# Diplomarbeit

# Transformation von Geschäftsprozessen auf SOA Plattformen mit Hilfe Semantischer Technologien


## Christian Stamber


## Datum: 03/2008


**Fachbereich Informatik, Technische Universität Kaiserslautern**

# Transformation of Business Processes onto SOA Platforms by Means of Semantic Technologies

Diplomarbeit

von

Christian Stamber

29.03.2008

Betreuung: Prof. Dr. H. Dieter Rombach
Dipl.-Inform. Sebastian Adam, Fraunhofer IESE
M.Sc. SW Eng., Dipl.-Wirt.Inf.(FH) Sebastian Stein, IDS Scheer AG

# Erklärung

Hiermit erkläre ich, Christian Stamber, dass ich die vorliegende Diplomarbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe.

.....

Christian Stamber,

Kaiserslautern, den 29.03.2008

# Abstract

Business processes are very important for nowadays enterprises. The centralization of business processes requires a supporting IT infrastructure often arranged as Service Oriented Architecture. Business analysts are modelling processes on a very high functional level. These functional process models have to be transferred onto the underlying IT infrastructure, which is arousing problems caused by very time consuming implementation work accomplished by IT experts and inconsistencies between model and implementation. The initial cause for the mentioned problem can be broken down to a communication problem between business analysts and IT experts which is often referred to as the Business-IT Divide.

Semantic technologies aim at eliminating this gap by providing a common communication basis through ontologies. The present work examines the applicability of semantics in context of Business Process Management by developing a new method for business process modelling. It further focuses on the automatic transformation of business process models into the implementation with help of semantic descriptions. The method is validated concerning its usability by an empirical study.

**Keywords:** BPM, Business Process, EPC, ARIS, ARIS SOA Architect, EPC2BPEL, SOA, Web Service, WSDL, BPEL, Business-IT Divide, Semantic, Ontology, Semantic Web Service, Goal, WSMO, WSML, WSMX, SBPM, Semantic Business Process, SUPER, BPMO, sBPEL, BPEL4SWS, SISi, Semantic Invocation Service

# Preface

## About the Thesis


The diploma thesis at hand was conducted externally under supervision of Fraunhofer IESE in Kaiserslautern, Germany and IDS Scheer AG in Saarbrücken, Germany. Mentoring professor was Prof. Dieter Rombach of AG Software Engineering at TU Kaiserslautern.

The thesis was initially announced by IDS Scheer AG and takes place within scope of European SUPER[1] research project, which will be discussed in chapter 3.2.3.

## Fraunhofer IESE


The Fraunhofer "Institut Experimentelles Software Engineering"[2] was founded in 1996 in Kaiserslautern, Germany. The institute is lead by Prof. Dieter Rombach and Prof. Peter Liggesmeyer and employs about 180 researchers.

Fraunhofer IESE is today one of the worldwide leading research institutes in the area of software and systems development.

## IDS Scheer AG


IDS Scheer AG[3] is a software and consulting company that was founded in 1984 by Prof. August Wilhelm Scheer as a spin-off of "Institut für Wirtschaftsinformatik" of Saarland University, Germany.

Today, IDS Scheer is the market leader in Business Process Management software, serving more than 7,000 customers in over 70 countries in the world. IDS Scheer is employing more than 3,000 people.

---

1 http://www.ip-super.org/

2 http://www.iese.fraunhofer.de/

3 http://www.ids-scheer.com/

## Publications

During the accomplishment of the diploma thesis at hand, some ideas of it could already be published:

- As a chapter of the book *"Semantic Service Provisioning"*[4], which will be released at Springer in April 2008 (see [BCD+08]).

- As a conference paper for 11th Business Information Systems[5], which will take place in 2008 between 5th and 7th May in Innsbruck (Austria). Name of the contribution will be *"Prototypical Implementation of a Pragmatic Approach to Semantic Web Service Discovery During Process Execution"* (refer to [SSK08]).


## Acknowledgment

First of all, I would like to thank Prof. Dr. H. Dieter Rombach for giving me the possibility to accomplish my diploma thesis in cooperation with Fraunhofer IESE and IDS Scheer AG.

Then I would like to express my appreciation to Sebastian Adam and Sebastian Stein for their excellent support and guidance throughout my thesis.

Furthermore, I thank all participants of the conducted case study, who helped me to empirically fortify my work.

I also want to thank Albert Stamber and Patrick Schneider for proofreading as well as Tamara Ernst for her valuable linguistic advices.

Last but not least, I would like to say thank you to my family and friends for being so patient and supportive throughout my studies.


4 Homepage of the book "Semantic Service Provisioning": http://kuropka.net/ssp-book/

5 11th BIS conference: http://bis.kie.ae.poznan.pl/11th_bis/

# Table of Contents

# 1 Introduction

The following chapter introduces the reader in the field addressed by the present thesis. Therefore, it motivates the reader by highlighting the special importance of business processes and problems in their application.

Then, the general goals of the thesis are summarized on a very abstract level.

In the following section the showcase process is presented, which accompanies the reader through the whole thesis.

The chapter is closed by an overview of this thesis.

## 1.1 Motivation

The competitiveness of an enterprise is determined by three criteria: how fast can it offer a product, how good the product is, and how cheap it can offer the product [Car04]. Only enterprises fulfilling these goals will succeed (faster, better, cheaper).

Business processes are nowadays in the centre of attention [Car04] and the key asset in every enterprise. Business processes determine the procedure of creation of the product of an enterprise. They are therefore directly affecting the mentioned three criteria and hence the success of the company.

To keep the products of an enterprise fast on market, of good quality and with a competitive price, the enterprise has to deal with changing market conditions, like customer demands and competition, which define these values (what is fast, good and cheap). Only if the processes can permanently be adapted to these changing requirements, the enterprise can stay competitive.

Process management must therefore ensure the efficiency and flexibility of business processes. Flexibility and efficiency is the key for the survival of an enterprise.

Information technology becomes more and more important to enterprises. Although IT is in no way a guarantor for any competitive advantage [Car04], having no IT will surely cause business failure. The special importance of business processes and their central role in business management also leads to a change in architecture of supporting software systems, in which processes are at the centre of attention [Sch96]. A wide spread paradigm

and according to Leymann [Ley04] very important to development of future software systems is Service Oriented Architecture (SOA). A SOA is most commonly implemented by means of Web Services and process orchestration languages like BPEL (Business Process Execution Language). Therefore, this thesis analyses current methodologies involving those technologies.

Unfortunately, the mentioned IT infrastructure cannot cope directly with the business process models created by business analysts during discovery and design phases of business process life cycle. It is therefore necessary that the modelled business processes are being transformed into executable implementations. This transformation into executable software is accomplished by software engineers. Hereby, the eminent difficulty lies in the so called Business-IT Divide [SF03, DvdA04, KHSW05], which separates the business world from the IT world, and complicates the process implementation. This divide is due to the very different foci and knowledge backgrounds of business analysts and IT experts on the one hand and different notations on the other.

The transformed executable process is now executed and analysed by business analysts, which subsequently make changes to the initial process model if it does not fulfil the requirements. Unfortunately, business analysts cannot modify directly the executable process as its representation is too technical for them to understand. Hence, the initial model has to be modified and consequently also the implementation.

This procedure is very time intense, costly and error prone [SF03, DvdA04, KHSW05] as human intervention is needed in the transformation from business model into executable process. Due to its complexity, the resulting software cannot be verified by business analysts anymore and it is not possible to ensure that the initial business intent is actually implemented. A loss of consistency is often inescapable. The procedure reminds of some kind of “Chinese Whisper”, in which the starting objective is tampered in each station and a completely different message is received [SF03]. The core cause for the scenario can be reduced to the general communication problem of the Business-IT Divide.

In recent years increased endeavours in research of semantic technologies in context of business applications were made. Core idea is the usage of ontologies, which shall provide a common communication base and therefore eliminate the cause of the “Chinese Whisper” scenario.

## 1.2 Goals of the Thesis

The present diploma thesis is accomplished to analyze the applicability of semantic technologies in field of Business Process Management (BPM). Therefore, the current situation in BPM is examined and common procedures and technologies are highlighted. Furthermore, current research aspects regarding semantic technologies are summarized as well as their concrete visions about their application.

The main part of the thesis is the development of a new method, which extends the present process modelling method by enabling the annotation of semantic information.

The developed method is validated through empirical evaluation and compared to the state of the art approach.

## 1.3 Showcase

The work of this thesis was aligned on a real-life industrial process. This process was developed by Telekomunikacja Polska[6] (TP), the leading telecommunication provider in Poland and a research partner of SUPER.

Beside the process model, there were also several dummy Web Service implementations provided, which were modified and extended during the thesis.

The conducted empirical study was based on this business process, as well.

### 1.3.1 Telekomunikacja Polska (TP)

Telekomunikacja Polska (TP) is the dominant player in Polish telecommunications market. Headquarter of the company is Warsaw, Poland. TP was founded 1991 and had in beginning of 2007 about 10.6 million fixed-line subscribers including about 1.8 million broadband Internet access subscribers and about 12 million mobile customers.

In 2005, TP earned revenues of about PLN 18.3 billion (about EUR 5.17 billion) with EBITDA at about PLN 8 billion (about EUR 2.26 billion) and employed about 28.000 people.

6 http://www.tp.pl/

### 1.3.2 VoIP Activation Process

In the following section the chosen business process is explained in detail and its single steps are elaborated. Further, an overview of the different involved IT systems is shown.

The business process describes the business case if a TP customer wants to activate Voice over IP (VoIP) service. This service enables telephony via Internet protocol instead through conventional telephone protocols.

The process consists of the following steps:

1. Customer makes a request for VoIP telephony service on the TP web portal.

2. The portal system checks if customer exists in customer database. If not, he is informed about the situation and the order process comes to a premature end.

3. The system checks, whether the customer fulfils all requirements for the desired service. This check includes technical requirements and formal requirements. The customer must for instance have an appropriate DSL connection (technical) and must not have open bills to pay (formal). If the customer does not fulfil all requirements, he is informed, which steps he has to do to get VoIP and the order process terminates.

4. After successful verification, a new order is created in the system and detailed information about the desired service, conditions, and pricing are presented to the customer.

5. The customer confirms the order or rejects it. In the latter case the order process ends.

6. After confirming the order, the contract for the customer order is prepared and printed.

7. At the same time, the system checks whether the customer has already the necessary hardware (live box) to use VoIP. If not, the necessary hardware is prepared for shipping.

8. If the contract and the optional hardware are ready, it is shipped by a courier service to the customer. He signs the receipt of the hardware and the contract. The contract is shipped back to TP.

9. When the signed contract arrives, it is archived.

10. The billing system is informed and the VoIP service is activated. Now, the order process is successfully completed.

There are five software systems involved in the order process:

1. The web portal, which serves as a front end to customer and initializes the order process.

2. The Customer Relationship Management System (CRM), which takes care about customer specific issues like managing customer data.

3. The Order Management System (OMS), which handles all customer orders.

4. The Hardware Management System (HWM), which handles all hardware specific topics.

5. The extern Courier Service, which takes care of exchanging necessary hardware and documents.

Figure 1 illustrates the VoIP ordering scenario with the involved roles and software systems.

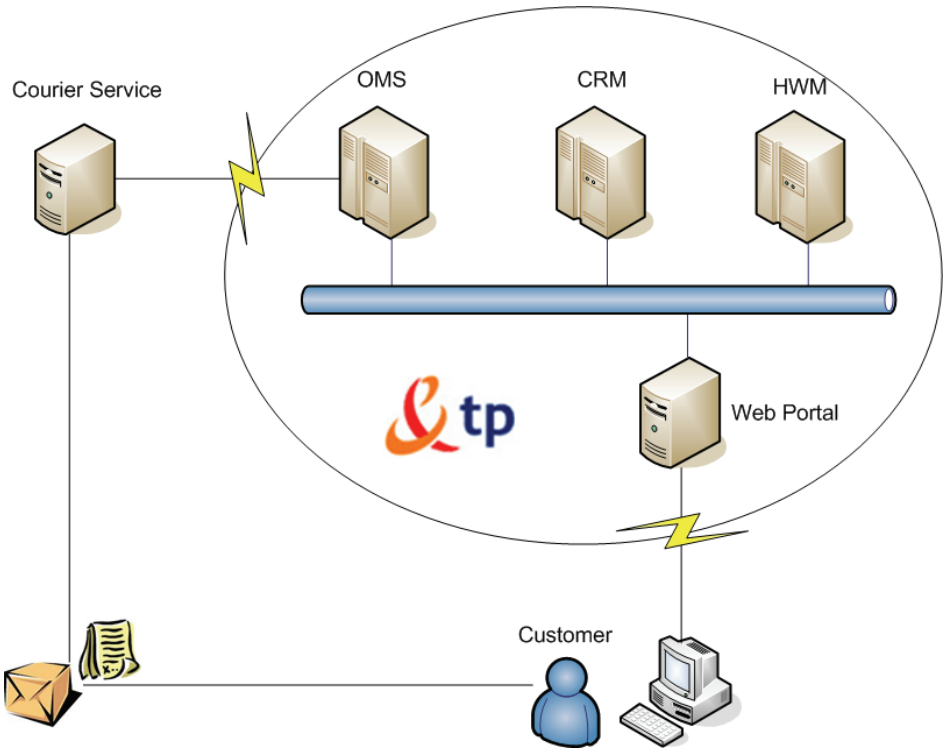



Figure 1 VoIP Activation Scenario

## 1.4 Structure of Thesis

The subsequent part of the present diploma thesis consists of five chapters.

Chapter 2 serves as introduction for the topic of the work. It hereby overviews the state of the art in Business Process Modelling and points out commonly used methods and technologies. It further summarizes shortcomings and problems in those approaches.

Chapter 3 summarizes the current research towards semantic technologies and shows the general idea of eliminating the mentioned shortcomings.

Chapter 4 forms the main part of the thesis. It explains which of the identified problems shall be addressed and shows the idea for the concrete solution. After, this solution is discussed in detail. This includes the developed method as well as the necessary software implementations.

Chapter 5 reviews the developed method by means of an empirical evaluation. At first, this chapter elaborates the criteria for selecting the appropriate type of empirical research method. Then, it explicates the chosen study and the set up for conduction. The last part of the chapter summarizes and interprets the results of the study.

Chapter 6 finally concludes the results and experiences with respect to the conduction of the thesis.

# 2 State of the Art in BPM

The following chapter shows an overview of common modelling practises in industry that deal with management of business processes. Especially concerning the modelling of business processes are introduced and widespread technologies about implementation of processes are presented.

The first part of the chapter points out the conceptual aspects of Business Process Management while the second part shows the technical aspects.

## 2.1 Business Level

This section depicts conceptual considerations concerning business processes. This includes especially the modelling and representation of business processes.

### 2.1.1 Enterprise Model

Although processes are in the centre of each enterprise a complete business model requires more than just processes. According to General Systems Theory [vB76] an enterprise can be considered as an open system consisting of the system core and its elements and relationships between them as well as the system boundary, which separates the system to its environment.

The core of the system “enterprise” can be divided in structural and behavioural elements. The structure is determined by organizational elements like hierarchies, functional elements, and data elements. The behavioural aspects of the business are captured by business processes as a part of the value chain.

This environment outside the system boundary consists of customers, competitors, regulators, suppliers, and the market.

#### 2.1.1.1 ARIS Methodology

A common methodology to capture all necessary aspects of a business model is ARIS, the Architecture of Integrated Information Systems [Sch96]. Central framework of ARIS is the “House of Business Engineering” (ARIS HOBE, see figure 2), which expresses the five dimensions of the business model:

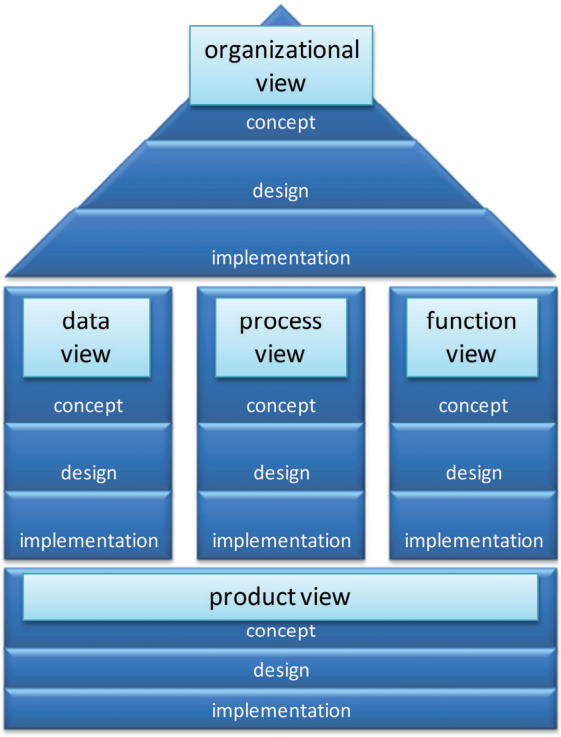



Figure 2 ARIS House of Business Engineering [Sch96]

- The *organizational view* describes the business by means of its organizational structure. In this view reside roles and relationships between those roles (for example hierarchies). Models in this view are for instance organizational charts.
- The *data view* represents relevant information objects and relationships between them. Models of this dimension can for instance be represented by ERM diagrams.
- The *function view* represents all relevant business functions and activities. These functions are arranged by means of function trees to represent their relationships.
- The *product view* describes all products of the enterprise. These products can be either material or immaterial.

Located in centre of ARIS HOBE is the process view, underlining its importance [KtHvdA03]:

- The *process view* contains the dynamic aspects of the enterprise. It links together and integrates the elements of the four static views by means of business processes. Processes in this view can be modelled by using Event-Driven Process Chains (see chapter 2.1.1.2).

Each of those five dimensions consists of the three layers “concept”, “design”, and “implementation”.

- The *concept layer* provides models that have a long term and pure functional meaning
- The models of *design layer* deal with realization of these concepts by software systems, without constituting to any specific technology.
- Finally, the *implementation layer* consists of concrete implementations of the design models by means of specific hard- and software.

The relationship of these three layers to the approach of Model Driven Architecture (MDA) is elaborated in detail in chapter 2.3.1.

#### 2.1.1.2 Event-Driven Process Chains

Event-Driven Process Chains [KNS92] (EPC) is a language for business process modelling, which was developed under supervision of Prof. August-Wilhelm Scheer at Saarland University, Germany in 1992. The EPC language is an accepted industry standard in business process modelling. EPCs are business oriented. They are in general not suitable for modelling of executable processes, because of missing implementation specific details like exception handling. An EPC is an ordered graph consisting of six types of elements:

- **Events** describe the state of the process. Events activate business functions. Each EPC starts and ends with at least one event.
- **Functions** symbolise a certain business activity, which leads to a change of state and therefore to a certain event. In general, a function is preceded by an event and followed by an event.
- **Relations** connect functions and events specifying the actual process.
- **Operators** are used to split or join the control flow. Different operators like AND, XOR, and OR are available. Operators, relations, events, and functions together form the control flow of the process model. The EPC language allows modelling of the most important workflow patterns [vdAtHKB03]. A detailed discussion of the workflow patterns supported by the EPC language can be found in [MNN05].
- **Organizational Units** can be assigned to a function to specify that the given function is executed by the organisational element. This is known as human task in other modelling languages.
- **Information Objects** can be assigned to a function to specify the input and output data of the given function. The information objects represent

parts of a logical (conceptual) enterprise data model. Typical elements of such an enterprise data model are customer, contract and invoice.

The graphical representation of the described element types is shown in figure 3.

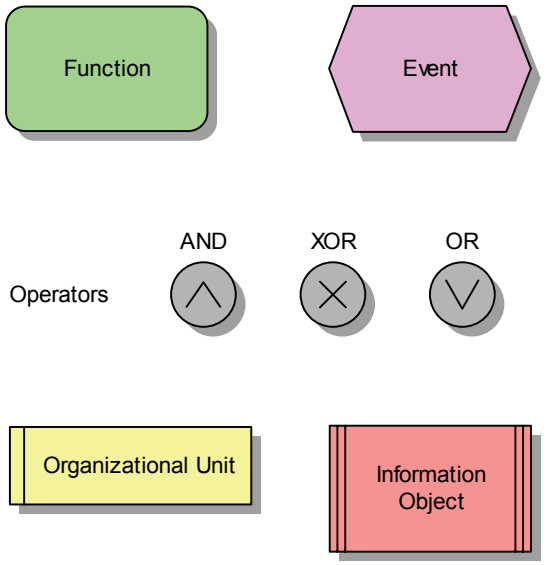



Figure 3 EPC Element Types

Figure 4 and figure 5 show the introduced showcase process modelled as EPC. The process itself consists only of events, functions and operators.

There is often a rigorous restriction, that EPCs must follow an alternation of events and functions. Nevertheless, so called trivial events (that is events, which only confirm the execution of a function) can be disclaimed.

In the rest of the thesis EPCs are used at various points, for instance to describe the developed method. In these illustrations trivial events are often omitted for reason of clarity.

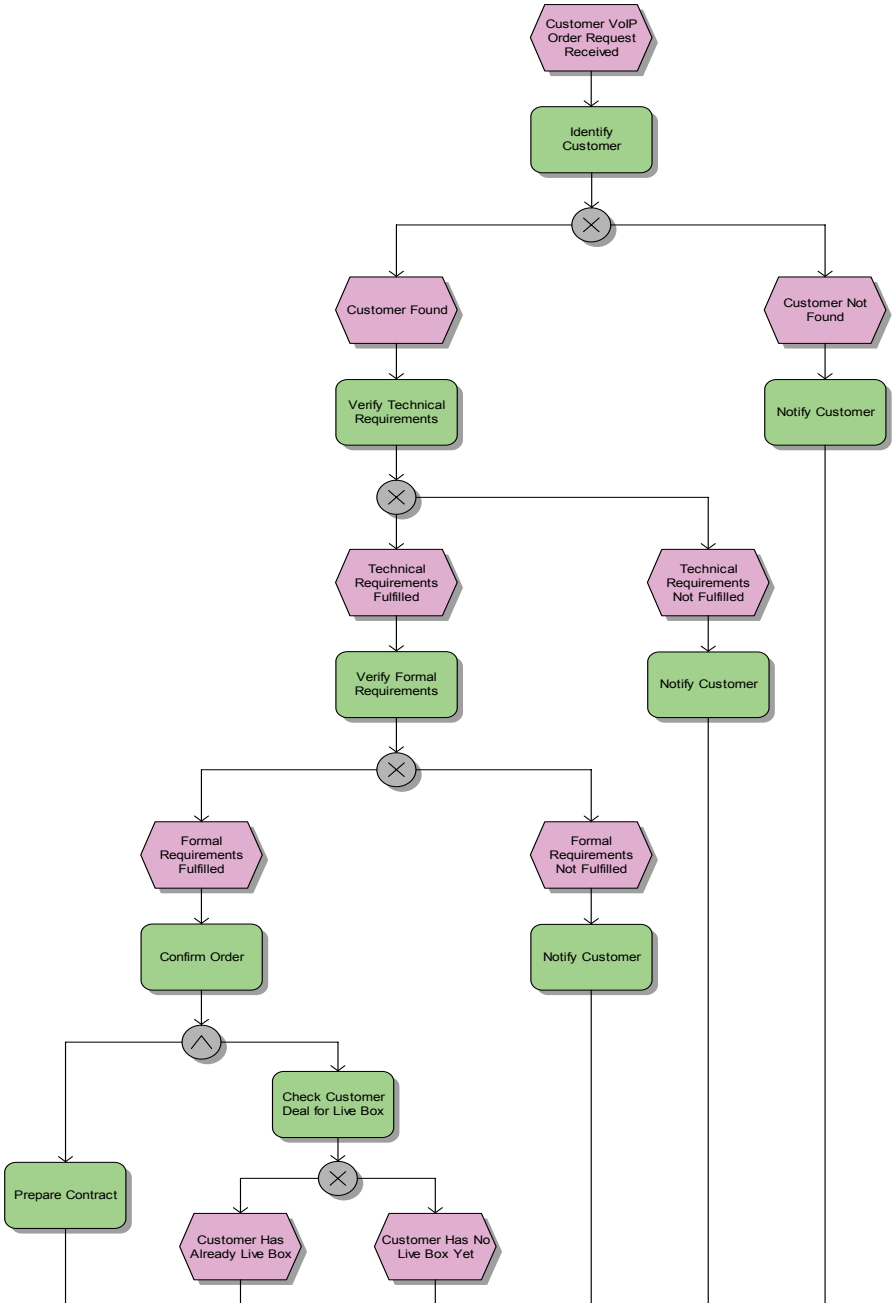



Figure 4 TP Showcase EPC 1/2

### 2.1 Business Level

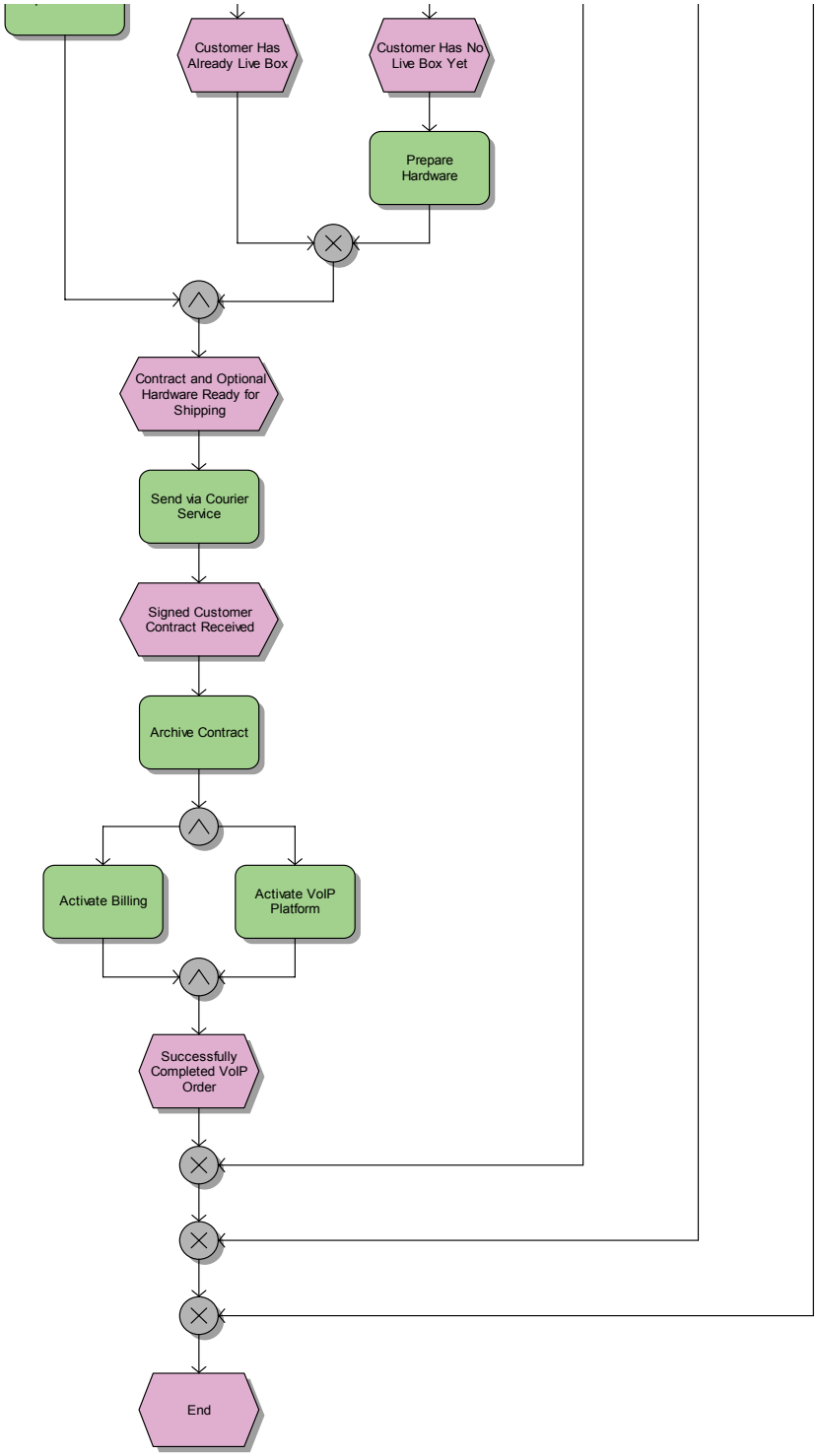



Figure 5 TP Showcase EPC 2/2

### 2.1.2 Business Process Management

Business Process Management (BPM) is an approach for continuous improvement of business processes in enterprises. It represents an iterative procedure of identifying, modelling, documenting, and optimizing business processes, which can be described by the Deming Cycle (see figure 6, according to [Dem82]). It consists of four steps, which are repeated:

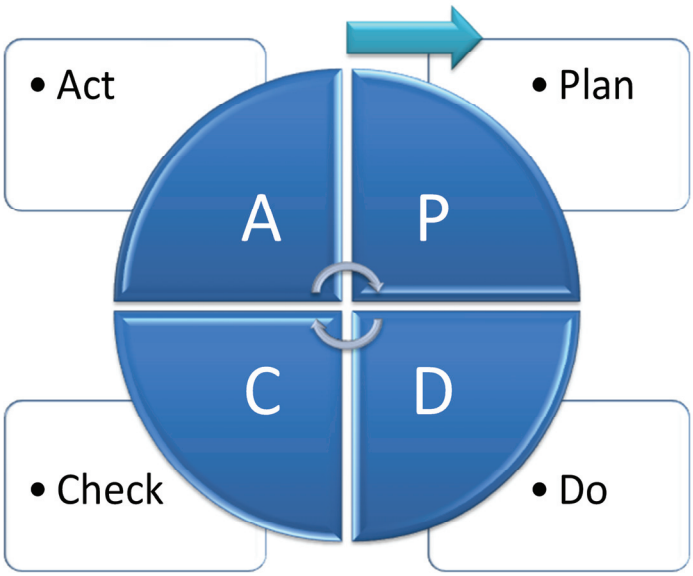



Figure 6 Deming Cycle

**Plan:** the actual process has to be planed and modelled according to the desired objectives

**Do:** the modelled process is implemented and executed

**Check:** the execution of the process is monitored and the results are analyzed with respect to differences from planed results

**Act:** the cause of deviance is determined and the elapsed steps are reviewed and adopted before the next iteration

The iterative procedure represents the need for businesses to continuously adapt their processes to accomplish the permanently changing requirements of their environment. These changes can have many reasons:

- **Dynamics of the market:**
  Businesses permanently need to improve time to market due to competition. Therefore, it is necessary to improve and restructure processes like changing suppliers, outsourcing or automation of specific tasks.

- **Regulations and compliance:**
  Businesses have to cope with new laws like Basel II[7] or the Sarbanes-Oxley Act (SOX, refer to [otUS02]). It is therefore possibly necessary to change existing processes to comply with these guidelines.

Although this procedure (Plan-Do-Check-Act) resides on business level and is in principle not dependent of any underlying information technology, it actually takes place in both layers.

In order to implement an executable business process, the abstract business model has to be enriched with further technical information and normally be transformed into another format, which is not understandable anymore for business analysts. However, the improvement and adoption of the business process is handled on business level, whereby the corresponding process on IT level has also to be adopted or even completely recreated. In fact, such a bipartite approach can harm consistency of model and implementation, as human intervention is necessary.

## 2.2 IT Level

The modelled business processes must be implemented in order to be enacted by the company. Thus, there must be a supporting software infrastructure in which the processes can be executed. Typical examples of such supporting software are middleware products of SAP, IBM, Oracle, and Microsoft. This section describes characteristics of those software systems.

### 2.2.1 Service Oriented Architecture

Service Oriented Architecture (SOA) became very popular during the last years. The term itself is very difficult to define as the core concepts have been obscured through marketing activities [Rau06] and there are many different and even partially contradicting understandings of the SOA terminology. A common definition for SOA is provided by the OASIS SOA Reference Model [MLM+06]:

*"Service Oriented Architecture is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations."*

The definition makes use of the term "capability". Entities (people or organizations) can have certain *capabilities* to provide a specific task. Other entities may have *needs*, which have to be fulfilled. A *service* – the central concept of

[7] http://www.bundesbank.de/bankenaufsicht/bankenaufsicht_basel.php

a SOA – represents a mechanism, which can bring capabilities and needs together [MLM+06]. Hereby, the entity which has a certain capability is called *service provider* and the entity, which has a certain need is called *service consumer*. The capabilities provided through a certain service are accessed through the *service interface*. In order to use the service, the service consumer needs besides the service interface a *service description*. The service provider offers both service interface and service description to the service consumer.

Figure 7 depicts the dependencies between the single roles. The service registry is no part of the reference model, although it is common in most architecture.

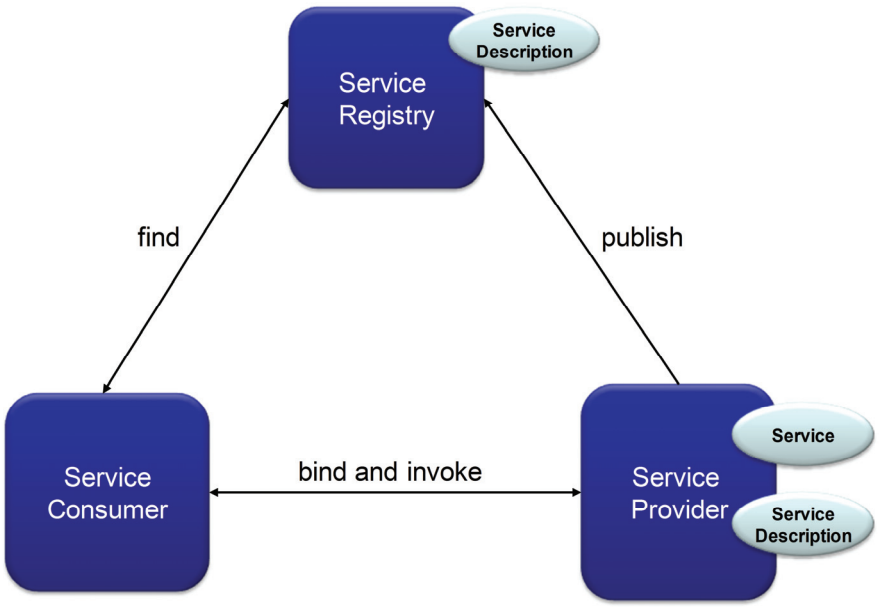



Figure 7 SOA – Roles [CFNO02]

Moreover, the figure visualizes the following procedure:

1. The service provider publishes its capabilities in form of service interfaces and descriptions in the service registry.

2. The service customer searches a service, which matches its needs by means of the service description and interface.

3. When service consumer found a suitable service, it looks up in service registry, where the service provider is located. It uses the service interface to access the capabilities of the service provider through the service to fulfil its needs.

Although the OASIS SOA Reference Model consciously disclaims of mentioning ambiguous attributes in order to provide a more abstract view on SOA, common SOA implementations can be characterized as follows (see [Erl07] and [EAA+04], list not complete):

- **Loose coupling**
  Service interfaces are decoupled by its actual implementation. Therefore, service consumers do not depend on the service implementation.

- **Location transparency**
  As service consumers use a service registry to find a specific service of a service provider, the actual location of the service implementation is hidden to the consumer.

- **Reusability**
  Services are self-contained with an agnostic functional context. Therefore, they can be reused in many different scenarios.

- **Composability**
  Services can be composed to realize business processes.

Software systems implementing SOA bring business processes composed of services in the centre of attention [KL04]. Especially through the "Composability" property, SOA is able to provide the necessary flexibility for Business Process Management.

Service oriented architectures as defined in OASIS SOA Reference Model [MLM+06] are in no way aligned with any specific technology. Nevertheless, many implementations of SOA make use of Web Service technology as described in chapter 2.2, which is often seen as the most important realization form of SOA. According to Leymann [Ley04] business processes implemented by orchestrated Web Services are the future direction for business application development.

Nevertheless, aside from Web Services also other technologies like CORBA, REST, or J2EE can be used to realize a SOA [EAA+04, Fie00].

### 2.2.2 Common Technologies

When talking about SOA and BPM, there is often a connection to a specific technology stack, which involves Web Services (on service level) and Business Process Modelling Language (on process level). The following chapters concentrate on the main aspects of those technologies and summarize especially those facets which are important for this thesis.

#### 2.2.2.1 Web Services

The W3C[8] defines the term “Web Service” as follows [HB04]:

*“A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.”*

The two cornerstones of Web Services are the standards WSDL and SOAP, which are introduced in the following.

**SOAP**

SOAP is a XML based protocol which defines the interaction of systems by means of messages in a distributed environment. Besides the structure of the message, it also defines the way of processing these messages [BEK+00]. A common scenario is the realization of RPC (*Remote Procedure Calls*, refer to [Sri95]) over HTTP (*Hyper Text Transfer Protocol*, see [FGM+99]).

A typical SOAP message is shown in listing 1. The root element of the message is envelope. This element encapsulates (optionally) the header, which contains status information and the body, which contains the actual information of the message.

```
<env:Envelope xmlns:env = http://www.w3c.org/soap-envelope>
    <env:Header> ... </env:Header>
    <env:Body>
          ...
    </env:Body>
</env:Envelope>
```

Listing 1 SOAP Message

**Web Service Description Language (WSDL)**

WSDL is a XML based language for defining interfaces of Web Services. It was submitted in 2001 in version 1.1 (see [CCMW01]) to W3C but is until now no official standard. Nevertheless, it is widespread and plays an important role in many industrial projects (de factor standard). The latest version 2.0 of WSDL (see [FL07]) on the other hand was accepted as official standard in 2007 by W3C. The following section overview WSDL 1.1, as it was used throughout the entire thesis.

8 http://www.w3.org/

Figure 8 gives an overview of the elements of a WSDL description for a Web Service.

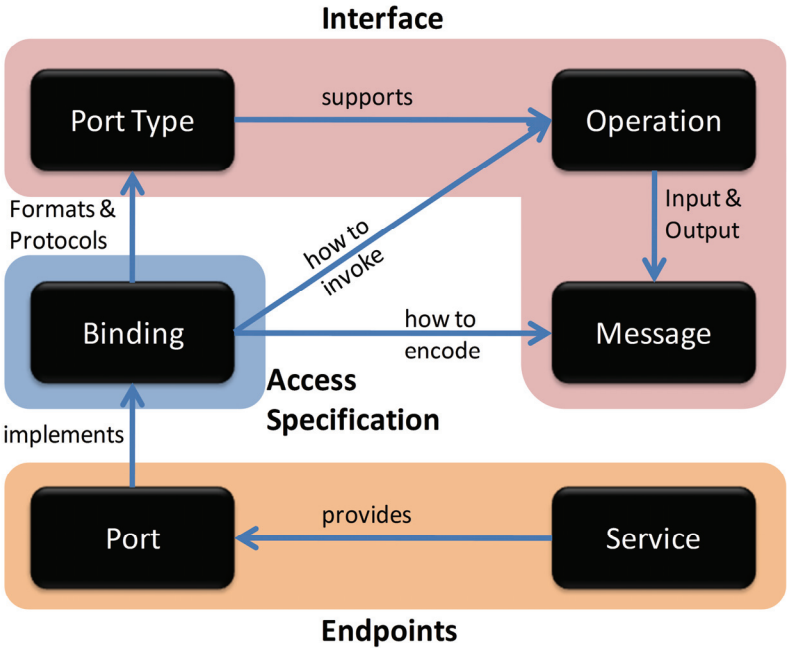



Figure 8 Structure of WSDL Document [WCL+05]

The structure of a WSDL document can be divided in three main parts [WCL+05]:

- **Interface ("what")**
  The interface can be seen as the logical part of the WSDL description. It consists of port types, operations, and messages. A port type consists hereby of operations, which can itself have input and output messages. These elements describe *what* messages a Web Service consumes and produce and *what* operations it offers.

- **Access Specification ("how")**
  The binding of the WSDL description specifies how to invoke operations and how to interchange messages of the Web Service by specifying the required protocols.

- **Endpoints ("where")**
  Consisting of the elements ports and services, this section describes the name of the Web Service and specifies *where* the messages must be addresses in order to invoke the Web Service operation.

#### 2.2.2.2 Business Process Execution Language (BPEL)

The Business Process Execution Language for Web Service (also known as WS-BPEL, BPEL4WS or just BPEL) is an orchestration language for composing processes out of Web Services. Latest version of BPEL is 2.0 (see [Bea05]), which is since April 2007 an official OASIS standard. Nevertheless, this thesis makes use of version 1.1 (see [ACD+03]) as it is more spread and supported by most orchestration engines.

BPEL is used to model executable business processes. Contrary to EPC, it provides a more concrete and very detailed technical representation of a business process. The implemented BPEL processes can be executed by orchestration engines like Oracle BPEL Process Manager[9] or IBM Websphere Process Server[10].

BPEL provides various elements to describe the workflow of a process. These elements are divided in structured activities to model the control flow and atomic activities, which stand for the interaction with a specific Web Service. To be more precisely, the BPEL process references through the `partnerLink` to the WSDL file of the Web Service and particularly to its operations to realise the single activities of the process.

Atomic activities are for example:

- **Invoke**

This activity calls a concrete operation of a Web Service. The WSDL interface of the Web Service is referenced by the `partnerLink`. The operation of this interface is identified over specification of the `portType` and the actual name of the `operation`. Furthermore, input and output variables are stated. Listing 2 shows the invocation of `operation1`, which is available at `portType1` of `partnerLink1`.

```
<invoke   partnerLink="partnerLink1"
          portType="portType1"
          operation="operation1"
          inputVariable="input1"
          outputVariable="output1"/>
```

Listing 2 BPEL Invoke Activity

- **Assign**

Assign activities are necessary to fill the variables of the process. The assign activity therefore nests one or more copy activities that copy the content of one variable into another variable.

[9] http://www.oracle.com/technology/products/ias/bpel/

[10] http://www-306.ibm.com/software/websphere/

```
<assign name="assign1">
   <copy>
      <from variable="var_from" part="part1"/>
      <to variable="var_to" part="part1"/>
   </copy>
</assign>
```

Listing 3 BPEL Assign Activity

Examples of structured activities are:

- **Sequence**

As the name of the activity suggests, the sequence activity is just sequentially executing its nested activities. Listing 1 for instance would invoke consecutively `activity 1`, `activity 2`, until `activity n`.

```
<sequence>
   <!-- activity 1-->
   <!-- activity 2-->
   ...
   <!-- activity n-->
</sequence>
```

Listing 4 BPEL Sequence Activity

- **Switch**

The switch activity specifies, in which direction the control should carry on. Listing 5 shows a switch statement. If the condition specified as XPath expression (refer to [CD99]) is evaluated to true, then `activity 1` is called, else the process control is calling `activity 2`.

```
<switch name="Switch_1">
   <case condition="...">
      <!-- activity 1-->
   </case>
   <otherwise>
      <!-- activity 2-->
   </otherwise>
</switch>
```

Listing 5 BPEL Switch Activity

- **Flow**

BPEL flow activity is used whenever the execution of particular activities does not depend of a specific order and can be quasi accomplished simultaneously. Listing 6 depicts an example where in a flow the `activities 1` to n can be executed simultaneously. The executions engine if free to determine the exact order.

```
<flow name="Flow_1">
   <!--activity 1-->
   <!--activity 2-->
   ...
   <!--activity n-->
</flow>
```

Listing 6 BPEL Flow Activity

## 2.3 Alignment of Business and IT

As BPM addresses the business view on processes and SOA addresses the technical view, an alignment between both views must take place. This generally includes the transfer of the business process model into an implementation, which is executable on the underlying infrastructure.

### 2.3.1 Business Process Automation

The field of business process automation generally aims to reduce cost for business processes by raising the efficiency of certain tasks through automation. With respect to the introduced topic of business process management, the desired area of automation described here is to reduce the effort for creating an implementation out of the abstract business process model.

In order to better understand the exact procedure of transforming the business process model to an executable implementation the concept of Model Driven Architecture is used.

*Model Driven Architecture* (MDA, refer to [OMG03]) is a modelling approach in computer science aiming on clear separation of functionality and technology. The model is structured in three levels of abstraction, which can be adopted based on the area of business process modelling as follows:

- **Computation Independent Model (CIM)**
  Models in this level provide a very abstract view on the process. It is independent of any kind of technology. It represents the business view on the process. This model is produced by business analysts. That means that the process model consists of business functions supported by abstract services, but it is not further specified if these abstract service are later provided by a human or implemented using any kind of software. The CIM is the basis for the other two models (PIM and PSM) and therefore the foundation for a later technical implementation. Typical modelling languages in context of business process modelling are EPC (see chapter 2.1.1.2) or BPMN (*Business Process Modelling Notation*, see [OMG06]).

- **Platform Independent Model (PIM)**
  This model is a refinement of the CIM. It is enriched with general technical details and the model provides already an awareness of software in general. The PIM is typically created by IT experts but is still abstract enough, that business analysts can cope with it. The added details are necessary for a technical implementation but on a very generic level, so that the model itself is in no way dependent on any specific technology. For example, the business process model on the PIM level consists of business functions, which are supported by software services. However, the model does not contain any details on the implementation technology used, so it is not said that the software services must be implemented using Web Service technology. This platform independence allows using the same business process model as a base for different implementations. This is important in enterprise computing, because usually a big company has many different middleware systems. In such a case having a platform independent model is an advantage, because different implementations can be derived from this single source. Therefore, the PIM model is the base for implementation (PSM).

- **Platform Specific Model (PSM)**
  The PSM is beside the instances the most concrete model. Compared to the PIM, it is refined in terms of a specific technology platform. For instance, the introduced software services of the PIM are mapped to concrete Web Services (which are specified by WSDL interfaces). The usage of concrete Web Services includes the clear definition of messages exchanged between the Web Services, as well. The control flow is represented with languages like BPEL or XPDL (*XML Process Definition Language*, refer to [OMG03]), as EPC or BPMN are not suitable for these technical descriptions. Although there is tool support for the transformation from PIM to PSM (for example Stein and Ivanov [SBEK07] describe a semi-automated EPC to BPEL transformation) there is still manual work necessary to create an executable BPEL process. For example, the mentioned data transformations between the different message types has to be done manually and also some parts of the control flow have to be further detailed like adding conditions to split and join statements.

As discussed above, the pure functional CIM is firstly transformed to PIM and then into PSM. Business process automation here aims at lowering manual effort when creating the PSM out of the PIM. Despite the fact that partial automatic transformations are available, eliminating manual work is the goal. As mentioned in the previous section, the business process models are permanently subject of change. Business analysts constantly change the CIM and PIM models to adapt to the business environment. The underlying implementation (PSM models) must be regenerated each time the CIM or PIM is changed. In reality, the PIM is the starting point for modifications as it is abstract enough to be handled by business analysts. Nevertheless, as of to-

day the transformation from PIM to PSM has still to be done partially manually, which is slow and may harm the consistency of the models.

### 2.3.2 ARIS SOA Architect

ARIS SOA Architect[11] by IDS Scheer AG is a business process modelling tool supporting multi-user repository based collaborations. The tool is targeted towards the field of Service Oriented Architectures. Processes are modelled using EPC. Services represented as WSDL interfaces can be selected and assigned to the process [SBEK07]. With help of the build-in transformation algorithm called EPC2BPEL, the EPC can be transformed into an executable format like BPEL. To do so, services have to be identified in EPC. Service descriptions are included on the basis of WSDL descriptions imported in the tool.

#### 2.3.2.1 EPC2BPEL Transformation

The mentioned EPC2BPEL transformation of ARIS SOA Architect enables the transformation of specially formed EPC processes into a corresponding BPEL model.

The principle structure of an EPC differs to the structure of BPEL. While EPC is a graph oriented language with a lot of flexibility to create a control flow, the desired BPEL process is block oriented. That means that the structure of a BPEL process is typically of hierarchical nature that does not provide this flexibility. For the EPC2BPEL transformation this means that the underlying EPC must fulfil certain requirements in order to be transformable. A detailed explanation of those requirements is out of scope for the thesis at hand and can be found in [SI07b, SI07c].

Concerning the VoIP showcase process, the most important restrictions are for example that every splitting XOR must be joined by a XOR again. The same holds for splitting and joining AND rules.

Although these requirements mean that not all possible EPCs can be transformed into BPEL, they are usually no restriction to the modelling of an EPC as they are part of best modelling practices anyway.

The transformation can be sketched as follows[12]:

[11] http://www.aris.com/soa

[12] Note that this is a simplified description of the algorithm as it only shows relevant aspects for the thesis and the underlying showcase process. For the full description of EPC2BPEL refer to [SI07c].

### 1. Function

For every function in the EPC process a new invoke activity is created in the corresponding BPEL process. This invoke statement is after transformation not complete yet, as for instance the concrete operation is not specified. Figure 9 shows that the functions of the EPC (on the left) are transferred one on one into BPEL invoke activities (on the right).


Figure 9 EPC Function into BPEL Invoke

### 2. XOR Block

For every XOR block in the EPC, that means a section of the EPC which starts with a XOR and ends with an XOR, a new switch activity in BPEL is created (see figure 10). Note that *function1* and *function2* in the EPC can be seen as recursive placeholders for any constructs consisting of functions, XOR blocks and AND blocks.

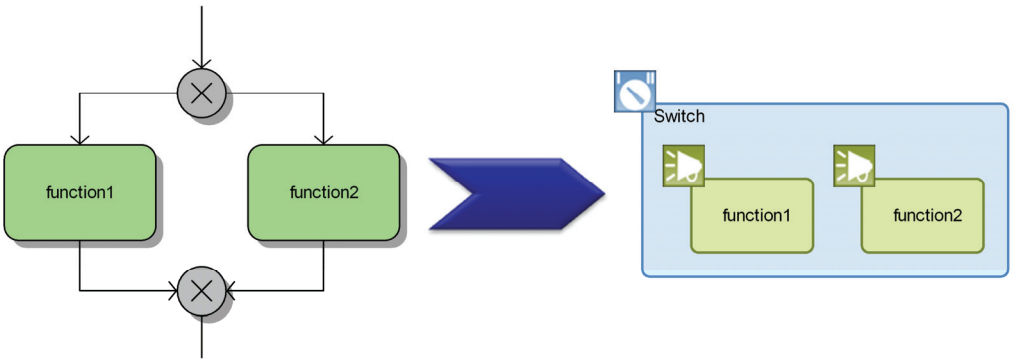



Figure 10 XOR Block into BPEL Switch

### 3. AND Block

Similar to the XOR block, for every AND block (section which starts with an AND rule and ends with an AND rule) is transformed into a flow activity. This is shown in figure 11, where *function1* and *function2* can again be seen as recursive placeholders.

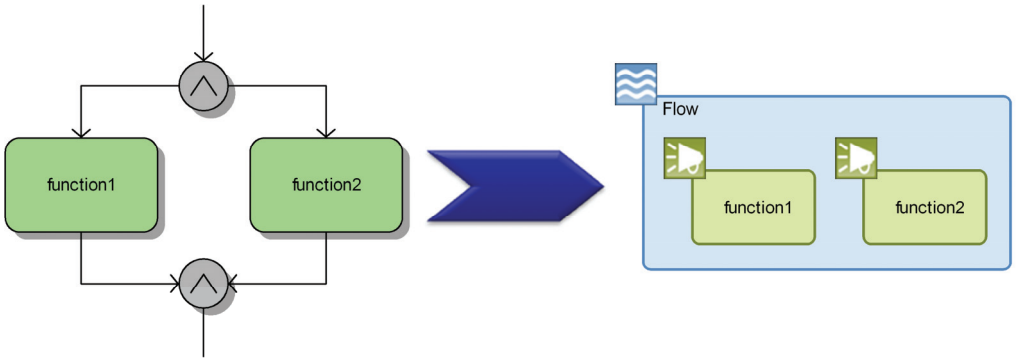



Figure 11 AND Block into BPEL Flow

The resulting BPEL process is still quite incomplete as neither the dataflow is modelled nor decision conditions on control flow branches (splitting XOR rules) are specified.

### 2.3.3 Standard Method

A common method, which covers the procedure of modelling a business process to the point of its implementation in the area of BPM and SOA, is illustrated in figure 12. The following method mostly corresponds to the procedure described in [SI07a] and is actually based on ARIS SOA Architect (see chapter 2.3.2). However, the method is of significant importance for the thesis, so it became its own headline.

The single steps of the procedure are described in detail:

#### 1. Model Business Process

The business analyst models the business aspects of the process. At this point the process contains pure functional aspects and resides on CIM level.

Goal: The functional aspects of the business process are modelled

Input: General description of the process

Output: Abstract model of the process represented as EPC. This model only consists of events, functions and operators. Result is an EPC process as shown in figure 4 and figure 5.

Roles: Business analyst

### 2. Refine Model with Web Services

In this step, the business analyst has to decide, which of the functions of the EPC can be fulfilled by Web Services. ARIS SOA Architect therefore provides a selection menu, in which all available Web Services are listed. The business analyst can search in these services by means of different criteria:

- Syntactical data input and output variables of the software service. Furthermore, also the name of the software service can be used for service identification.
- Stored textual descriptions and tags of the service. When storing Web Service descriptions in SOA Architect, it is possible to deposit also textual descriptions or tags that include information about the capabilities of the Web Service.

Further details about this service selection can be found in [SBEK07]. Despite these criteria it might be impossible for business analysts to make a decision which Web Service to choose. This can have a few reasons:

- It the textual descriptions are to imprecise or ambiguous.
- If more than one Web Service might fit to the concrete need.
- If there are no tags assigned to the Web Service yet.

These problems take especially effect if there are a lot of Web Service descriptions available. This can make it necessary for the business analyst to consult the integration engineer, who has deeper knowledge of the capabilities of the Web Services. The resulting process of this step contains already technical details and resides therefore on PIM/PSM level.

| | |
|---|---|
| Goal: | It has to be determined, which functions of the EPC can be covered by which Web Service |
| Input: | Business model of the process, Web Service descriptions |
| Output: | Process model with assigned Web Services |
| Roles: | Business analyst, integration engineer |

### 3. Transform Into Executable Process

The refined process model has to be transformed in an executable format. Therefore, the described EPC2BPEL transformation (see chapter 2.3.2.1) is applied and creates a corresponding but yet incomplete BPEL representation of the EPC process. The business analyst or the integration engineer is hereby just triggering the transformation, as the procedure itself runs automatically. This BPEL process is located on PSM level of MDA.

| | |
|---|---|
| Goal: | Transfer the EPC process into a yet incomplete BPEL representation. |
| Input: | Process model with assigned Web Services. |
| Output: | Incomplete BPEL representation of process (BPEL closure). |
| Roles: | Business analyst or integration engineer (just triggering the transformation) |

**4. Finalize Executable Process**

After running the transformation in step 3, the resulting BPEL code has to be completed. This code only represents at this moment the basic control flow with the invocation statements of the identified Web Services. Still missing in the code is:

- Concrete operation of each assigned Web Service
- Branch conditions for switch activities
- Complete data flow, that is assign and copy activities to map the input and output messages on each other
- Exception handling

These information has to be manually added by integration engineer, which is generally tool supported, for instance by IDEs like Oracle JDeveloper[13].

| | |
|---|---|
| Goal: | The BPEL process closure from the EPC2BPEL transformation has to be filled with the missing implementation specific details. |
| Input: | Incomplete BPEL process |
| Output: | Completed BPEL process |
| Roles: | Integration engineer |

**5. Deploy BPEL Process**

The resulting process can be deployed on application server and is ready for invocation. This step can be automated, for instance by execution of an Ant[14] script and needs only to be triggered.

[13] http://www.oracle.com/technology/products/jdev/

[14] http://ant.apache.org/

| | |
|---|---|
| Goal: | Deploy the final BPEL process on a orchestration engine |
| Input: | Completed BPEL process implementation |
| Output: | Successfully deployed process, which can be used productively |
| Roles: | Integration engineer (just triggers the deployment) |

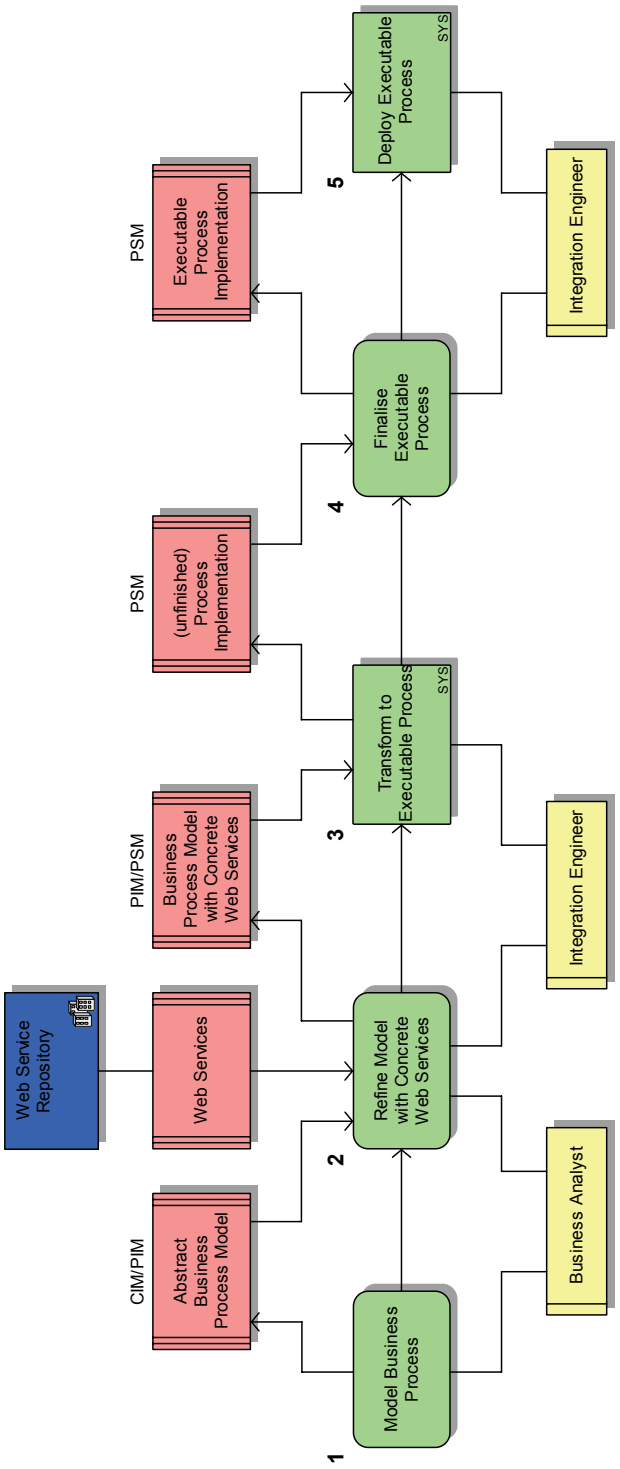



Figure 12 Standard Method

# 3 Semantic Approaches

The standard method described in chapter 2.3.3 revealed some shortcomings like the difficulty of Web Service selection and much manual work in process transformation. These problems are well known in current research and topic of various scientific essays and uncountable research projects. The core idea of many of these projects is the introduction of semantic technologies.

This chapter provides an overview of the current research concerning semantic technologies. The common basis for all these research projects are ontologies, which are discussed in chapter 3.2.1.

Semantic Web Services, which are topic of chapter 3.2.2, are founded on ontologies and introduce semantics on service level.

The latest trend copes with semantics on process level and the enrichment of BPM with semantic technologies. The so called Semantic Business Process Management is examined in chapter 3.2.3.

Chapter 3.3 deals with the technical aspects necessary for the introduction of the previously described semantic concepts.

Finally chapter 3.4 shows, which of the introduced ideas and identified problem are exactly addressed by the thesis.

## 3.1 Motivation for Semantic Technologies

As already mentioned in description of the state of the art method in chapter 2.3.3, there are several difficulties when trying to combine the concepts of the business level with the underlying IT infrastructure. The general cause can be broken down to a communication problem which separates the business view from the IT view.

This communication problem is founded on the different levels of abstraction and different languages used on both sides. That is the pure functional view of the business analysts using EPC to describe the business processes on a very high level and the technical process implementations by means of BPEL code on the side of IT experts.

The resulting problem can be divided into two parts as shown in figure 13. Part one addresses the direction from business level to the IT level (top-down problem), which deals with the transformation of the business process

model into an executable implementation. Part two of the problem is addressing the opposite direction (bottom-up problem), which copes with querying of the process space [HLD+05, HR07a]. Figure 13 shows that as of today still manual work is needed to mitigate between both levels. This is known as the Business-IT Divide.

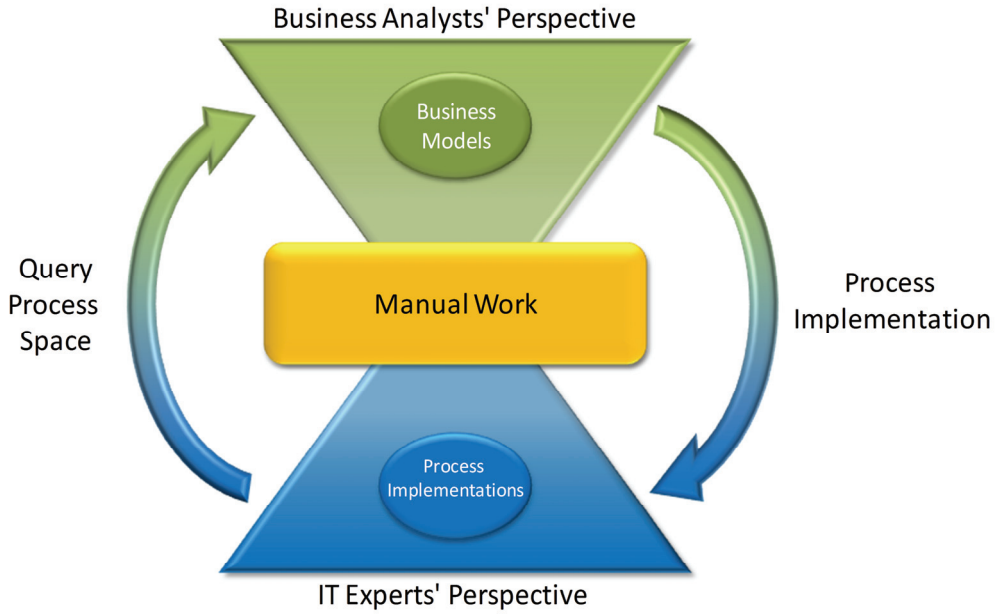



Figure 13 Business-IT Divide [HLD+05]

### 3.1.1 Top-Down Problem

The introduced state of the art method for business process modelling shows the complexity of business process automation. The Business-IT Gap appears especially in step 3 and step 4/5 of the method explained in chapter 2.3.3, which are the assignment of Web Services to the EPC functions and the transfer of the model into the implementation (compare to figure 13, “Process Implementation”).

The task of service selection (step 3) places business analysts in the difficult position to select Web Services and possibly their operation by means of their syntactic description (WSDL files) or maybe an informal textual description, which are often ambiguous and lack of unified semantics. Business analysts in general do not have wide knowledge in computer science as their job is to concentrate on business logic. Therefore, this step generally has to be accomplished with the help of IT experts.

The transformation of the model into the executable process (step 4/5) is indeed partially automated (EPC2BPEL), but leaves still pretty much work

open for IT experts to finalize the implementation. The reason for this lies in the used notations on business level like EPC or BPMN. These languages describe processes from an abstract business point of view. They abstract from technical details like exception handling or explicit modelling of data flow. Therefore, these representations in general are not able to provide directly an executable process.

Result is the mentioned lack of automation and a need for IT experts to deal with service selection as well as the implementation of business processes. When modifying an existing business process model, the corresponding and already implemented business process has to be modified. The adapted model has to be passed to IT department, which starts the software development process. As elaborated in motivation of chapter 1, this practise is very costly, slow and may result in inconsistent process implementations as the resulting program cannot be verified by business analysts. Hence, the desired and necessary flexibility of BPM may suffer.

In terms of the used technology another drawback lies in the design-time binding of the used Web Services. Although the WSDL description is decoupled from the actual service implementation, the location of the desired interface has already been specified. This influences directly the reliability of the system for instance if a Web Service fails.

### 3.1.2 Bottom-Up Problem

Part two of the general problem is about the direction from IT level to business level (compare to arrow “Query Process Space” of figure 13).

Business analysts often want to monitor and analyze process space of the enterprise (*check* and *act* of Deming cycle, refer to chapter 2.1.2). For example if the average throughput time of all processes involved with raw meat must be listed. For this task often implicit knowledge is needed, that is not available in machine readable representation. If for instance a machine is processing pork it must be known that pork is a sub category of meat.

This makes it necessary to manually check the process space, and to locate processes dealing with all sub forms of meat, like pork, poultry, beef, and so on. This procedure is very costly and error prone. For example if the process space is very big it is likely that some processes might be missed, especially if not all sub forms are known.

### 3.1.3 Semantics to Solve Both Problems

The introduction of semantics eliminates the communication problem. This is accomplished by providing a common communication basis by means of a unified vocabulary – an ontology (see chapter 3.2.1).

This communication basis can help to solve the bottom-up problem. Applied on the meat query example, the ontology explicitly defines all specialized forms of meat. This makes it possible to automatically consider every relevant process, even if it processes pork.

The focus of this thesis lies in the top-down approach. The subsequent part of the thesis shows in detail, how semantics can help with Business Process Automation and to overcome the mentioned shortcoming of chapter 3.1.1.

## 3.2 Conceptual Aspects

The following section describes the conceptual aspects of semantic technologies. The section first introduces the foundation for semantic technologies. With respect to this foundation, semantic technologies can be divided in service levels, in which the elementary services are provided and business process level which orchestrates the services of the service level.

### 3.2.1 Semantic Basis: Ontologies

Common to all semantic technologies is the same fundamental concept: ontologies. There exist a wide range of definitions for the term ontology (see [MI96] and [UG96]). A common definition is provided by Tom Gruber [Gru93]:

*“In the context of computer and information sciences, an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members). The definitions of the representational primitives include information about their meaning and constraints on their logically consistent application. …”*

Gruber also gave a famous short definition of ontology [Gru93]:

*"An ontology is an explicit specification of a conceptualization"*

The here discussed ontologies are so called domain ontologies, which captures the terminology and relationships of a specific domain. Ontologies describe an explicit and formal conceptualized view on the domain. They describe the entities, their attributes and the existing relationships between those entities of the domain.

Domain ontologies are closely related with the data models of data bases and conceptual modelling techniques like ERM (*Entity Relationship Model*,

refer to [Che76]) and UML (*Unified Modelling Language*, see [OMG07]) but on a higher more abstract and pure functional level, with no technology specific intend.

Goal of an ontology is to provide a common and clearly semantically specified terminology or vocabulary of a specific domain (compare to chapter 3.1.3). Ontologies therefore offer a common communication base. Important for the field of enterprise computing is the ability that ontologies are machine readable, which is indirectly covered by the definition (*explicit* specification).

The procedure of creating ontologies is a difficult undertaking. When creating an ontology for instance for the domain of a company, there are already many different, partially overlapping and often imprecise vocabularies, which have to be combined to a common denominator. Involved in this process are many different roles and hence different points of view, which need to be moderated.

There are different formal languages concerning the representation of ontologies. The most popular ones are *Resource Description Framework Schema* (RDFS, see [BG04]), *Web Ontology Language* (OWL, refer to [MvH04]) and the *Web Service Modelling Language* (WSML, see [dBLK+05]). A characteristic of the WSML language is the human readable syntax, which differs from other languages because it does not have a XML based syntax.

WSML comes in combination with WSMO (*Web Service Modelling Ontology*) and WSMX (*Web Service Modelling Execution*), which are introduced in chapter 3.2.2. All further shown ontologies are based on WSML.

Listing 7 shows an extract of the TP ontology used for the showcase process. The ontology consists of various elements such as *concepts* and their *attributes*. For instance, the concept Customer has three attributes `hasName`, `hasTelephoneNumber`, and `hasAccount` which represent the customer's name, telephone number, and account number. Each element can have non functional properties, which are used to title the element or to provide an informal description. Attributes can either be primitive types like integer, or can have a concept as type like the attribute `hasName` of concept Name. Furthermore, WSML supports inheritance. For example, the concept `VoipService` is a sub concept of `TelcoService`, which itself extends the general concept `Service`.

```
ontology _"http://org.ipsuper.composition.tp/tpOntology"
     nonFunctionalProperties
          dc#title hasValue "Polish Telcom Ontology"
          dc#language hasValue "English"
     endNonFunctionalProperties
```

```
concept Customer
     nonFunctionalProperties
          dc#description hasValue "An abstract notion for so
                         meone who pays for goods or services."
     endNonFunctionalProperties
     hasName impliesType Name
     nonFunctionalProperties
          dc#description hasValue "the name of the customer"
     endNonFunctionalProperties
     hasTelephoneNumber ofType _integer
     nonFunctionalProperties
          dc#description hasValue "a telephone number which
                         customer pointed for contacts with him"
     endNonFunctionalProperties
     hasAccount ofType _integer
     nonFunctionalProperties
          dc#description hasValue "arrangement of subscribed
                         services, payments, profile, etc. to a cus-
                         tomer; it is also used for billing purposes;
                         each customer account must have unique iden-
                         tification number"
     endNonFunctionalProperties

concept Service
     nonFunctionalProperties
          dc#description hasValue "An abstract notion for va
                         lue provisioning in some domain."
     endNonFunctionalProperties
     hasName impliesType Name
     hasID ofType _decimal
     nonFunctionalProperties
          dc#description hasValue "the name of the service."
     endNonFunctionalProperties
     isHardwarePrepared ofType _boolean
     isActive ofType _boolean

concept TelcoService subConceptOf Service
     nonFunctionalProperties
          dc#description hasValue "A service in the telcom do
                         main."
     endNonFunctionalProperties
     hasOptions impliesType _string
     nonFunctionalProperties
          dc#description hasValue "possible options which may
                         be chosen by a customer."
     endNonFunctionalProperties
     hasLines impliesType Line
     nonFunctionalProperties
          dc#description hasValue "a (set of) line(s) dedica
                         ted to the service"
     endNonFunctionalProperties

concept VoipService subConceptOf TelcoService
     nonFunctionalProperties
          dc#description hasValue "A VoIP service"
     endNonFunctionalProperties
```

```
concept Name
     forename impliesType _string
     surname impliesType _string
```

Listing 7 Extract of TP Ontology

Elements missing in the example above are *axioms*, which represent conditions that have to be true all the time and *instances*. Instances incorporate a concrete specificity of a concept. An instance of the concept `Customer` is shown in listing 8.

```
instance PeterLustig memberOf Customer
     hasName hasValue petersName
     hasTelephoneNumber hasValue 4711
     hasAccount hasValue 1234

instance petersName memberOf Name
     forename hasValue "Peter"
     surname hasValue "Lustig"
```

Listing 8 Exemplary Ontology Instance

As already mentioned, listing 7 provides only a small section of the actual ontology. Ontologies in general can be very comprehensive. Figure 14 shows the complete TP ontology with all defined concepts. The picture just aims to illustrate the size of the ontology and the number of concepts, so it is not necessary to recognize each single concept. This figure was generated using the graphical visualizer of WSMO Studio (see chapter 3.2.2 for further details about WSMO Studio).


Figure 14 Concept Overview of TP Ontology

### 3.2.2 Semantic Web Services

The general idea of Semantic Web Services is to describe the functionality of a Web Service semantically. This is accomplished by means of logical expressions and with the vocabulary of the underlying ontology. The Semantic Web Service then can be found and invoked with help of these semantic descriptions [BLHL01, FLP+06].

The notion of Semantic Web Services was subject of many European research projects in recent years, like *DIP*[15], *ASG*[16] and *SemanticGov*[17].

All these projects make use of the

**Web Service Modelling Ontology (WSMO)**

In the mentioned projects the Web Service Modelling Ontology (WSMO) (refer to [FLP+06]) was developed and enhanced. WSMO is considered as a meta model to describe and formalize the various aspects of Semantic Web Services. All aspects of WSMO are specified using WSML (refer to chapter 3.2.1).

The four main elements of WSMO are:

### 1. Ontologies

Ontologies are the key elements of WSMO and the foundation for the Web Services, Goals and Mediators. They consist of concepts, attributes, relations, axioms and instances as detailed in chapter 3.2.1.

### 2. Web Services

WSMO Web Services describe a specific service semantically but independent of a concrete implementation. Basis for the description is the ontology respectively their concepts and attributes. Web Services consist of one `capability` and one `interface`.

The `capability` of a Web Service describes necessary `preconditions` that need to be fulfilled in order to call upon the service. Furthermore, it defines `postconditions`, which specify the state of the world after the invocation of the Web Service. Both conditions are specified as predicate logical expressions.

Listing 9 shows an example of a WSMO Web Service called `FormalVerification`. It has the `precondition` that the input variable `customer` must be of type `TP_Customer` and attribute `hasCustomerID` must be of type `CustomerID`. The `postcondition` of the service is that the passed `customer` must have attribute `hasVerification` specified and the attribute `hasVerficationResult` of `hasVerfication` must be `true`.

[15] Data, Information and Process Integration with Semantic Web Services: http://dip.semanticweb.org/

[16] Adaptive Service Grid: http://asg-platform.org/

[17] SemanticGov: http://www.semantic-gov.org/

The `interface` can be divided in `Choreography` and `Orchestration`. Section Choreography is specifying, how the Web Service is being called. Therefore, it specifies the in- and output variables of the interface. Listing 9 defines one input variable of type `FormalVerificationRequest` and one output variable of type `FormalVerificationResponse`. With help of grounding (`withGrounding`) of the input variable it is further determined, which concrete Web Service implementation has to be invoked if an instance of type `FormalVerificationRequest` is being received. The exact procedure is illustrated in chapter 3.3.2.

```
webService FormalVerification
   nonFunctionalProperties
      wsmostudio#version hasValue "0.7.2"
   endNonFunctionalProperties

   importsOntology
      _"http://org.ipsuper.composition.tp/tpOntology"

   capability FormalVerificationCapability
      sharedVariables ?customer

      precondition FormalVerificationPrecondition
         definedBy
            ?customer[hasCustomerID hasValue ?cID] memberOf
                tp#TP_Customer
            and ?cID memberOf tp#CustomerID.

      postcondition FormalVerificationPostcondition
         definedBy
            ?customer[hasVerification hasValue ?verification]
                memberOf tp#TP_Customer
            and ?verification[hasFormalVerificationResult
            hasValue _boolean("true")] memberOf
                tp#FormalVerification.


   interface FormalVerificationInterface
      choreography wsFormalVerificationChoreography
         stateSignature wsFormalVerificationStateSignature
            importsOntology
               _"http://org.ipsuper.composition.tp/tpOntology"

               in concept tp#FormalVerificationRequest withGrounding
                   {_"http://localhost:8080/axis2/services/
                   CRMService?wsdl
                   #wsdl.interfaceMessageReference(
                   CRMService/verifyFormalRequirements/in0)"}
                out concept tp#FormalVerificationResponse
```

Listing 9 WSMO Web Service

### 3. Goals

Goals describe the need of a service customer. It consists analogue to a WSMO Web Service of exactly one capability (although the term “need” would be cleaner, compared to OASIS SOA Reference Model) and possibly one interface.

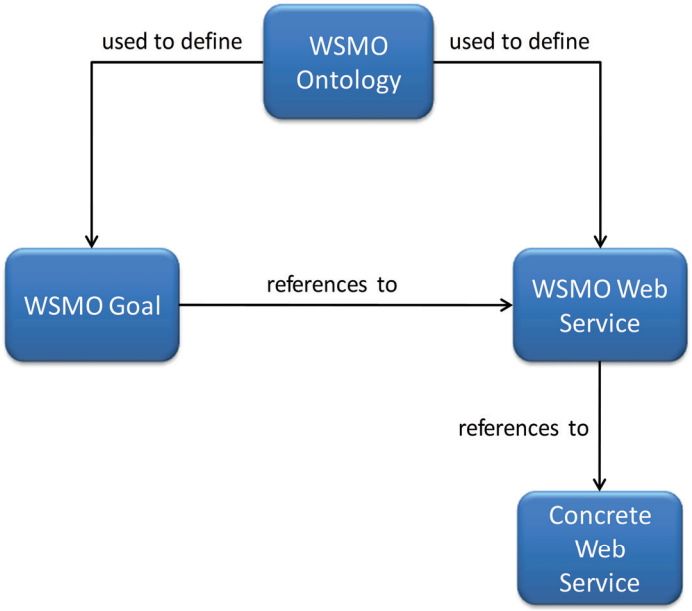



Figure 15 WSMO Ontologies, Web Services and Goals

Figure 15 visualizes the relationship between ontologies, Web Services and Goals described so far. Ontologies serve as basis for the Goals as well as the Web Services. The connection of Goal to (WSMO) Web Service is indirectly drawn by semantic reasoning, which compares the descriptions by means of predicate logic. The reference to the concrete Web Service is realized by the grounding as described above.

### 4. Mediators

Mediators provide a connection between different ontologies, Web Services, Goals and other mediators. The major task of mediators is to enable communication of partners even if these are using different ontologies to describe their domain.

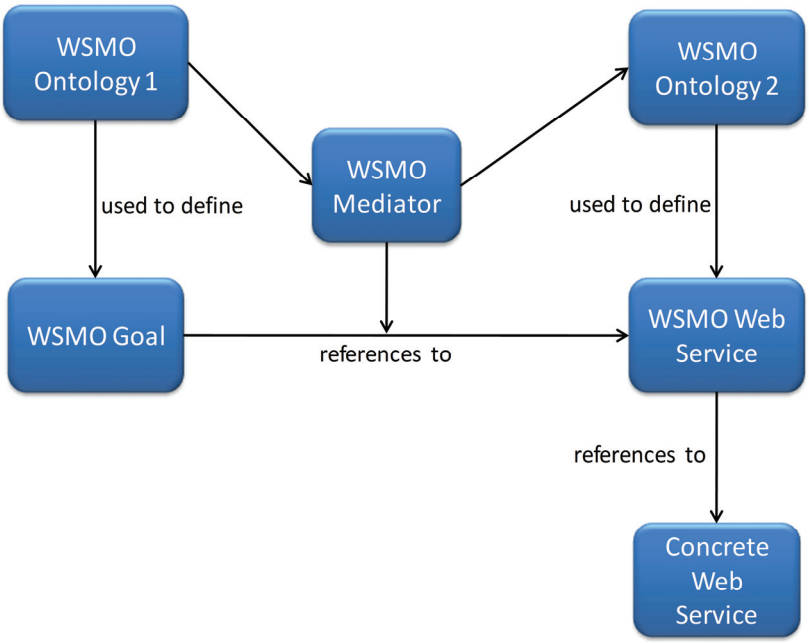



Figure 16 Role of WSMO Mediators

Figure 16 illustrates this. Imagine there is one ontology which is used to define the Goal and another ontology defining the Web Service. As the ontologies and therefore the Goal and the Web Service contain different concepts and attributes, the semantic reasoner cannot directly identify the Web Service by means of the Goal. The mediator in the scenario mediates between Goal and Web Service by mapping the single concepts and attributes of both ontologies.

Mediators benefit to WSMO by providing a lot of flexibility. It can be ensured that two business partners can communicate with each other although they are using different ontologies to describe their domain and their Web Services and Goals respectively. Nevertheless, it is on the other hand necessary to create those mediators, which is additional work to do.

Mediators are not applied during this thesis. Here they are shortly sketched for the sake of completeness. WSMO also provides different types of mediators, which is out of scope of the present work. For more details refer to [MC05, MCS+05].

### 3.2.3 Semantic Business Process Management

The term Semantic Business Process Management (SBPM) was introduced by Hepp [HLD+05] in 2005. The general idea is to combine Semantic Web Service technologies of chapter 3.2.2 with BPM in order to address the topics of chapter 3. Hepp et al. aim at *"raising BPM from the IT level where it mostly resides to the business level where it actually belongs"* [BCD+07, HLD+05, HR07a] with help of semantic technologies.

Their vision is basis for the European integrated project SUPER[18] (Semantics Utilized for Process Management within and between Enterprises).

The goal of SUPER is to provide a consolidated architecture and methodology for Semantic Business Process Management. SUPER tries to cover the complete Deming Cycle (see chapter 2.1.2) with its methodology and supports it as well as with its architecture. This makes it necessary to provide support for modelling (*plan*), execution (*do*), monitoring (*check*) and analysis (*act*).

There is a large set of ontologies used and developed in SUPER. All these ontologies are based on the WSML standard. SUPER provides an ontological representation of common process languages like EPC, BPMN, and BPEL. The corresponding ontologies sEPC, sBPMN, and sBPEL provide a meta model for the concepts of the mentioned languages (refer to [BCD+07]).

The central ontology of SUPER is the *Business Process Management Ontology* (BPMO, see following chapter 3.2.3.1), which subsumes common concepts of the ontologies sEPC, sBPMN and sBPEL and can be seen as an intermediary process representation [BCD+07].

Figure 17 shows the intended process lifecycle in SUPER. On the left, business process are modelled by means of standard modelling tools like ARIS SOA Architect. The resulting process model is annotated with semantic information ("ontological lifting") and transformed into the intermediary representation in BPMO and stored in a central repository. Additionally to the use of traditional modelling tools, SUPER also aims to provide its own tools for creating BPMO processes like WSMO Studio (see chapter 3.3.1).

The BPMO representations of the processes in the repository serve on the one hand as source for process space queries (compare to the bottom-up problem of chapter 3.1.2) and on the other hand for the automatic generation of executable process implementations (see top-down problem of chapter 3.1.1).

[18] Integrated Project SUPER: http://www.ip-super.org/

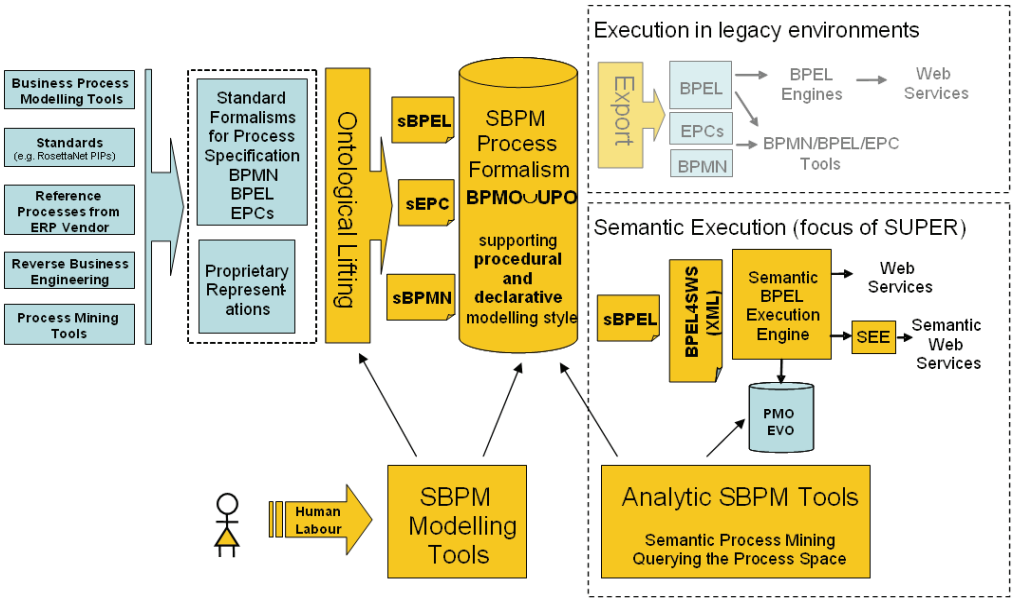



Figure 17 SUPER Process Lifecycle [BCD+07]

### 3.2.3.1 Business Process Modelling Ontology (BPMO)

As elaborated above BPMO subsumes common concepts of process description languages EPC, BPMN, and BPEL. Hereby, it covers the graph oriented as well as the block oriented workflow patterns of these languages (refer to [BCD+07]).

The most relevant concepts of BPMO for the underlying thesis and the introduced showcase process are now highlighted.

- **Goal Task**

A goal task is representing a semantic service of the modelled process. This service is specified by a WSMO Goal, which is located in attribute `hasWSMOGoal`. The actual content of the Goal is referenced by its URI and is stored in a repository of the SUPER architecture. Listing 10 shows an example goal task instance. A goal task can be compared to a BPEL invoke activity.

```
instance GoalTaskInstance memberOf GoalTask
     hasName hasValue "Task"
     hasWSMOGoal hasValue "http://example.org/Goal"
```

Listing 10 BPMO Goal Task

- **Sequence**

A sequence can be directly compared to the sequence activity of BPEL. It links elements of the process which are executed one after the other (refer to listing 11). The elements of the sequence can be goal tasks, again sequences or all other presented elements.

```
instance SequenceInstance memberOf Sequence
     hasOrderedElement hasValue {Element_1,
                                 Element_2,
                                 ...
                                 Element_n }
```

Listing 11 BPMO Sequence

- **Exclusive Choice-Merge**

Exclusive Choice-Merge instances are used to model decisions in control flow. Depending on which of the specified condition is true at runtime, the corresponding branch is executed. These constructs of BPMO can be compared to the switch activities of BPEL. Listing 12 shows an example of an Exclusive Choice-Merge instance with two branches.

```
instance ExclusiveChoiceMergeInstance memberOf ExclusiveChoiceMerge
     hasConditionalBranch hasValue {ConditionalBranch_a,
                                    ConditionalBranch_b }

instance ConditionalBranch_a memberOf ConditionalBranch
     hasBranch hasValue Instance_a
     defined_By hasValue "WSMO ConditionExpression a"

instance ConditionalBranch_2b memberOf ConditionalBranch
     hasBranch hasValue Instance_b
     defined_By hasValue "WSMO ConditionExpression b"
```

Listing 12 BPMO Exclusive Choice-Merge

- **Parallel Split-Synchronize**

This BPMO construct is used in process whenever two or more elements can be executed simultaneously. It is comparable to flow activities of BPEL. Listing 13 shows an example with two concurrent executable branches.

```
instance ParallelSplitSynchroniseInstance memberOf
                                          ParallelSplitSynchronise
     hasConditionalBranch hasValue {ConditionalBranch_a,
                                    ConditionalBranch_b }

instance ConditionalBranch_a memberOf ConditionalBranch
     hasBranch hasValue GoalTaskInstance_a

instance ConditionalBranch_b memberOf ConditionalBranch
     hasBranch hasValue GoalTaskInstance_b
```

Listing 13 BPMO Parallel Split-Synchronize

Figure 18 depicts a small example process modelled in BPMO. It consists of a sequence of one start event, one goal task, an exclusive choice-merge instance and an end event. The process in figure 18 was created and visualized using WSMO Studio (see chapter 3.3.1).

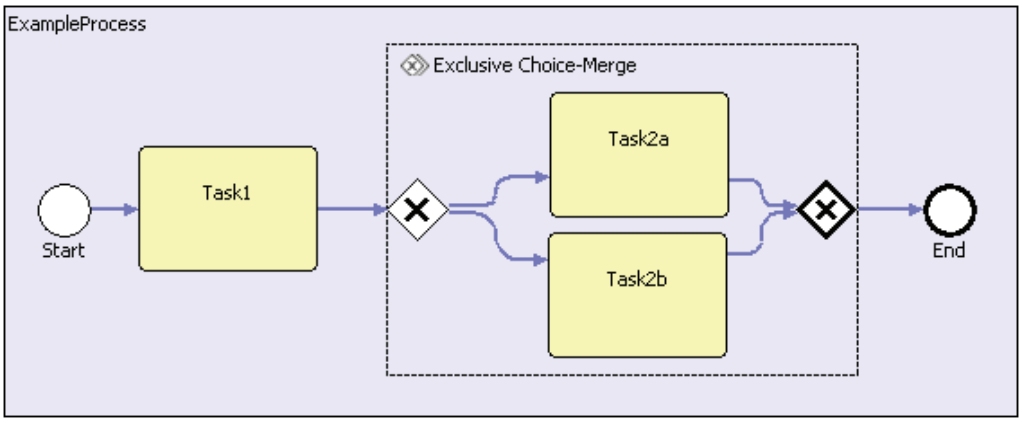



Figure 18 Example BPMO Process

The corresponding WSML description of the process can be found in listing 14.

```
ontology processInstanceOntology
     nonFunctionalProperties
          wsmostudio#version hasValue "0.7.2"
     endNonFunctionalProperties

     importsOntology
           _"http://www.ip-super.org/ontologies/BPMO/20070620#"

instance ProcessInstance memberOf Process
     hasName hasValue "ExampleProcess"
     hasWorkflow hasValue SequenceInstance

// Main Sequence
instance SequenceInstance memberOf Sequence
     hasOrderedElement hasValue {OrderedElement_1,
                                 OrderedElement_2,
                                 OrderedElement_3,
                                 OrderedElement_4 }

instance OrderedElement_1 memberOf OrderedElement
     hasOrder hasValue 1
     hasElement hasValue StartEventInstance

instance OrderedElement_2 memberOf OrderedElement
     hasOrder hasValue 2
     hasElement hasValue GoalTaskInstance_1
```

```
instance OrderedElement_3 memberOf OrderedElement
     hasOrder hasValue 3
     hasElement hasValue ExclusiveChoiceMergeInstance

instance OrderedElement_4 memberOf OrderedElement
     hasOrder hasValue 4
     hasElement hasValue EndEventInstance

//
instance StartEventInstance memberOf StartEvent
     hasName hasValue "Start"

instance GoalTaskInstance_1 memberOf GoalTask
     hasName hasValue "Task1"
     hasWSMOGoal hasValue "http://example.org/Goal1"

instance ExclusiveChoiceMergeInstance memberOf ExclusiveChoiceMerge
     hasConditionalBranch hasValue {ConditionalBranch_2a,
                                    ConditionalBranch_2b }

instance ConditionalBranch_2a memberOf ConditionalBranch
     hasBranch hasValue GoalTaskInstance_2a
     defined_By hasValue "WSMO ConditionExpression 2a"

instance ConditionalBranch_2b memberOf ConditionalBranch
     hasBranch hasValue GoalTaskInstance_2b
     defined_By hasValue "WSMO ConditionExpression 2b"


instance GoalTaskInstance_2a memberOf GoalTask
     hasName hasValue "Task2a"
     hasWSMOGoal hasValue "http://example.org/Goal2a"

instance GoalTaskInstance_2b memberOf GoalTask
     hasName hasValue "Task2b"
     hasWSMOGoal hasValue "http://example.org/Goal2b"

instance EndEventInstance memberOf EndEvent
     hasName hasValue "End"
```

Listing 14 Example BPMO Process

## 3.3 Technical Aspects

The introduced conceptual aspects of semantics need to be supported by software. This affects the creation of semantic documents as well as the actual execution of semantic Web Services. These technical aspects are explained in the following section.

### 3.3.1 WSMO Studio

As there are a lot of artefacts to be created, a modelling tool called WSMO Studio[19] was developed. This tool supports the creation of the necessary ontologies, Web Services, and Goals. This is particularly necessary as the writing of the textual WSML files would be very uncomfortable and time consuming. WSMO Studio therefore supports the modeller by providing adapted input forms and a text editor with auto completion and syntax highlighting. The tool itself is based on Eclipse[20] IDE. A screenshot of WSMO Studio is shown in figure 19.

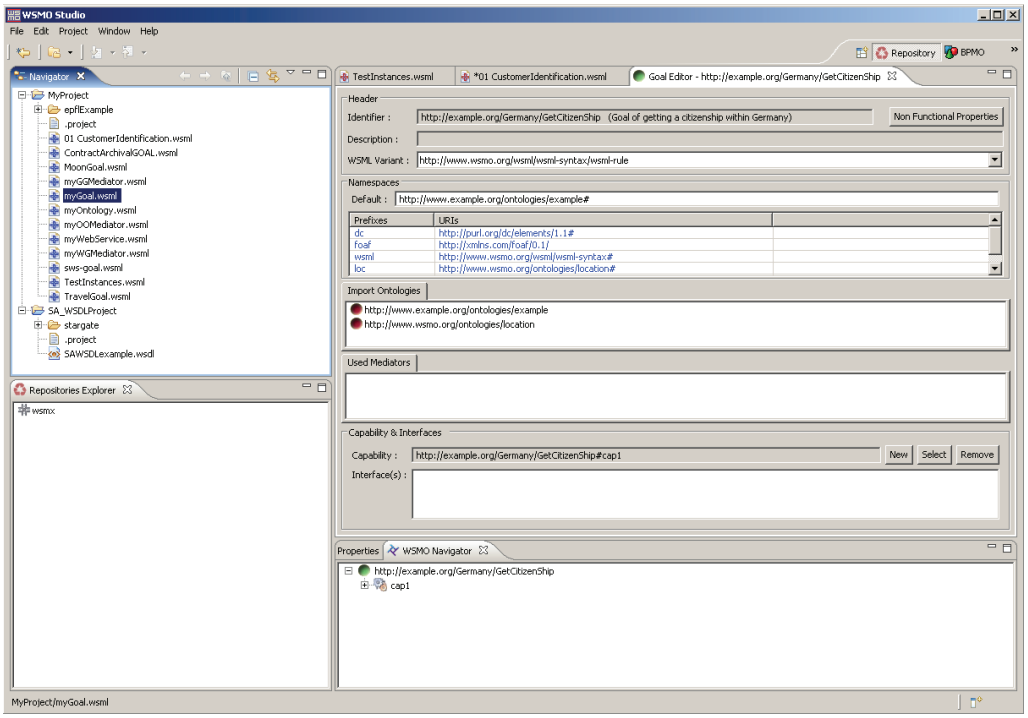



Figure 19 WSMO Studio

### 3.3.2 Web Service Execution Environment (WSMX)

WSMX is the official reference implementation for WSMO. It provides an execution environment for Semantic Web Services described in chapter 3.2.2 and to realize the reference relationships of figure 15. Therefore, it realizes algorithms for:

19 http://www.wsmostudio.org/

20 http://www.eclipse.org/

- Semantic Service Discovery
- Semantic Service Selection
- Semantic Service Invocation

#### 3.3.2.1 Semantic Service Discovery

Semantic service discovery is used for finding an appropriate WSMO Web Service for a given WSMO Goal. There are three different strategies implemented to realize service discovery (refer to [KLP+04]). There is one strategy that uses simple string matching algorithms to discovery appropriate Web Services (*keyword-based discovery)* and two strategies that use predicate logic for discovery (*discovery based on semantic descriptions).* Generally, WSMX is starting with keyword based discovery and is then applying the other strategies on the results as discovery based on semantic descriptions is very time consuming.

#### 3.3.2.2 Semantic Service Selection

After discovering eligible WSMO Web Services, semantic service selection takes care of selecting one. There are also different approaches available. A common one is simply selecting the first discovered Web Service. Another strategy could be the inclusion of non-functional properties of the Web Services in the selection process. Those non-functional properties can be administered in the Goal description as well as in the semantic Web Service description. A possible scenario could be found in the VoIP showcase process. When the hardware and the contract have to be shipped by a courier service, non-functional properties like price or reliability of the courier could be adducted. If more then one courier Web Service was found, the one with the lowest price and the highest reliability would be chosen.

#### 3.3.2.3 Semantic Service Invocation

Having selected a WSMO Web Service description, depending on the usage of WSMX this Web Service is also invoked by means of semantic instances that serve as input parameters. For this purpose it uses the provided grounding information included the `interface` section of the semantic Web Service description (see WSMO Web Service in listing 9 of chapter 3.2.2). The grounding information is usually attached to the input concepts with the key word `withGrounding` (see listing 15):

```
in concept tp#CustomerIdentificationRequest withGrounding
{_"http://localhost:8080/axis2/services/CRMService?wsdl
#wsdl.interfaceMessageReference(CRMService/identifyCustomer/in0)"}
```

Listing 15 Grounding Information

Here, the first part of the string (until the `#`) points to the WSDL location of the concrete Web Service (`http://localhost:8080/axis2/CRMService?wsdl`).

The second part refers to the portType (`CustomerIdentification`) and its concrete operation to invoke (`identifyCustomer`) as well as to the parameter of the operation (`in0` specifies the first input parameter).

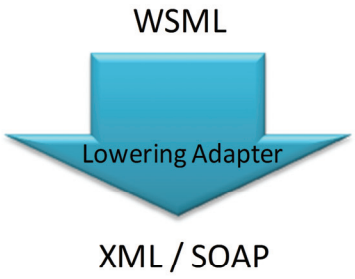
In order to enable semantic invocation, the semantic input data has to be transformed into a XML format, which is processable by the Web Service. In WSMX this is accomplished by so called *Lowering Adapters*. As WSMX has still alpha status, at the moment these lowering adapters are realized by hand through Java code, which have to be manufactured and hard coded for every Web Service in WSMX. Later versions of WSMX will probably provide a more comfortable and tool supported way to generate these adapters.

```
ontology CustomerIdentificationInput

   importsOntology  _"http://org.ipsuper.composition.tp/tpOntology"

   instance customerIdentificationRequest memberOf tp#CustomerIdentificationRequest
      tp#customer hasValue customerInstance

   instance customerInstance memberOf tp#TP_Customer
      tp#hasName hasValue "Laurie"
```




```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <ns:identifyCustomer xmlns:ns="http://crm.portal.telekomunikacja.pl/>
         <ns:return type="pl.telekomunikacja.portal.types.Customer">
            <ns:lastName>Laurie</ns:lastName>
         </ns:return>
      </ns:identifyCustomer>
   </soapenv:Body>
</soapenv:Envelope>
```

Listing 16 Lowering Adapter

The task accomplished by the lowering adapter of Web Service `CustomerIdentification` is shown in listing 16.

After the invocation of the concrete Web Service operation, the result has to be transformed the opposite way into its semantic representation. This is realized in WSMX by means of *Lifting Adapters*. These adapters are usually

realized using XSLT [Cla99], as it is the most convenient way and the result is in a XML format. Alternatively, again a Java based solution is possible.

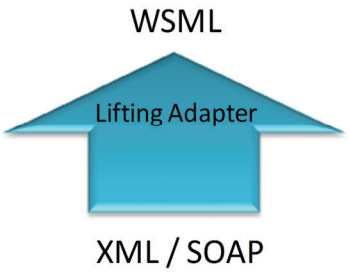
The work done by lifting adapters is exemplified in listing 17, which shows the transformation of the response of the invocation of operation `identifyCustomer` into the output ontology.

```
ontology CustomerIdentificationOutput

   importsOntology  _"http://org.ipsuper.composition.tp/tpOntology"

   instance customerIdentificationResponse memberOf tp#CustomerIdentificationResponse
      tp#customer hasValue customerInstance

   instance customerInstance memberOf tp#TP_Customer
      tp#hasName hasValue "Laurie"
      tp#hasCustomerID hasValue 151215
```




```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <ns:identifyCustomer xmlns:ns="http://crm.portal.telekomunikacja.pl/>
         <ns:return type="pl.telekomunikacja.portal.types.Customer">
            <ns:lastName>Laurie</ns:lastName>
            <ns:id>151215</ns:id>
         </ns:return>
      </ns:identifyCustomer>
   </soapenv:Body>
</soapenv:Envelope>
```

Listing 17 Lifting Adapter

### 3.3.3 SUPER Execution Environment

The software architecture of SUPER is shown in figure 20. The central component is the *Semantic Service Bus* (SSB), which serves as a backbone for the resident components.

Connected components are for example essential services that are responsible for transformation or composition of process (see *SUPER Platform Services* in figure 20) as well as components for modelling and monitoring processes (refer to *SUPER Tooling* in figure 20). Furthermore, repositories for storing semantic documents like semantic Web Service descriptions and

semantic process descriptions are attached to the SSB (*SUPER Repositories*) as well as components responsible for execution (*SUPER Execution*).

These execution components can be divided into two levels: the Web Service level covered by *Semantic Execution Engine* (SEE) and the process level covered by *Semantic BPEL Execution Engine* (SBPELEE). In terms of the SEE, SUPER uses already available semantic execution engines like WMSX (refer to chapter 3.3.2).

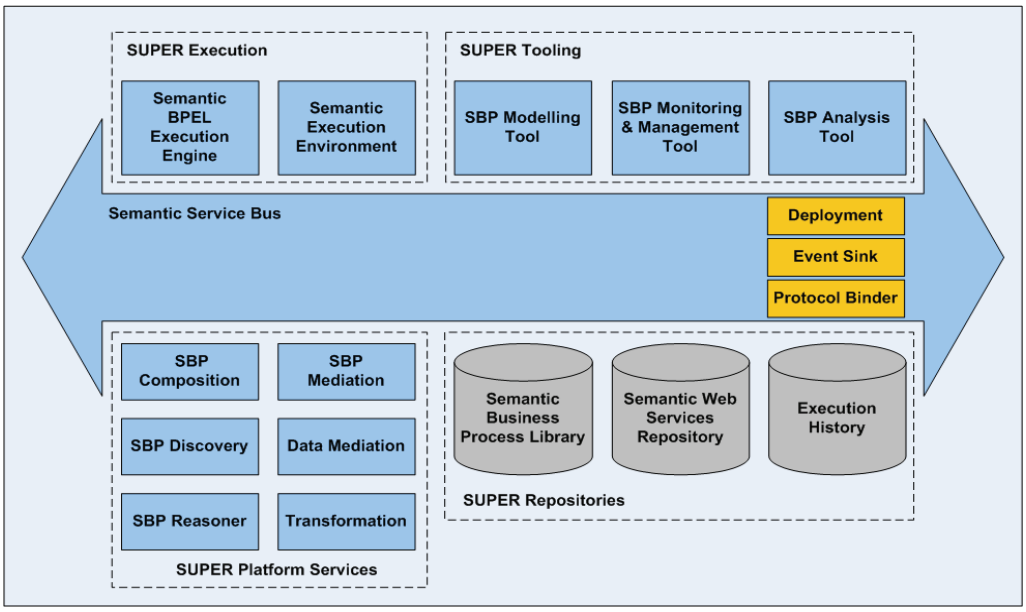



Figure 20 Architecture of SUPER [BCD+07]

The SBPELEE is being developed in SUPER and based on Apache ODE[21], an orchestration engine which is able to process BPEL code. SBPELEE is an extension of ODE by means of the processing of BPEL4SWS (*BPEL for Semantic Web Services*). BPEL4SWS itself is the XML representation of sBPEL and therefore equivalent through a provided automatic transformation. It can be summarized that SUPER supports only execution of sBPEL processes, which means that it must provide a transformation from the common and intermediary format BPMO into sBPEL.

This transformation will be available by the final delivery of SUPER so it can be assumed for the rest of the thesis that BPMO processes are deployable and executable on the SUPER SBPELEE (via transformation from BPMO over sBPEL into BPEL4SWS).

[21] http://ode.apache.org/

## 3.4 Problems Addressed by the Thesis

The shortcomings mentioned in chapter 3.1 are not covered entirely in this work. The focus of this thesis lies on the top-down problem described in chapter 3.1.1, whereas the bottom-up problem of chapter 3.1.2 is not explicitly addressed.

One main topic of this thesis is to enable the usage of semantic technologies in the area of EPC modelling. Although SUPER basically supports EPCs through the sEPC ontology it does not make suggestions how to embed semantic annotations in EPC modelling. The thesis therefore should analyze how and which semantic annotations can be included in order to make use of the promised benefits of semantics.

The focus hereby lies on the top-down approach eliminating manual work and improving quality in process lifecycle. This includes on the one hand the modelling phase in terms of assigning services to the EPC functions. Another important point is the creation of executable code out of this semantically enriched process, which helps to bridge the gap between business model and implementation.

The straightforward solution would be entirely building upon the SUPER infrastructure. But although SUPER provides a complete execution environment for semantic business processes as well as the possibility to make semantic queries to process space there are a few drawbacks of this solution.

SUPER is at the moment in an early research phase and there is currently no available software infrastructure. There is currently neither a ready SBPELEE nor a transformation of BPMO into sBPEL that generates the executable process code. In the final release however the engine and the sBPEL transformation will be available. In order to support the SUPER platform, the transformation produces BPMO code.

The major drawback of SUPER is its enormous complexity. On the one hand it provides a complete solution for SBPM on the other hand it introduces a lot of new ontologies, technologies and software systems. This needs heavy investment into employee training and remodelling existing business processes and semantic descriptions of existing Web Services. Furthermore, existing orchestration engines are not applicable any more as a special one is needed (SBPELEE). This leads to the problem that there is no possibility for companies to accomplish a simple migration from their old process infrastructure including the used modelling tools as well as the process and service engines to the environment envisioned in SUPER. Although this approach faces no problem for companies that may start from scratch, this is especially important as companies made heavy investments into their process environment. This includes the acquired employee competence in this

area, the running software systems as well as the present process models and implementations. All these assets might more or less get lost during migration.

Therefore, a more pragmatic solution has to be found that enables a step by step migration from conventional to semantic processes and supports as far as possible available software, process models and the method to create them.

# 4 Solution

The following chapter presents the actual solution of the substantiated problems of chapter 3.4. To make the solution more comprehensible, this chapter first specifies the general description of the solution followed by a detailed presentation.

Afterwards, the solution idea is described. Therefore, it is shown first which requirements and constrains the presented solution has to comply with. The following sections show in detail of which parts the whole solution consists.

The chapter is closed by a summary of the benefits that result from the solution.

## 4.1 General Description

During the thesis a method was developed to apply semantic technologies in context of business process modelling with help of EPCs. The method makes it possible to enrich semantic annotations to the process model and to make use of some of the promised benefits of semantics. These annotations affect especially the assignment of semantic Web Services to the functions of the EPC process.

Furthermore, the presented solution addresses Business Process Automation and actually tries to overcome the Business-IT Divide [SF03] by enabling business analysts to model the business process in a form that enables the automatic transformation of it into an executable representation. This representation does not need to be refined anymore and is directly deployable in the execution environment.

The solution is integrated in the SUPER architecture, by providing a transformation of the enriched EPC process into BPMO.

Another major focus of the solution is the development of another execution platform than SUPER. For the reasons explained in chapter 3.4 this alternative execution environment is introduced to avoid the additional complexity of SUPER. The support of this environment also requires another transformation to transfer the enriched EPC in an understandable format.

The developed solution is finally evaluated in terms of applicability and to verify the highlighted benefits (see chapter 5).

## 4.2 Solution Idea

This section shows the requirements and constraints the solution obeys. Furthermore, the big picture of the concrete solutions is presented as well as the composition of the different parts.

### 4.2.1 Requirements for the Approach

The developed solution has to comply with the following requirements and constraints, which where partially predetermined and partially added to cope with the enormous complexity of the solution space.

#### 4.2.1.1 Method

A method was developed with respect to the following constrains:

1. The method has to be based on the standard approach of chapter 2.3.3.

2. The business model of the process shall be provided in EPC.

3. The method must contain the possibility to add semantic information to the process model.

4. The business analyst must be able to model the process as far as possible. This means the modelled processes should require as little manual work as possible to be executable in the execution environment.

5. The method should support the idea of dynamic binding of the respective services during runtime.

6. It should be possible to consider non-functional properties of the services during this dynamic binding.

The provided solution makes use of miscellaneous applications and standards that enable the process modelling as well as the process execution, for which there are also several constraints:

#### 4.2.1.2 Modelling Tool

1. As modelling tool, ARIS SOA Architect 7.02 shall be used. It has to be extended to cover the developed method.

2. The tool must provide transformations that transfer the semantically enriched business process into the supported execution environments.

#### 4.2.1.3 Execution Environments

1. The semantic enriched process must be usable by the SUPER architecture.

2. Additionally, an alternative execution environment should be developed.

3. This alternative execution environment should provide a more pragmatic solution to enable the execution of semantic processes. This solution shall not make a rigorous change in available technology stack but extend it instead.

4. Is shall be assumed that the available technology stack is made up of a standard application server that can deal with Web Services (see chapter 2.2.2.1) as well as an orchestration engine that enables the execution of BPEL processes (see chapter 2.2.2.2).

5. The extension of the technology stack should only include WSMX

#### 4.2.1.4 Used Standards

1. As ontology framework WSMO shall be used. WSMO provides WSML as language to describe the semantic documents. Furthermore, WSMX has to be used as semantic execution engine.

2. On technical level, the common standards WSDL for Web Services and BPEL on process level shall be applied.

## 4.3 Concrete Solution

The concrete solution consists of several parts:

- **Method**

In order to benefit from semantics, a method has to be developed, how semantic information are used in the functional modelling process. The method must include a step to add semantic annotations to the business process model.

- **Adaptation of ARIS SOA Architect**

Based on the developed method, ARIS SOA Architect as the used modelling tool has to be adapted. This extension of ARIS SOA Architect must enable a business analyst to semantically annotate a business process model based on the EPC notation.

- **Set up of an Alternative Environment**

A conventional SOA environment with a Web Service execution engine and an orchestration engine cannot handle semantic annotations by default. Therefore, it is necessary to extend this environment to support semantics. The central concept that enables the usage of semantic processes in such a conventional infrastructure is the *Semantic Invocation Service (SISi)*. SISi can be seen as a proxy Web Service, which takes the semantic information and passes them to a semantic execution engine.

- **Transformations**

According to the requirements, two environments have to be supported. That is SUPER on the one hand and also the alternative environment for a more pragmatic solution on the other.

Therefore, two transformations have to be developed, which export the process model into the suitable format for the particular environment. The two transformation approaches are shown in figure 21.

*Transformation 1* transfers the annotated EPC process into BPMO to support SUPER. *Transformation 2* produces standard BPEL code, which can be deployed on every standard orchestration engine. This BPEL code makes use of SISi from alternative execution environment.

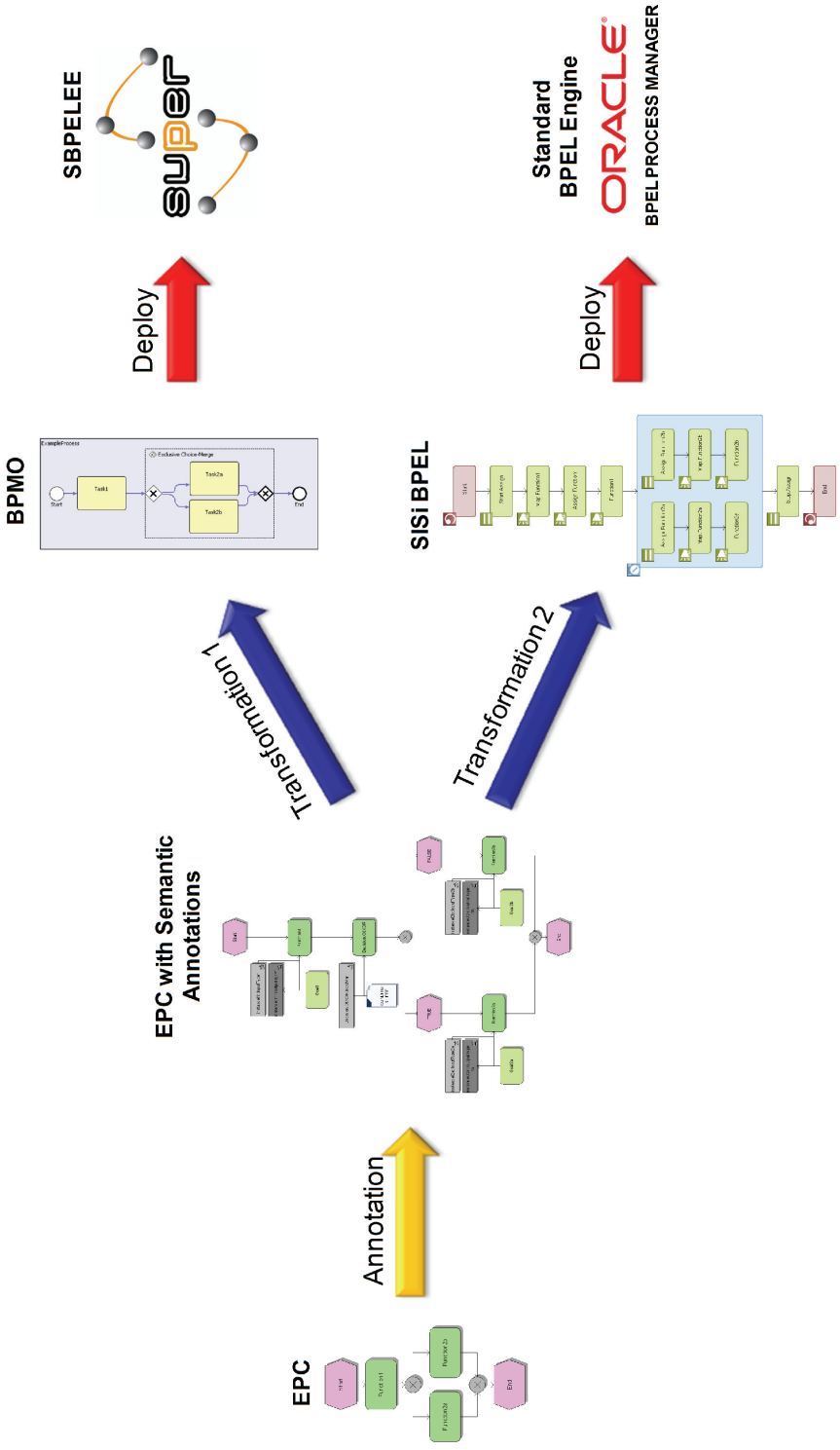



Figure 21 Overall Approach

## 4.4 Semantic Method

The following section explains the developed method for applying semantic technologies in business process modelling in detail.

The developed method consists of two phases. The first phase must take place before the actual process modelling can start.

### 4.4.1 Initial Preparation

Before the actual process can be modelled, the basis for semantic enrichment must be created. Figure 22 visualizes this procedure consisting of two steps.

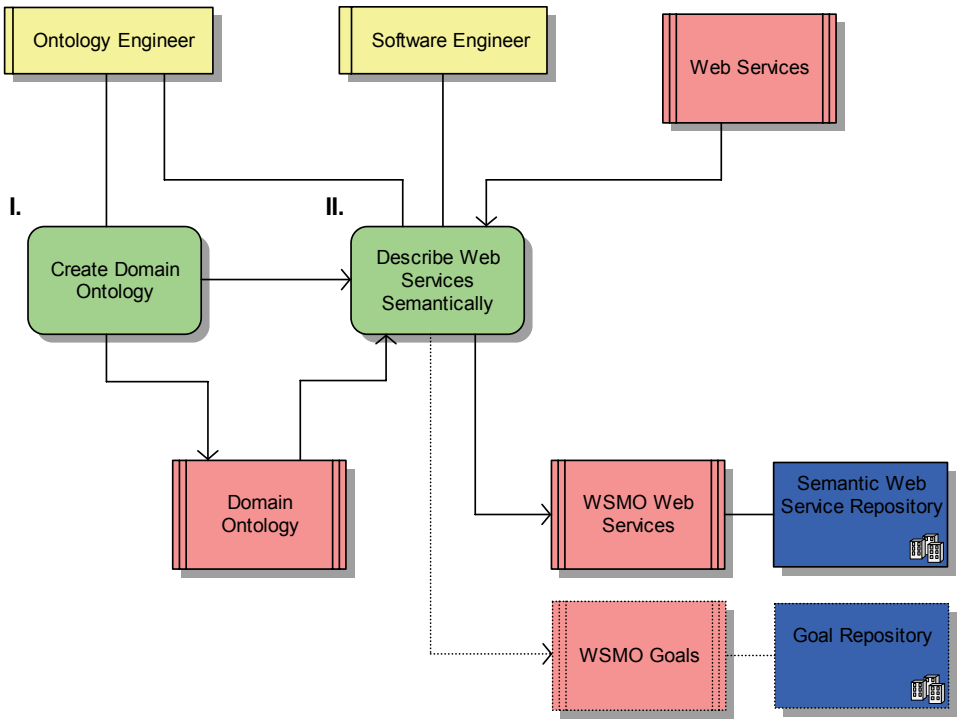



Figure 22 Semantic Method: Initial Preparation

#### I. Create Domain Ontology

The domain ontology is the foundation of semantic enrichment of the process models. It provides the unified vocabulary. The creation of the ontology is usually carried out by ontology engineers, which are experts of the targeted domain. Different methods exist to support the ontology engineering step [GPCFL04]. Result of the creation process is a domain ontology. This step

only has to be accomplished once, but is usually very complex as it is very difficult to capture all relevant aspects of the domain. Nevertheless, it can be necessary from time to time to adapt the ontology, for example due to changes in domain. Responsible for this step is the ontology engineer which is consulting various domain experts that have a deep knowledge in the domain.

Goal: Creating the domain ontology that serves as foundation for all other semantic descriptions like WSMO Web Services or WSMO Goals.

Input: Descriptions of the domain, views/notions of domain experts

Output: Domain ontology

Roles: Domain experts, ontology engineer

### II. Describe Web Services Semantically

After creation of the domain ontology, the existing (syntactic) Web Services can be described semantically. This is accomplished by WSMO Web Service descriptions, in which the functionality of the considered Web Service is specified by means of preconditions and postconditions (compare to chapter 3.2.2). The resulting document is stored in a WSMO Web Service repository. Note that the repository only contains the semantic description of the Web Service, not the implementation itself. The actual Web Services remain on application servers. This step has to be repeated whenever new Web Services are implemented or changes on existing Web Services are done. Optionally to the creation of the WSMO Web Services also the corresponding WSMO Goals can be already created. These Goals are stored in Goal repository. The business analysts later assign appropriate services to the EPC functions on basis of these Goals (see step 2a of chapter 4.4.2). Both WSMO Web Services and Goals are created by the ontology engineer who has expertise in domain ontology and the software engineer of the (syntactic) Web Service who brings along knowledge of its exact capabilities.

Goal: Provide semantic descriptions for the available Web Services.

Input: Domain ontology, actual Web Services, (informal) descriptions of the Web Services

Output: Semantic WSMO Web Service descriptions and WSMO Goals

Roles: Ontology engineer, software engineer

### 4.4.2 Modelling the Business Process

After the domain ontology, the WSMO Web Services and WSMO Goals are available the actual process modelling can take place. The full procedure is shown in figure 23:

#### 1. Model Business Process

The business analyst models the EPC of the business process as usual and analogue to chapter 2.3.3. In this step, the process only contains the pure functional business aspects.

Goal: The functional business aspects of the process are being modelled.

Input: General description of the process

Output: Abstract model of the process represented as EPC. This model only consists of events, functions and operators

Roles: Business analyst

#### 2. Semantically Annotate Business Process

This step contains the core idea of the new method. Here, the semantic annotations are added to the business process model. This annotation actually consists of three steps:

#### 2a. Add Goals

After creating the business process, it is enriched with the WSMO Goal descriptions taken from the Goal repository. Therefore, the business analyst selects an appropriate WSMO Goal for each EPC function. Hereby, the business analyst selects the Goal on basis of:

- The pre- and postconditions of the Goal. These conditions have to be compared with the desired intention of the corresponding EPC function.
- The input and output instances consumed/produced by the Goal. This information is especially helpful, when considering the output and input instances of the preceding and succeeding functions if these are already annotated.

If no matching Goal was found by the business analyst, there are different possibilities:

- Maybe a new Web Service implementation has to be created. In this case both WSMO Web Service and WSMO Goal have to be created, which can happen independent of the actual Web Service implementation. If the WSMO Goal is ready, the process modelling can be continued.
- Assuming there is a Web Service implementation including the corresponding WSMO Web Service but no corresponding WSMO Goal, and then only a corresponding WSMO has to be created.
- It could be that a Goal matches in general but not exactly to the need of the function. For instance if there is the need to buy a train ticket but a WSMO Goal is only available for buying tickets in general. This Goal then has to be refined in terms of modifying the pre- and postconditions.

It is intended that business analysts are able to select appropriate Goals on their own. During the selection of the Goal it is likely that the business analyst has to look at the domain ontology as it describes the concepts of the domain and relationships among them. Depending on the experience of the business analyst it can even be necessary to consult the ontology engineer, who brings expertise in semantic descriptions. The consultancy of the ontology engineer can be inevitable especially if new WSMO Goals and Web Services have to be created.

| | |
|---|---|
| Goal: | Select for each function of the EPC a matching WSMO Goal. |
| Input: | EPC process, WSMO Goals, domain ontology as implicit input |
| Output: | EPC process, which annotated functions |
| Roles: | Business analyst, possibly ontology engineer |

**2b. Complete Control Flow**

After the step 2a, the annotated EPC process is still missing control flow conditions of splitting XOR rules. The conditions are WSMO expressions which are specified by the business analyst. During the creation of this WSMO expression again the domain ontology can be useful or the ontology engineer can be consulted in case of doubt.

| | |
|---|---|
| Goal: | Specify for each splitting XOR rule of the EPC the corresponding decision conditions. |
| Input: | EPC process, WSMO condition expressions, domain ontology as implicit input |

Output: Annotated EPC process with all relevant condition expressions

Roles: Business analyst, possibly ontology engineer

### 2c. Model Data Flow

After the previous two steps, each function of the EPC is assigned to input and output instances. Furthermore, each splitting XOR rule has an input instance. In this step these instances have to be mapped on each other. To be more precisely, the output instances have to be mapped on the input instances of succeeding functions.

Goal: Find for each input instance a corresponding output instance in order to complete the data flow.

Input: EPC process, WSMO Goals

Output: Fully semantically annotated EPC process

Roles: Business analyst, possibly ontology engineer

### 3. Transform to Executable Process

After the EPC process was enriched with all necessary semantic annotations, the transformation can be triggered by the business analyst. Depending on the chosen options the transformation algorithm either produces BPMO or standard BPEL code. The latter code includes consequently calls of SISi.

Goal: Automatically transform the enriched EPC process into an executable representation.

Input: Fully semantically annotated EPC process

Output: BPMO process or standard BPEL process for alternative execution environment

Roles: Business analyst (only triggers transformation)

### 4. Finalise Executable Process

The optimal situation would be that this step can be completely omitted. That would actually meet the vision of eliminating the Business-IT Divide of [SF03]. Nevertheless, it may be possible that the transformed process must

be refined though additional information, for instance concerning the exception handling.

| | |
|---|---|
| Goal: | Add necessary and yet missing implementation specific details in the process. |
| Input: | Uncompleted BPMO or SISi BPEL process code |
| Output: | Completed BPMO or SISi BPEL process code |
| Roles: | Integration engineer |

**5. Deploy Executable Process**

Depending on the chosen transformation, the resulting process can be either deployed into the SUPER environment or on the application server of the alternative execution environment. The process is then ready for execution. This step can be automated, for instance by execution of an Ant[22] script and needs only to be triggered.

| | |
|---|---|
| Goal: | Deploy the final process (BPMO or SISi BPEL) into the respective execution environment (SUPER or alternative execution environment). |
| Input: | Completed BPMO or SISi BPEL process implementation |
| Output: | Successfully deployed process, which can be used productively |
| Roles: | Integration engineer (just triggers the deployment) |

[22] Apache Ant: http://ant.apache.org/

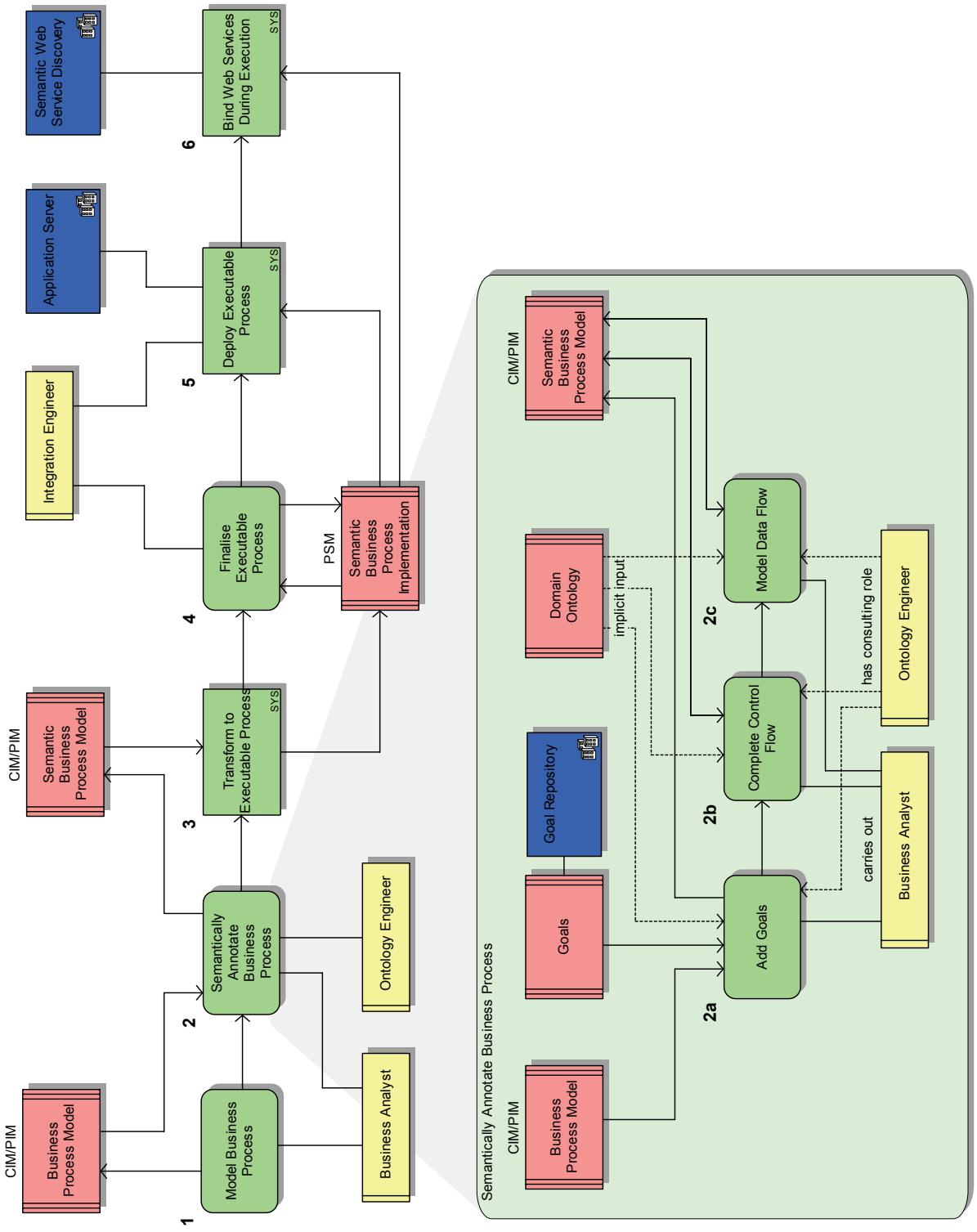



Figure 23 Semantic Method

## 4.5 Adaptation of ARIS SOA Architect

To provide the solution outlined above, it is necessary that the business analyst can enrich the EPC model of the business process with semantic annotations.

It is further necessary that the annotated EPC can be transformed and exported. As described above, there are overall two transformations to implement. That is one transformation which results in conventional BPEL code and one which leads to BPMO to support the SUPER infrastructure.

The following section shows how the mechanism to add semantic annotations was implemented as well how the two transformations were realized.

### 4.5.1 Semantic Annotations

The central artefact that contains the semantic information is the WSMO Goal. For this artefact there must be an appropriate representation in ARIS SOA Architect. Additionally, it must be ensured, that the business analyst can model the data flow and to complete the control flow.

For these tasks, the tool was extended by means of macros. Macros provide an easy way of extending functionality of SOA Architect on the basis of JavaScript code [Ecm99], which can be executed without changing the source code of SOA Architect itself. In essence there are:

#### 4.5.1.1 Add Goal Macro

This macro is executed by business analyst for each EPC function. It enables the selection of WSMO Goals and generates the necessary artefacts. This macro fulfils step 2a of the semantic method (see chapter 4.4). For this topic the business analyst selects the function of the EPC and runs the macro. In the appearing dialog (see figure 24), it is possible to look at the content of the Goal files and make an informed selection.

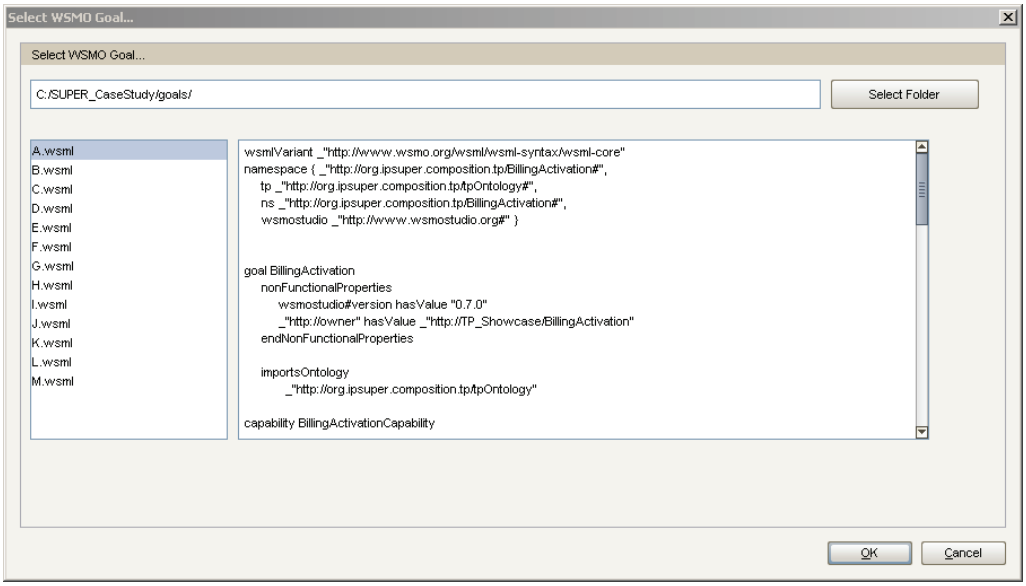



Figure 24 Goal Selection Dialog

The functionality of macro *"Add Goal"* is illustrated in figure 25. It shows the result of adding Goal "`Goal`" of listing 18 to the function. For each input and output variable of the Goal description a new "Technical Term" object is added to the model. The name of theses objects assembles through the name of the variable and the type separated by "::".

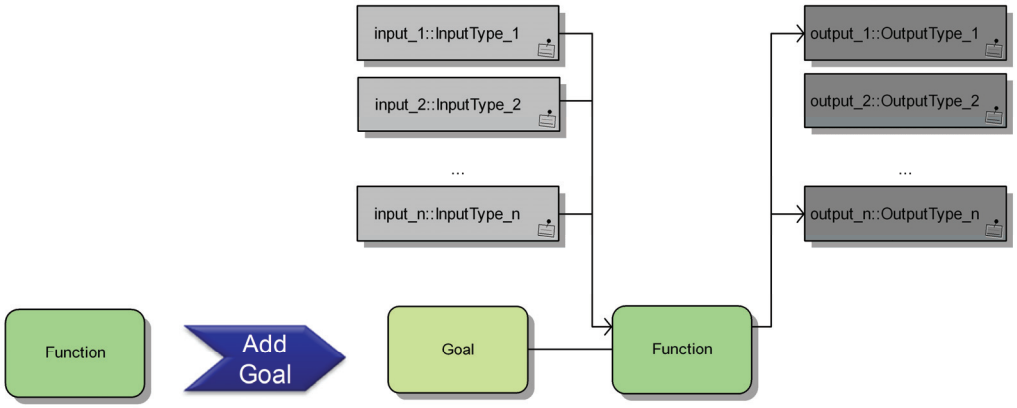



Figure 25 Add Goal

```
goal Goal
     importsOntology
            _"http://org.ipsuper.composition.tp/tpOntology"

capability GoalCapability
//... semantic description of the capability
```

```
interface GoalInterface
     choreography OrderCreationChoreography
     stateSignature OrderCreationStateSignature
          importsOntology
            _"http://org.ipsuper.composition.tp/tpOntology"

           in concept ns#GoalRequest
          out concept ns#GoalResponse

ontology GoalInput
     importsOntology
            _"http://org.ipsuper.composition.tp/tpOntology"

concept GoalRequest
     input_1 impliesType InputType_1
     input_2 impliesType InputType_2
     // ...
     input_n impliesType InputType_n

concept GoalResponse
     output_1 impliesType OutputType_1
     output_2 impliesType OutputType_2
     // ...
     output_n impliesType OutputType_n
```

Listing 18 Goal

#### 4.5.1.2 Map Instances Macro

The previous macro generates the data objects of the process. In order to model the data flow of the process, a mapping of the output instances onto the input instances must take place. This task is accomplished by macro *“Map Instances”* and directly corresponds to step 2c of the developed method in chapter 4.4. Figure 26 visualizes the functionality. To map output variable of a preceding function onto the input variable of the currently considered function, one has first to select these variables. Note that the dotted arrow indicates that the preceding function lies in the control flow somewhere before the current function but not necessarily directly before. Then, the macro is executed. The macro creates a new model (visible on the right of figure 26) in which the relationship of the variables is shown and further links this newly model with the input variable visible by the small symbol in the lower right of the variable (see red ellipse in figure 26).

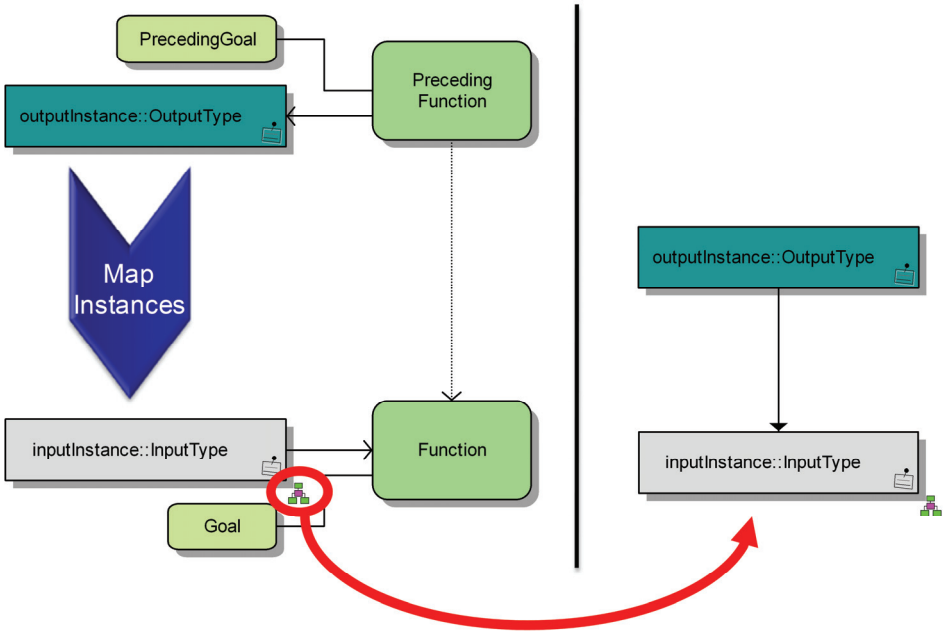



Figure 26 Map Instances

#### 4.5.1.3 Add Decision Macro

The EPC process typically includes XOR rules, which represent a split of the control flow depending of a prior decision. The condition of such a decision must be specified, which is covered by macro *"Add Decision"*. This macro is implementing step 2b of the developed method (see chapter 4.4).

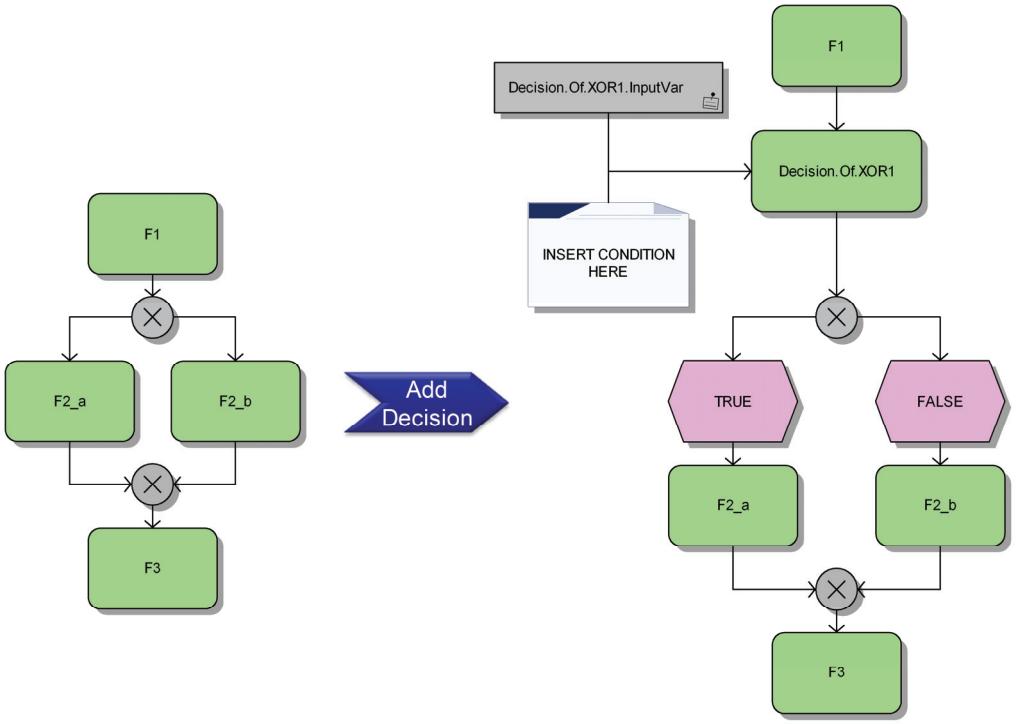



Figure 27 Add Decision

The modeller selects the (splitting) XOR rule and executes macro "Add Decision". The macro then creates new artefacts. A new function with name "Decision.Of.XOR" with one input variable and an information object is added. The information object serves as carrier for the condition expression. Initially, it contains only "INSERT CONDITION HERE", which must be replaced by the actual condition expression. The expression itself must be a valid WSMO expression, for instance "`technicalVerificationResult hasValue true`". The input variable has to be assigned via macro *"Map Instances"*. At runtime, the entered condition is verified against the value of this variable. Additionally to these two objects, the macro also inserts two events "TRUE" and "FALSE". When the condition holds at runtime, the path with the "TRUE" event is being executed otherwise the path with the "FALSE" event.

Note that this procedure currently only enables the execution of binary decisions, as the present work only represents a prototypic implementation and

the used showcase process can be covered with it. Although principally all kinds of decision could be covered by nested application of binary decisions, a general usage would require an extension towards arbitrary decisions for reasons of modelling convenience.

#### 4.5.1.4 Miscellaneous Macros

Beside the three main macros above, some additional macros where implemented:

- Macro **"Set as Process Input Instance"**. This macro is executed on an input instance of the EPC process. The macro declares the selected input instance to be the input parameter for the process. This means that the specified instance has to be passed by the caller if the process is invoked. This macro is part of step 2c ("Model Data Flow") of the semantic method (refer to chapter 4.4.2).

- Macro **"Set as Process Output Instance"**. Analogue to macro *"Set as Process Input Instance"* this macro is executed on an output instance of the EPC process. This declares the chosen instance to be an output parameter of the process. The specified instance is then passed back to the caller after the process was executed. This macro is also part of step 2c of the developed method of chapter 4.4.2.

- Macro **"Transform Process"**. Here, the user can specify which of the two transformations should be executed. The user can either select *"into BPMO"* to transform process into its BPMO representation (transformation 1 of figure 21) or *"into SISi BPEL"* to transfer the process into standard BPEL code (transformation 2 of figure 21). Note, that this macro fulfils exactly the task of step 3 of the developed semantic method (refer to chapter 4.4.2).

- Macro **"Deploy Process"**. Depending on the previously performed transformation, the transferred process is being deployed on the underlying execution environment. Therefore, the macro produces the necessary documents and copies them into the environment. If SUPER is used as execution environment (transformation 1 of figure 21) then a text file with the BPMO process is being generated. If the alternative execution environment is used (transformation 2 of figure 21), the macro generates various files containing standard BPEL code and the WSDL interface of the process (do not confuse this interface with the WSDL interface of SISi in chapter 4.7.1.3). The WSDL interface represents the offered interface of the process used for invocation (refer to listing 30 for an example). This macro corresponds to step 5 of the developed method (see chapter 4.4.2).

## 4.6 Transformations

As explained before, there were two partially different transformations implemented. The first transformation supports the SUPER environment by producing a process representation in BPMO. The second transformation aims at creating standard BPEL code, which can be run on every standard BPEL execution engine. This transformation makes use of the *Semantic Invocation Service (SISi)*, which is characterized in chapter 4.7.

Figure 28 shows the two transformation approaches. Common to both approaches is the usage of the EPC2BPEL transformation (compare to chapter 2.3.2.1), to provide a BPEL process, which provides an initial framework of the control flow.

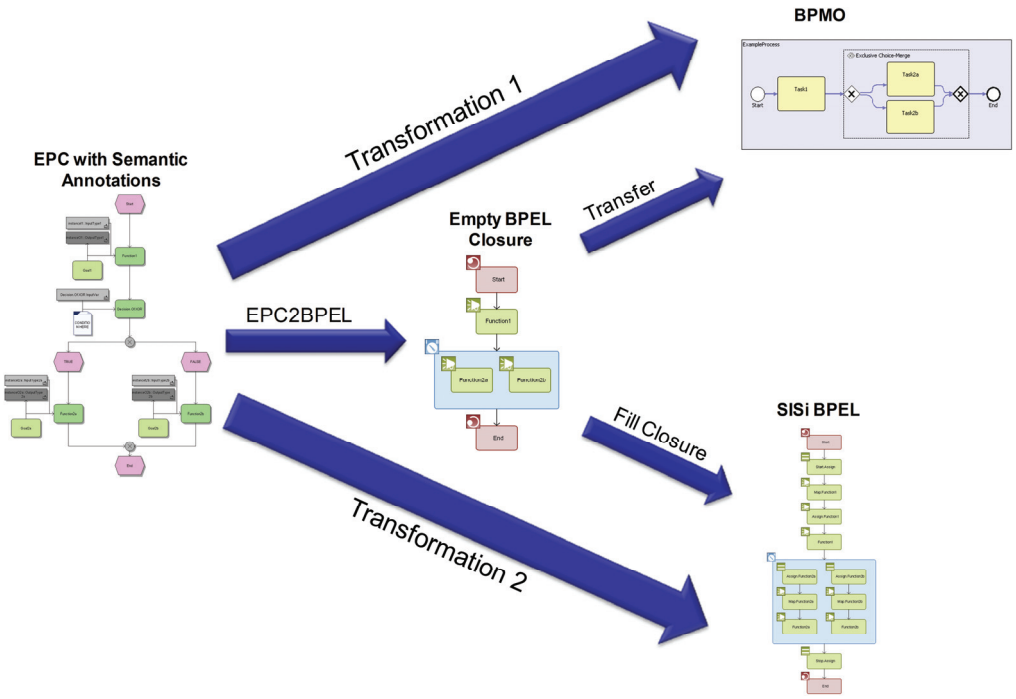



Figure 28 Transformation Approaches

### 4.6.1 Transformation into BPMO

As explained in chapter 3.2.3, BPMO contains graph oriented workflow patterns of EPC as well as block oriented patterns of BPEL. In theory the EPC process with annotations could directly be transferred into BPMO.

Nevertheless, the presented transformation here first transfers the EPC into BPEL (via EPC2BPEL) and then transfers the BPEL code into BPMO (compare to figure 29). The reason for this is that BPEL code is already nearer to execution and SUPER provides only a transformation of the block oriented patterns into the executable sBPEL format.

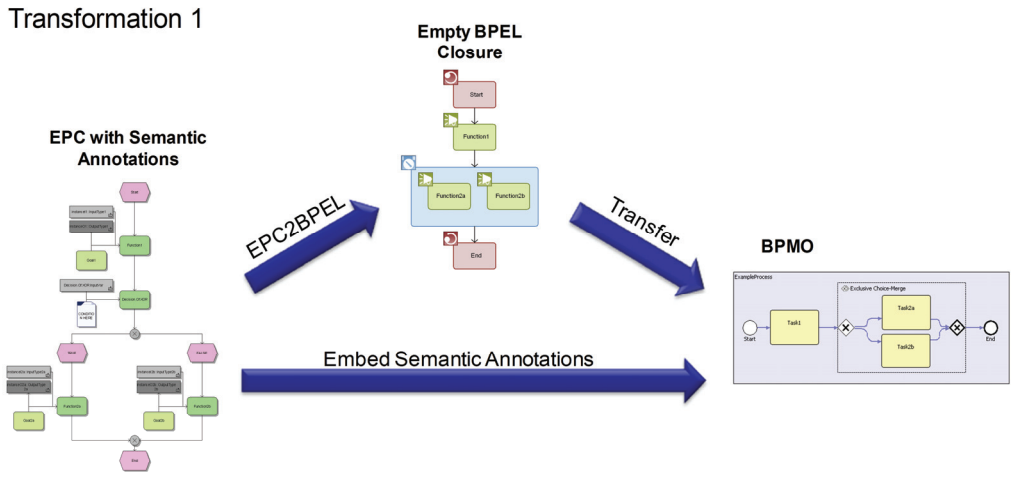



Figure 29 Details of Transformation 1

The transformation was realized using the scripting functionality of ARIS SOA Architect.

The actual transformation is very straightforward. The algorithm iterates over the activities of the BPEL process produced by EPC2BPEL. Depending on which activity is currently considered, there are now different cases. The algorithm is starting on the top level sequence of the BPEL process.

### 1. Sequence

For each sequence activity the transformation creates also a sequence instance in the BPMO process. The algorithm then iterates through the sequence and processes each activity until end.

### 2. Invoke

If an invoke activity is found, the algorithms creates a new BPMO goal task instance. As described in chapter 3.2.3, the attribute `hasWSMOGoal` of the goal task needs the URI of the annotated goal. This can be found in the corresponding function of the EPC process (compare to arrow “Embed Semantic Annotations” of figure 29).

### 3. Switch

If a switch activity is found the algorithm is creating a new Exclusive Choice-Merge instance is created in the BPMO process as well as the ConditionalBranch instances. Furthermore, the conditions have to be specified for these branches. As explained before, the developed method only supports binary decisions. The conditions of the branch instances are therefore restricted to the specified condition in the EPC for the TRUE branch and its negation for the FALSE branch.

Additionally, the algorithm is called recursively for each of the cases of the switch activity in order to transfer all activities to BPMO.

### 4. Flow

If the algorithm detects a flow activity it creates a new Parallel Split-Synchronize instance in the BPMO process and analogue to switch/Exclusive Choice-Merge also the ConditionalBranch instances are applied. But in difference to the last step there is no need to specify branch conditions.

Then, the algorithm is again called recursively for each nested sub activities of the flow.

The transformation finally results in a text file with the BPMO representation of the process analogue to listing 14.

### 4.6.2 Transformation into SISi BPEL

The implementation of the transformation was also achieved using the scripting functionality of ARIS SOA Architect. It further makes also use of the above introduced EPC2BPEL transformation (see chapter 2.3.2.1).

The transformation into SISi BPEL corresponds to the lower path of figure 28.

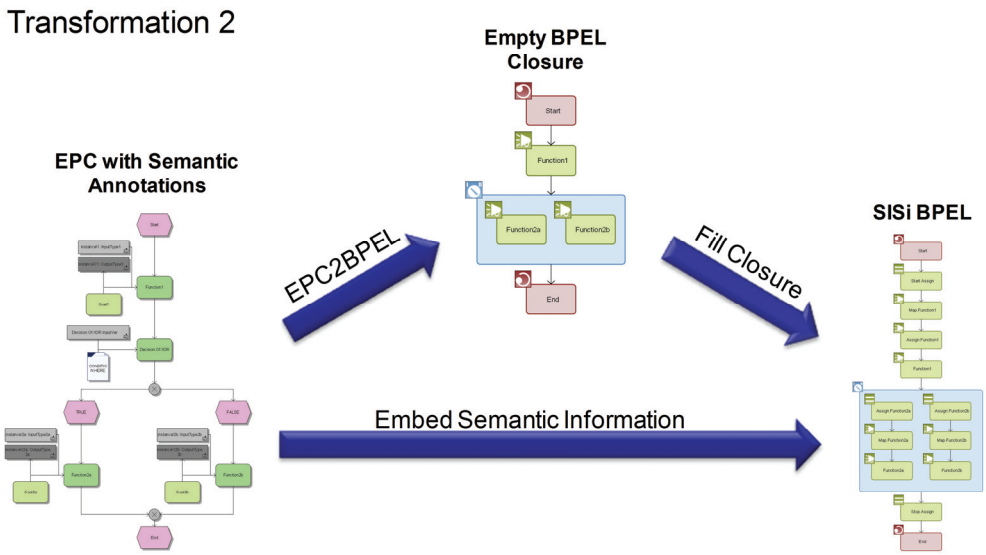



Figure 30 Details of Transformation 2

The annotated EPC serves as a starting point for the EPC2BPEL transformation. This generates an “empty BPEL closure” of the EPC process. In general it creates for each function an invoke activity and links these activities according to the EPC. Further, it generates for each XOR block and for each AND block a corresponding “switch” and “flow” activity in the BPEL process. Further details to EPC2BPEL are described in chapter 2.3.2.1.

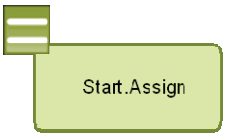
This empty closure of the BPEL process has to be extended by means of the semantic annotations of the enriched EPC process (see “Fill Closure” in figure 30). This information has to be transferred separately from EPC to BPEL (see bottom arrow of figure 30).

In contrast to the transformation into BPMO, the algorithm used here is more complex. The filling of the BPEL process closure is aligned onto the alternative execution environment developed in chapter 4.7.

#### 4.6.2.1 Goals

The Goal descriptions added to EPC functions have to be transferred to the BPEL process. This task is accomplished as follows:

For each function of the EPC process a global input variable is created in the BPEL process. At the beginning of the BPEL process a starting assign activity is created. This assign activity copies the Goal descriptions into the right part of the input variables (part `Goal`) of the corresponding function. In listing 19 the assign activity “Start.Assign” contains several copy activites. Here, the first copy activity assigns the textual content of the Goal (that is `Goal_1`) attached to function `Function_1` to the input variable of this function (`Function_1.InputVar`).




```
<assign name="Start.Assign">
       <copy>
              <from expression=...Content_Of_Goal_1.../>
              <to variable="Function_1.InputVar"
               part="Goal"/>
       </copy>
...
       <copy>
              <from expression=...Content_Of_Goal_n.../>
              <to variable="Function_n.InputVar"
               part="Goal"/>
       </copy>
</assign>
```

Listing 19 Transformation of Goals

#### 4.6.2.2 Functions with 1:1 Instance Mapping

For each function of the EPC process additional BPEL activities must be created. Figure 31 visualizes the procedure by function “Function”. Note that this procedure only holds for functions whose input instances are assigned to exactly one output instance. The next section shows the procedure if there is more than one output instance assigned to the input instance.

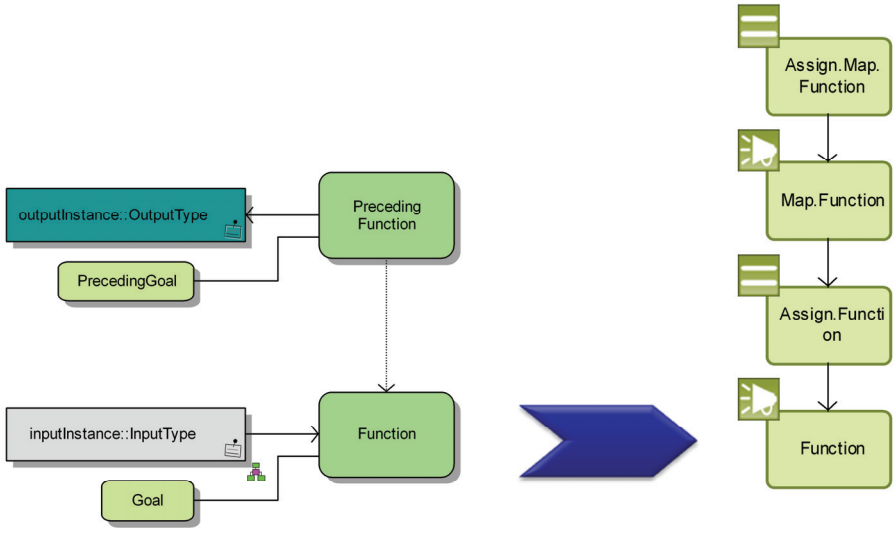



Figure 31 Transformation of Function with 1:1 Instance Mapping

The transformation generates for each function a total of four BPEL activities (see figure 31), two assign activities and two invoke activities.

The sequence of the first three activities fulfils the task of mapping the output instances of the preceding function onto the input instances of the current function. The invoke activity between the two assigns invokes the map operation of SISi. The detailed description of the functioning of this operation can be found in chapter 4.7.1.

The content of the first assign statement can be found in listing 20. It contains the necessary copy activities to map the single parts of the output message of the preceding function onto the right parts of the input message of the considered function.


```
<assign name="Assign.Map.Function">
       <copy>
              <from expression="PrecedingGoal"/>
              <to variable="Map.Function.InputVar"
               part="sourceGoal"/>
       </copy>
       <copy>
              <from expression="outputInstance"/>
              <to variable="Map.Function.InputVar"
               part="sourceInstance"/>
       </copy>
       <copy>
              <from expression="Goal"/>
              <to variable="Map.Function.InputVar"
               part="targetGoal"/>
       </copy>
       <copy>
              <from expression="inputInstance"/>
              <to variable="Map.Function.InputVar"
               part="targetInstance"/>
       </copy>
       <copy>
              <from variable="PrecedingFunction.OutputVar"
               part="result"/>
              <to variable="Map.Function.InputVar"
               part="sourceInput"/>
       </copy>
</assign>
```

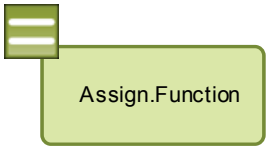
Listing 20 Assign.Map.Function

According to the WSML, the instances are represented during runtime as a textual document and passed as a string between the single activities. The starting document for the first assign activity is the message part `result` of output variable `PrecedingFunction.OutputVar` of the BPEL invoke activity `PrecedingFunction`. As not all instances in this variable are required and the instance names are not necessarily matching, the string is passed through operation “map”. This operation extracts the necessary instance and names it according to the requirements of the current function. In listing 20 the source instance has the name `outputInstance` and the target instance has name `inputInstance`. Additionally, the name of the WSMO Goal is passed, as it is needed to form a message which meets the defined naming conventions of the exchanged WMSL messages (for those conventions see chapter 4.7.2).


```
<invoke        xmlns:imp1="http://sisi.externalInterface/"
               name="Map.Function"
               inputVariable="Map.Function.InputVar"
               operation="map"
               outputVariable="Map.Function.OutputVar"
               partnerLink="SISiPL"
               portType="imp1:SISiPortType"/>
```

Listing 21 Map.Function

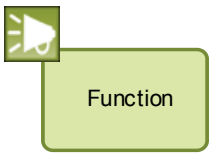
After filling the input message of the `map` operation, its invocation takes place. The corresponding BPEL code of this invoke activity is shown in listing 21. The output of the map is a message which contains the input instances of the function “Function”, which is correctly named according the naming conventions. This message now has to be assigned to the input variable of function “Function”. This is accomplished by listing 22.




```
<assign name="Assign.Function">
      <copy>
            <from variable="Map.Function.OutputVar"/>
            <to variable="Function.InputVar"/>
      </copy>
</assign>
```

Listing 22 Assign.Function

After the input instances are properly named and assigned to the input variable of the function, the actual invocation can take place. This is achieved by listing 23.




```
<invoke      xmlns:imp1="http://sisi.externalInterface/"
             name="Function"
             inputVariable="Function.InputVar"
             operation="invokeSemantically"
             outputVariable="Function.OutputVar"
             partnerLink="SISiPL"
             portType="imp1:SISiPortType"/>
```

Listing 23 Function

### 4.6.2.3 Functions with 2:1 Instance Mapping

If the input instance of the considered function is made up of two output instances, the transformation constructs a marginal different BPEL structure.

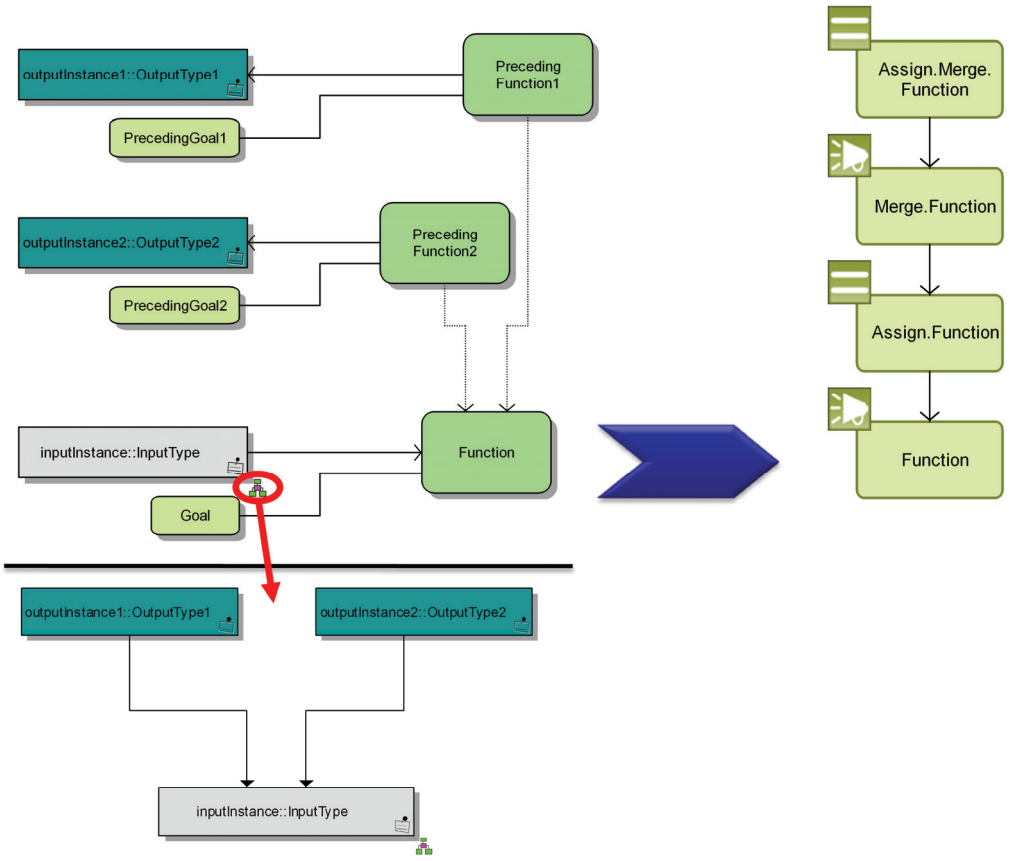



Figure 32 Transformation of Function with 2:1 Instance Mapping

Figure 32 visualizes such a situation where “outputInstance1” and “outputInstance2” are assigned onto “inputInstance” of function “Function”.

The main difference to the previous transformation of function lies in invocation of operation `merge` instead of `map` before the actual invocation of the function (see listing 24).


```
<invoke       xmlns:imp1="http://sisi.externalInterface/"
              name="Merge.Function"
              inputVariable="Merge.Function.InputVar"
              operation="merge"
              outputVariable="Merge.Function.OutputVar"
              partnerLink="SISiPL"
              portType="imp1:SISiPortType"/>
```

Listing 24 Merge.Function

As also the parameters of these operations vary, the two assign activities differ as well. The content of these activities can be derived from listing 20 and listing 22, these are omitted here.

#### 4.6.2.4 Decisions

The transformation further must transfer the splitting XOR decisions of the enriched EPC process. The principle functioning is depicted by figure 33. Starting point is the decision construct of the EPC process, which consists of a preceding decision function (Decision.Of.XOR1) with input variable (Decision.Of.XOR1.InputVar) and the condition expression as well as the actual XOR block in which the TRUE and FALSE paths are demarked by events. Transformation 2 has to insert various assign and copy activities as well as the invocation of the map operation of SISi analogue as during transformation of functions. Note that depending on the mapping of the input variable (Decision.Of.XOR1.InputVar) is may be also necessary to replace the map invocation by a merge invocation. Then, also the assign activities have to be changed which is along the lines of chapter 4.6.2.3.

The assign activity Assign.Map.Decision.Of.XOR1 and the invocation of Map.Decision.Of.XOR1 are again responsible for bringing the output instances of the preceding function into an input message conform to the naming conventions of chapter 4.7.2. Their content is therefore omitted as it is analogue to chapter 4.6.2.2.

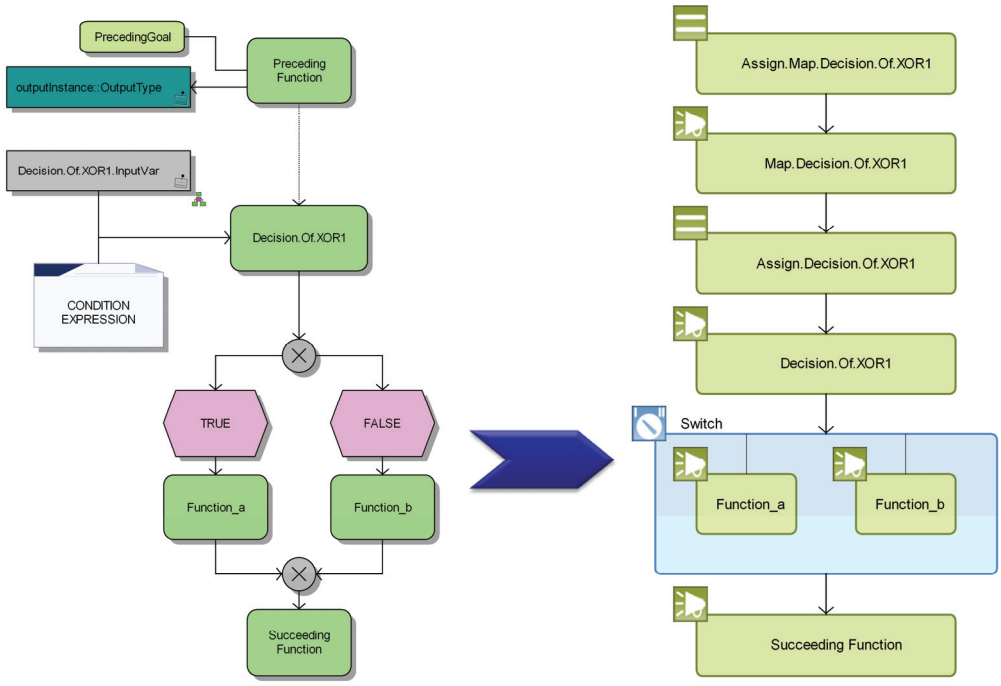
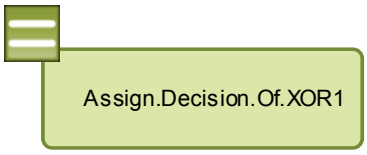



Figure 33 Transformation of Decision

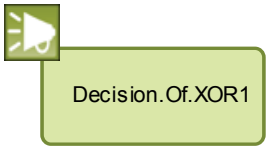
The content of Assign.Decision.Of.XOR1 can be found in listing 25. This activity copies the output of operation `map` as input message of the operation `decide`. Additionally, the actual condition expression ("CONDITION EXPRESSION" on the left of figure 33) is passed as parameter for operation `decide`.




```
<assign name="Assign.Decision.Of.XOR1">
	<copy>
		<from variable="Map.Decision.Of.XOR1.OutputVar"/>
		<to variable="Decision.Of.XOR1.InputVar"/>
	</copy>
	<copy>
		<from expression="CONDITION EXPRESSION"/>
		<to variable="Decision.Of.XOR1.InputVar"/>
	</copy>
</assign>
```

Listing 25 Assign.Decision.Of.XOR1

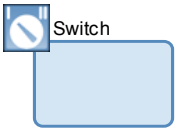
When the input message for invocation of operation decide is properly filled, the invoke activity of listing 26 takes place. The resulting message (`Decision.Of.XOR1.OutputVar`) contains the Boolean value of the decision.




```
<invoke       xmlns:imp1="http://sisi.externalInterface/"
              name="Decision.Of.XOR1"
              inputVariable="Decision.Of.XOR1.InputVar"
              operation="decide"
              outputVariable="Decision.Of.XOR1.OutputVar"
              partnerLink="SISiPL"
              portType="imp1:SISiPortType"/>
```

Listing 26 Decision.Of.XOR1

Depending on this returned value the corresponding case of the following switch activity is executed (see listing 27). Therefore, the case statements specify an XPath expression that compares the value of variable `Decision.Of.XOR1.OutputVar` to the Boolean value true or false. Only the nested code of the case with the fulfilled condition is being executed.




```
<switch name="XOR1">
       <case condition=
       "http://schemas.xmlsoap.org/ws/2003/03/business-process/
       :getVariableData('Decision.Of.XOR1.OutputVar') = true()">
              // further code of F2_a
       </case>
       <case condition=
       "http://schemas.xmlsoap.org/ws/2003/03/business-process/
       :getVariableData('Decision.Of.XOR1.OutputVar') = false()">
              // further code of F2_b
       </case>
</switch>
```

Listing 27 Switch

## 4.7 Alternative Execution Environment

To circumvent the complexity of the SUPER environment an alternative execution environment had to be developed. Goal was to give support of running semantic business processes on conventional orchestration engines, which can only handle standard BPEL code version 1.1. Thus, it is necessary to make the second transformation (refer to the lower path of figure 21) to produce standard BPEL code, but although consider the semantic information of the process. There were two possibilities identified to receive standard BPEL code out of the semantically annotated EPCs.

In the first case the code transformation resolves the semantic dependencies and produces standard BPEL code. To achieve this, the transformation has to use a semantic reasoner to discover the semantic Web Service description. This description is used to extract the grounding information of the assigned syntactic Web Service respectively the location of the WSDL file and the assigned operation. The gathered information is used to assemble the BPEL file.

A big drawback of this solution is the loss of run time binding of Web Services as the WSDL is directly referenced.

The remaining benefit of the annotated semantics lies therefore in Service Discovery that can be handled by business analysts. Positive to this approach is the run time speed as semantic discovery process has completely been carried out at design time.

The second approach keeps the semantic dependencies until runtime and preserves therefore late binding. Central idea hereby is a proxy Web Service that is invoked every time in BPEL code, when a semantic Web Service is requested. This proxy Web Service is called Semantic Invocation Service (SISi) and is described in detail in chapter 4.7.1.

Negative in this idea is the lower execution speed, as semantic Web Service discovery and selection process is started with every BPEL activity.

The first possibility provides due to the loss of late binding no good solution. This problem could be partially solved by triggering a new transformation and deployment of the EPC process if a new Web Service implementation is available. Nevertheless, also this solution is not appropriate for instance if services would change very often.

According to the loss of dynamic binding the best solution is the second idea with the described proxy Web Service, which therefore was implemented during this thesis[23].

### 4.7.1 Semantic Invocation Service (SISi)

The *Semantic Invocation Service* (SISi) provides the possibility to run semantic business processes on conventional orchestration engines as already available in environment of the enterprise. SISi serves as a proxy between the BPEL process and the Semantic Web Service engine.

#### 4.7.1.1 Runtime Invocation Procedure

There were two alternative scenarios identified for the application of SISi concerning the exchanged data and the underlying semantic software.

Depending on the used semantic software, one must make some difference. Semantic reasoners are used to identify semantic Web Services by means of the Goal description. They only cover semantic service discovery and probably semantic service selection but not semantic service invocation (compare to chapter 3.3.2). Semantic execution engines like WSMX however can handle all three mentioned tasks.

One has further to distinguish whether the BPEL process should handle semantic data or syntactic data. In the first case, SISi receives data in form of ontology instances through a textual representation. This data has to be first transformed into an appropriate syntactic representation in order to serve as parameter for the actual Web Service operation. In the second alternative, SISi receives directly syntactical data (that is for instance SOAP messages, see chapter 2.2.2.1).

Hereby, the semantic reasoner alternative is combined with the variant of syntactic input data, while the semantic execution engines are ideally composed with the use of semantic input data. The reason for that becomes clear in the following section.

Note that the current implementation of SISi only supports the combination semantic execution engine and semantic input data. The usage of semantic reasoners and syntactic data would further need modifications in the three macros and in the transformation. The possibility is illustrated for the sake of completeness.

[23] The source code of the Semantic Invocation Service (SISi) is hosted on Google Code and can be obtained under the following URL: http://code.google.com/p/semanticinvocationservice/

- **Usage of Semantic Reasoner**

If a semantic reasoner is used, SISi delegates the semantic descriptions to the reasoner for the discovery process of the Web Service. The invocation of the resulting Web Services is then completely handled by SISi. The exact procedure is shown in figure 34.

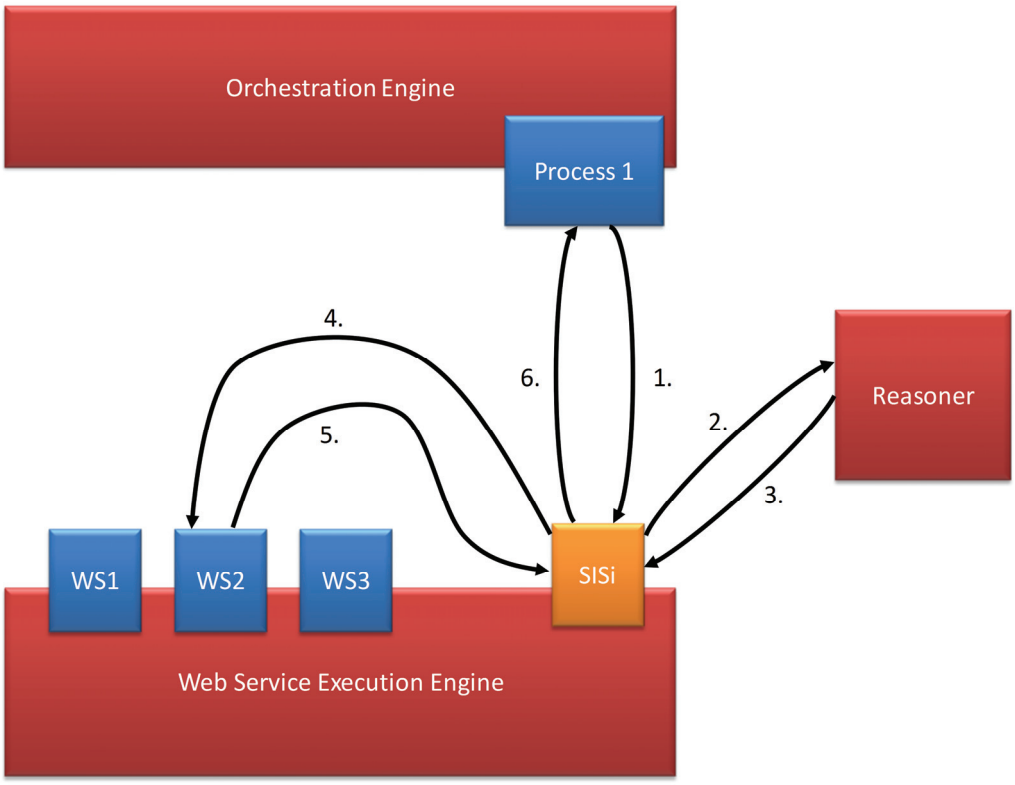



Figure 34 SISi: Invocation with Syntactic Input Data

Every time the BPEL process reaches an invoke statement, the following steps are worked off:

1. The BPEL process is executed on a standard orchestration engine like Oracle BPEL Server or IBM Websphere. In case a semantic Web Service should be bound during run-time, the request is forwarded to SISi. SISi is executed on a JAVA servlet container like Apache Tomcat.

2. SISi receives the semantic discovery request and passes the semantic Goal description to the semantic reasoner.

3. The reasoner uses semantic discovery algorithms to find matching semantic Web Service descriptions. The best fitting Web Service description is selected and passed back to SISi.

4. SISi uses this semantic Web Service description (WSMO Web Service) and finds through the contained grounding information the underlying concrete Web Service implementation. Then, this Web Service is invoked with the data received from the BPEL process as input parameter. Note that this variant requires that SISi becomes syntactical input data.

5. The Web Service is executed and the output data returned to SISi.

6. SISi forwards the output data back to the BPEL process.

- **Usage of Semantic Execution Engine**

The other alternative is covered by the usage of a complete Semantic Execution Engine (SEE) like WSMX. The difference to the procedure described above is, that now not SISi is invoking the actual Web Service but the SEE itself. The order of the single steps therefore differs:

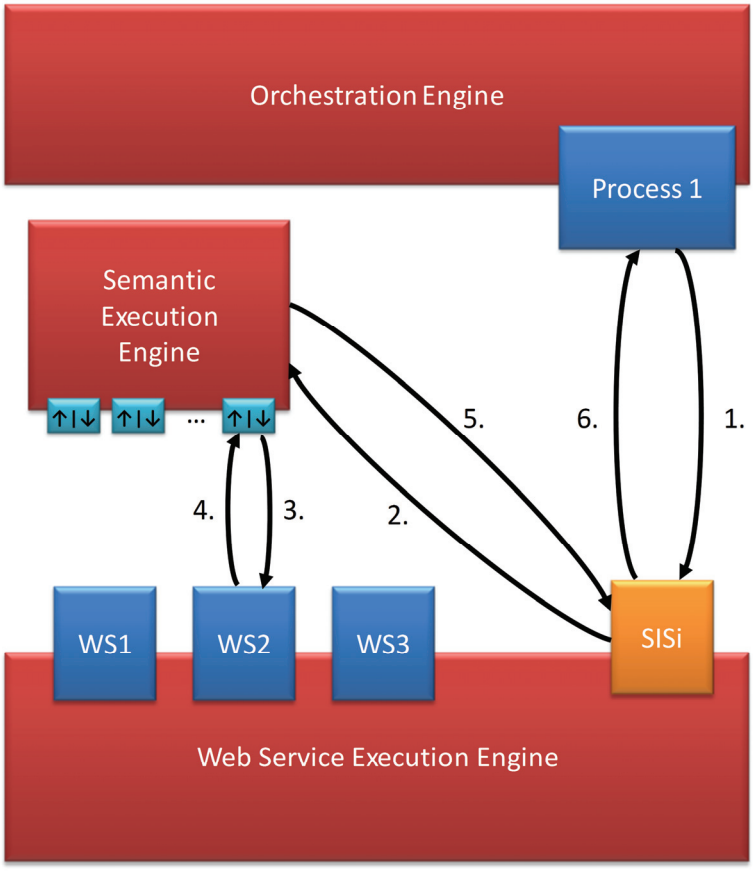



Figure 35 SISi: Invocation with Semantic Input Data

1. Again the BPEL process is executed and invokes a semantic Web Service.

2. SISi receives the semantic discovery request. SISi passes the semantic instance data (input for the Web Service) as well as the semantic Goal description to the semantic execution engine.

3. The engine discovers a semantic Web Service with help of the semantic description selects one service and handles also its invocation. As WSMX is designed to receive ontological instances as input for Semantic Web Service invocation, the SEE now also has to transfer the instances into a syntactical format like XML. This is in WSMX accomplished by means of lowering adapters (refer to chapter 3.3.2).

4. After the invocation of the concrete Web Service, the result is passed back to the SEE.

5. The SEE transfers the syntactic output data into its ontological representation. In WSMX, this procedure is handled by lifting adapters (again refer to chapter 3.3.2).

6. SISi receives the ontological output data of the Web Service and delegates it back to the BPEL process.

For the underlying thesis, only the last usage procedure using semantic execution engines was implemented. There are several reasons for this decision. There are time restrictions on the one hand and the circumstance that the second procedure has some advantages on the other hand. In fact, the usage of semantic data simplifies the integration with the developed method of chapter 4.4. Step 2c of this method requires data modelling based on the input and output instances from the attached Goals. The usage of semantic data (in combination with semantic execution engines) allows the direct transfer of this mapping. However, the usage of syntactic data (in combination with semantic reasoners) would require additional mirroring of this (semantic) mapping onto syntactic data.

#### 4.7.1.2 Architecture

The software architecture of SISi is a classical three layer design [BMR+96] and is shown in figure 36.

The top layer *External Interface Component* is exposed to the BPEL process and accessible via a standard Web Service interface offered as WSDL description. Although at the moment only the Web Service interface is pro-

vided, this layer provides room for other access possibilities. Details about this offered Web Service interface can be found in chapter 4.7.1.3.

The middle layer is the *Core Component* of SISi. The included *Controler Module* provides a unified interface for invoking Semantic Web Services. Hereby, both identified invocation procedures are covered. These are encapsulated in two methods `invokeSyntactically` and `invokeSemantically`, which are exposed by *External Interface Component*.

Method `invokeSyntactically` carries out the invocation of a semantic Web Service with a semantic reasoner and syntactic data (see chapter 4.7.1.1, part “Usage of Semantic Reasoner”). It therefore discovers the Semantic Web Service first (accomplished by *Semantic Interaction Module*, see figure 36), then extracts the grounding information, and then invokes the grounded Web Service implementation (covered by *Web Service Interaction Module* in figure 36).

Method `invokeSemantically` implements the second invocation procedure that envisions the usage of a semantic execution engine and semantic data (refer to chapter 4.7.1.1, part “Usage of Semantic Execution Engine”). Hereby, it needs to trigger the semantic service invocation of this execution engine (accomplished by *Semantic Interaction Module,* compare to figure 36). This method is described more closely in chapter 4.7.1.3.

The bottom layer *Semantic Abstraction Component* abstracts from the actual implementation of the used semantic reasoner or semantic execution engine and provides a unified interface for the overlying *Core Component*. New semantic engines are plugged by providing a corresponding adapter, which matches the interface of the semantic engine to this unified interface offered to *Core Component*. Currently, SISi implements only a WSMX adapter (see *WSMX Adapter Module* of figure 36).

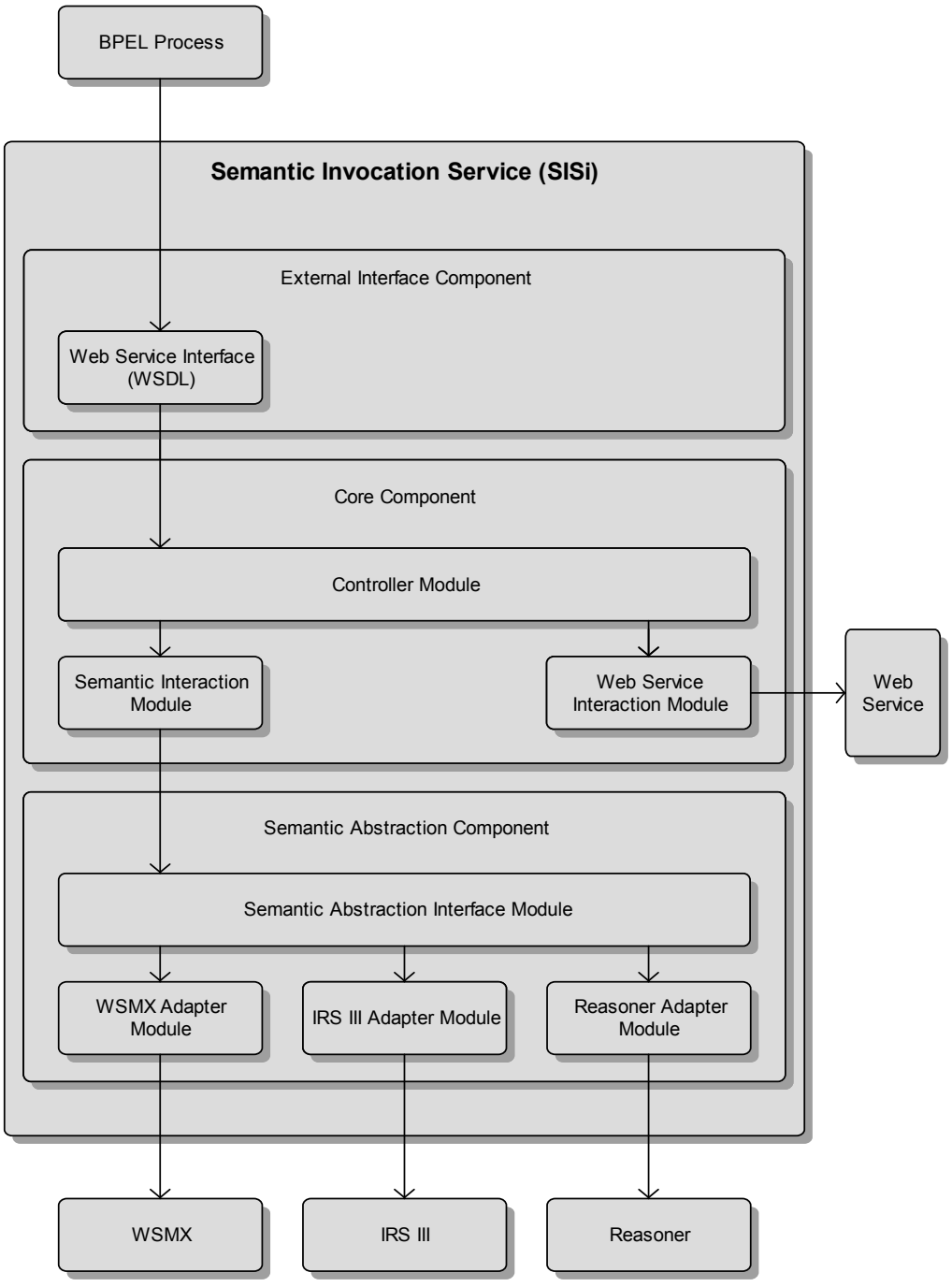



Figure 36 Architecture of SISi

#### 4.7.1.3 WSDL Interface

The external way to access SISi is provided in form of a WSDL interface. The interface contains five operations, which are:

- **map**

This operation creates the right input instance ontology out of the assigned output instance. Basis for the operation is the annotation created in the EPC process through macro "*Map Instances"* (refer to chapter 4.5.1) on 1:1 instance mapping (see chapter 4.6.2.2). The intent for the operation is to provide a correctly formatted input for the subsequent invocation activity. Figure 37 visualizes the functionality of the operation. It gets the name of the source instance (`instance_x`), the name of the source Goal (`GoalX`), as well as the desired name of the target instance (`instance_y`), the desired name of the target Goal (`GoalY`) and finally the actual source ontology (`ontology GoalXOutput`) as input paramteters. The operation then returns the properly named output ontology (`GoalYInput`) as shown in figure 37.

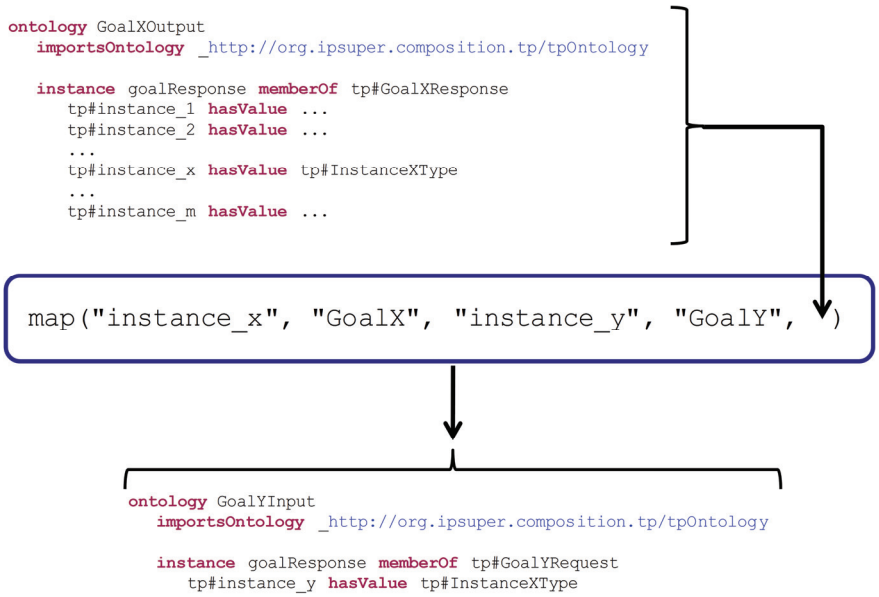



Figure 37 Operation Map

- **merge**

Similar to operation "*map"*, this operation creates appropriate instance ontology. In contrast to the previous operation, "*merge"* has two output instances as source that need to be joined into one ontology that also matches the naming conventions of chapter 4.7.2. This operation is related to macro *"Map Instances"* and is applied if a 2:1 instance mapping (refer to chapter 4.6.2.3) exists. The operation is depicted in figure 38. Input parameters or operation "merge" are the instance name of the first ontology (`instance_xi`), the

name of the first ontology (`GoalX`) and the ontology itself as well as the instance name of the second ontology (`instance_yj`), the name of the second ontology (`GoalY`) and the actual second ontology. Last input parameter is finally the desired name of the resulting ontology (`GoalZ`). The operation then generates an ontology which contains both instances and formats it according to the naming conventions.

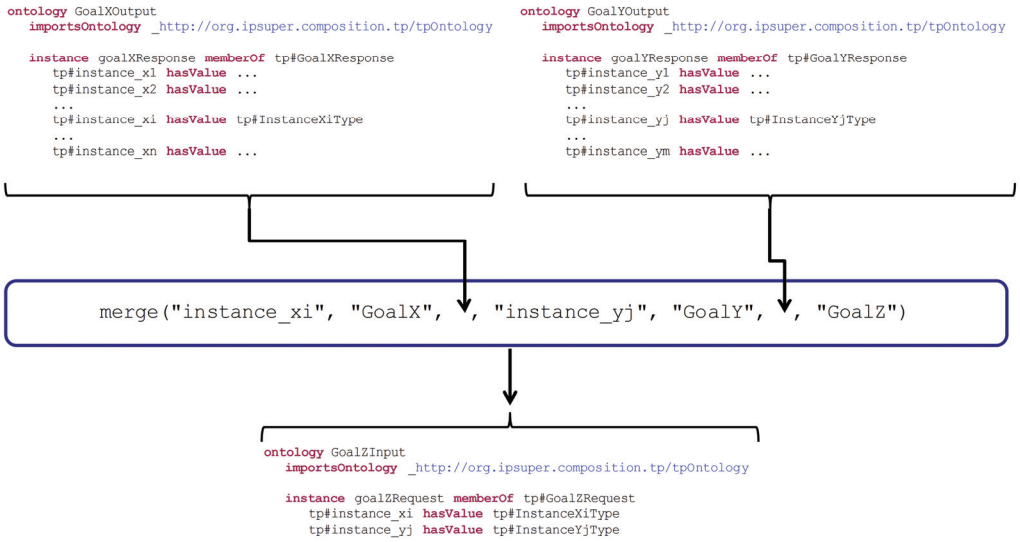



Figure 38 Operation Merge

- **decide**

Operation “*decide”* is used to realize decisions in control flow. As elaborated above, decisions are represented by XOR rules in EPC and by switch activities in corresponding BPEL process. Before every switch activity, operation “decide” is invoked. This operation receives an instance ontology as well as a WSMO condition expression as parameters. The operation then checks whether the condition holds for the passed instances and finally returns true or false. Figure 39 shows this procedure. Note that the evaluation of the condition expression is currently implemented as a simple string matching algorithm. Nevertheless, it is possible to delegate the evaluation to a semantic reasoner that calculates the result by means of predicate logic algorithms.

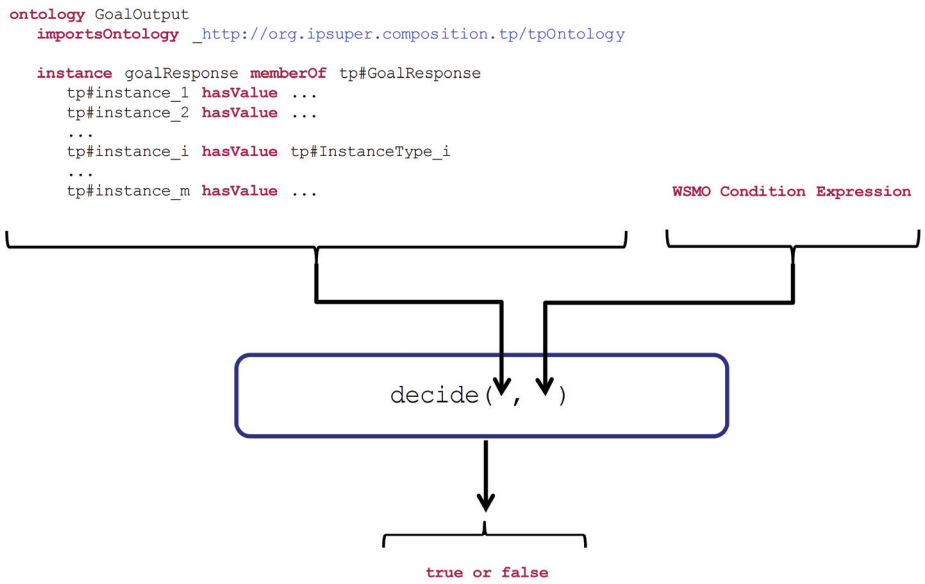



Figure 39 Operation Decide

- **invokeSemantically**

The operation *"invokeSemantically"* fulfils the main task of SISi: the invocation of the Web Service. Each invoke activity of the transformed SISi BPEL code that corresponds to an EPC function contains an invocation of this operation. The functionality of the operation is shown in figure 40. Input parameters are the WSMO Goal (`Goal`) and the ontology containing the input instances (`GoalInput`). The operation then delegates this information to the semantic execution engine (here WSMX). The engine uses the Goal for finding an appropriate WSMO Web Service (compare to service discovery and service selection of chapter 3.3.2). Then, it transfers the input instances into the XML representation by means of the lowering adapter (see chapter 3.3.2.3). After that, it invokes the grounded Web Service implementation of the found WSMO Web Service with the XML data as input parameter. The result is transferred to WSML (lifting adapter, see also chapter 3.3.2.3) and return to SISi, which delegates it to the caller of operation "invokeSemantically". The described procedure corresponds exactly to chapter 4.7.1.1 (part "Usage of Semantic Execution Engine").

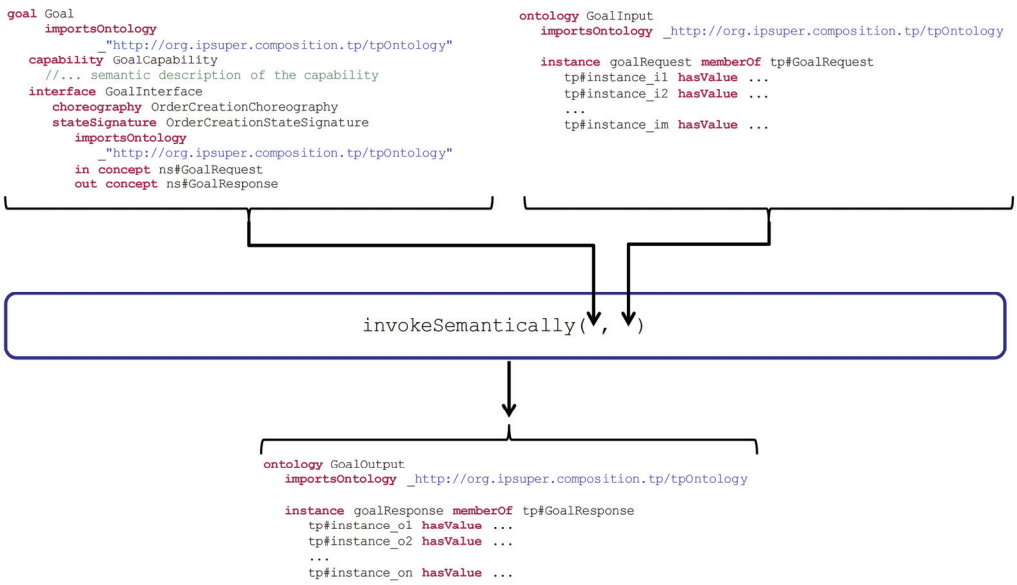



Figure 40 Operation InvokeSemantically

### 4.7.2 Naming Conventions

The messages that contain in operation `invokeSemantically` the input instances as well as the produced messages with the output instances follow a certain pattern. Listing 28 and listing 29 exemplify these patterns by means of one input and one output ontology.

The convention appoints that the target namespace for the input and output ontology have to start with `http://org.ipsuper.composition.tp/` followed by the name oft he consuming or producing Goal. This means `Goal` as this is the name of the Goal in the examples.

```
namespace { _"http://org.ipsuper.composition.tp/Goal",
            tp _"http://org.ipsuper.composition.tp/tpOntology" }

ontology GoalInput

    importsOntology "http://org.ipsuper.composition.tp/tpOntology"

          instance goalRequest memberOf tp#GoalRequest
                 tp#instance_1 hasValue instance_1_Value
                 tp#instance_2 hasValue ...
                 ...
                 tp#instance_n hasValue ...

          instance instance_1_Value memberOf ...
          ...
```

Listing 28 Naming Conventions of Input Messages

Furthermore, the name of the Goal takes also place in the actual name of the enclosing ontology followed by `Input` or `Output` depending whether the encapsulated instances are input or output instances.

```
namespace { _"http://org.ipsuper.composition.tp/Goal",
          tp _"http://org.ipsuper.composition.tp/tpOntology" }

ontology GoalOutput

    importsOntology _"http://org.ipsuper.composition.tp/tpOntology"

    instance goalResponse memberOf tp#GoalResponse
           tp#instance_1 hasValue ...
           tp#instance_2 hasValue ...
           ...
           tp#instance_m hasValue ...
```

Listing 29 Naming Conventions of Output Messages

As last convention, all relevant instances of the ontology are referenced by a main instance, which is named by the Goal plus `Request` or `Response` depending on the type of the message (input or output).

WSMX itself makes no restrictions concerning the assembly of the instance messages. Nevertheless, these conventions make it easier to handle this flexibility when constructing the lifting and lowering adapters.

## 4.8 Exemplary Procedure with Showcase Process

This chapter exemplifies the developed method of chapter 4.4 by means of the showcase process of chapter 1.3.2. The example particularly shows the method in combination with the alternative execution environment of chapter 4.7.

The showcase VoIP scenario consists of overall three Web Services (compare to chapter 1.3.2):

- **CRMService**

This service is responsible for all operations concerning the customer. It contains the operations:

- `notifyCustomer`
- `identifyCustomer`
- `verifyFormalRequirements`

- **OMSService**

This service contains all operations which are somehow involved when processing an order:

- `createOrder`
- `confirmOrder`
- `prepareContract`
- `sendByCourierService`
- `archiveContract`
- `activateBilling`

- **HWMService**

This service provides operations concerning hardware specific tasks:

- `verifyTechnicalRequirements`
- `checkForLiveBox`
- `prepareHardware`
- `activateVoIP`

The functionality of these operations is straightforward and corresponds to their names.

Compared to the description of chapter 1.3.2, only these three systems are relevant for the example. The mentioned portal system is only calling the modelled process here and the task of the courier service is encapsulated in operation `sendByCourierService` of OMSService.

### 4.8.1 Applying the Method

#### I. Create Domain Ontology

The relevant extract of the domain ontology for the showcase can be found in listing 7.

#### II. Describe Web Services Semantically

Each of these Web Services contains different operations. For the application of semantics for each of those operations a corresponding WSMO Web Service has to be created that describes this operation semantically. There are overall 13 WSMO Web Service descriptions:

- *CustomerNotification*, which is grounded by operation `notifyCustomer` of CRMService
- *CustomerIdentification*, which is grounded by operation `identifyCustomer` of CRMService
- *FormalVerification*, which is grounded by operation `verifyFormalRequirements` of CRMService
- *TechnicalVerification*, which is grounded by operation `verifyTechnicalRequirements` of HWMService
- *OrderCreation*, which is grounded by operation `createOrder` of OMSService
- *OrderConfirmation*, which is grounded by operation `confirmOrder` of OMSService
- *LiveBoxCheck*, which is grounded by operation `checkForLiveBox` of HWMService
- *ContractPreparation*, which is grounded by operation `prepareContract` of OMSService
- *HardwarePreparation*, which is grounded by operation `prepareHardware` of HWMService
- *CourierService*, which is grounded by operation `sendByCourierService` of OMSService
- *ContractArchival*, which is grounded by operation `archiveContract` of OMSService
- *BillingActivation*, which is grounded by operation `activateBilling` of OMSService
- *ServiceActivation*, which is grounded by operation `activateVoIP` of HWMService

For reason of simplicity the WSMO Goal are named exactly like their corresponding WSMO Web Services.

For the showcase it was further necessary to implement the lifting and lowering adapters of WSMX. These had to be implemented according to chapter 3.3.2.2 for each of the WSMO Web Services to be able to call them semantically.

### 1. Modelling the Process

The result of this step is the EPC of figure 4 and figure 5.

### 2. Semantically Annotate Business Process

The three steps of the semantic annotation are only sketched shortly. The actually Goal annotation, the control flow completion as well as the data flow modelling is very straightforward. Nevertheless, additional details to these steps can be also found in the tutorial of appendix I.

### 2a. Add Goals

During this step the 13 WSMO Goals are assigned to the single functions of the EPC process. These WSMO Goals are referencing on the WSMO Web Services and the WSMO Web Services are itself grounded by the operations of the concrete three Web Services (CRMService, OMSService and HWMService). For this step macro *"Add Goal"* (see chapter 4.5.1.1) is being executed for every assignment.

### 2b. Complete Control Flow

Here, the branch conditions of the splitting XOR operators in the EPC are attached. In the showcase process, there are overall two splitting XOR operators that need to be refined (see figure 41 and figure 42). Macro *"Add Decision"* (chapter 4.5.1.3) is responsible for this step.

### 2c. Model Data Flow

Now, the output instances have to be mapped on the corresponding input instances and the process input and output has to be assigned. Responsible for these tasks are macros *"Map Instances"* (see chapter 4.5.1.2), *"Set as Process Input Instance"* and *"Set as Process Output Instance"* (both chapter 4.5.1.4). To mention all mappings would go beyond the scope of this example. The process input instance is "customerName" in figure 41 (red arrow), while the process output instance can be found in figure 42 ("customerCase", green arrow).

### 3. Transform to Executable Process

After all semantic annotations were added the process can be transformed into SISi BPEL code, which is accomplished by running macro *"Transform Process"* (see chapter 4.5.1) and then choosing "SISi BPEL" as the alternative execution environment shall be supported. This macro carries out the transformation of chapter 4.6.2.

This results in the corresponding SISi BPEL model of the process which is shown by figure 43 and figure 44.

### 4. Finalise Executable Process

This step can be omitted as the transformed process is completely finished.

### 5. Deploy Executable Process

The process (SISi BPEL model generated in step 3) can now directly be deployed by means of macro *“Deploy Process”* of chapter 4.5.1.4. The exposed WSDL interface of the deployed process can be seen in listing 30. The interface provides only operation `execute_VoIP-Order` to invoke the process. This operation receives as input parameter a string with the ontology, which includes the input instances of the process. The definition of these input instances happened in step 2c by means of the macro *“Set as Process Input Instance”*. Accordingly, the process needs an ontology containing instance “customerName” (red arrow in figure 41) as input. The output message of the operation also contains a string. This string includes the output instance “customerCase” (green arrow in figure 42) specified in step 2c with macro *“Set as Process Output”*.

```
<definitions      ...
                  targetNamespace="http://sisi.externalInterface/
                  name="VoIP-Order-Process"">
   <message name="stringMT">
      <part name="stringMP" type="xsd:string"/>
   </message>
   <portType name="processPT">
      <operation name="execute_VoIP-Order">
         <input message="stringMT"
                name="execute_VoIP-Order_Request"/>
         <output message="stringMT"
                 name="execute_VoIP-Order_Response"/>
      </operation>
   </portType>
  ...
</definitions>
```

Listing 30 Exposed WSDL Interface of Showcase Process

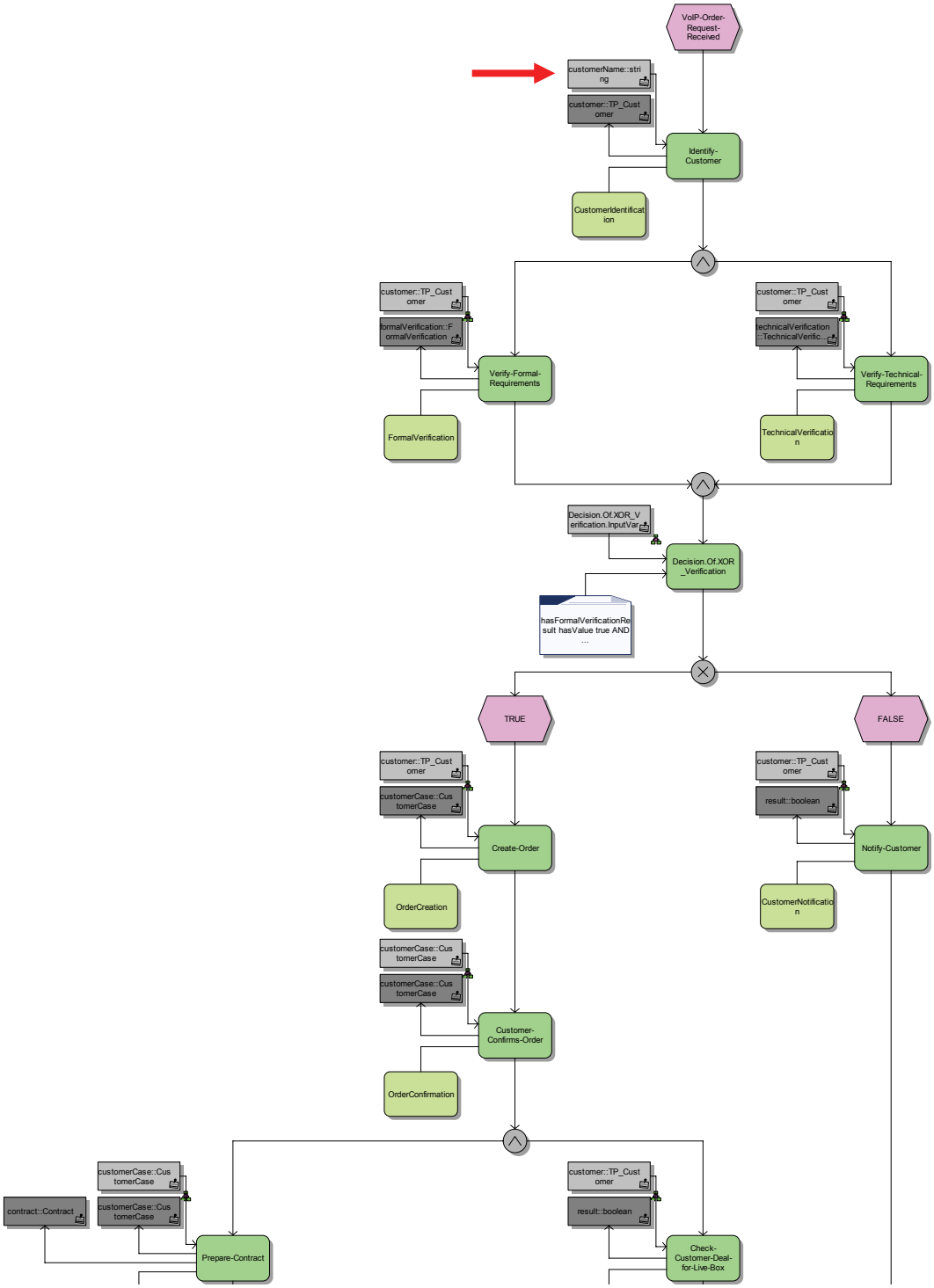



Figure 41 Enriched TP Process 1/2

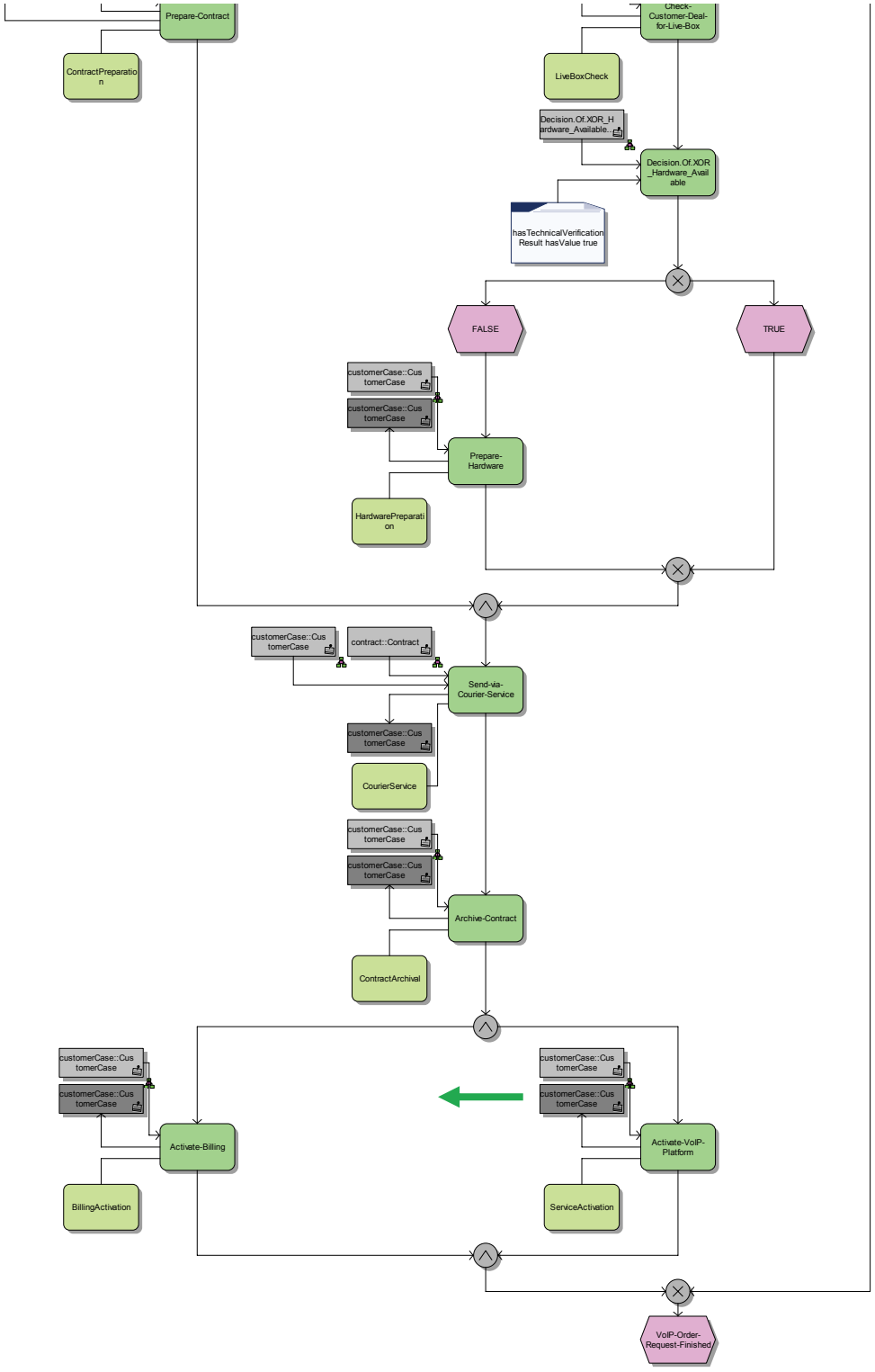



Figure 42 Enriched TP Process 2/2

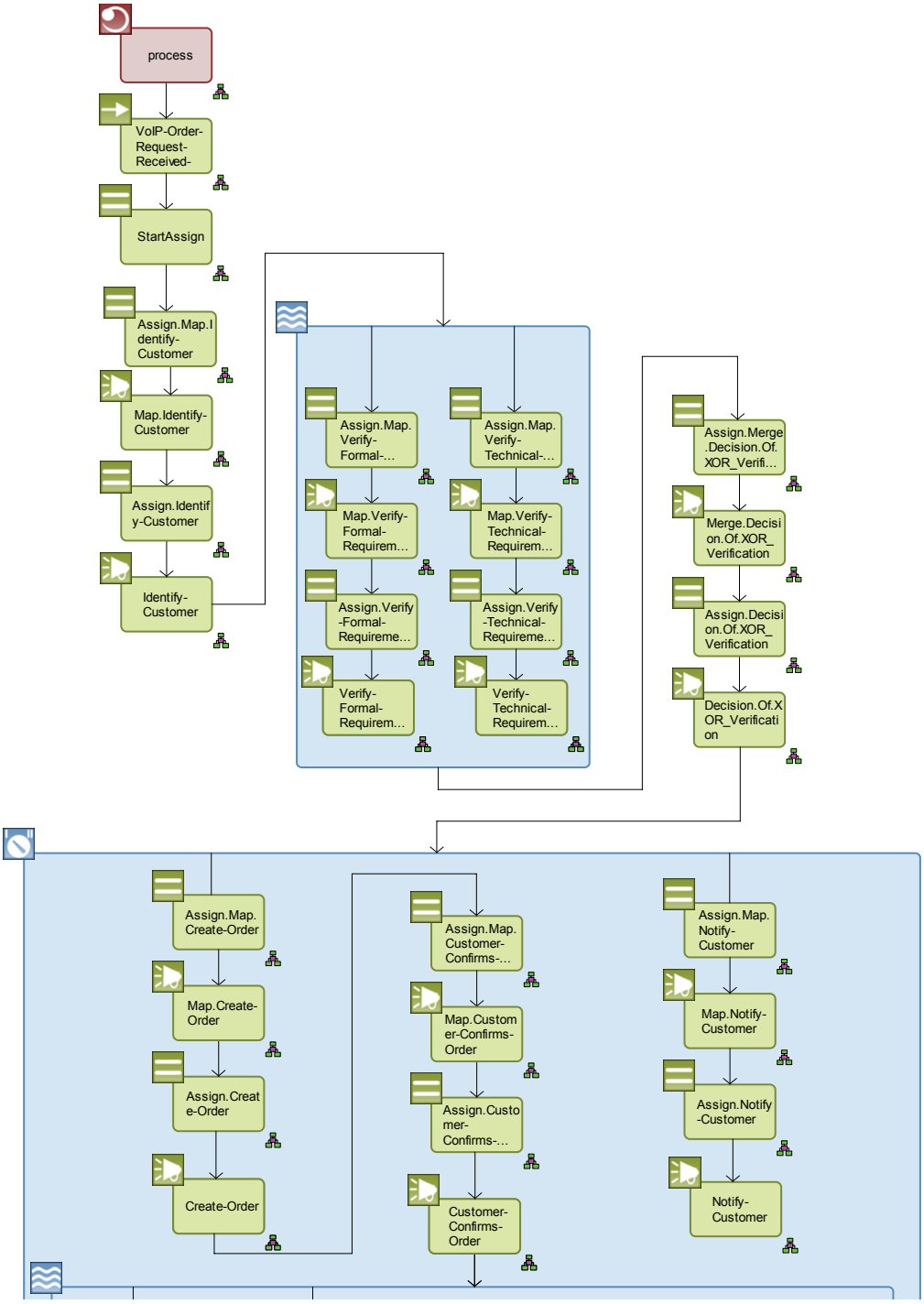



Figure 43 Transferred TP Process 1/2

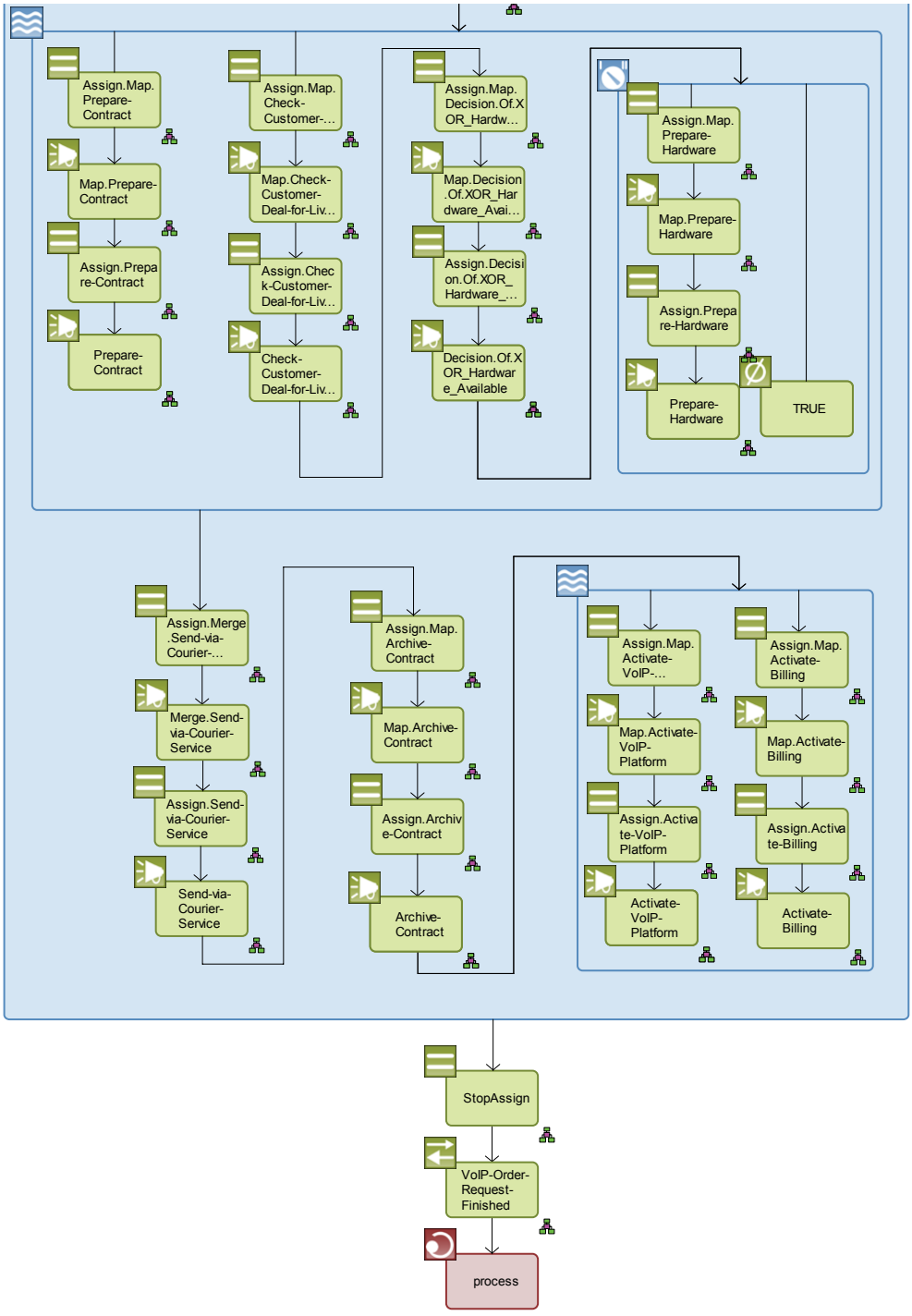



Figure 44 Transferred TP Process 2/2

### 4.8.2 Runtime Scenario

The runtime scenario with the involved software systems and the distribution of the single artefacts is shown in figure 45. The three application Web Services (CRMService, OMSService and HWMService) as well as SISi are deployed on an Apache Tomcat server. The modelled showcase process is deployed on an Oracle BPEL Process Manager server. The semantic backbone is formed by WSMX. It includes a WSMO Web Service repository, where the 13 WSMO Web Service descriptions are stored and further contains the corresponding Lifting and Lowering Adapters to these descriptions. Note that the WSMO Goal descriptions, which reference to the WSMO Web Services (see figure 15), are included in the SISi BPEL code of the showcase process.

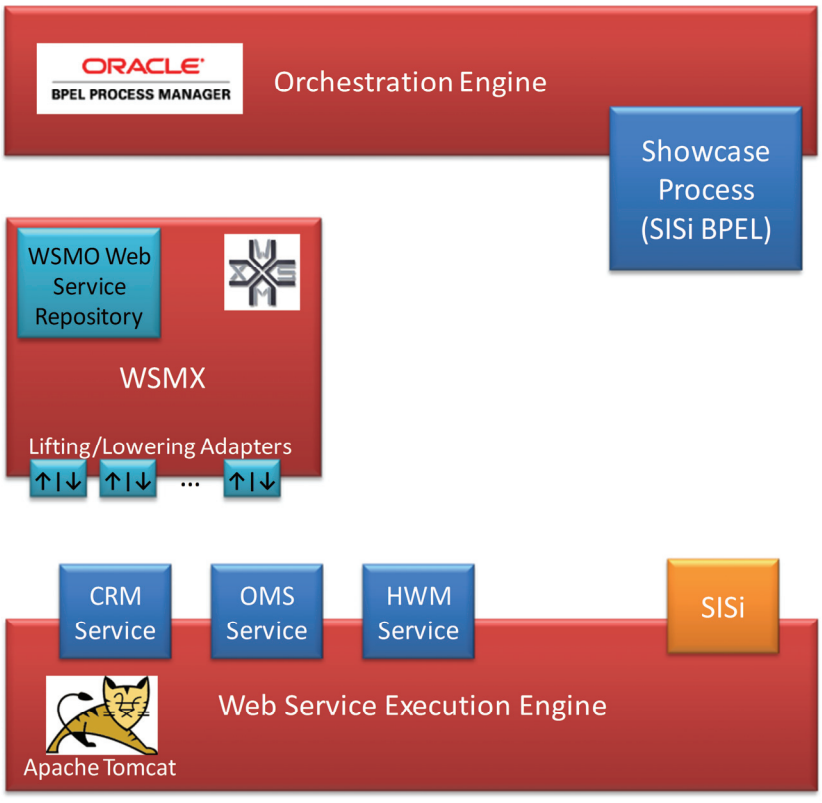



Figure 45 Showcase Scenario Systems

The invocation procedure of the showcase procedure follows exactly the description of chapter 4.7.1.1.

## 4.9 Benefits

The following chapter is separated as follow: first the general benefits due to the usage of the developed method are summarized. Then, the benefits depending on the used semantic infrastructure (SUPER or the developed alternative environment) are shown. The first part hereby contains particularly benefits which are common to both infrastructures while the two other parts respond to the characteristic benefits of the respective environment.

### 4.9.1 General Benefits

The developed method provides three principle advantages, namely:

- **Service Selection**

Service selection denominates step 2 of the standard method (see chapter 2.3.3) and respectively step 2a of the semantic method (refer to chapter 4.4). This step requires in the standard method possibly the examination of technical details of the considered service by the business analyst or in case of doubt the consultation of the integration engineer. In the new semantic method however the needs for the service are described on a pure functional level without any technical details (WSMO Goals). Although the business analyst needs knowledge in the domain ontology to be able to properly select a service by means of these Goals, this makes in general no additional requirement on the business analyst as it is anyhow reasonable for him to know his domain very well. Therefore, the business analyst should be able to select the desired Goal for the considered function of the EPC out of the repository or to refine a general Goal to match his specific needs. Nevertheless, it is indeed necessary for the business analyst to get familiar with WSML and the logical representation of the pre- and postconditions of the WSMO Goals. He must be at least able to understand these descriptions and he is in the ideal case also capable of specify a Goal on his own. In cases of doubt he must consult the ontology engineer, who brings a deep knowledge in these areas.

But even if the business analyst should need the help of the ontology engineer the new method has still the advantage that everything happens on a pure functional business level without any involvement of the technical Web Service.

- **Full Process Modelling on Business Level**

The developed method allows modelling of business processes on a pure functional level. The whole process can be modelled completely in one tool and in one diagram. This affects the complete control flow as well as data flow modelling. The method abstracts during the modelling phase entirely from technical details of the underlying software services.

- **Process Transformation**

After the modelled process contains already all details about control and data flow, it can be automatically transformed into an executable representation. This representation in general only requires little manual work (for instance in terms of exception handling) to be finished and ready to use. In terms of the showcase process it is actually possible that the process model can be completely automatically transformed, deployed and directly executed, admittedly due to the fact that exception handling is not being considered. Compared to the standard method this reduces manual work significantly as here is still a lot of effort necessary for rounding off the transferred BPEL process (incomplete control flow, missing data flow, and missing exception handling), which has to be accomplished by an integration engineer.

The method can therefore help to ensure consistency of business model and executable process, as less manual work is necessary. Furthermore, it enables faster reaction on changing requirements as modification have mainly to be accomplished on business model and the implementation is for the most part auto generated.

- **Service Binding**

During the modelling phase of the developed semantic method the business analyst assigns WSMO Goals to the functions. These Goals can be considered as a reference of the actual Web Service, which stands in contrast to the standard method in which services are hard wired to the EPC functions.

The actual resolution of these Web Service references (Goals) happens at runtime through the SEE (for instance WSMX, see chapter 3.2.3). This fact introduces some additional reliability and flexibility as for example failed Web Services can be dynamically replaced. Furthermore, this enables the consideration of non functional or quality properties of Web Services during service selection (refer to chapter 3.3.2.2)

### 4.9.2 Benefits of SUPER

The provided solution supports SUPER, which provides a consolidated environment for SBPM. The created BPMO can be deployed in the central repository of SUPER where it is then possible to continue the complete process life cycle with execution (*do*) over monitoring (*check*) to analysis (*act*) inside the SUPER environment (compare to chapter 3.2.3).

That means that SUPER provides additionally to the general advantages of chapter 4.9.1 further benefits. These benefits affect in particular bottom up features, as querying the process space or monitoring the process execution by means of semantic information (see chapter 3.2.3).

### 4.9.3 Benefits of Alternative Execution Environment

As explained in chapter 3.4, SUPER has despite those benefits several disadvantages that hinders companies from introduction.

The developed alternative environment (see chapter 4.7) on the other hand provides a pragmatic possibility to introduce semantic technologies in the present standard technology stack of an enterprise. This prevents various assets of the enterprise to get lost. For example existing process implementations can be executed on the same environment as the semantic processes. This mixed execution allows enterprises a step by step migration towards SBPM.

Additionally, the investments for introduction are lower as employees do not have to be trained about the enormous complexity of SUPER with all its meta ontologies and its sophisticated functionality.

# 5 Empirical Evaluation

In order to evaluate the developed method and to validate the highlighted benefits, empirical techniques shall be considered.

Empiricism in general aims at applying theoretical concepts in practise and to gain experience from their appliance. Empirical research can be applied whenever it is necessary to prove theories or to chose among different techniques [WRH+00].

In order to underlay the accomplished research with empirical data, empirical studies shall be considered. According to [WRH+00], there are three types of empirical studies: experiments, case studies, and surveys.

## 5.1 Choosing an Appropriate Empirical Study

The general intend for conducting the empirical study is to evaluate the developed method. Therefore, it is necessary to cover all steps of the method, especially the core features of service discovery and process transformation. To provide a better understanding, it is desirable to compare the new method with the state of the art approach.

According to [WRH+00] the proper ways to compare two methods are case studies or experiments. For the present thesis, two possible scenarios for empirical studies were identified:

### 1. Experiment

The imagined experiment involves two groups of participants. The first group uses the state of the art method and the second one uses the newly developed semantic method in ARIS SOA Architect. Both groups have to model the EPC of the process and enrich the additional information to the EPC functions. During the modelling phase also the two service discovery approaches are evaluated. After adding all services or semantic descriptions, the transformations are triggered. Target is to receive an executable process. Therefore, the first group has to refine the resulting BPEL process with the missing information. The experiment could evaluate if one of the approaches produces a better quality or is more effective.

### 2. Case Study

The conceived case study compares two methodologies by means of two tutorials. In the first tutorial the subject is modelling an exemplary process according to the standard non-semantic method. In the second tutorial, the subject is modelling the introduced showcase process according to the semantic method. In contrast to the experiment this case study does not tend to collect quantitative data. This would not be a reasonable undertaking as the used processes differ and the results would therefore be not comparable. Instead the case study collects qualitative data by interviewing the modelling experts.

For the present work the case study was favoured for the following reasons:

In order to conduct the comparison as an experiment, very much effort has to be undertaken for preparation and execution. To provide a valid experiment setup, two impartially comparable scenarios have to be developed. It must be ensured, that neither the first nor the second group has benefits due to scenario design. To provide a realistic scenario therefore a lot of different Web Services have to be provided. Only a few of these Web Services are actually used during modelling, the other “dummy” Web Services ensure that the service discovery is not “too easy” and more realistic. In addition, also semantic descriptions (WSMO Goals) have to be provided for these services as well as some dummy semantic descriptions for the same reason.

As an experiment was initially planned also efforts in searching participants were made. The available participants are from the peripherals of Fraunhofer IESE[24] and are usually students of Computer Science with a strong technical background.

The background of the participants is directly affecting the external validity [WRH+00] of the experiment. As the main parts of the methodologies are usually covered by business analysts with a strong background in business administration and management, the results would not be representative and hence could not be generalized.

Furthermore, the participants of the experiments would have to be instructed in both methods in prior. Especially refining the implementation with help of an IDE like Oracle JDeveloper is a complex task which requires experience. Although the available participants have a technical background, it is likely that they would be over challenged by this task, particularly due to time restrictions. A possible solution could be to downsize the process in order to reduce complexity, but this would lead to an unrealistic scenario and would again narrow the external validity of the experiment [WRH+00]. It would also not help to fulfil the research goal of evaluating the complete approach.

[24] http://www.iese.fraunhofer.de/

Besides these reasons against the experiment there are additionally reasons supporting the choice of the case study:

The availability of the tutorial for the standard approach as well as access to experts who are familiar with the approach requires only the creation of the semantic tutorial and the experts only have to conduct this semantic tutorial. This reduces the effort for the test persons.

The results expected from the case study are of qualitative nature and promise more insight in evaluation of the method as pure quantitative numbers gained in the experiment. At the current point, explorative insight is needed so that identified areas of interest can be later investigated using controlled experiments. Using qualitative research to identify open research questions later investigated by quantitative research is a common approach in software engineering research [SSS01].

All these points suggest that the case study is the right choice for applying empirical evaluation.

The design, implementation, conduction, and interpretation of the entire case study were aligned to the definitive book “Case Study Research” of Robert Yin [Yin02].

## 5.2 Case Study Design

The following chapter explains how the case study was designed and points out which research objectives are aimed.

### 5.2.1 Study Context, Research Aim, and Research Questions

The research discipline semantic business process management (SBPM) investigates how to successfully leverage technologies taken from the semantic web and ontological engineering to enhance business process management. Business process management is concerned with all activities needed to document, define, design, analyse, implement, and monitor business processes. From a broad perspective, business process management combines top-down as well as bottom-up approaches. In a top-down approach business processes are defined and modelled and an implementation is derived. In a bottom-up approach, existing business processes are extracted and analysed to derive a more abstract view on what is going on in the analysed enterprise.

Doing business process management involves a transition between the business and the IT domain. This is one of the biggest challenges in busi-

ness process management, because people with a wide variety of skills are involved. As a result, different modelling languages and formalisms are used and a transition between them must be provided. Here, semantic technologies especially defining a mapping between different domain vocabularies using ontologies might help.

The research aim of the case study can be specified as follows:

**The case study examines if the highlighted benefits can be confirmed. Hereby, the focus lies on the general benefits described in chapter 4.9.1**.

The research questions are therefore aligned to the structure of the alleged benefits of chapter 4.9.1:

- **Service Selection**
  a. Do business analysts manage to select the WSMO Goals according to the method?

  b. Are business analysts capable of understanding the content of the WSMO Goals (in particular pre- and postconditions)?

  c. Are business analysts able to refine given Goals?

  d. Are business analysts capable of writing Goals completely on their own?

  e. Does the service selection of the semantic method provides a benefit or are there disadvantages?

- **Full Process Modelling on Business Level**
  f. Do business analysts get along with completing the control flow (step 2b of semantic method)?

  g. Are business analysts able to model the data flow as planned in the semantic method (step 2c)?

- **Process Transformation**
  h. Do business analysts think that the process transformation provides benefits?

- **Service Binding**
  i. Do business analysts see an advantage of dynamic service binding?

Beside the strict evaluation of the mentioned benefits, also general insight in the method and the application of semantics shall be gained. The research aim is therefore extended.

The corresponding research questions are formulated as follows:

- **Method in General**
  j. Do business analysts understand the method in general?

  k. Do business analysts come along with the provided documents and their representation?

  l. Is the semantic approach ready for practical adoption?

  m. Do business analysts see the advantages of the new method?

  n. Are there disadvantages connected to the semantic approach?

### 5.2.2 Case Study Propositions

The propositions represent the expected results of the case study. According to Yin [Yin02], it is important to specify these expectations before the actual conduction and interpretation. This helps to avoid unconscious manipulation as well as misinterpretation of the results and contributes to internal validity of the case study [Yin02] (see chapter 5.5.1).

The propositions of the case study in general directly correspond to the benefits stated in chapter 4.9.1. Nevertheless, not all of those benefits are expected to be confirmed without restrictions.

The use of semantics promises simplifying the generation of executable models, because there is a mapping between the business and IT world on the meta modelling level through the use of ontologies. Also, using mediators promises a simplified data handling through automatic transformation between different data formats. Semantics are expected to help simplifying data handling by introducing an additional abstraction level. In addition, using semantics allows advanced mechanisms to discover services, helping business analysts to select appropriate services. Therefore, it is expected that the use of semantics will simplify the implementation of business processes.

Semantic Web Service discovery is based on the assumption that semantic requests (that is WSMO Goals) are formulated and that services are described semantically. It is expected that creating such semantic descriptions is easier or requires less resource than selecting services manually.

Connected to the prior point is the promise of semantics to enable late binding of Web Services. Instead of hardwiring services during business process design, appropriate Web Services are selected during execution based on semantic descriptions (WSMO Goals). Even though this sounds like an interesting possibility, it is expected that this is no essential feature for business analysts. Web services represent partners and in a business environment partners are selected and contracted carefully, but they are not dynamically allocated during runtime without further interaction.

Other authors like Hepp et al. [HR07a] already pointed out that one major problem in any ontology based approach is the development, communication, and usage of ontologies. For example, creating an ontology accepted by all involved stakeholders is a challenging or maybe even impossible task as Hepp et al. have shown. Therefore, it is estimated that significant effort is needed for creating and understanding the needed ontologies.

The introduction of semantics requires additional knowledge like modelling ontologies and especially defining logical expressions. In addition, adding semantics also means extending the existing enterprise computing infrastructure with semantic execution engines. Installing and maintaining this additional infrastructure might require additional skills. Therefore, an increased learning curve is expected, which might hinder the adoption of semantics. On the other hand, using semantics might simplify the implementation so that business analysts do not need extensive IT knowledge anymore. This might reduce the learning curve and balance out the increase.

The mentioned propositions show that it is critically considered if the use of semantics will pay off. It is estimated that semantics will simplify or even eliminate certain steps, but it is also expected that additional steps will be required. It is estimated that at the current point an investment into semantics cannot be justified from an economic point of view. However, it is also expected that the case study will give concrete pointers what needs to be done to fully leverage the benefits of semantics in Business Process Management.

### 5.2.3 Unit of Analysis

Yin [Yin02] requests a clear definition for the unit of analysis of the case study. In the present case, the unit of analysis are two tutorials guiding a user through the process of implementing a business process. One tutorial covers the approach taken today without using semantics. This is called the non-semantic tutorial or approach. The other tutorial covers the approach applying semantics. This is called the semantic tutorial or approach.

Both tutorials are based on realistic processes taken from real-life projects. The case of our case study comprises conducting at least the semantic tutorial if the test person is already familiar with the non-semantic approach, or to

conduct first the non-semantic tutorial and afterwards the semantic tutorial if the test person is not familiar with the current approach.

The test persons are interviewed after conducting the tutorial(s) by the researcher in a semi-structured interview. It is intended to replicate the case with different test persons several times to get to know different view points and to increase the validity of the research results.

As the thesis, the tutorials only focus on a top-down approach to business process management, in particular business process implementation. This further narrows the scope of investigation and defines more clearly the unit of analysis.

### 5.2.4 Analysis of Results

The experience gained by the test persons will be gathered through semi-structured interviews. Also the work artefacts produced by the test persons will be reviewed and the tests persons will be observed while doing the tutorials if feasible.

It is not intended to record the interviews, but the interviews will be conducted by two interviewers, of which one can concentrate on taking notes. Each interviewer will write a short summary of the main points discussed during the interview immediately after conducting the interview. This will ensure to not loose important ideas, which came up during the interviews. This increases the validity of the interviews.

In addition, after conducting all interviews, the interviewers will reflect on the propositions stated before in a written report reusing the single reports created before. The written material will be reused later on to create the final research report, which will be submitted as a research paper to an international conference or journal.

### 5.2.5 Participants of Case Study

The following participants were identified and grouped according to their knowledge of the non-semantic approach (in alphabetical order):

With little or without prior knowledge about non-semantic approach:

- 1 participant of FHTW Berlin[25]
- 1 participant of Fraunhofer IESE

[25] http://www.fhtw-berlin.de/

With prior knowledge about non-semantic approach:

- 2 participants of FHTW Berlin
- 1 participant of Fraunhofer IAO[26] in Stuttgart
- 1 participant of IDS Scheer AG
- 1 participant of IOWI[27] of FH Rosenheim

Most of the found participants have a background in Economic Science or Business Informatics and can therefore be compared to a typical business analyst. Furthermore, they are able to rate also the implementation specific tasks, as some of them also have knowledge in Computer Science. None of the identified participants is directly involved in the SUPER research project.

## 5.3 Case Study Implementation

The following chapter describes how the case study was actually implemented.

### 5.3.1 Overview

It is intended to cover the complete process of implementing a business process. For example, the case study should include creating the business process model, annotating it so that it can be automatically transformed into an executable one, doing the transformation, refining the executable model, deploying, and finally executing it.

### 5.3.2 Non-Semantic Tutorial

The non-semantic tutorial introduces the participants by means of a step by step instruction how to accomplish the entire standard method (see chapter 2.3.3) by means of a real-life business process.

The non-semantic tutorial can be obtained from IDS Scheer AG (mailto:sebastian.stein@ids-scheer.com).

[26] http://www.iao.fraunhofer.de/

[27] http://www.iowi.fh-rosenheim.de/

### 5.3.3 Semantic Tutorial

The semantic tutorial introduces the participants into the semantic method (see chapter 4.4). Here, the showcase process (refer to chapter 1.3.2) is used by the participants to exemplarily conduct the method.

The semantic tutorial can be found in appendix I of the present work.

### 5.3.4 Interview Questionnaire

The questionnaire used for conducting the interviews can be found in appendix II. The questionnaire consists of 19 questions, which are formulated open-ended. The open-ended questions are meant to stimulate a discussion about advantages and disadvantages of the two investigated approaches.

## 5.4 Results

The following section summarizes the given answers of the participants in the interview after conducting the tutorial(s). The order discussed here slightly differs from the order of the questions in the questionnaire (see appendix II) and sometimes also questions were merged or omitted. The corresponding question number(s) of the questionnaire can be found in brackets.

1. *Participants were asked to recapitulate the method in their own words (question 2 and 7).*

   All participants were asked to repeat the method in their own words. Hereby, 6 of the 7 participants could give detailed and correct descriptions. One participant was not able to reproduce the method correctly.

   Most have understood the general procedure.

2. *The participants were asked to explain the concept of Goals/semantic descriptions (question 3 and 8).*

   4 of 6 asked participants were able to describe the main aspects of a Goal correctly. 2 participants had different understandings of what a Goal is.

   3 of the participants were unclear, where those Goals come from and who creates them.

   The question was posed to only 6 of the 7 participants.

3. *Participants were asked to define the term “ontology” (question 4).*

The question, what an ontology is, was correctly described also by 6 of the 7 participants, although 1 participants was insecure about whether he understood it correctly. 1 participant had problems to classify the term.

4. *Participants were asked if they used the provided ontology during application of the method (question 5).*

   The participants all agreed that the provided ontology was not helpful during the accomplishment of the tutorial. Although all participants took a look at the ontology during introduction to gain a better understanding of the domain, the ontology was not consulted anymore during the actual tutorial. According to the participants the reason for this can be found in the small size of the described domain.

5. *Participants were asked which one of the ontology representations (see appendix I.1.3) was the most helpful (question 6).*

   The question which of the three provided interpretations (star, UML, WSML) of the ontology was the most useful was answered as follows.

   The star diagram was found useful for a first overview by 3 participants and also 3 participants thought this interpretation was not useful at all as it contains too few information.

   The UML class diagram however was rated useful by 6 of the participants, as it provides a good compromise between clarity and information.

   The worst impression left the WSML interpretation. All participants found this representation as bad to read, very cryptic and with the need to get used to first. 1 participant even refused to take a closer look at it.

6. *The participants were asked about the selection and annotation of the WSMO Goals (questions 9 and 10).*

   All asked participants were pretty sure about the assignment of the Goals to the functions. Nevertheless, all participants used only the name of the Goal but not the semantic description of the pre- and postconditions. Doubtful Goals could however be assigned though process of elimination.

   Only 1 participant would have considered the conditions during selection, whereas 2 participants were not even conscious about the existence of pre- and postconditions in the Goals.

7. *The participants were asked to describe the modelling of data flow (question 11).*

   6 of the 7 participants correctly described the general idea of the procedure. The answers for the procedure exposed positive and negative judgements. 4 of the participants denoted it as intuitive or comprehensible. Furthermore, 1 participant expressed that it is a great benefit that the input and output instance are immediately assigned after the addition of a Goal. Nevertheless, the procedure was also criticised by 2 participants, because of the necessary scrolling activity in order to map the instances. 4 participants also criticized that the procedure itself has a very technical touch.

8. *The participants were asked about the benefits and shortcomings of the new method compared to the standard approach (questions 12 and 14).*

   The participants were asked to specify the main difference to the old method. All participants mentioned that in the new method the implementation work can be removed or at least narrowed. Furthermore, 2 participants mentioned the dynamic binding of service during runtime. 1 participant found it helpful that the new approach is completely accomplished in one diagram whereas the old approach required manual work in many different diagrams. Moreover, 1 participant sees a clear separation of business and IT view in the new approach.

   The biggest disadvantage of the new approach was seen by 2 participants in the fact that it is necessary to cope with the cryptic WSML descriptions and that many unknown things happen in the background.

   1 participant agreed that the new approach provides benefits, but that it has to be checked exactly whether the new approach is worthwhile overall due to the additional costs with regard to the creation of semantic information.

9. *The participants were asked about the relevance of dynamic service binding (question 13).*

   The participants were asked about the sense of dynamic binding of services. 2 participants did not see any advantages or relevance of dynamic binding at all. 1 participant saw an advantage in principle but not directly in the context of an enterprise but in more dynamic environments (for example a pocket computer that uses dynamically a specific service). The other 2 participants saw a benefit in dynamic binding, which is expressed by a higher reliability (for instance if a service fails)

and flexibility. 1 participant even thought that it is essential to have dynamic service binding, especially in large systems.

10. *The participants were asked if the new method can be accomplished by a business analyst (question 14).*

    The question if the new method can be accomplished by a typical business analyst was in principle acknowledged by 3 of the participants. 3 participants saw a hurdle in gaining knowledge in WSML and especially in the creation of the Goals. 1 participant was confident to be able to refine a given Goal but thought that it is impossible to create a Goal completely by his own. 2 participants thought that the method is on a too low level for a typical business analyst and should be carried out by IT engineers.

11. *The participants were asked how much time they needed to conduct the tutorial (question 16).*

    The tutorial was accomplished by the participants in a time span from 0,5h to 2h.

## 5.5 Interpretation

The following section interprets the summarized results of chapter 0 towards the raised research questions of chapter 5.5.

Concerning the positioned benefits of chapter 4.9, the case study comes to the following conclusion (the addressed research question of chapter 5.2.1 can be found in brackets):

- **Service Selection**

The benefits promised for service selection cannot be hold completely. The case study showed that business analysts offhand are not able to cope with the semantic descriptions to the full extent. Although it is in general possible to select the appropriate Goal out of the repository, this task gets complicated if there are a lot of available Goals (**a**).

The business analysts used mostly the names of the Goals. The pre- and postconditions were ignored as most analyst were not able to understand these descriptions (**b**).

The case study further showed, that business analysts do not think that they are able to specify a Goal by means of pre- and postconditions completely

on their own (**c**), although some think they would be able to refine a given Goal (**d**).

It further became clear, that business analysts will need heavy training in order to use the semantic descriptions, as the analysts found the WSML description very cryptic and hard to understand. It turned out that there must be some graphical representation of the Goals, which would simplify their application.

Concerning the service selection it can be said, that business analysts cannot accomplish the method in practice on their own. In general, they will need the help of ontology engineers (**e**).

- **Full Process Modelling on Business Level**

In the case study, step 2b of the semantic method (completing the control flow) could be accomplished by business analysts. However it must be admitted that the branch conditions follow the same restrictions as the mentioned pre- and postconditions. Without help of ontology engineers or further training, business analysts will not be able to specify these on their own (**f**).

Step 2c of the new method (modelling the data flow), was found intuitive and comprehensible by most business analysts. Nevertheless, there were also critical voices concerning this task. Some participants criticised that this approach has a very technical touch. They suggested that the data flow modelling has to be accomplished by IT experts in a later step (**g**).

- **Process Transformation**

Almost all participants agreed that the process transformation into an executable implementation is significantly simplified by the method and a lot of work can be saved (**h**).

- **Service Binding**

Most participants see contrary to the expectations, an advantage of dynamic service binding (**i**).

- **Method in General**

It can be summarized, that almost all participants understood the semantic method. Nevertheless, a few participants had problems to understand the term ontology and also the concepts behind the WSMO Goals. This is critical as they provide the foundation of the developed method (**j**).

Business analysts were not satisfied with the provided WSML documents. These are too cryptic and hard to read and understand for the analysts. The case study made clear that business analysts prefer graphical illustration and demand for such representations for ontologies and especially Goals (**k**).

It can be summarized that additional training and an adequate graphical representation will be necessary for business analysts in order to apply the method in practice (**l**).

Business analysts see advantages in the new approach especially in the process transformation and the dynamic service binding (**m**). The mentioned issues concerning service selection and the need to learn are the main disadvantages seen by the business analysts (**n**).

### 5.5.1 Validity and Reliability of the Case Study

General validity must be divided in internal and external validity. While internal validity is concerned about existence of side effects that distorts the result, external validity cares about the ability to generalize the gained results. Reliability on the other side ensures that a case study will return the same results if it is repeated under the same conditions. The goal of reliability is to minimize biases and errors [Yin02].

#### 5.5.1.1 Internal Validity

The participants of the case study were instructed in the two methods by means of two tutorials. These tutorials are independent of each other and are written as objective as possible and do not give any arguments for or against any method. Before the case study, the participants received only the most relevant information about the topic in order to avoid influences.

Further, the case study was designed and implemented using the checklists of Host et al. [HR07b] and Kitchenham et al. [KPP95]. These checklists provide best practises for case study research and help to capitalize experience gained from other researchers.

The expectations of the case study were explicitly formulated in prior of the actual conduction (see chapter 5.2.2, propositions). According to Yin [Yin02] this helps to prevent from side effects caused by subliminal misinterpretation of the results.

All these facts help to eliminate side effects and to assure internal validity.

#### 5.5.1.2 External Validity

The whole case study was aligned to the book of Yin [Yin02]. Therefore, it can be assured that major methodical mistakes were avoided during the conduction of the case study.

Furthermore, the used processes in the tutorials are industrial business processes taken from real-life, which contributes to generalisability of the results.

Also the participants of the case study were chosen carefully. Even though some of the participants had a more technical background, the majority of the participants meet the picture of a typical business analyst.

These circumstances allow generalising the results and hence benefit to a good external validity.

Nevertheless, it must be admitted that external validity suffers from the fact, that the methods were only covered by respectively one showcase process, even so these were taken from real life. In fact, none of the two processes covers all available EPC modelling constructs (there are for instance no loops in the processes), which also narrows external validity.

#### 5.5.1.3 Reliability

According to Yin [Yin02], a case study protocol is a good way to ensure reliability of a case study. The protocol for the present case study corresponds to chapter 5.2 and 5.3. These chapters guide the investigator through the conduction of the case study and ensure that repeating the study would produce the same results.

For this reason the reliability of the case study can be guaranteed.

# 6 Conclusion & Outlook

This chapter finally concludes the gained results and shows how the developed ideas could be enhanced in future work.

## 6.1 Conclusion

The field of semantic technologies became very popular in the last years not only in the field of BPM and SOA. This is especially visible through the large amount of research projects in this area. Hereby, it is remarkable that not only universities and research institutes but also industry with big companies like IBM or SAP are active in those projects.

Semantic technologies in the area of BPM have theoretically a great potential concerning the Business-IT Divide. They can help to solve the top-down problem as well as the bottom-up problem and can contribute to efficiency and to improve quality.

However, it has to be kept in mind that the introduction of semantic technologies also requires additional work and expenses. Furthermore, there are established standard methods that do not make use of semantics and work as well. The question is why should companies spend money in a technology if they come along the traditional way?

An interesting feature of semantic technologies is the ability to link systems of enterprise even if they are using different ontologies (refer to Mediators of chapter 3.2.2). This inter-organizational application could be a reason for companies to introduce semantic technologies. The more companies semantic technologies adopt, the more interesting it will be for other companies to follow. A kind of network effect could help to excess the “critical mass” and to help semantic technologies to break through especially with the mentioned big companies above.

The developed method combined with the alternative execution environment based on SISi provides a pragmatic approach for introducing semantic technologies into BPM of an enterprise. Although it is still far away to productive usage it at least presents a starting point that can be enhanced, especially regarding the open issues revealed by the case study.

## 6.2 Outlook

The designed method provides a first toehold for applying semantic technologies in process landscape of an enterprise. The conducted case study showed that there are some possibilities for improvement:

Concerning the Goal selection it showed that business analysts demand for some graphical representation of the Goals and a kind of Goal browser.

One participant of the case study proposed the visualization of the Goals with common concepts of EPCs (see figure 46). It is very likely that such a visual representation would be better accepted and understood by business analysts than the textual WSML representation.

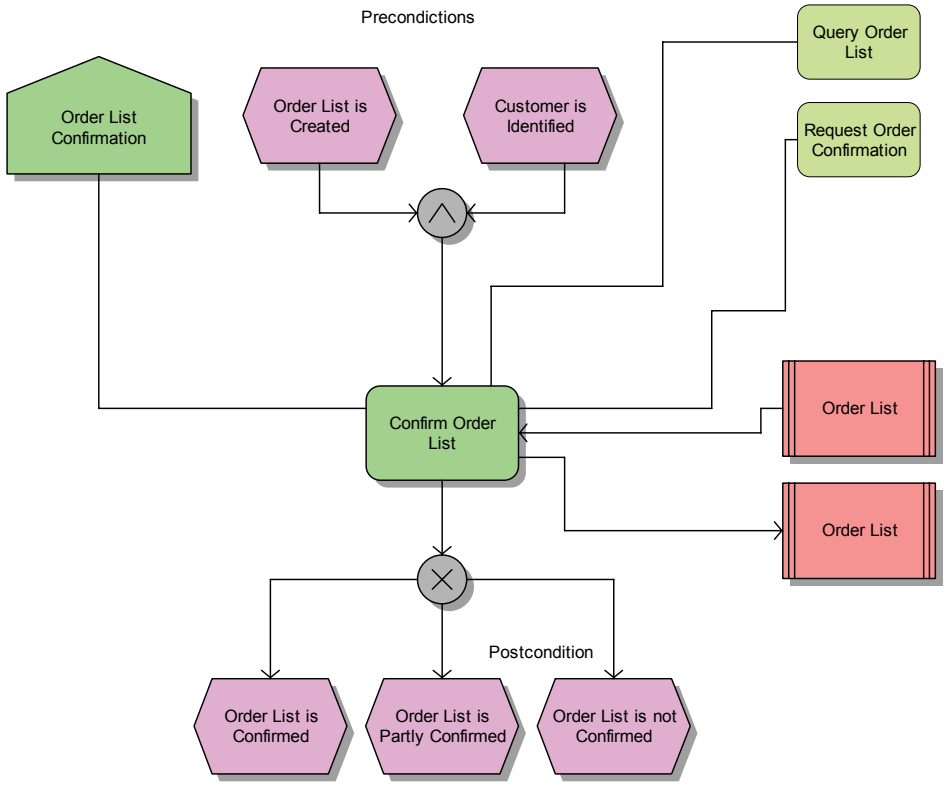



Figure 46 Envisioned Representation of WSMO Goals

This envisioned service browser shall also simplify the Goal selection. The idea is to provide a search engine that is aware of the concepts of the underlying ontology and allows a comfortable and guided search for appropriate Goals. It would also be possible to embed the idea of Goal visualization in this browser.

The developed method was developed along the provided VoIP showcase process. Although this process is taken from real-life it is not guarantied that the method and especially the performed implementations can be applied for the general case. The showcase process indeed covers many common EPC modelling constructs but for instance it does not include loop constructs. Also the restriction to binary decisions (see chapter 4.5.1.3) is not practical for a general usage and would have to be extended.

Furthermore, the method so far abstracts from exception handling. However, for a productive application of the method, exceptions have to be supported.

Solving these issues will help to raise usability of the developed semantic method and contribute to practical applicability of semantic technologies in Business Process Management.

# Appendix

## I. Case Study Semantic Tutorial

The following chapter contains the tutorial that was used by the participants to get involved with the new method. After accomplishing the tutorial the participants were interviewed according to the questionnaire of appendix II.

### I.1. General Information

Before the actual case study can be conducted, the following section provides some preparing information and general information about the used process and method.

#### I.1.1 Process Description

In the following section, the chosen process is explained in detail and its single steps are elaborated. Further, an overview of the different involved systems is shown.

**The process used in this tutorial was contributed by Telekomunikacja Polska (TP).** The process describes how a TP customer orders and activates the Voice over IP (VoIP) service. This service enables telephony via Internet protocol instead through conventional telephone protocols.

The process consists of the following steps:

1. Customer makes a request for VoIP telephony service on the TP web portal.

2. The portal system checks if the customer exists in customer database. If not he is informed about the situation and the order process is cancelled.

3. The system checks if the customer fulfils all requirements of the desired service. This check includes technical requirements and formal requirements. The customer must for instance have an appropriate DSL connection (technical) and must not have open bills to pay (formal). If the customer does not fulfil all requirements, he is informed,

which steps he has to do to get VoIP and the order process is cancelled.

4. After successful verification, a new order is created and detailed information about the desired service, conditions, and pricing are shown to the customer.

5. The customer confirms the order or rejects it. In the latter case the order process ends.

6. After confirming the order, the contract for the customer order is prepared and printed.

7. At the same time, the system checks if the customer has already the needed hardware (live box) to use VoIP. If not, the necessary hardware is prepared for shipping.

8. If the contract and the optional hardware are ready, it is shipped by a courier service to the customer. He signs the receipt of the hardware and the contract. The contract is then shipped back to TP.

9. When the signed contract arrives, it is archived and the order process continues.

10. After that, the billing system is informed and the VoIP service is activated. Now the order process is successfully completed.

There are five software systems involved in the order process.

1. The web portal, which serves as a front end to customer and initializes the order process.

2. The Customer Relationship Management System (CRM), which takes care about customer specific data and issues.

3. The Order Management System (OMS), which handles all customers' orders.

4. The Hardware Management System (HWM), which cares about all hardware specific topics.

5. The external Courier Service, which takes care of exchanging necessary hardware and documents.

Figure A 1 illustrates the VoIP ordering scenario with the involved roles and software systems.

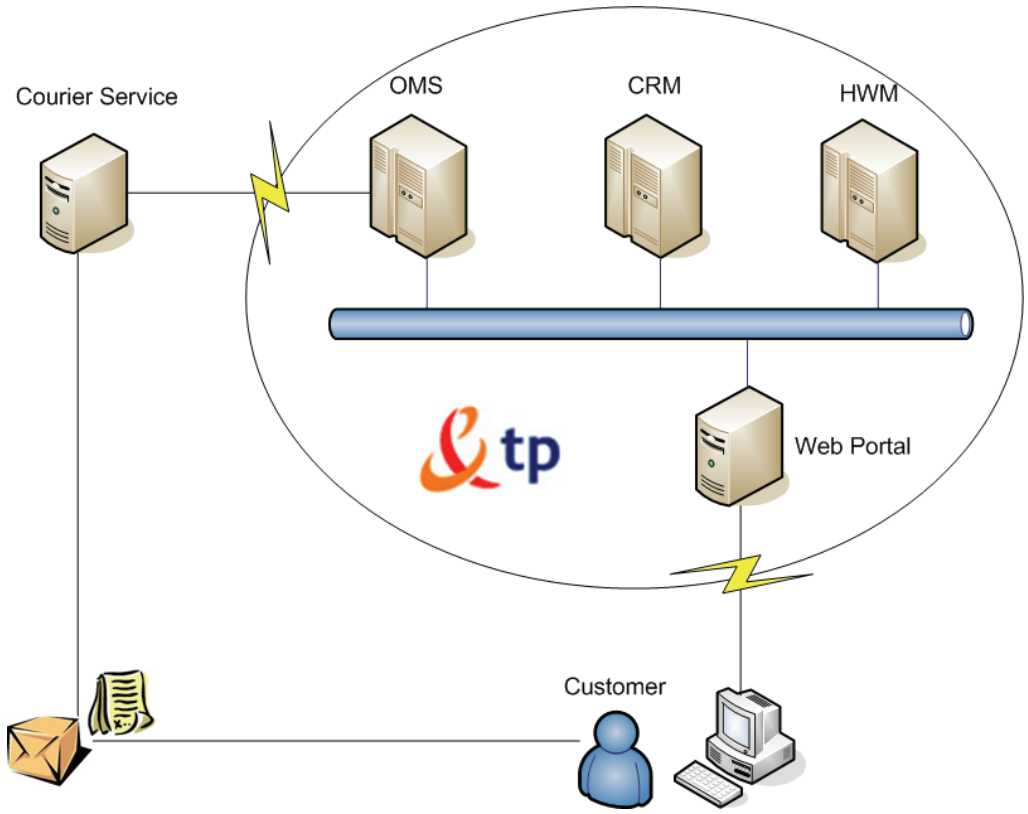



Figure A 1 System Overview

## I.1.2 EPC of Process

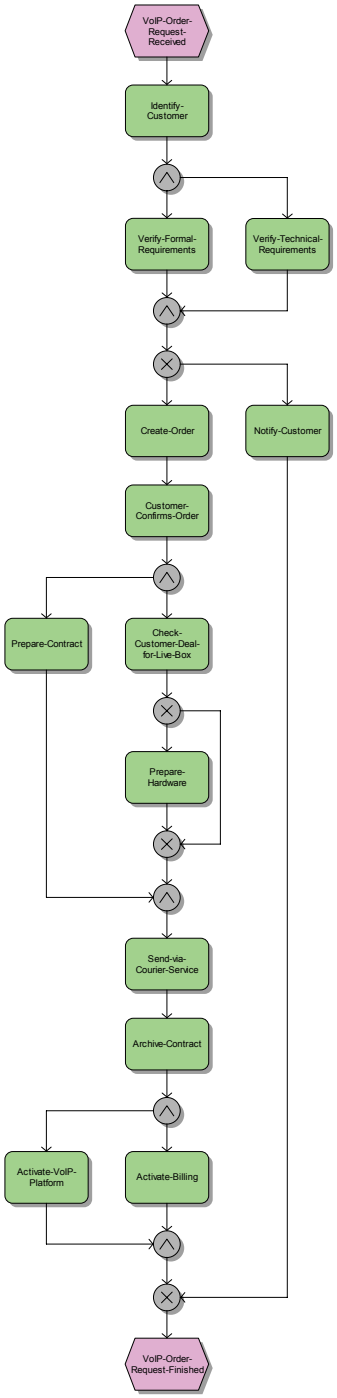



Figure A 2 EPC Process

| | |
|---|---|
| **Identify-Customer** | The system identifies the customer by means of the last name. |
| **Verify-Formal-Requirements** | The system checks if the customer has unpaid bills, etc. |
| **Verify-Technical-Requirements** | It is checked, whether the customer has the required technical requirements, e.g. an appropriate Internet connection for the specified desire of the customer. |
| **Create-Order** | The system creates a new order/customer case. |
| **Notify-Customer** | The customer is informed by the system. |
| **Customer-Confirms-Order** | The customer confirms the selected order. |
| **Prepare-Contract** | The contract for the confirmed order is prepared. |
| **Check-Customer-Deal-for-Live-Box** | It is checked if the customer has already an appropriate modem (Live Box), which is ready for VoIP. |
| **Prepare-Hardware** | The necessary hardware (Live Box) is prepared for shipping. |
| **Send-via-Courier-Service** | The necessary items (contract, possibly hardware) are being shipped. |
| **Archive-Contract** | The signed contract returns from customer and is archived by the system. |
| **Activate-Billing** | The system sends the bill to the customer. |
| **Activate-VoIP-Platform** | The desired VoIP service is activated for the customer. |

### I.1.3 Ontologies

The introduced semantic approach is founded on *ontologies*. An ontology can be seen as a collection of concepts of a specific domain comparable to a UML class diagram but on a pure functional level without any technical aspects. It provides a common vocabulary of the domain with clear semantics and serves as a shared communication basis. The main domain described for the case study process is the Polish Telecom (TP). The following listing shows a small extract out of the entire TP ontology. It only contains concepts, which are somehow relevant to the tutorial. It is very helpful to know the main concepts of this extract (especially *TP_Customer*, *CustomerCase* and *Contract*).

The extract of the ontology extract is shown as diagram in Figure A 3 and Listing A 1 shows it in textual form.


Figure A 3 Relevant Extract of TP Ontology

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-flight"
namespace { _"http://org.ipsuper.composition.tp/tpOntology#",
     wsmostudio _"http://www.wsmostudio.org#",
     dc _"http://purl.org/dc/elements/1.1#",
     wsml _"http://www.wsmo.org/wsml/wsml-syntax#"
}

ontology _"http://org.ipsuper.composition.tp/tpOntology"
     nonFunctionalProperties
          dc#title hasValue "Polish Telcom Ontology"
          dc#language hasValue "English"
          dc#subject hasValue "Some telecom terms as used in Tp"
          wsmostudio#version hasValue "0.6.0"
          dc#description hasValue "Polish Telco Ontology"
          wsml#version hasValue "0.1"
     endNonFunctionalProperties

concept Customer
     nonFunctionalProperties
```

```
            dc#description hasValue "An abstract notion for someone who
                                     pays for goods or services."
      endNonFunctionalProperties
      hasName impliesType Name
      nonFunctionalProperties
            dc#description hasValue "the name of the customer"
      endNonFunctionalProperties
      hasTelephoneNumber ofType _integer
      nonFunctionalProperties
            dc#description hasValue "a telephone number which customer
                                     pointed for contacts with him"
      endNonFunctionalProperties
      hasAccount ofType _integer
      nonFunctionalProperties
               dc#description hasValue "arrangement of subscribed ser-
                                     vices, payments, profile, etc. to a
                                     customer; it is also used for billing
                                     purposes; each customer account must
                                     have unique identification number"
      endNonFunctionalProperties

concept TP_Customer subConceptOf Customer
      nonFunctionalProperties
            dc#description hasValue "A customer who uses some TP ser-
                                     vices or intends to order additional
                                     TP service (products offered by TP)."
      endNonFunctionalProperties
      hasPIN impliesType _decimal
      nonFunctionalProperties
            dc#description hasValue "an id provided by TP for the
                                     customer"
      endNonFunctionalProperties
      hasCustomerID impliesType CustomerID

concept Service
      nonFunctionalProperties
            dc#description hasValue "An abstract notion for value pro
                                     visioning in some domain."
      endNonFunctionalProperties
      hasName impliesType Name
      hasID ofType _decimal
      nonFunctionalProperties
            dc#description hasValue "the name of the service."
      endNonFunctionalProperties
      isHardwarePrepared ofType _boolean
      isActive ofType _boolean

concept ServiceOrderRequest
      nonFunctionalProperties
            dc#description hasValue "A customer wish for a telcom
                                     service submitted to a contact cen-
                                     tre."
      endNonFunctionalProperties
      isCancelled impliesType _boolean
      hasStatus impliesType Status

concept CustomerCase subConceptOf ServiceOrderRequest
```

```
        description impliesType _string
        hasCaseID impliesType CaseID
        caseType impliesType _string
        customer impliesType TP_Customer
        requestedService impliesType Service

concept Contract
        nonFunctionalProperties
            dc#description hasValue "An abstract notion for an agreement
                                     between two or more parties, espe-
                                     cially one that is written and en-
                                     forceable by law."
        endNonFunctionalProperties
        hasID ofType _integer
        nonFunctionalProperties
             dc#description hasValue "the ID of the contract."
        endNonFunctionalProperties
        hasAgreement impliesType _string
        nonFunctionalProperties
             dc#description hasValue "a formal part of a contract con
                                     taining rights and duties for telecom
                                     company and a customer that are agreed
                                     between them as parties of an agree-
                                     ment"
        endNonFunctionalProperties
        isSigned ofType _boolean
        isApproved ofType _boolean

concept Verification
        nonFunctionalProperties
             dc#description hasValue "An abstract notion for the act of
                                     verifying or the state of being veri-
                                     fied."
        endNonFunctionalProperties
        hasVerificationID impliesType _decimal

concept TechnicalVerification subConceptOf Verification
        nonFunctionalProperties
             dc#description hasValue "Checking if there are technical
                                     possibilities to install a service in
                                     location pointed by a customer."
        endNonFunctionalProperties
        hasTechnicalVerificationResult impliesType _boolean
        nonFunctionalProperties
             dc#description hasValue "whether or not the result of tech-
                                     nical verification is positive (i.e.
                                     if there are technical possibilities
                                     to install a service)"
        endNonFunctionalProperties
        causedBy impliesType TechnicalProblem

concept FormalVerification subConceptOf Verification
        nonFunctionalProperties
             dc#description hasValue "Checking if there are formal pos-
                                     sibilities to install a service for a
                                     certain customer."
        endNonFunctionalProperties
```

```
        hasFormalVerificationResult impliesType _boolean
        nonFunctionalProperties
            dc#description hasValue "whether or not the result of for-
                                     mal verification is positive (i.e. if
                                     there are formal possibilities to in-
                                     stall a service i.e. a customer has no
                                     debts)."
        endNonFunctionalProperties

    concept Problem
        nonFunctionalProperties
            dc#description hasValue "An abstract notion for a question
                                     to be considered, solved, or ans-
                                     wered."
        endNonFunctionalProperties
        hasID impliesType _integer
        nonFunctionalProperties
            dc#description hasValue "an id associated with this prob-
                                     lem."
        endNonFunctionalProperties
        customerInformed impliesType _boolean
        customerToBeInformed impliesType Customer

    concept TechnicalProblem subConceptOf Problem
        nonFunctionalProperties
            dc#description hasValue "A lack of technical possibilities
                                     for installing a service requested by
                                     a client"
        endNonFunctionalProperties
```

Listing A 1 Relevant Extract of TP Ontology

As these illustrations lack of clarity (Listing A 1) and details (Figure A 3) additionally a UML class diagram is provided in Figure A 4.

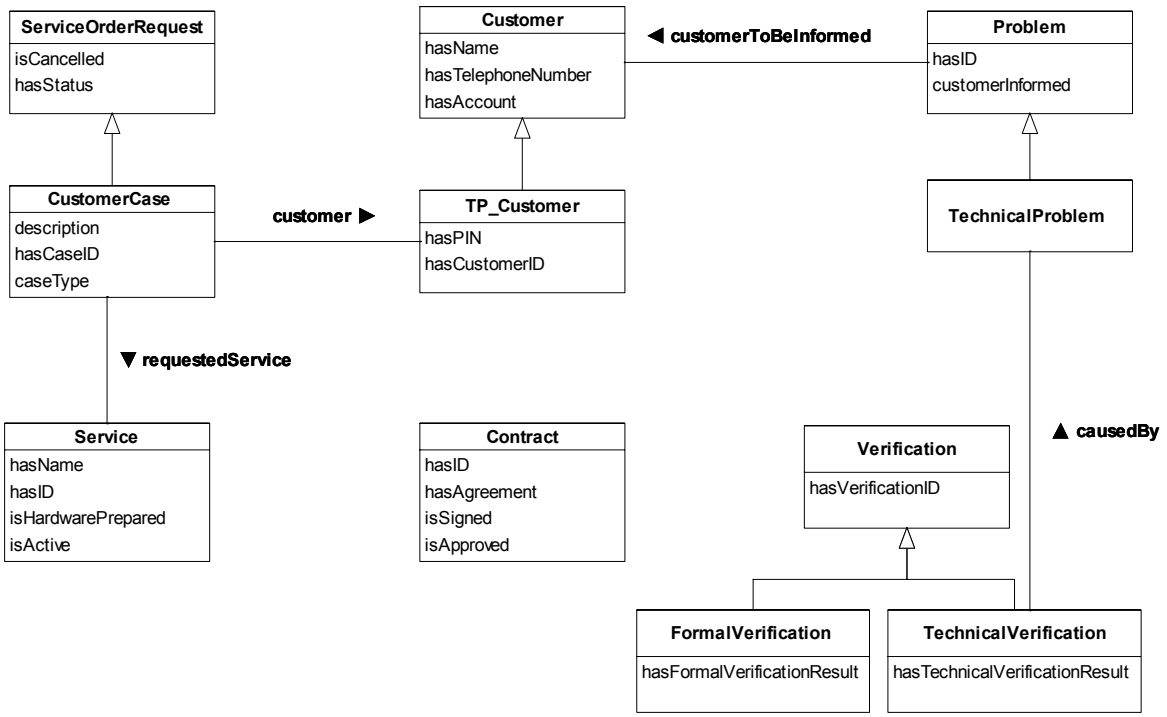



Figure A 4 Extract of TP Ontology - UML Class Diagram

### I.1.4 Semantic Goals

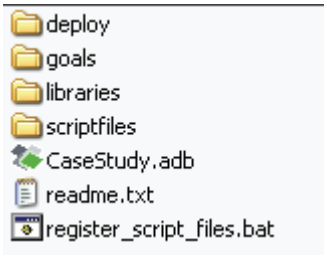
According to the non-semantic approach, the functions of the EPC process are represented by Web Services, which have to be assigned by the modeller. This core idea sill holds for the semantic approach, but instead of directly assigning concrete Web Services so called Goals have to be assigned to each function.

A Goal is a semantic description of a specific capability that a Web Service has to have in order to fulfil the needs of the function. These descriptions make use of the concepts defined in the TP ontology shown above.

## I.2. Preparation

**Requirements:** This tutorial is based on **ARIS SOA Architect 7.02**

1. Unzip the file into the root of drive *"C:\"*. The folder *"C:\IDS_CaseStudy\"* should have the following content:




2. Please note that is important to store the folder exactly at the mentioned position!

3. Make sure that you have closed ARIS SOA Architect.

4. Execute batch file "*register_script_files.bat"* to register the necessary macros and reports in ARIS SOA Architect

5. Start ARIS SOA Architect as usual.

6. Import the case study database. Therefore, switch to explorer view and right click on your installed server. Choose *"Restore"* and select database *"CaseStudy.adb".*

7. Make sure that you log in database in language *„English (US)"* with filter *„Entire Method".*

## I.3. Execution

The following sections contain a step by step description of the actual tutorial.

### I.3.1 Adding Semantic Annotations

The key idea of the semantic approach is to describe the requirements of the EPC functions with semantic descriptions as characterized in the Section above.

Therefore, it is necessary to enrich each function of the EPC with a semantic Goal. For this purpose right click on the function and select *“Evaluate”* => *„Start Macro…* (see Figure A 5). Select macro *“Add Goal”* in the next window (see Figure A 6), click *“Next”* and finally *“Finish”*. In the now appearing dialog (refer to Figure A 7) you are asked to select an appropriate WSMO Goal description file. After selecting a Goal additional artefacts are generated out of the Goal description and added to the EPC model. In general, there will appear input (*light gray*) and output instances (*dark grey*) and an object representing the capability (see Figure A 8). You can see the name of the instance and the type of the instance separated by “::”.

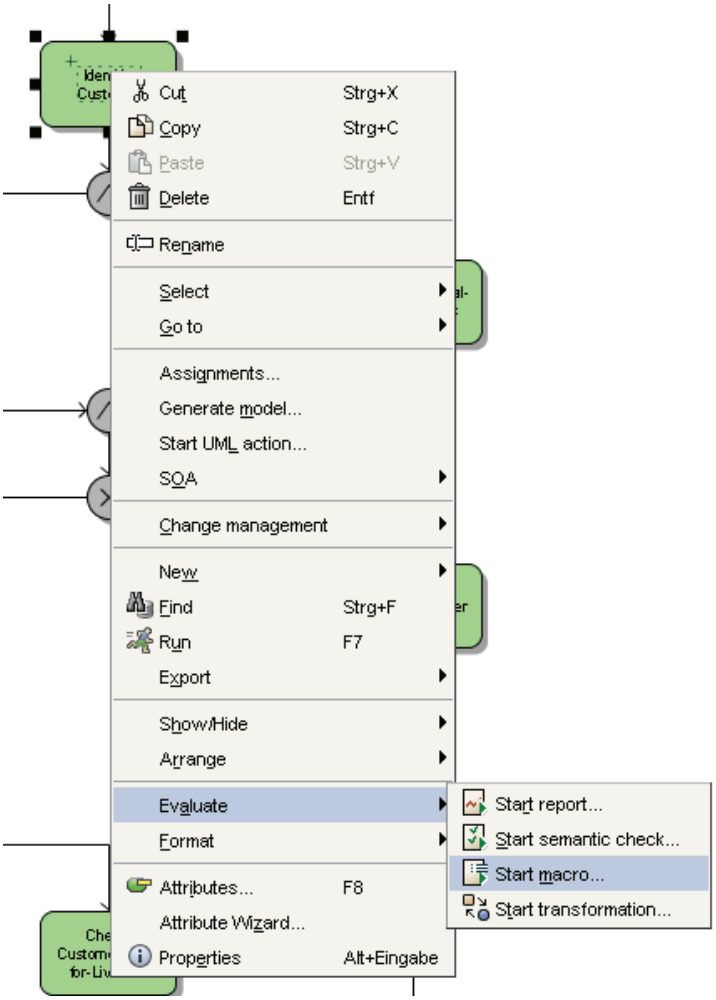



Figure A 5 Context Menu: Start Macro

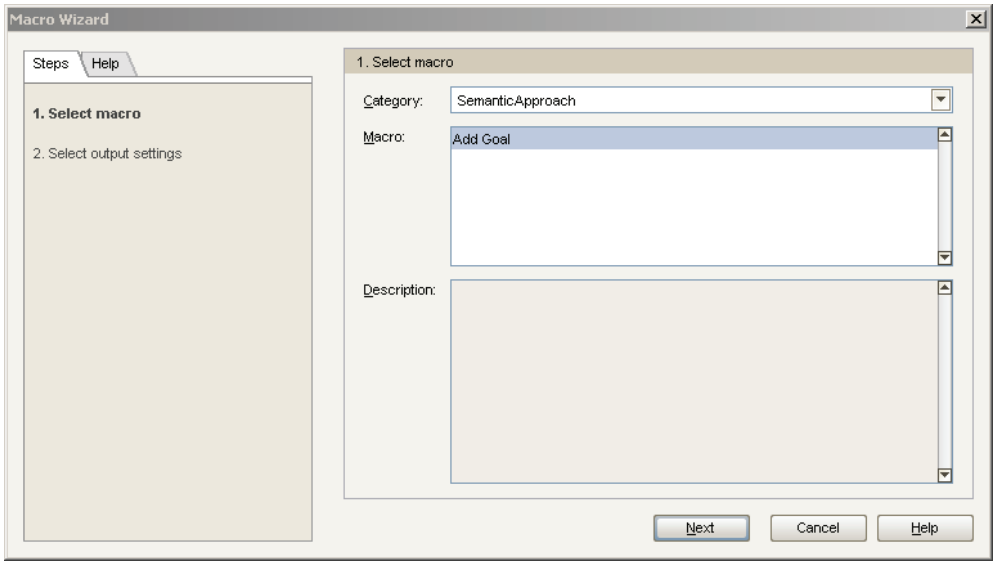



Figure A 6 Select Macro "Add Goal"

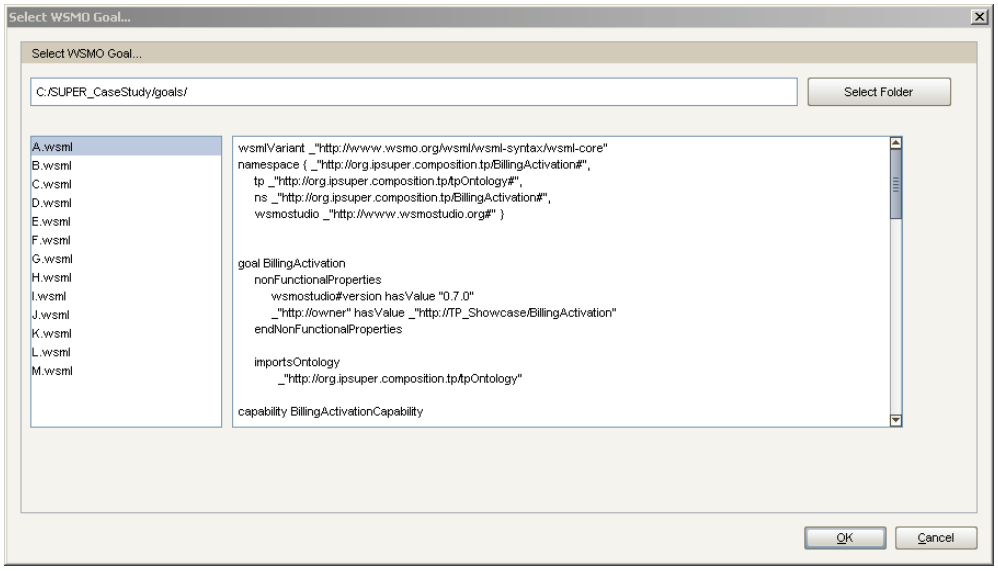



Figure A 7 Goal Selection Dialog

Selecting the right Goal for a given function is a non-trivial task. On the one hand, you must be able to link the informal description of the process function with the vocabulary of the domain ontology. On the other hand, you must compare the context of the function with the content of a specific Goal.

For the present showcase process, this procedure is explained by means of the first function *"Identify-Customer"*. The informal description of the EPC function was given in the introducing chapter: *"The portal system checks if the customer exists in customer database"*. Now you need to compare this requirement with the available Goal descriptions. The file *"E.wsml"* looks as follows:

```
goal CustomerIdentification
     nonFunctionalProperties
          wsmostudio#version hasValue "0.7.0"
          _"http://owner" hasValue
_"http://TP_Showcase/CustomerIdentification"
     endNonFunctionalProperties

     importsOntology
            _"http://org.ipsuper.composition.tp/tpOntology"

capability CustomerIdentificationCapability

sharedVariables ?customer

precondition pr
     definedBy
          ?customer[hasName hasValue ?name] memberOf tp#TP_Customer
  and ?name memberOf tp#Name.

postcondition po
     definedBy
          ?customer[hasCustomerID hasValue ?customerID] memberOf
tp#TP_Customer
  and ?customerID memberOf CustomerID.

interface CustomerIdentificationInterface
     choreography CustomerIdentificationChoreography
     stateSignature CustomerIdentificationStateSignature
          importsOntology
            _"http://org.ipsuper.composition.tp/tpOntology"

       in concept ns#CustomerIdentificationRequest
          out concept ns#CustomerIdentificationResponse
...
```

Listing A 2 Content of File E.wsml

For selecting the Goal especially the pre- and postcondition information are important. In the current case the precondition only says, that in the instance customer the field *"hasName"* must be specified. The postcondition ensures however that the customer was identified correctly, as his corresponding ID is now filled, which is exactly what we need here.

So the corresponding Goal for this function is called *"CustomerIdentification"* and is located in file *"E.wsml"*.

Note that the Goal descriptions make use of the introduced domain ontology of TP. If the meaning of a concept or an attribute is unclear, please take a look at the provided ontology extract.

After adding Goal *"CustomerIdentification"* to function *"Identify-Customer"* the situation looks like in Figure A 8. As you can see, the function was enriched with one input instance *"customerName"* of type *"string"* and one output instance *"customer"* of type *"TP_Customer"* as well as with a symbol *"CustomerIdentification"* that represents the Goal itself.

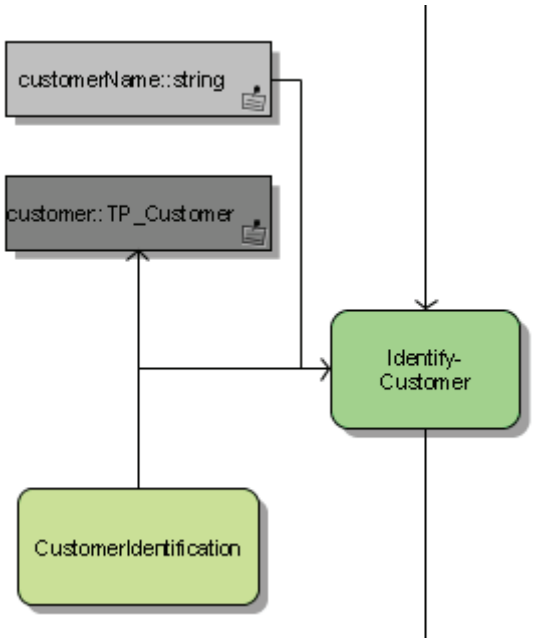



Figure A 8 After Adding Goal to Identify-Customer

As next tasks, the other functions of the EPC must be enriched, which works exactly as described above.

### I.3.2 Modelling the Data Flow

If all functions are enriched with semantic annotations, you have to **model the data flow** in the model. There are various steps to do here:

1. You have to **map the output instances** of functions to **the corresponding input instances** of other functions. Therefore, you have to

select exactly one input (light gray) and one output instance (dark gray; hold “Control” to select second instance) and then start macro *“Map Instances”* (see Figure A 9 and Figure A 10). As result a new model is created and assigned with the selected input instance, recognizable by the small icon in the lower right of the instance (refer to Figure A 11). By double clicking the symbol, you can take a look at the linked model (see Figure A 12), which details the created instance mapping.

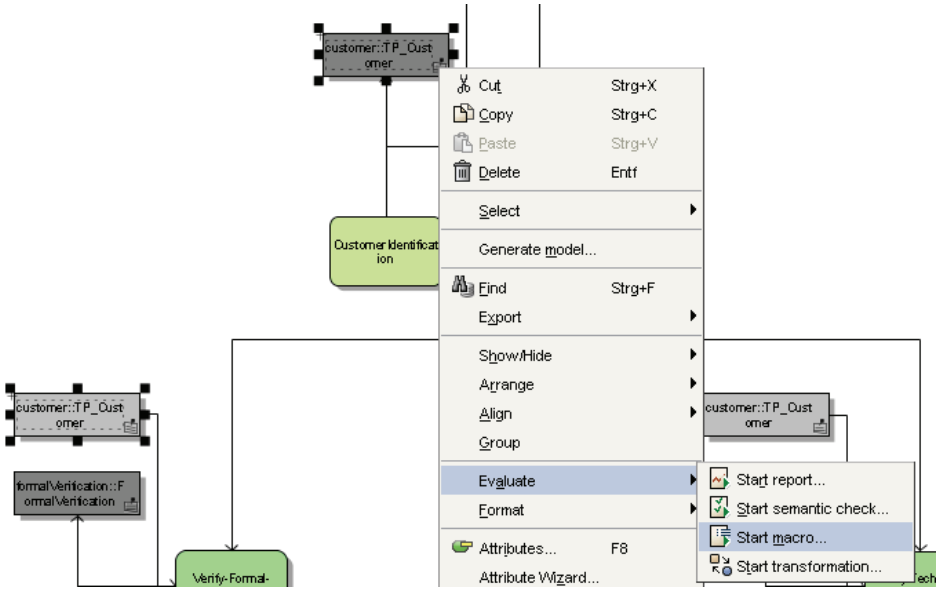



Figure A 9 Map Instances Context Menu

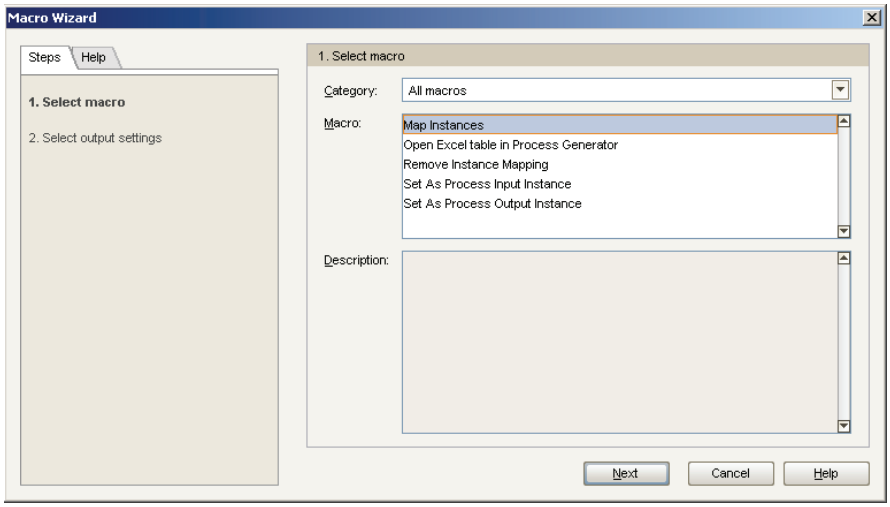



Figure A 10 Macro Wizard "Map Instances"

The explained procedure is now exemplified by means of the showcase process:
When looking at the function *"Verify-Formal-Requirements"*, you can see that this needs *"customer"* of type *"TP_Customer"* as input instance. Such an object is delivered by the preceding function *"Identify-Customer"*. Therefore, select the output instance *"customer"* of *"Identify-Customer"* and the input instance *"customer"* of function *"Verify-Formal-Requirements"* and run macro "*Map Instances*".
Note that in a real process the mapping will be revealed from the context. Although it is not explicitly required, you should only assign instances of the same type.

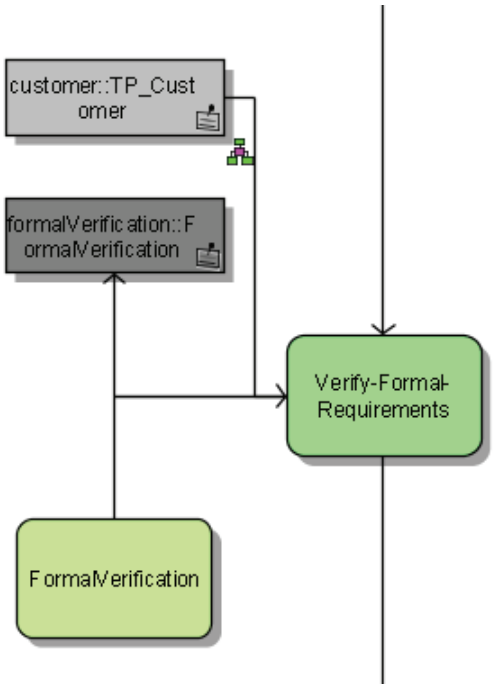



Figure A 11 Verify-Formal-Requirements after Instance Mapping

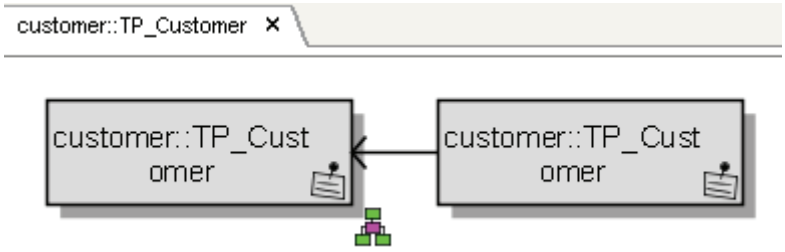



Figure A 12 Assigned Model of Input Instance

2. Then, you have to **assign the input instance** of the whole process. Therefore, select an input instance and start the macro *“Set As Process Input Instance”*.

3. In the current process, you need to assign instance *“customerName”* of function *“Identify-Customer”* as process input instance.

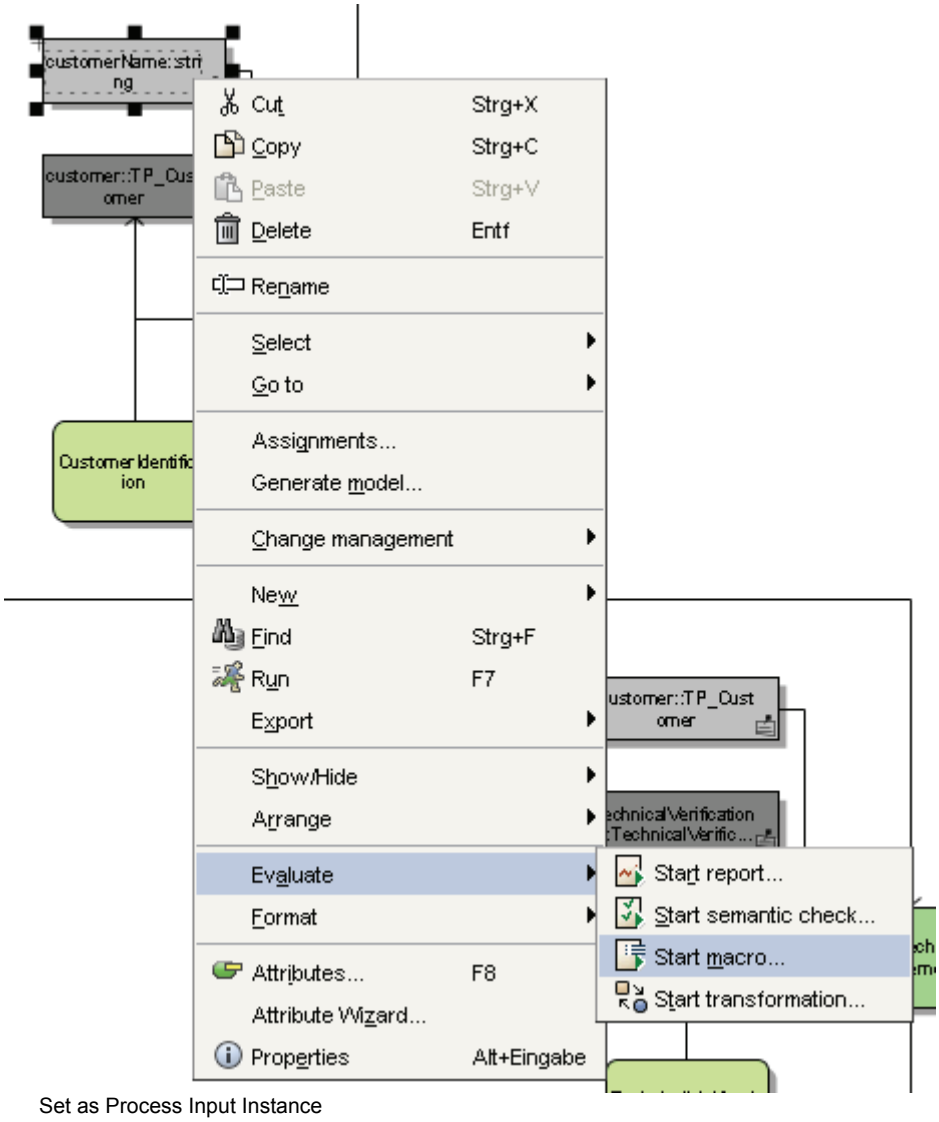



Figure A 13 Set as Process Input Instance

4. As last step here, you have to **assign the process output instance**. Analogously to the previous step, you have to select an output instance and select *“Set As Process Output Instance”*.

5. In the current process, you need to assign the output instance *"customerCase"* of function *"Activate-VoIP-Platform"* as process output instance.

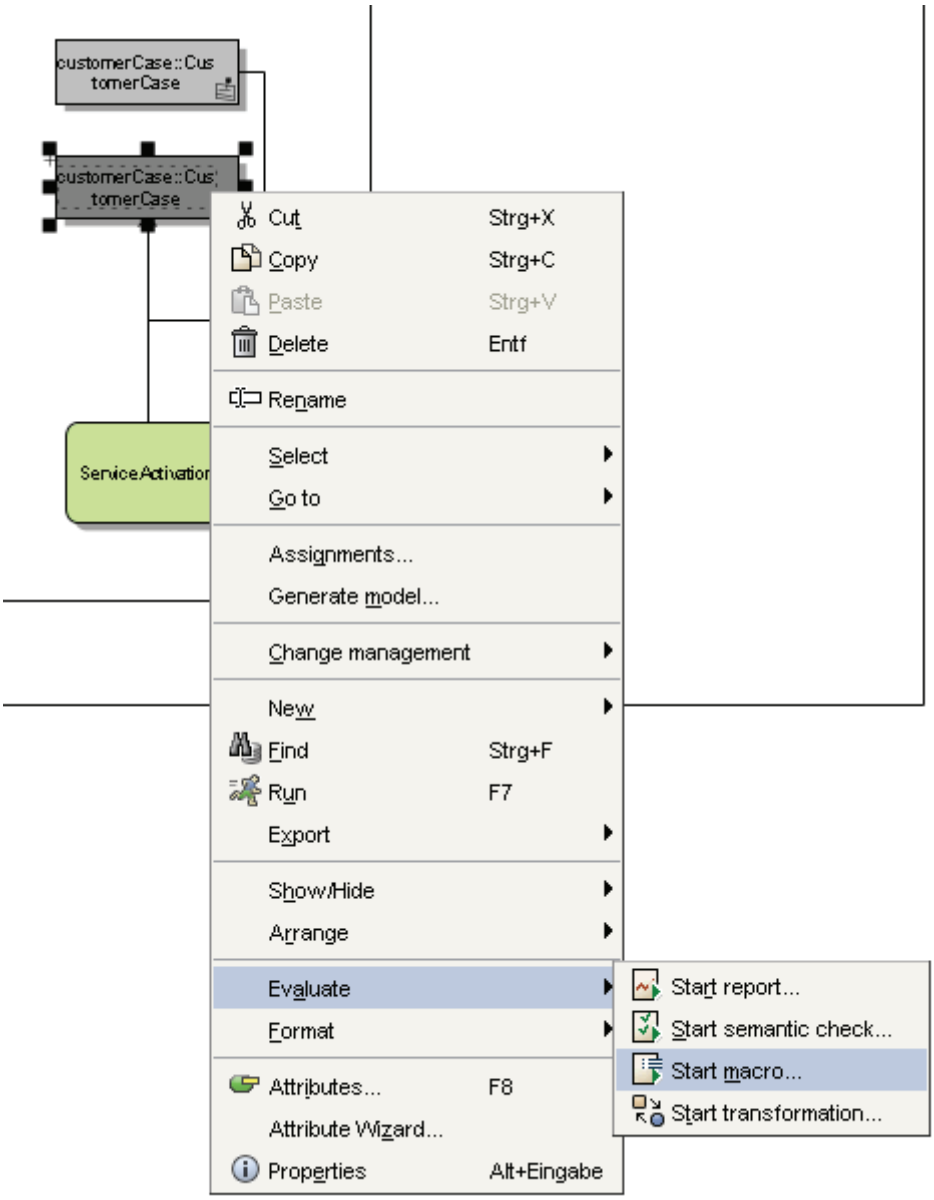



Figure A 14 Set as Process Output Instance

6. If you **have assigned two instances by mistake**, you can select *the input instance* and run macro *"Remove Instance Mapping"*. The same holds for incorrectly assigned process input or output instance. There you also have to the select the corresponding instance and run the mentioned macro.

### I.3.3 Completing the Control Flow

As the switch conditions for splitting paths of the EPC are still missing, the control flow is not complete yet. You now add further information onto splitting XOR operators to **complete the control flow**. To do this right click on the respective XOR and run macro *"Add Decision"*.

After running the macro, you will see a situation like in Figure A 15. The macro generates a new function *("Decision.Of.XYZ")* with one input instance *("Decision.Of.XYZ.InputVar")* and an information object, which represents the condition of the XOR. You now need to assign the input instance of the decision and the condition expression. Assigning the instance works analogously to chapter I.3.2.

The condition expression is set by simply replace *"INSERT CONDIDTION HERE"* with the respective condition expression. At runtime, it will be checked if the condition holds for the specified instance. If it holds, the path including the *"TRUE"* event will be executed, otherwise the path with the *"FALSE"* event. As the macro assigned randomly these two events, it may be necessary to exchange them. In order to do so simply rename the events from *"TRUE"* to *"FALSE"* and vice versa in the other event.

In the present showcase process, there are two XOR operations, on which we have to assign decisions. The first one is located after functions *"Verify-Formal-Requirements"* and *"Verify-Technical-Requirements"* and has the name *"XOR_Verification"*. After running *"Add Decision"* you need to do here:

- Map the output instances "formalVerification" and "technicalVerification" of functions "Verify-Formal-Requirements" and "Verify-Technical-Requirements" onto input instance "Decision.Of.VERIFICATION.InputVar" of the decision. As you have here **two** input instances that have to be mapped on **one** input instance, you must run macro "Map Instances" twice, one time on "formalVerification" and "Decision.Of.VERIFICATION.InputVar" and one time on "technicalVerification" and "Decision.Of.VERIFICATION.InputVar".
- Add the condition expression. As the mechanism for evaluating this expression is a dummy implementation, it is given here and can be simply copied and pasted into the information object:
- "`hasFormalVerificationResult hasValue 1 AND hasTechnicalVerificationResult hasValue 1`".

- The inserted expression is shown in Figure A 16.
- Verify that the events (TRUE and FALSE) are placed on the right paths of the XOR. In Figure A 15 this is not the case. Hence, it is necessary to swap the events by renaming them. The result is shown in Figure A 16.

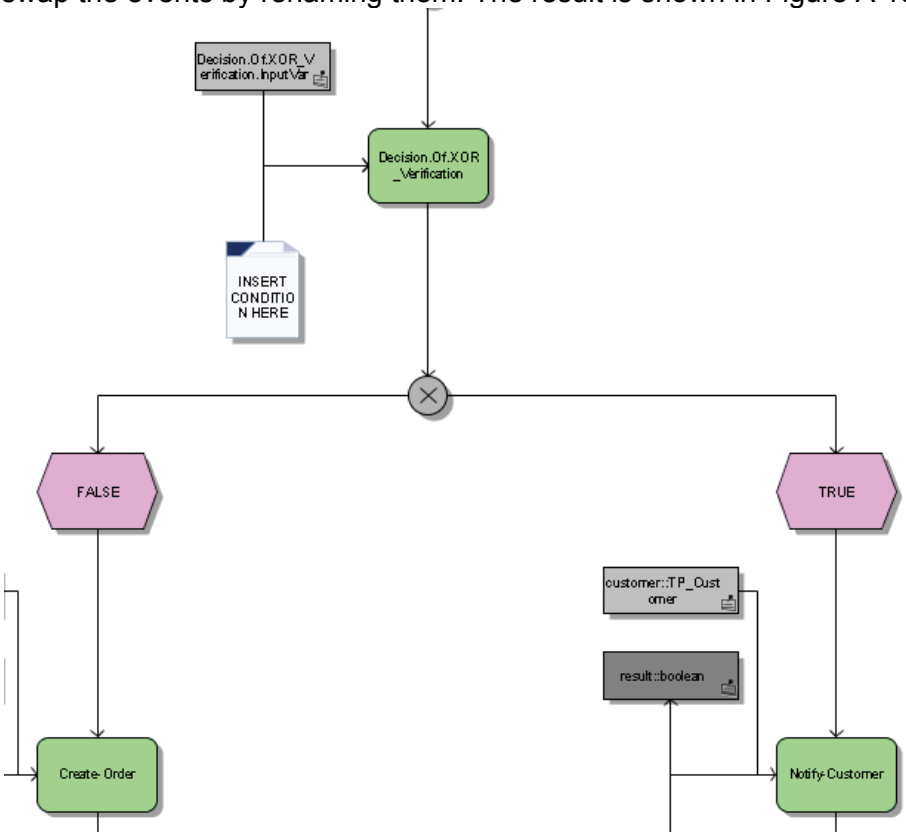



Figure A 15 After "Add Decision" 1/2

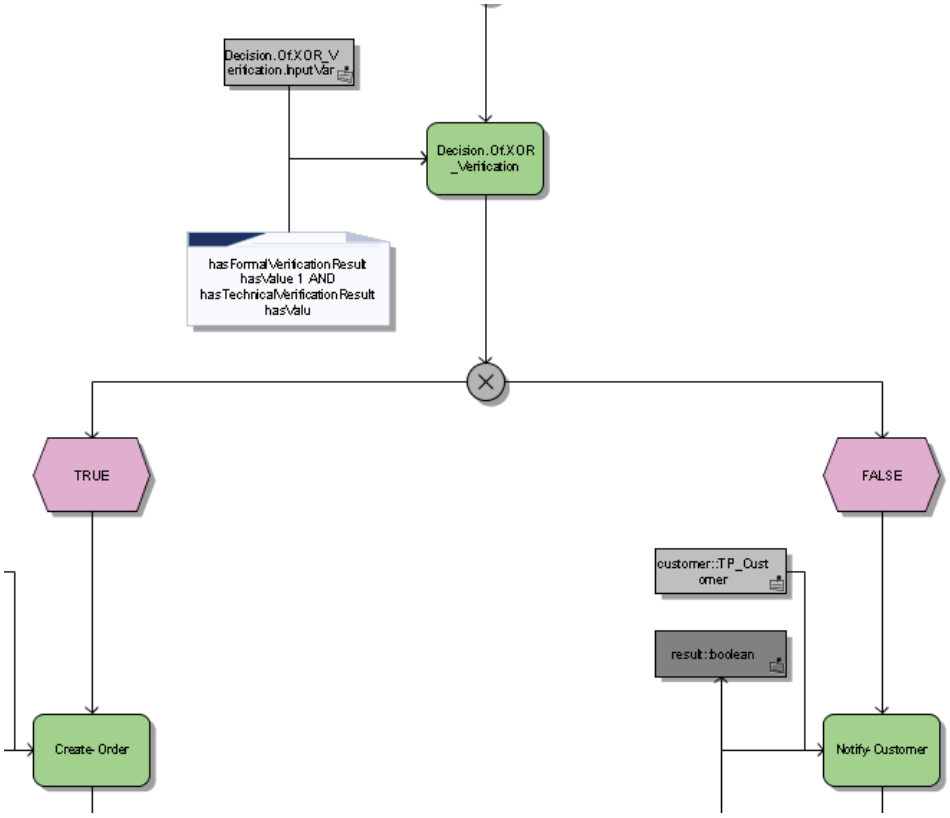



Figure A 16 After "Add Decision" 2/2

The second XOR with name “XOR_Hardware_Available” is located after function “Check-Customer-Deal-for-Live-Box”:

- Map output instance “result” of function “Check-Customer-Deal-for-Live-Box” onto input instance “Decision.Of. XOR_Hardware_Available.InputVar” of the decision.
- Add the condition expression. It is also completely given for copy & paste:
- “`hasTechnicalVerificationResult hasValue true`”
- Check whether the events are located in the right paths.

### I.3.4 Running the Transformation

After you assigned all Goals and decisions and you mapped all instances, you can **start the transformation** of the EPC into the executable BPEL code. For this purpose you have to make sure that no object is selected in EPC model. Then, click right and run macro *“Transform Into SISi BPEL”*. As result, you will receive a corresponding BPEL model.

### I.3.5 Export the BPEL Code

**Export the generated BPEL model**. To do so, go to the new model and select in context menu *“SOA” => “Export BPEL process”*. Now specify a path where to export the zipped code files of the process. This zip file includes the actual BPEL code (*“showcase.bpel”*) and the interface description to invoke the process (*“showcase.wsdl”*). Furthermore, the zip file contains all partner-linktypes of the invoked services.

**End of Tutorial**

## I.4. Outlook

After accomplishing the actual tutorial, this section shall provide an outlook how the modelled process is being deployed and executed.

The exported BPEL code can be automatically deployed on an orchestration engine like ORACLE BPEL process manager, which is covered by a macro called *"Deploy generated SISi BPEL"*.

The overall runtime scenario can be sketched as follows:

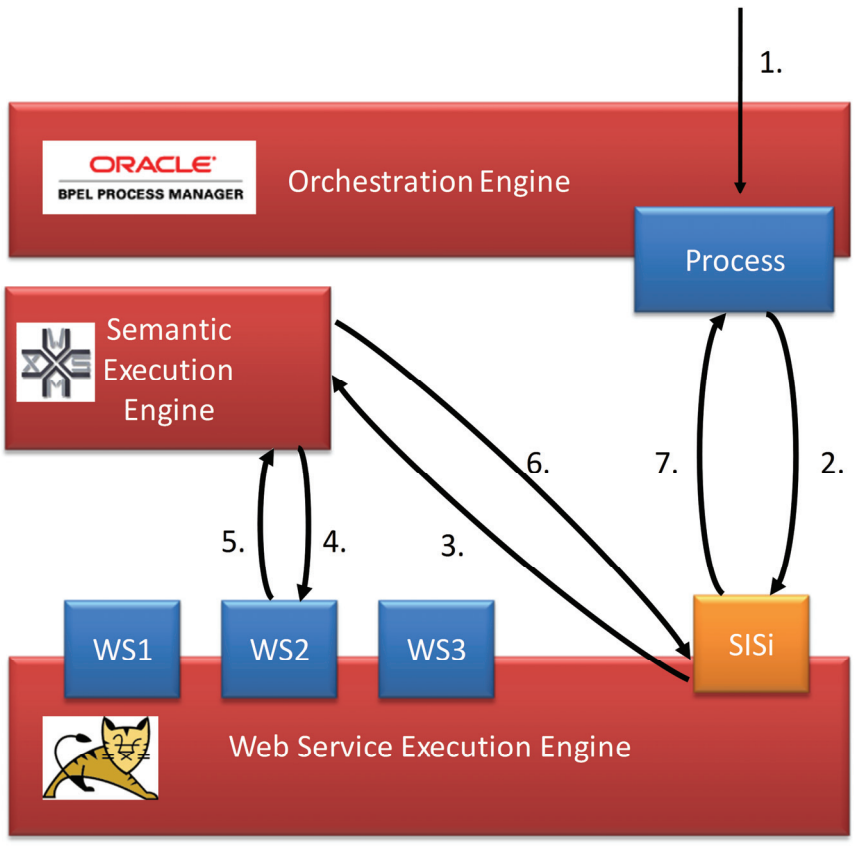



Figure A 17 Overall Runtime Scenario

1. After the process was deployed on the orchestration engine as described above, the execution of the process is triggered and a new process instance is started.

2. When the instantiated process reaches an invoke activity (corresponds to a function of the EPC process), a proxy Web Service (Semantic Invocation Service, SISi) is called.

3. SISi receives the semantic discovery request and additionally the semantic instance data (input for the Web Service) passes the semantic Goal description to the semantic execution engine (SEE).

4. On basis of the semantic Goal, the semantic execution engine discovers an appropriate Web Service and invokes it with the semantic instance data. The semantic data therefore has to be transferred into syntactical data, which is process-able by the Web Service.

5. After the invocation of the concrete Web Service, the result is passed back to the SEE.

6. The SEE transfers the syntactic output data into its ontological representation and passes it back to SISi.

7. SISi receives the ontological output data of the Web Service and delegates it back to the BPEL process.

# II. Case Study Interview Questionnaire

This section contains the questionnaire, with which the interviews were conducted.

## II.1. Introduction

- Introduce yourself if this is unknown
- Thank for participation and invested time
- Summarize the goal of the case study: compare the standard method with the semantic method
- Mention context: study is part of the big European research project SUPER, which works on the vision of Semantic BPM
- This study is only a small part of the project and is concerned about the evaluation of the first approaches

## II.2. Notification of Confidentiality

- Content of interview will be handled confident
- Notes oft he interviews are not stored personalized
- There is no voice recording
- There will be no evaluation concerning age, gender, or educational background
- Only two scientists have access to the interview notes
- Goal oft he interview:
  - Evaluation of the new approach by the interviewee
  - The interview shall not be considered as a test
  - A open discussion is as far as possible desired
  - Open critique is desired
- Benefit for the interviewee:
  - If interested, the interviewee can have the published scientific paper and the book chapter, which describes the background of the case study more closely. *These documents were consciously deprived of the participants in order to not influence them by the included argumentation and to raise internal validity!*
  - The case study is going to be replicated about 10 times and the results will be evaluated in a scientific paper

## II.3. Questions

1. Then, we can start. Do you have questions concerning the interview or the background?

2. For the beginning, I would like you to describe the procedure of the tutorial from your point of view.

3. How would you define “semantic description” or “Goal”?

4. In your own words, what is an ontology?

5. The tutorial introduced the domain ontology. Did you fall back on this ontology during accomplishment of the study?

6. Which one of the three ontology representation in the tutorial (star, WSML, UML) did you use?

7. How works the assignment of services to the functions in tutorial?

8. How are the semantic descriptions/Goals represented? What contains this description?

9. How sure were you whether you selected the right Goal?

10. How helpful were the pre- and postconditions of the Goals during the selection?

11. One step of the tutorial was about the data flow. What was the sense of this step?

12. Did you lost track during the procedure?

13. When looking at the overall procedure, what is the main difference to the standard method in ARIS SOA Architect?

14. Does dynamic binding makes any sense for you? Is there a business motivation for this feature?

15. Does the new method provide a simplification compared to the standard approach?

16. Do you think that the new semantic method can be accomplished by a typical EPC modeller/business analyst? Why?

17. How long did you need to accomplish the tutorial?

18. Was the description of the tutorial sufficient?

19. Can you send us the used database? We would like to see, whether there were divergences from the intended procedure.

**Thank you for your participation!**

# Table of Figures

# Table of Listings

# References


[ACD+03] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. Business Process Execution Language for Web Services Version 1.1. Technical report, BEA, IBM, Microsoft, SAP AG, Siebel Systems, 2003.

[BCD+07] Roxana Belecheanu, Liliana Cabral, John Domingue, Walid Gaaloul, Martin Hepp, Agata Filipowska, Monika Kaczmarek, Tomasz Kaczmarek, Jörg Nitzsche, Barry Norton, Carlos Pedrinaci, Dumitru Roman, Michael Stollberg, and Sebastian Stein. SUPER d 1.1 - Business Process Ontology Framework, 2007.

[BCD+08] Joachim Bayer, Emilia Cimpian, Steffi Donath, Michael Eisenbarth, Thomas Hering, Dominik Kuropka, Marek Kowalkiewicz, Andre Ludwig, Guido Laures, Theresa Lehner, Harald Meyer, Mariusz Momotko, Andreas Polze, Kai Petersen, Christoph Ringelstein, Dumitru Roman, Adina Sirbu, Christian Stamber, Jan Schaffner, Nathalie Steinmetz, Steffen Staab, Sebastian Stein, Ioan Toma, Peter Tröger, and Mathias Weske. *Semantic Service Provisioning*. Springer, 2008.

[Bea05] Ben Bloch and et al. Web Services Business Process Execution Language Version 2.0. Technical report, OASIS, 2005.

[BEK+00] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. Simple Object Access Protocol (SOAP) 1.1. W3C Note, 2000.

[BG04] Dan Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, February 2004.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–44, 2001.

[BMR+96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. John Wiley &

Sons, August 1996.

[Car04] N. G. Carr. *Does IT matter? Information Technology and the Corrosion of Competitive Advantage*. Harvard Business School Press, 2004.

[CCMW01] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web Services Description Language (WSDL) 1.1. W3C Note, 2001.

[CD99] James Clark and Steve DeRose. XML Path Language (XPath) version 1.0. W3C Recommendation, November 1999.

[CFNO02] Michael Champion, Chris Ferris, Eric Newcomer, and David Orchard. Web services architecture. W3C Working Draft, November 2002.

[Che76] Peter Pin-Shan Chen. The Entity-Relationship Model–Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1976.

[Cla99] James Clark. XSL Transformations (XSLT) Version 1.0. W3C Recommendation, November 1999.

[dBLK+05] Jos de Bruijn, Holger Lausen, Reto Krummenacher, Axel Polleres, Livia Predoiu, Michael Kifer, and Dieter Fensel. *The Web Service Modeling Language WSML—WSML Final Draft 5 October 2005*. Digital Enterprise Research Institute (DERI), October 2005. WSMO Final Draft D16.v0.21.

[Dem82] W. E. Deming. *Out of the Crisis*. MIT Press, 1982.

[DvdA04] J. Dehnert and W.M.P. van der Aalst. Bridging Gap between Business Models and Workflow Specifications. *International Journal of Cooperative Information Systems 13*, pages 289 – 332, 2004.

[EAA+04] Mark Endrei, Jenny Ang, Ali Arsanjani, Sook Chua, Philippe Comte, Pål Krogdahl, Min Luo, and Tony Newling. *Patterns: Service-Oriented Architecture and Web Services*. IBM, 2004.

[Ecm99] Ecma International. Ecmascript language specification. Standard ECMA-262, December 1999.

[Erl07] Thomas Erl. *SOA: Principles of Service Design*. Prentice Hall, 2007.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer protocol – HTTP/1.1. Network Working Group Request for Comments: 2616, June 1999.

[Fie00] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, USA, 2000.

[FL07] J. Farrell and H. Lausen. *Semantic Annotations for WSDL and XML Schema (SAWSDL)*. World Wide Web Consortium (W3C), August 2007.

[FLP+06] Dieter Fensel, Holger Lausen, Axel Polleres, Jos de Bruijn, Michael Stollberg, Dumitru Roman, and John Domingue. *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Springer, 2006.

[GPCFL04] Asuncion Gomez-Perez, Oscar Corcho, and Mariano Fernandez-Lopez. *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-commerce and the Semantic Web (Advanced Information and Knowledge Processing)*. July 2004.

[Gru93] Thomas Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 2(5):199–220, 1993.

[HB04] Hugo Haas and Allen Brown. Web Services Glossary. W3C Working Group Note, 2004.

[HLD+05] Martin Hepp, Frank Leymann, John Domingue, Alexander Wahler, and Dieter Fensel. Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. In *IEEE International Conference on e-Business Engineering (ICEBE 2005)*, pages 535–540, Beijing, China, 2005.

[HR07a] Martin Hepp and Dumitru Roman. An Ontology Framework for Semantic Business Process Management. In A Oberweis, C. Weinhardt, H. Gimpel, A. Koschmider, V. Pankratius, and B. Schmizler, editors, *eOrganisation: Service-, Prozess, Market-Engineering*, volume 1 of *Proceedings of the 8th International Conference Wirtschaftsinformatik 2007*, pages 423–440, Karlsruhe, February 28 - March 2 2007. Universitaetsverlag Karlsruhe.

[HR07b] Martin Host and Per Runeson. Checklists for Software Engineering Case Study Research. *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, pages 479–481, 2007.

[KHSW05] J. Koehler, R. Hauser, S. Sendall, and M. Wahler. Declarative Techniques for Modeldriven Business Process Integration. *IBM Systems Journal 44*, pages 47 – 65, 2005.

[KL04] Donald Kossmann and Frank Leymann. Web Services. *Informatik Spektrum*, 27(2):117–128, 2004.

[KLP+04] Uwe Keller, Ruben Lara, Axel Polleres, Ioan Toma, Michael Kifer, and Dieter Fensel. WSMO D5.1 v0.1 WSMO Discovery. WSML Working Draft, September 2004.

[KNS92] G. Keller, M. Nüttgens, and August-Wilhelm Scheer. Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK). Arbeitsbericht Heft 89, Institut für Wirtschaftsinformatik Universität Saarbrücken, 1992.

[KPP95] Barbara Kitchenham, Lesley Pickard, and Shari Lawrence Pfleeger. Case Studies for Method and Tool Evaluation. *IEEE Software*, pages 52–62, 1995.

[KtHvdA03] B. Kiepuszewski, A.H.M. ter Hofstede, and W.M.P. van der Aalst. Fundamentals of Controlflow in Workflows. *Acta Informatica 39*, pages 143 – 209, 2003.

[Ley04] F. Leymann. *The Influence of Web Services on Software: Potentials and Tasks*. Springer, 2004.

[MC05] Adrian Mocan and Emilia Cimpian. WSMX D13.3v0.2 WSMX Data Mediation, 2005.

[MCS+05] Adrian Mocan, Emilia Cimpian, Michael Stollberg, Francois Scharffe, and James Scicluna. WSMO - D29v0.2 WSMO Mediators, 2005.

[MI96] Riichiro Mizoguchi and Mitsuru Ikeda. Towards Ontology Engineering, 1996.

[MLM+06] C. Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter F Brown, Rebekah Metz, and Booz Allen Hamilton. OASIS Reference Model for Service Oriented Architecture 1.0. OASIS, 2006.

[MNN05] Jan Mendling, Gustaf Neumann, and Markus Nüttgens. Towards Workflow Pattern Support of Event-Driven Process Chains (EPC). In M. Nüttgens and J. Mendling, editors, *XML4BPM 2005, Proceedings of the 2nd GI Workshop XML4BPM – XML Interchange Formats for Business Process Management at 11th GI Conference BTW 2005, Karlsruhe Germany, March 2005*, pages 23–38, http://wi.wu-wien.ac.at/mendling/XML4BPM2005/xml4bpm-2005-proceedings-mendling.pdf, March 2005.

[MvH04] Deborah L. McGuinness and Frank van Harmelen. Owl web ontology language overview. W3C Recommendation, February 2004.

[OMG03] OMG. MDA Guide Version 1.0.1. Object Management Group Secification, June 2003.

[OMG06] OMG. Business Process Modeling Notation Specification. Object Management Group Secification, February 2006.

[OMG07] OMG. OMG Unified Modeling Language (OMG UML), Infrastructure, V2.1.2, November 2007.

[otUS02] Congress of the United States. Public company accounting reform and investor protection act (sarbanes-oxley act), 2002. Pub. L. No. 107-204, 116 Stat. 745.

[Rau06] Till Rausch. Service Orientierte Architektur - Übersicht und Einordnung, 2006.

[SBEK07] Sebastian Stein, Katja Barchewitz, and Marwane El Kharbili. Enabling Business Experts to Discover Web Services for Business Process Automation. In Cesare Pautasso and Thomas Gschwind, editors, *2nd Workshop on Emerging Web Services Technology*, pages 19–35, Halle, Germany, November 2007.

[Sch96] August-Wilhelm Scheer. ARIS-House of Business Engineering: Von der Geschäftsprozeßmodellierung zur Workflow-gesteuerten Anwendung, September 1996.

[SF03] Howard Smith and Peter Fingar. *Business Process Management: The Third Wave*. Meghan-Kiffer Press, Tampa, FL, USA, 1st edition, 2003.

[SI07a] S. Stein and K. Ivanov. Vorgehensmodell zur Entwicklung von

Geschäftsservicen. In Klaus-Peter Fähnrich and Maik Thränert, editors, *Integration Engineering – Motivation, Begriffe, Methoden und Anwendungsfälle*, Leipziger Beiträge zur Informatik VI. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, 2007.

[SI07b] Sebastian Stein and Konstantin Ivanov. EPK nach BPEL Transformation als Voraussetzung für praktische Umsetzung einer SOA. In Wolf-Gideon Bleek, Jörg Raasch, and Heinz Züllighoven, editors, *GI Software Engineering 2007*, volume 105 of *Lecture Notes in Informatics (LNI)*, pages 75–80, Hamburg, Germany, March 2007. Gesellschaft für Informatik (GI).

[SI07c] Sebastian Stein and Konstantin Ivanov. Industrial EPC to BPEL Transformation, 2007.

[Sri95] R. Srinivasan. RPC: Remote Procedure Call Protocol Specification Version 2. Network Working Group Request for Comments: 1831, August 1995.

[SSK08] Christian Stamber, Sebastian Stein, and Marwane El Kharbili. Prototypical Implementation of a Pragmatic Approach to Semantic Web Service Discovery During Process Execution. *BIS 2008*, pages 201 –212, 2008.

[SSS01] Susan Elliott Sim, Janice Singer, and Margaret-Anne Storey. Beg, Borrow, or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research. *Empirical Software Engineering*, 6(1):85–93, 2001. http://www.csr.uvic.ca/icse2000/.

[UG96] Mike Uschold and Michael Gruninger. Ontologies: Principles, Methods and Applications, February 1996.

[vB76] Ludwig von Bertalanffy. *General System Theory*. Braziller (George) Inc., New York, USA, 1976.

[vdAtHKB03] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(3):5–51, 2003.

[WCL+05] Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey, and Donald F. Ferguson. *Web Services Platform Architecture*. Prentice Hall PTR, 2005.

[WRH+00] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson,

Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[Yin02] Robert K. Yin. *Case Study Research : Design and Methods (Applied Social Research Methods)*. SAGE Publications, December 2002.