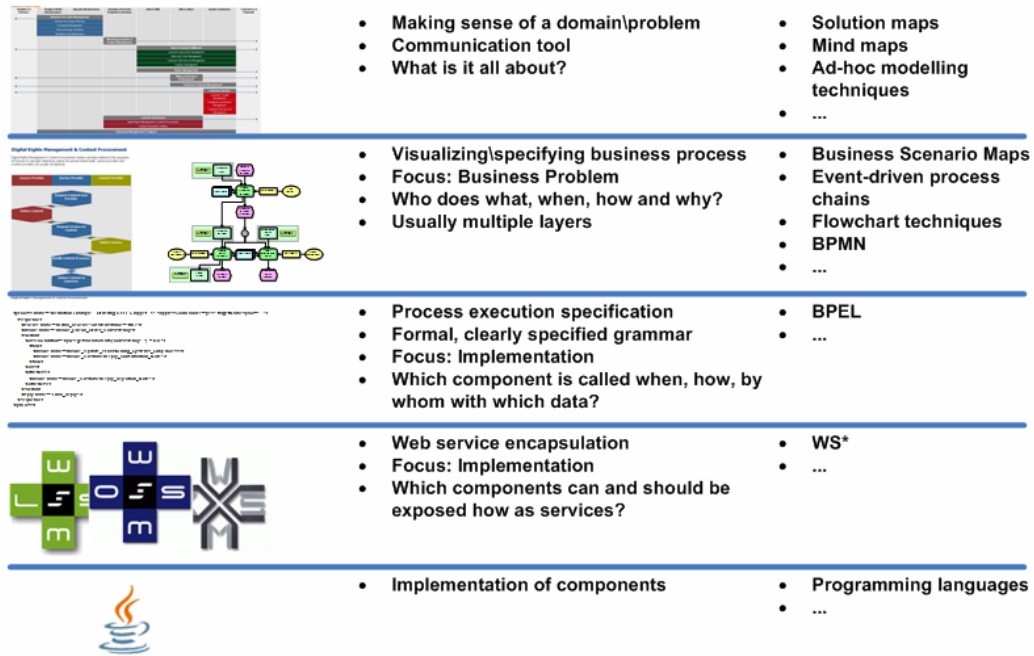
This deliverable is part of the overall research project SUPER. In this section we describe the position of the Process Ontology Language in the SUPER modelling stack and summarise the choice of BPEL as the underlying process model and its implications for the use cases. As this deliverable describes only a language specification, the alignment with the SUPER architecture is outlined in deliverable 6.2, where the SBPELEE engine, taking BPEL4SWS as its input language, is developed.

### 1.1.1 Modelling Stack Alignment

Processes in SUPER will be represented using a set of five ontologies i.e. Upper Process Ontology (UPO), Business Process Modelling Ontology (BPMO), semantic Business Process Modeling Notation (sBPMN), semantic Event Driven Process Chain sEPC and sBPEL. Figure 1 presents the SUPER ontology stack as described in D.1.1. Process Modelling Ontology and Mapping to WSMO.

The Process Ontology Language defined in this deliverable represents an XML serialization of the sBPEL ontology to be executed by the SBPELEE engine. A service requester does not model in BPEL4SWS directly, but in its abstracted ontological format. A bi-directional XML to WSML Transformation Service is utilized for executing mappings between the ontological level (BPMO, sBPEL) defined in WSML and the XML Schema level BPEL4SWS is defined in. As seen in Figure 1 BPEL4SWS is purely designed as an execution language with a formal grammar and operational semantics.

IP- Project / Programme	SUPER	Project - No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	13.05.07



**Figure 1: SUPER Modelling Stack**

### 1.1.2 Use case Alignment

BPEL is the de facto standard for describing Web Service Flows. It is widely accepted and is already supported by major software vendors in their products. Deliverable 11.1 strongly recommends the use of BPEL as the underlying process model. The Process Execution Engine, SBPELEE, is based on an existing open-source BPEL engine, Apache ODE. SBPELEE uses a BPEL4SWS model, an SA-WSDL document describing the service defined by the process model and a WSDL file containing the service descriptions to import and deploy the process model.

IP- Project / Programme	SUPER	Project - No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	13.05.07

## 2 Introduction

The basic idea of Semantic Business Process Management is to combine Semantic Web Service technologies and Business Process Management technologies. Both scientific domains have established standards that the SUPER project is building upon. This deliverable provides an intersection of what is needed for standards in both worlds to work together. It introduces BPEL4SWS, a conservative set of language extensions to BPEL [1] to allow the reference of Semantic Web Services within a business process description. BPEL4SWS represents an XML serialisation of its ontological counterpart, sBPEL. However, sBPEL is not only a translation of BPEL4SWS, but it extends it to a reasonable notion of a semantic business process. As such this ontology is not a restricted XML schema, but an open ontology where the partners involved and the messages exchanged in the process can be arbitrarily extended with annotations which can be used by a Reasoner to be interpreted and to derive new facts. However, BPEL4SWS includes reference points to the ontological description in sBPEL in order to relate the data exchanged to its ontological schema and to keep a semantic audit trail after execution.

### 2.1 Problem definition

The Business Process Execution Language (BPEL) is one of the most popular languages and de facto standard for modelling a business process. However, it only allows using hard-coded syntactical interfaces for partners and the process itself, i.e. semantic descriptions of services cannot be used within a process model. The lack of an ontological description of the process elements cause limitations in the way services are used within a process. A service providing the same functionality as the one referenced in the process model, but via a different syntactical interface, cannot be used instead.

### 2.2 Objectives

A business process model is used by business experts for different purposes. Thus, it is important to describe processes and its context clearly. In this deliverable, a generic language to be developed that describes different aspects of a business process. The major objective of this deliverable is to specify a Process Ontology Language and its operational semantics to represent processes, including activities for service and goal invocations composed in a particular order. These activities will be supported through the extension of some of the features of BPEL such as Data Handling, Interaction Activities, and Partner Links. The deliverable also describes the operational semantics of the language, which is inherited from BPEL.

### 2.3 Document Structure

This deliverable is structured as follows. Section 3 presents possible business process modelling choices driving the requirements on the process ontology language. Section 4 presents background technologies such as the languages BPEL4SWS builds upon and the SUPER ontology stack which

IP- Project / Programme	<b>SUPER</b>	Project - No	<b>026850</b>
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	<b>13.05.07</b>

defines some of the requirements for the process language. Section 5 describes the proposed extensions required to use Semantic Web Services within a BPEL process and section 6 concludes.

IP- Project / Programme	SUPER	Project - No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	13.05.07

### 3 Requirements for a Process Ontology Language

#### 3.1 Scenario

The business analyst models business processes using the SUPER Modelling Tool. The output of this modelling phase is the Business Process Modelling Ontology (BP MO). After completion of all appropriate steps (among others the composition of executable processes and the transformation from BP MO into sBPEL) the executable part of the model is deployed on the Semantic BPEL Execution Engine and ready for execution.

From an execution point of view such executable business processes may be abstracted to the following cases:

- a) processes in which SWS are stateless,
- b) processes in which SWS are stateful,
- c) processes that are solely based on SWS's,
- d) processes that mix invocation of SWS's and (ordinary) Web Services
- e) processes with a simple dataflow where output from one service is simply passed as the input to another service,
- f) processes with complex dataflow where the control flow is constrained by the content of a message returned by a service,
- g) processes with a simple or complex dataflow where the output from one service is passed to a mediator,
- h) processes that do not need compensation activities,
- i) processes that do not require any special conditions for SWS being used during execution (except those expressed in WSMO goals),
- j) processes in which two (or more) subsequent SWS's have to be provided by the same (but not known during process modelling) entity (e.g. RetrieveFlightDatabase, BookFlight),
- k) none of the SWS involved in the process requires any special access rights,

#### 3.2 Requirements

The role of the process execution language (BPEL4SWS) is to describe the modelled process in such a way that the description can be utilized by the SWS execution framework without resorting to descriptions of the process (or its constituents) in other formalisms. Thus, all the information required to invoke SWS is contained in the process execution language. The particular requirements for the language are given below.

IP- Project / Programme	SUPER	Project - No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	13.05.07

**Note:**

The key words "MUST", "REQUIRED", "SHALL", "SHOULD", "RECOMMENDED", "MAY", and "OPTIONAL" in the remainder of this document are to be interpreted in a manner similar to that described in IETF RFC 2119 [2].

MUST, REQUIRED, SHALL - absolute requirements. The language must address such requirements.

SHOULD, RECOMMENDED - there may exist valid reasons to ignore such requirements, but the implications of doing so must be understood and weighed before doing so.

MAY, OPTIONAL - the requirement is truly optional. The language may choose to omit the requirement for the sake of scope.

1. The language MUST be a conservative extension of BPEL in terms of its XML Schema:
  - be backwards compatible with BPEL,
  - and not violate any XML schema constraints defined in BPEL.
2. The language MUST allow referencing to SWS concepts (in particular related to goal-oriented execution and mediation).
3. The language MUST support loose coupling, i.e. no binding to concrete endpoints SHALL be required in the model.
4. The language MUST support invocation of stateless as well as statefull Semantic Web Services.
5. The language MUST allow imposing constraints on goals to be fulfilled by the same partner.
6. The language MUST allow for automatic instance creation - all the necessary information needed to create an instance must be carried in the description.
7. The language SHOULD support compensation activities for erroneous SWS invocations.
8. The language MUST have an unambiguous execution, i.e. well-defined operational semantics.

IP- Project / Programme	SUPER	Project - No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	13.05.07

## 4 Preliminaries

This section gives an introduction to the Meta Models and languages upon which the process ontology language, proposed in this document, is built. BPEL, the process specification language which represents the syntactical boundary and the operational semantics the process ontology language inherits. The Web Service Modeling Ontology as the Meta Model for Semantic Web Services and WSML, the ontology language sBPEL, as the ontological counterpart to BPEL4SWS, is defined in.

### 4.1 BPEL

The Business Process Execution Language for Web Services (WS-BPEL, BPEL4WS or BPEL for short) is the de facto standard for describing Web Service Flows. It enables orchestration of Web Services using their abstract interface definition, defined in the Web Service Description Language (WSDL) [3]. A process itself is also represented as a Web Service via a WSDL file. In BPEL there are two kinds of activities, basic activities and structured activities. The latter are used to define the control flow of the process. The most essential ones are: `<sequence>` enabling sequential execution of activities, `<flow>` enabling parallel execution, and `<while>` supporting loops.

Basic activities can be grouped into two categories: interaction activities [4] and others. The interaction activities (`<invoke>`, `<receive>`, `<reply>` and `<pick>`) enable communication with a partner Web Service referencing the `portType` and operation to be used. Additionally these activities reference a `<partnerLink>` which specifies the role the partner service and the process itself plays. The type of the `<partnerLink>` is defined as a WSDL extension, the so called `<partnerLinkType>` that specifies one or two roles and the `<portType>` each role has to implement. All of these interaction activities have at least one variable, input and/or output variable depending on their usage.

For a BPEL process multiple scopes can be defined using the `<scope>` activity. The root scope is the process itself. In such a `<scope>` variables and multiple kinds of handlers can be defined, e.g. `<faultHandler>` and `<compensationHandler>`. The `<faultHandler>` defines how to proceed when exceptions occur. The `<compensationHandler>` defines how to compensate individual or composite activities that already have been executed successfully in the case of an error.

In a BPEL process the dataflow is realized via access to globally shared data, i.e. data passing from one to another activity is achieved by accessing the variables defined in the surrounding scope. Note, that in BPEL there are no explicit constructs to define dataflow, i.e. dataflow is implicit. The `<assign>` activity can be used to copy data from one variable to another.

BPEL enables synchronous and asynchronous Web Service invocation. Synchronous invocation here means that the response message of the used Web Service is received by the invoking activity, i.e. the activity waits blocking for the response. In this case the `<invoke>` activity has one input and one output container. Asynchronous invocation means that an `<invoke>` activity does not take a response directly, but the process handles the response message in a `<receive>` activity later on. The

IP- Project / Programme	SUPER	Project - No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	13.05.07

<correlationSet> enables the correlation of messages to the corresponding process instance and the appropriate activity. The <correlationSet> is of utmost importance when using asynchronous invocation because by defining a <correlationSet> it is assured that the response message is associated to the right process instance.

## 4.2 WSMO

The Web Service Modelling Ontology (WSMO) [5] is a formal ontology for describing various aspects related to Semantic Web Services. WSMO defines four main modelling elements for describing several aspects of Semantic Web Services: ontologies, which provide the terminology used by other WSMO elements, Web service descriptions, which describe the functional and behavioural aspects of a Web service, goals that represent user desires, and mediators, which aim at automatically handling interoperability problems between different WSMO elements. Both, Web Service descriptions and Goals include an interface description consisting of a choreography and an orchestration. However, only the choreography is relevant in this deliverable, the BPEL4SWS supersedes the orchestration model proposed by WSMO. A choreography in WSMO describes the behaviour of the Web service from a client's point of view. The client may be either a human user or an agent acting on behalf of a user. Since WSMO allows more than one interface to be defined, multiple choreographies can be defined for a Web service and hence multiple models of interactions may exist for the same Web service.

The Web Service Modeling Language (WSML) [6] is an ontology including native support for Semantic Web Services modelled according to the Meta Model provided by WSMO. WSML offers a human readable syntax, as well as XML and RDF syntaxes for exchanging data between services. WSML clearly separates between conceptual and logical expression syntaxes. The conceptual syntax is used to distinctly model different conceptual aspects of WSMO such as Web Services, Ontologies, Goal and Mediators, whereas the logical expression syntax is used for describing additional constraints and axioms.

WSML consists of a number of variants based on different logical formalisms. The different variants of the WSML correspond to different levels of logical expressiveness and are both syntactically and semantically layered. These variants are WSML-Core, WSML-DL, WSML-Flight, WSML-Rule and WSML-Full. WSML-Core is the least expressive variant, but possesses the most preferable computational characteristics among the WSML variants. It is defined by the intersection of Description Logic and Horn Logic. The WSML-DL, WSML-Flight and WSML-Rule variants extend WSML-Core to provide increasing expressiveness in the direction of Description Logics and Logic Programming, whereas WSML-Full is the union of these two directions which makes it the most expressive WSML variant.

IP- Project / Programme	SUPER	Project - No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	13.05.07

## 5 BPEL for Semantic Web Services

This section introduces the Process Ontology Language (BPEL4SWS) and describes the extensions to traditional BPEL. All extensions are non-obtrusive to the XML Schema of BPEL. A valid BPEL4SWS document is also a valid BPEL document.

### 5.1 Extension of BPEL Data Handling

The exchange of messages is one of the cornerstones of collaborative business processes.

The `<variable>` element in BPEL4SWS defines a container within a BPEL `<scope>` to hold messages received from a partner or data defining state information in a process. Messages that are received are copied to these containers and data is read from these containers in order to send messages. The type of each variable in BPEL may be a WSDL message type, an XML Schema simple type or an XML Schema element. Since BPEL4SWS does not require a WSDL definition in the case of a Semantic Web Service invocation, it is not likely to use WSDL message types for the message exchange with a Semantic Web Service. Furthermore, WSDL 2.0 [7] dropped the notion of messages and it is recommended practise to reference a type schema, preferably XML Schema directly. When involved in semantic interactions and mediation a variable will necessarily be defined with an XML Schema type, since there an SA-WSDL [8] style of `modelReference` will be used to define the semantic representation for this data, and `liftingSchemaMapping` and `loweringSchemaMapping` attributes will reference services for lifting into a semantic form (for semantic invocation or mediation), and lowering from a semantic form (for syntactic invocation), respectively. Thus, the `<variable>` element in BPEL4SWS remains unchanged with the following syntax.

```
<variables>
  <variable name="ncname" messageType="qname"?
    type="qname"? element="qname"? />
</variables>
```

#### Listing 1: Variable element according to the BPEL specification [1]

The `<assign>` element is used to copy data from one variable to another. However, modelling the data transformation/manipulation using the assign activity does not consider semantics. Hence, the assign does not describe semantically how the data is transformed or copied.

Therefore we introduce the use of mediators applying the `<extensionAssignOperation>` defined in BPEL 2.0. The `<extensionAssignOperation>` in BPEL 2.0 allows one to include data manipulation elements which MUST belong to a namespace different from the BPEL namespace.

```
xmlns:b4s="http://ip-super.org/BPEL4SWS/"

<assign validate="yes|no"? standard-attributes>
```

IP- Project / Programme	SUPER	Project - No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	13.05.07

```

    standard-elements
    (
    <copy keepSrcElementName="yes|no"?>from-spec to-spec</copy>
    |
    <extensionAssignOperation>
      <s:mediate name="NCName"
        mediatorURI="anyURI"
        inputvariable="NCName"
        outputvariable="NCName"/>
    </extensionAssignOperation>
    )+
  </assign>

```

**Listing 2: Extension element to introduce semantic data manipulation**

## 5.2 Extension of BPEL Partner Links

To enable communication with other services or processes BPEL introduces the concept of a `partnerLinkType` which is an extension to WSDL. This `<partnerLinkType>` describes a contract between two partners in terms of roles and corresponding WSDL `portTypes` the partners have to provide. If only one partner has to provide one or multiple WSDL operations the `partnerLinkType` defines only one role. However, if both of the partners require WSDL operations of each other the `partnerLinkType` defines two roles. An example is given in the following listing.

```

<wsdl:definitions
  targetNamespace=...
  xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype">
  ...
  <plnk:partnerLinkType name="salesPLT">
    <plnk:role name="buyer"
      portType="buyerPT" />
    <plnk:role name="seller"
      portType="sellerPT" />
  </plnk:partnerLinkType>
</wsdl:definitions>

```

**Listing 3: Example partnerLinkType in BPEL**

This WSDL extension is used within a BPEL process definition in the context of a `partnerLink`. The `<partnerLink>` references this `partnerLinkType` and defines which role is taken by the process itself and which role is taken by the partner. Thus, the `partnerLink` somehow defines which goal the process wants to achieve making a conversation with a partner using this `partnerLink`. In the example given below (see Listing 4) the process takes the role “buyer” and the partner takes the role “seller”. That suggests that the process has the goal “buy something”.

```

<partnerLink name=""
  partnerLinkType=""
  myRole="buyer"

```

IP- Project / Programme	SUPER	Project - No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	13.05.07

```
partnerRole="seller"/>
```

#### Listing 4: Example partnerLink in BPEL

The interaction activities [8] (<receive>, <reply>, <invoke>, <pick>) define the communication directions of the message exchange in order to achieve the goal specified by the partnerLink by referencing the partnerLink and a WSDL operation. Thus, the interaction with partners heavily relies/depends on WSDL.

BPEL4SWS defines a new mechanism to describe the communication between two partners without dependency on WSDL. Therefore BPEL4SWS introduces a new element, the <b4s:conversation> element, which is not dependant on a <partnerLinkType> and as such is not dependant on WSDL. This element enables grouping of interaction activities and thus enables defining a complex message exchange between two partners.

Similarly to the <partnerLink> which is defined in the <partnerLinks> block, every <b4s:conversation> is defined within a <b4s:conversations> element. The syntax is as follows:

```
<b4s:conversations>
  <b4s:conversation name="NCName" WSMOGoal="anyURI"?/>+
</b4s:conversations>
```

#### Listing 5: Conversation element

If the conversation is an outgoing interface, i.e. the first interaction activity is sending a message to the partner, the conversation links to a WSMO Goal by means of the WSMOGoal attribute. The WSMO Goal describes what the process expects from a service that plays the partner role in this particular interaction by means of the requested capabilities. It also defines the requested message exchange pattern between the two partners by means of the choreography.

### 5.3 Extension of BPEL Interaction Activities

In the previous section the <conversation> element which is independent of WSDL was introduced. In addition, interaction activities that do not depend on WSDL are needed. Due to the fact that the partnerLink and operation attribute of the interaction activities defined in BPEL (<receive>, <reply>, <invoke> and <onMessage> within a <pick>) are mandatory, these activities do have a WSDL dependency. Consequently, BPEL4SWS introduces new interaction activities without a WSDL dependency. A set of interaction activities that form a message exchange with a partner are grouped using the <conversation> element.

IP- Project / Programme	SUPER	Project - No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	13.05.07

In BPEL 1.1 [1] the extensibility is limited, i.e. there is no way to introduce new activity types without violating the BPEL Schema and specification, and thus losing BPEL compliance. In order to eliminate this drawback BPEL 2.0 [9] introduces the `<extensionActivity>` that is the designated extension point for new activity types. BPEL4SWS introduces new activity types using the `<extensionActivity>` as specified in BPEL 2.0. The new types of activities are presented in the following listing:

```

xmlns:b4s="http://ip-super.org/BPEL4SWS/"

<extensionActivity>
  <b4s:receive variable="NCName"
    conversation="NCName"
    standard-attributes>
    standard-elements
  </b4s:receive>
</extensionActivity>

<extensionActivity>
  <b4s:reply variable="NCName"
    conversation="NCName"
    standard-attributes>
    standard-elements
  </b4s:reply>
</extensionActivity>

<extensionActivity>
  <b4s:invoke inputVariable="NCName"
    outputVariable="NCName"?
    conversation="NCName"
    standard-attributes>
    standard-elements
  </b4s:invoke>
</extensionActivity>

```

#### **Listing 6: Extension activities of BPEL4SWS**

The receive activity receives data and stores it in the specified variable whereas the reply activity reads data from the specified variable and sends it back to the partner that invoked the process. The invoke activity can be used either for synchronous invocation of a partner or for asynchronous communication in conjunction with a following receive activity. In the case of a synchronous invocation the invoke activity reads data from the specified input variable, sends the data to the partner, receives data from the partner and stores it into the specified output variable. In the case of asynchronous communication the invoke activity does not receive any data.

With respect to the WSDL dependency the pick activity plays a special role since it does not depend on WSDL itself but encapsulates the `onMessage` element which does. Listing 7 shows the syntax of the pick activity specified in BPEL 2.0.

IP- Project / Programme	SUPER	Project - No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	13.05.07

```

<pick createInstance="yes|no"? standard-attributes>
  standard-elements
  <onMessage partnerLink="NCName"
    portType="QName"?
    operation="NCName"
    variable="BPELVariableName"?
    messageExchange="NCName"?>+
    <correlations>?
      <correlation set="NCName" initiate="yes|join|no"? />+
    </correlations>
    <fromParts>?
      <fromPart part="NCName" toVariable="BPELVariableName" />+
    </fromParts>
    activity
  </onMessage>
  <onAlarm>*
  (
    <for expressionLanguage="anyURI"?>duration-expr</for> |
    <until expressionLanguage="anyURI"?>deadline-expr</until>
  )
  activity
</onAlarm>
</pick>

```

#### Listing 7: conventional pick activity

Note that there has to be at least one onMessage which includes the mandatory attributes partnerLink and operation. This means that the standard pick would not allow modelling a pick with one or multiple WSDL independent onMessage elements.

For this reason BPEL4SWS introduces a new pick activity that allows for the conventional onMessage and onAlarm as well as a new WSDL independent onMessage element which references a conversation and thus implements a part of the message exchange pattern specified by a WSMO goal or Web Service. This is illustrated in the following Listing 8.

```

<b4s:pick createInstance="yes|no"? standard-attributes>
  standard-elements
  (<b4s:onMessage conversation="NCName"
    variable="BPELVariableName">
    activity
  </b4s:onMessage> |
  <onMessage partnerLink="NCName"
    portType="QName"?
    operation="NCName"
    variable="BPELVariableName"?
    messageExchange="NCName"?>*
    <correlations>?
      <correlation set="NCName" initiate="yes|join|no"? />+
    </correlations>
    <fromParts>?
      <fromPart part="NCName" toVariable="BPELVariableName" />+
    </fromParts>
  activity

```

IP- Project / Programme	SUPER	Project - No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	13.05.07

```

    </onMessage>)+
    <onAlarm>*
      (
        <for expressionLanguage="anyURI"?>duration-expr</for> |
        <until expressionLanguage="anyURI"?>deadline-expr</until>
      )
      activity
    </onAlarm>
  </b4s:pick>

```

### Listing 8: redefined pick activity in BPEL4SWS

The pick activity executes when one of the specified events happens. This might be a conventional WSDL dependant onMessage, a WSDL independent onMessage or a timer event.

Note that there has to be at least one WSDL-less onMessage or one conventional WSDL dependant onMessage. The same applies for the <b4s:pick> activity. If the <b4s:pick> activity is used to instantiate a process, all the event in the <b4s:pick> activity MUST either be <b4s:onMessage> events or <onMessage> events.

Using this WSDL independent interaction activities BPEL4SWS enables expressing complex message exchange patterns with a partner service and does not restrict the communication with a partner to the message exchange patterns which are supported by one single interaction activity. Thereby BPEL4SWS enables the interaction with stateful Semantic Web Services. The invocation of stateless Semantic Web Service can still be modelled by simply having a conversation with one single two-way <b4s:invoke>, i.e. an <b4s:invoke> with one input and one output message.

## 5.4 Extension of BPEL Business Partners

BPEL 1.1 includes a <partner> element to express what capabilities a specific partner has to offer. A partner in BPEL 1.1 is defined as a subset of the partner links defined within the process.

BPEL4SWS introduces a new partner element that enables grouping of the WSDL independent conversations. Thus it can be defined that several conversations have to take place with one business partner. The new <b4s:partner> element, which is referring to a <b4s:conversation> instead of a <partnerLink> is illustrated in Listing 9.

```

<b4s:partners>?
  <b4s:partner name="NCName">+
    <b4s:conversation name="NCName">+
  </b4s:partner>
</b4s:partners>

```

### Listing 9: Partner element extension

This gives us a way to impose constraints on a partner to fulfil multiple goals. A common scenario for example is a case where the first goal request is to check the price for a specific service (e.g. book a

IP- Project / Programme	SUPER	Project - No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	13.05.07

flight) and the next goal is to utilize this service. In this case it is required that both services that are invoked are services of the same partner because it would make no sense to check the price e.g. at Lufthansa and then book a flight at British Airways.

## 5.5 Relationship between the BPEL4SWS processes and their Ontological Representation

BPEL4SWS is coupled with its ontological representation which is called sBPEL. This ontological representation can be used for reasoning across different process models. To enable reasoning on instance data, BPEL4SWS has to preserve the relationship to its ontological representation in order to enable a semantically enriched execution history. In order to relate the semantics pertaining to one element in the BPEL4SWS description we introduce an additional attribute which identifies the corresponding ontological instance in the sBPEL process model.

Since the identification of the corresponding ontological instance within an sBPEL process model is similar to referring XML messages to their ontological representation we use the `modelReference` attribute defined in the SA-WSDL [8] specification.

According to the BPEL specification, this `modelReference` attribute can be used for extending any element of a process model. However, we recommend using it only for elements that are related to events that occur during the execution of a process model and that are relevant to the execution history, especially the following BPEL4SWS elements:

- `<process>`
- basic and structured activities
- `<source>` element that encapsulates the transition condition for a link
- `<variable>`
- `<partnerLink>`
- `<correlationSet>`

The following example illustrates the usage of the semantic annotation of the extended `<b4s:invoke>` activity.

```
xmlns:sa=" http://www.w3.org/2002/ws/sawsdl/spec/sawsdl#"
<extensionActivity>
  <b4s:invoke inputVariable="NCName"
    outputVariable="NCName"?
    conversation="NCName"
    sa:modelReference="anyURI"?
    standard-attributes>
    standard-elements
  </b4s:invoke>
</extensionActivity>
```

### Listing 10: Semantic Annotations element in BPEL4SWS

IP- Project / Programme	SUPER	Project - No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	13.05.07

## 6 Conclusions

In this deliverable we have defined the Process Ontology Language BPEL4SWS, a conservative extension to BPEL. The language inherits the operational semantics of BPEL, which are commonly agreed upon and their formal semantics are given in different formalisms [10] [11] [12]. The language introduces a number of extensions to cater the inclusion of Semantic Web Services within a BPEL process model.

For data manipulation on the ontological level BPEL4SWS reuses the `<extensionAssignOperation>` defined in BPEL 2.0 [9] and the SA-WSDL [8] style of `modelReference` to define the semantic representation for this data, and `liftingSchemaMapping` and `loweringSchemaMapping` to reference services for lifting and lowering into a semantic form.

The Process Ontology Language introduces new activity types using the `<extensionActivity>` as specified in BPEL 2.0 to relax the definition of interaction activities (`<receive>`, `<reply>`, `<invoke>` and `<pick>`) in order to omit explicit specification of the `<portType>` and `<operation>` in their definition.

Further, BPEL4SWS defines a new element, `<b4s:conversation>`, to describe the communication between two partners without dependency on WSDL. The new `<b4s:partner>` element, which is referring to a `<b4s:conversation>` instead of a `<partnerLink>`, enables the grouping of the WSDL independent conversations.

BPEL4SWS is tightly coupled with its ontological representation, sBPEL. BPEL4SWS introduces an additional attribute, the `modelReference` attribute as defined in the SA-WSDL [8] specification, which identifies the corresponding ontological instance in the sBPEL process model.

IP- Project / Programme	SUPER	Project - No	026850
	Process Ontology Language and Operational Semantics for Semantic Business Processes	Work Package 1	
Document	Deliverable 1.3	Date	13.05.07

## 7 References

- [1] T. Andrews, F. Curbera, H. Dholakia, Y. Golland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana, "Business Process Execution Language for Web Services Version 1.1," OASIS 2003.
- [2] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," Harvard University RFC 2119, 1997.
- [3] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1.," W3C, Note 2001.
- [4] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. Ferguson, *Web Services Platform Architecture*: Prentice Hall, 2005.
- [5] D. Roman, U. Keller, L. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel, "Web Service Modeling Ontology," *Applied Ontologies*, vol. 1, pp. 77 - 106, 2005.
- [6] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel, "The web service modelling language WSML," DERI, WSML Final Draft D16.1v0.2 2006.
- [7] R. Chinnici, J. J. Moreau, A. Ryman, and S. Weerawarana, "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language," W3C, Working Draft 2007.
- [8] J. Farrell and H. Lausen, "Semantic Annotations for WSDL and XML Schema," W3C, Working Draft 2006.
- [9] D. Jordan, J. Evdemon, A. Alves, A. Arkin, S. Askary, B. Bloch, F. Curbera, M. Ford, A. Guízar, Y. Golland, N. Kartha, R. Khalaf, D. König, M. Marin, V. Mheta, D. van der Rijn, C. Kevin Liu, S. Thatte, P. Yendluri, and A. Yiu, "Web Services Business Process Execution Language Version 2.0.," OASIS 2006.
- [10] C. Ouyang, W. M. P. van der Aalst, S. Breutel, M. Dumas, A. H. M. ter Hofstede, and H. M. W. Verbeek, "Formal Semantics and Analysis of Control Flow in WS-BPEL," BPMcenter.org BPM-05-13, 2005.
- [11] D. Fahland and W. Reisig, "ASM-based semantics for BPEL: The negative Control Flow," presented at 12th International Workshop on Abstract State Machines, Paris, France, 2005.
- [12] R. Lucchia and M. Mazzara, "A pi-calculus based semantics for WS-BPEL," *The Journal of logic and algebraic programming*, vol. 70, pp. 96-118, 2007.