Modelling Method Extension for Service-Oriented Business Process Management

Dissertation

zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

der Technischen Fakultät
der Christian-Albrechts-Universität zu Kiel

Sebastian Stein

Kiel
Dezember 2008

1. Gutachter:	Prof. Dr. Andreas Speck
2. Gutachter:	Prof. Dr. Witold Abramowicz
Datum der mündlichen Prüfung:	11. Dezember 2009

Abstract

The introduction of service-oriented architectures (SOA) in the enterprise context promises many advantages. For example, by composing existing services new capabilities can be provided quickly allowing a fast and agile reaction to changing market conditions.

In order to support companies in a successful adoption of SOA, service-orientation must be integrated in their enterprise architecture. Many companies made high investments in the past modelling their enterprise architecture based on different modelling methods. The introduction of SOA is fostered if existing models can be reused and investments are preserved. Therefore, existing modelling methods must be extended by service-oriented concepts.

This thesis extends the modelling method ARIS with concepts for service-oriented business process management. It contributes a graphical modelling language, which is tightly integrated with the existing ARIS modelling method.

Besides a modelling language, a modelling method also consists of algorithms and applications using the content captured in the models. Therefore, this thesis develops three distinct applications based on the contributed modelling language. First, service discovery enables identifying services needed for business process automation. Second, the automated EPC to BPEL model transformation allows transforming a business process into an executable service orchestration. Third, semantic business process management formalises enterprise models so that they are machine processable.

To evaluate the usefulness of the designed modelling language and the developed applications, two empirical case studies are conducted. The first case study evaluates the modelling language together with the applications service discovery and process transformation. The second case study evaluates the application semantic business process management.

Both case studies demonstrate the usefulness and relevance of the modelling language as well as its applications. Hence, companies introducing service-oriented concepts can use the extended ARIS modelling method to document and analyse their service-oriented enterprise architecture.

Contents

Chapter

1 Introduction

This chapter introduces the thesis by first providing the motivation for the research conducted in section 1.1. Based on this motivation, section 1.2 describes the main contributions of this thesis. Section 1.3 first clarifies the epistemological standpoint of the author and briefly describes the research methods applied. This is further extended in section 1.4 by a discussion of measures taken to ensure a high validity of the research results achieved. At the end of this chapter, the overall structure of this thesis is presented in section 1.5.

1.1 Motivation

Besides employees and their qualification, business processes are a core asset of every company. The imitation of products by competitors is easier than imitation of business processes, because the complete design of business processes is not visible to an external observer [SS08a]. Therefore, companies can create a significant competitive advantage through excellent business processes. To accomplish this vision of business process excellence, companies invest in the management and improvement of their business processes by applying business process management [SS08a, BKR05, SF03].

As a discipline of business process management, business process automation tries to automate at least parts of a business process using IT. Here, the use of web services and combining them in a service orchestration is a promising approach [Ley04]. Such an approach to business process automation requires the introduction of a service-oriented architecture (SOA) [SN96, Erl05, KBS05, MSJL06]. IT capabilities available in the company's IT landscape are encapsulated as web services published on the company's network.

Different modelling languages like BPMN [OMG06] and EPC [KNS92, STA05] are used in business process management and SOA to document the business processes and service orchestrations. As a matter of fact, companies have invested in past years in documenting their business processes. While introducing SOA concepts to the company's

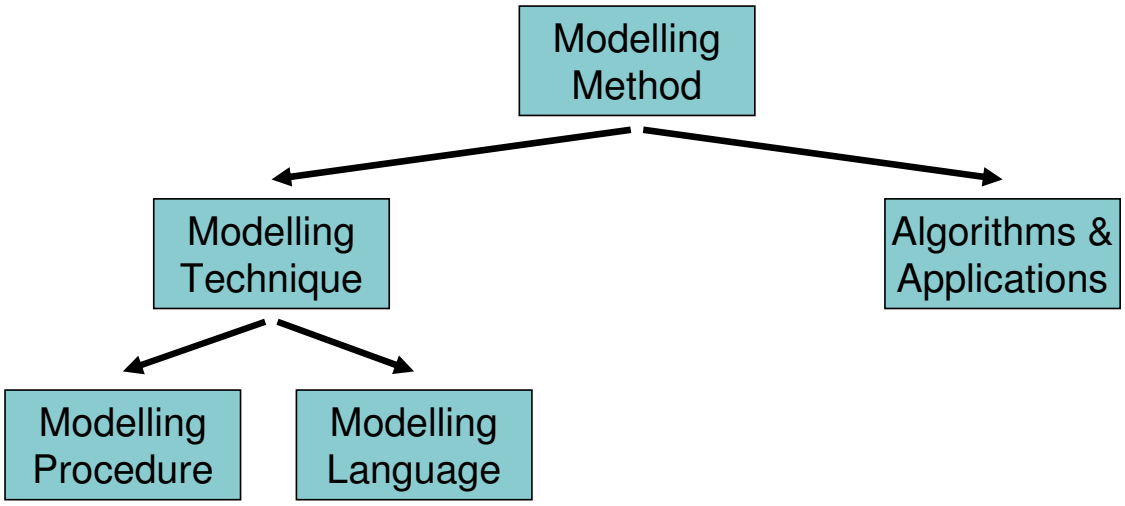

Figure 1.1: Elements of modelling method [KK02]

enterprise and IT architecture, such existing documents can be used to jump start the effort. To enable reuse of existing models, an integrated modelling method is required, which supports all phases beginning with strategy definition down to implementing a service orchestration.

According to [KK02] and as shown in figure 1.1, a modelling method consists of a modelling technique and algorithms and applications using the content captured by the modelling technique. A modelling technique is further divided in a modelling procedure (i. e. a methodology) and a modelling language [KK02]. Even though there are modelling languages, applications, and methodologies for business process management as well as SOA, they are not integrated forming no coherent modelling method for service-oriented business process management. For example, Papazoglou et al. [PvdH06] introduce a methodology for developing a service-oriented IT landscape, but they do not integrate their approach with existing business process management methodologies. Scheer [Sch99, Sch02] describes a modelling method for developing IT support based on business processes, but service-oriented technologies like web services are not considered. In [CBD07] a comprehensive meta model for describing a service-oriented enterprise architecture is introduced, but no modelling language and integration with existing modelling methods for business process management is provided.

This missing integration of business process management and SOA hinders a fast adoption in industry. Companies either have to adapt completely new modelling methods ignoring existing investments or they have to extend the modelling method used on their own. Especially the individual extension of modelling methods is questionable, because such extensions cannot be supported by standard software and therefore leads to high costs.

To overcome this problem, the present thesis extends the existing modelling method ARIS [Sch99, Sch02] to support service-oriented business process management. ARIS is selected, because it has according to [Pey07, Ble07] a broad adoption in industry. Thus the results of this thesis may be available to many potential users.

The ARIS extension must make the least possible changes to ARIS enabling reuse of

existing models. This lowers the adoption costs for companies and it preserves previous investments. In addition, the extension must ensure that a transition from a service-oriented enterprise architecture to a service-oriented IT architecture is possible and supported. This is required to prevent a new divide between business and IT.

1.2 Contribution

1.2.1 Theses

The motivation shows that an integrated approach to service-oriented business process management is missing. Therefore, this thesis aims at providing such an approach. The work presented is based on the following theses:

1. An integrated modelling method is required, which does not just cover technical SOA artefacts like web services, but which also provides modelling constructs for a service-oriented enterprise architecture.
2. The modelling language to be developed as part of the modelling method will consist of several layers and will feature different notions of service concepts.
3. The service concepts of the modelling language must be flexible enough to not just cover syntactical service information, but also semantic business descriptions.
4. An algorithm or application is needed to transform content on one layer of the modelling language to content of another layer.
5. An algorithm or application must be provided to support business experts in selecting an appropriate service for a specific modelling situation.
6. The modelling language must be extensible allowing custom description elements such as ontologies.
7. The modelling method must be integrated with existing modelling methods for business process and enterprise architecture management to ensure quick industrial adoption.

This thesis focuses on implementing those theses by providing several contributions. The following subsection presents the main contributions provided by this thesis.

1.2.2 Derived Contributions

This thesis extends the modelling method ARIS to also support service-oriented business process management. The scientific contribution of this thesis is structured according to the elements of a modelling method as defined by [KK02]. The contribution comprises the following points, which are mapped in figure 1.2 to the elements of a modelling method:

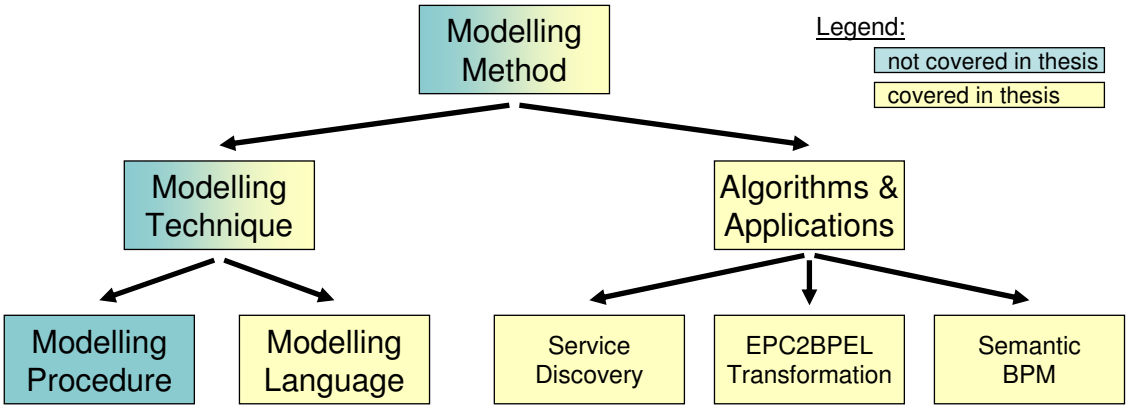

Figure 1.2: Extensions of ARIS modelling method provided by thesis

- The modelling method ARIS provides a modelling language for various aspects of an enterprise model. This modelling language is extended to support the definition of a service architecture and the representation of a service-oriented IT landscape. Existing modelling languages like EPC are extended to make use of service concepts. This contribution is described in chapter 3.
- Based on the modelling language, three applications and algorithms are developed demonstrating the applicability of the extended ARIS modelling language. Those applications generate additional value for users, because the content captured with the modelling language is not only used as documentation. The following applications are presented in this thesis:
  - The application service discovery supports business experts in selecting services to automate a function. Two different matching algorithms are used to identify relevant services and the results are presented in a user-friendly interface. This application is described in chapter 4.
  - The application automated EPC to BPEL process transformation generates a service orchestration out of a service-oriented business process model. In contrast to previous work, the transformation does not solely transform the control flow, but also considers other dimensions like data and service descriptions. This application is described in chapter 5.
  - The application semantic business process management formalises enterprise models so that their content is machine processable. This enables service discovery during runtime, which is also demonstrated by this application. This application is described in chapter 7.

As shown in figure 1.2, a new or extended modelling procedure (i. e. methodology) is not contributed by this thesis for several reasons. For example, the modelling method ARIS [Sch99, Sch02] comprises the modelling procedure ARIS Value Engineering (AVE), which is still applicable and valid. A SOA extension for AVE is currently a research topic of Ricken [Ric07, RP08]. Besides, several works [JM05, Jon06, PvdH06] on such modelling

procedures exist. In addition, it can be expected that companies define or extend their own internal methodologies.

The extended modelling language as well as the developed applications are evaluated in two empirical case studies. Those case studies are contributions, too. For example, the case study evaluating semantic business process management is the first scientific case study in this research area.

1.3 Epistemological Standpoint

Scientific rigor requires carefully designing and carrying out research [Cre02]. As an initial step in the research design process outlined by Creswell [Cre02], one has to define the own epistemological standpoint. This is important, because e. g. in case of a positivistic standpoint it is impossible to apply qualitative research methods.

In general, the author of this thesis follows a post positivism standpoint, implying that knowledge can be generated through empirical observations and measurements. However, the author extends this standpoint by taking pragmatic knowledge claims into account. The author is looking for the currently best possible solution in the given environment, but is aware that having found a good solution for a specific use case does not mean having found the best general solution.

While conducting the research presented in this thesis, the author applies the research tool falsification [Pop34], which was developed as part of critical rationalism. Falsification asks researchers to try to disprove their own theories and models (i. e. to falsify them). This leads to a critical view on the own work, because now the researcher focuses on disproving instead of proving the own contribution.

The chosen epistemological standpoint allows applying a “mixed-method” approach [Cre02, MB97]. In such a “multimethodology” research approach [MB97], it is possible to apply quantitative as well as qualitative research methods. Thus the author applies research methods from both paradigms in the research presented in this thesis. Development of the ARIS modelling method extension is done argumentative-deductive (i. e. quantitative), development of the three applications is done using prototyping (i. e. quantitative), and the two case studies are mainly qualitative in their nature. According to a study by Wilde and Hess [WH07], those three research methods are also used in 63% of all research works in business information systems.

This research is contributing to the research discipline business information systems[1] [SH05, HN05]. German speaking business information systems research mainly uses constructivist research methods, whereas the English speaking community prefers behaviourism [WH07, HMPR04]. However, Hevner et al. [HMPR04] emphasise that both approaches are not contradictory, but instead can be combined. For example, a solution for a problem can be “constructed” after identifying a possible theory through a behavioural

[1] In German speaking countries, this is known as “Wirtschaftsinformatik”. There are differences between “Wirtschaftsinformatik” and business information systems research. However, those differences do not influence the work presented in this thesis. Therefore, no detailed discussion of the differences is provided.

study [HMPR04]. This thesis follows the tradition of German speaking business information systems research. It uses quantitative as well as qualitative research methods to construct an extended modelling method. The measures taken to ensure a high validity of the research results are discussed in the following section.

1.4 Research Design

Methodologies for conducting a doctoral thesis propose to manage all belonging activities as a project [Daw00, Rob04]. Such a project consists of milestones like finishing the literature review or completing the writing of the thesis. In general, the author followed such an approach. However, the covered topics are too complex to be handled in a single project. Therefore, for each researched area a project was instantiated. There was at least one scientific publication at the end of each project to have an external evaluation of the work done. This project based approach enables research results with a high quality, but it must be complemented with additional measures.

As discussed in the previous section 1.3, this thesis first constructs artefacts and afterwards evaluates them through empirical case studies. Hevner et al. [HMPR04] term this kind of research "design science". They define seven guidelines, which must be implemented while designing and conducting design science research in the business information systems domain.

1. Guideline one defines what a valid design science result in business information systems research is. Design is understood as an artefact, but not as a process [HMPR04]. Valid design science results are all artefacts with a relation to information systems like: information systems, descriptions of information systems, models for describing information systems, methodologies for describing information systems, etc. Hevner et al. [HMPR04] do not limit an information system to the aspects implemented by software or hardware. Instead, information systems are seen as socio-technical systems [TB51, Tri63], which encompass not only technical artefacts, but also humans, organisations, and other social entities.

   By providing a modelling language for service-oriented enterprise architectures and by developing specific applications using it, this thesis clearly contributes new artefacts to the body of knowledge in business information systems research.

2. Guideline two demands that design science research focuses on relevant problems [HMPR04]. A problem is only relevant if there is a real business need for it. Only having a technical relevance is not enough.

   As the discussion in section 1.1 shows, this thesis clearly focuses on problems relevant for companies by providing a modelling method for service-oriented business process management.

3. Guideline three emphasises the importance of an evaluation of the design artefacts created [HMPR04]. Different methods like case studies, experiments or tests can

be applied for evaluation. Hevner et al. demand to follow well established evaluation methods.

The artefacts created in this thesis are evaluated manifold. For example, the modelling language created is challenged in modelling workshops with experts. Empirical case studies are conducted as well following Yin's [Yin03] methodology for case study research. In addition, several scientific publications were prepared to also have an external review of the research results.

4. Guideline four demands a clearly recognisable contribution to the body of knowledge in business information systems research [HMPR04]. Such a contribution can be either about an artefact, about the process to develop an artefact or about the evaluation of artefacts.

   The work presented in this thesis and summarised in section 1.2 belongs to the first category of contribution - knowledge about an artefact. Even though knowledge about creating artefacts and evaluating them was also gathered, this is not considered as an own contribution and is therefore not explicitly discussed in this thesis.

5. Guideline five defines that accepted research methods must be applied while conducting design science [HMPR04]. It applies to the construction of artefacts as well as to the evaluation of the created artefacts.

   This thesis applies accepted research methods such as argumentative-deductive method development, prototyping, and case study research (see also section 1.3). By using case study research, the artefacts created are evaluated in a real-world setting.

6. Guideline six characterises design science in business information systems research as a search process [HMPR04]. Artefacts are created and optimised in several iterations. This search process aims at solutions fulfilling all current requirements. The found solution does not have to be the final and optimal solution. If requirements change, the artefacts must be changed in a new development cycle.

   The development of the artefacts contributed in this thesis followed such an iterative search process. For example, the modelling language was discussed with users several times in order to gather their feedback and improve the modelling language accordingly. It can be expected that the artefacts presented in this thesis are not final, but that they must be changed if new requirements emerge.

7. Finally, guideline seven demands that the research results get published [HMPR04]. Yin [Yin03] defines a similar guideline for research results gathered through case studies. Hevner et al. [HMPR04] emphasise that the solution must be communicated to a technical audience like engineers and researchers as well as to a business oriented audience like business experts and managers.

   To implement this guideline, the different results were published. This led to an early external evaluation of the research conducted. The feedback gathered could be incorporated in the research, which helped to improve the research results further.

Table 1.1: Publications categorised by media type

Media Type	Number	Publications
Journal, conference, workshop (with peer review)	16	[SLEK09], [SLI08a], [SSEKR08], [EKAdMSvdA08], [EKSP08], [SKD+08], [SKI08], [SSEK08b], [FKS08], [EKSMP08a], [EKSMP08b], [SSEK08a], [FFS08], [SBEK07], [DSSK07], [SI07a]
Book chapter/book	5	[SDS+09], [SSEKR09], [SS08b], [SI07c], [SI07b]
Technical report (without peer review)	14	[FS09], [SH09], [SLI08b], [CSR08], [EKS08], [KSI08], [SKSD08], [Ste08c], [SKW06b], [SKW06a], [Ste06], [SI07d], [SR08], [Ste08b]
Tutorial	8	[FBS+08], [Ste08d], [Ste08a], [Ste07a], [DFK+07], [NS07], [PSS07], [Ste07b]
Sum	**43**	

To reach the technical as well as business oriented audiences, different media types were used. The technical audience was reached by publications in journals and at scientific conference and workshops. Those publications were only accepted after a scientific peer review. Potential users and business experts were addressed by publications in books, technical reports, articles in magazines, and tutorials. Table 1.1 categorises the different publications based on the media type used.

Following those seven guidelines for design science in the business information systems research ensures that the results presented in this thesis are valid, reliable, relevant, and follow scientific standards.

1.5 Outline

This thesis is structured according to the scientific contributions described in section 1.2. The outline is visualised in figure 1.3. The following chapter 2 describes the current state of the art in business process management, service-oriented architecture, business process transformation, service discovery, and semantic business process management. Chapter 3 describes the extended modelling language including the underlying meta model. Afterwards, the service discovery application is described in chapter 4. The transformation of service-oriented business process models into a BPEL service orchestration is described in chapter 5. The application service discovery and EPC to BPEL transformation as well as the extended modelling language are evaluated in a case study described in chapter 6. Afterwards, chapter 7 describes the application semantic business process management.

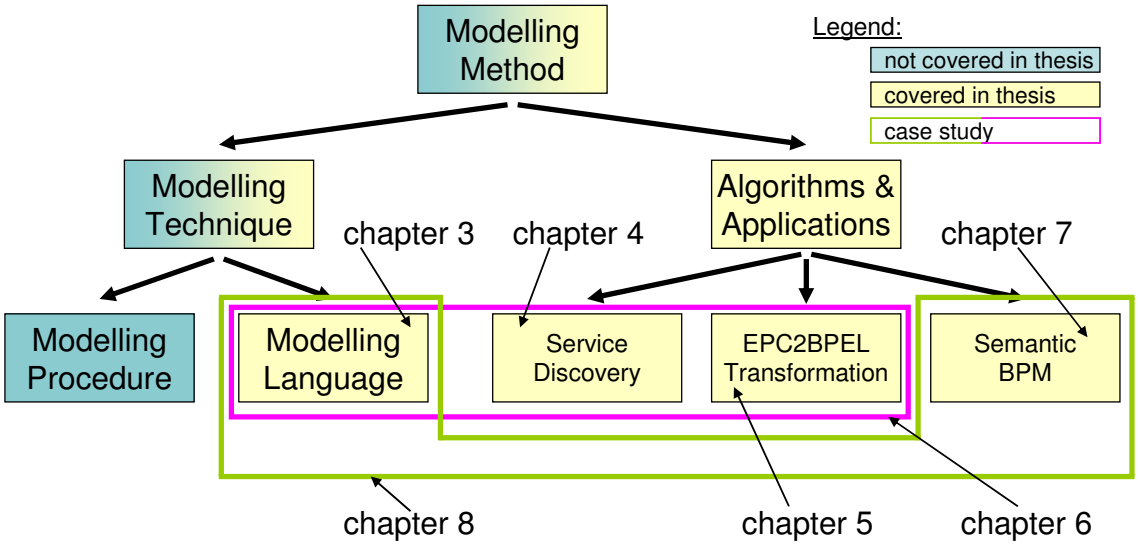

Figure 1.3: Outline of thesis

This application and the extended modelling language are evaluated in a case study as described in chapter 8. The thesis is summarised in chapter 9.

Chapter

2 State of the Art

This thesis is based on existing research. This chapter introduces this existing knowledge through a state of the art discussion. The chapter is structured according to the different areas addressed by this thesis. First, business process management is introduced in section 2.1. As this thesis is concerned with integrating the concept of service-oriented architecture (SOA) with business process management, section 2.2 provides an overview of relevant works on SOA. Such an integration can be achieved through model-driven engineering. Relevant techniques are discussed in section 2.3. One main contribution is an automated transformation to turn business processes into executable ones. Related work on process transformations is presented in section 2.4. Section 2.5 introduces the different approaches to web service discovery. Finally, section 2.6 describes the state of the art in semantic business process management.

2.1 Business Process Management

2.1.1 Overview

Different definitions and perceptions of business process management exist. Schmelzer and Sesselmann e. g. characterise business process management as an holistic management concept to design, control, and optimise business processes [SS08a, p. 5]. According to this definition, business process management comprises many other management methods like quality management, activity based costing, change management, six sigma, and balanced scorecards [SS08a, pp. 12]. Here, business processes are a means to implement a corporate strategy. Similar holistic definitions of business process management can be found in [BKR05, SF03]. Besides those works, other authors like [Wes07] view business process management mainly as a discipline focusing on the automation of business processes using IT. Those two views of business process management are confirmed by Scheer et al. [SKJK06] by dividing the area into IT focused business process management on one hand and business oriented business process management on the other

hand. There are also works like [All06] trying to integrate both views. This thesis adopts and follows the broader meaning by limiting business process management not to the implementation of business processes, but instead by understanding it as a broad concept influencing all parts of a company.

A business process tries to produce value expected by customers. Therefore, a business process is not only a collection of activities transforming some input into some output, but it starts with customer requirements and it produces a product consumed by customers [SS08a, p. 64]. A business process connects different elements of a company for value creation. Whereas the organisational structure of a company defines how formal power is distributed among people, roles, and departments, business processes define how elements of the organisational structure work together.

Business processes are shaped by the company strategy. The company strategy defines e. g., which customers should be addressed by which products. To measure the degree of strategy implementation, key performance indicators are defined, measured, and provided to top management, e. g. as a balanced scorecard. A company tries to optimise the performance of their business processes either by continuously improving [Dem82] them or by replacing them with completely new ones. The latter one is know as business reengineering [Ham90].

It is a common approach to use models to support business process management. As-is models reflect the current state of the business processes and to-be models are used to describe a state aimed for. The idea of enterprise modelling is introduced in the following subsection.

2.1.2 Enterprise Modelling and Enterprise Architecture Frameworks

In general system theory [vB76], a system is defined by its border, by its goal or purpose, by its elements, and by the relations among those elements. An enterprise is such a system, because it fulfils all those characteristics. An enterprise has a border to the environment (customers, competitors, market, etc.). It also has a goal like creating a high return on investment or maximising the shareholder value. An enterprise consists of many elements and the relations among those elements. During its lifetime, the enterprise is restructuring itself in order to adapt itself to a changing environment so that the overall goal can still be achieved.

Relevant aspects of a system can be captured in models. Models are used to abstract from the system and to focus on relevant aspects. An enterprise model captures all relevant aspects of an enterprise. It is created to document the structural and dynamic aspects of an enterprise, but also to plan and communicate possible changes internally and externally. The structural elements of the enterprise model are grouped according to their nature into different dimensions like organisational elements, functional elements, data elements, etc. Different diagram types are used to model the static as well as dynamic relations between elements of the same dimension. For example, organisational charts are used to model the formal hierarchy of power within the enterprise. In contrast, dynamic models like business process models define how the different system elements of the enterprise work together to achieve the enterprise's goals.

Table 2.1: Zachman framework

Abstraction Level	What	How	Where	Who	When	Why
Scope			Strategists			
Business			Executive Leaders			
System			Architects			
Technology			Engineers			
Component			Technicians			
Operations			Workers			

The enterprise model is usually structured according to an enterprise architecture framework like Zachman[1], ArchiMate [ARC04, ARC06], MODAF [MOD07], TOGAF [TOG07] or ARIS [Sch02, STA05]. Such an enterprise architecture framework defines the dimensions, abstraction levels, possible element types, and relation types. Table 2.1 shows the Zachman framework. According to the Zachman framework, an enterprise model must be structured into six levels of abstraction and six different dimensions. Each level of abstraction defines which role will create and work with the content on the abstraction level. In addition, the Zachman framework specifies several rules like that it is forbidden to add new rows or columns to the framework. Even though the Zachman framework defines the overall architecture of an enterprise model, it does not specify which notation to use for the different models. For example, one can freely choose between different business process notations like BPMN or EPC (see subsection 2.1.4) to document the content required in the "How" dimension on the "Business" abstraction level. This thesis is based on the modelling method ARIS, which includes an enterprise architecture framework among other things. ARIS is described in the following section.

2.1.3 Modelling Method ARIS

ARIS [Sch02, STA05] is a modelling method developed and maintained by IDS Scheer[2]. It is according to market research reports by Gartner [Ble07] and Forrester [Pey07] the leading framework in this market. It implements all elements of a modelling method as defined by Karagiannis and Kühn [KK02] (see also figure 1.1). Tools for creating, analysing, and communicating models using ARIS are released by IDS Scheer as commercial software packages under the brand "ARIS Platform"[3]. The software tools are grouped into six solutions addressing different topics like enterprise architecture management and SOA. The modelling procedure is called ARIS Value Engineering (AVE)[4]. AVE has six different flavours, one for each solution. However, all AVEs share a common core. ARIS supports various modelling languages like BPMN, EPC, Entity Relationship Model (ERM), organisa-

[1] http://www.zifa.com/ and http://www.zachmaninternational.com/
[2] http://www.ids-scheer.com/
[3] http://www.aris.com/
[4] http://www.aris.com/ave

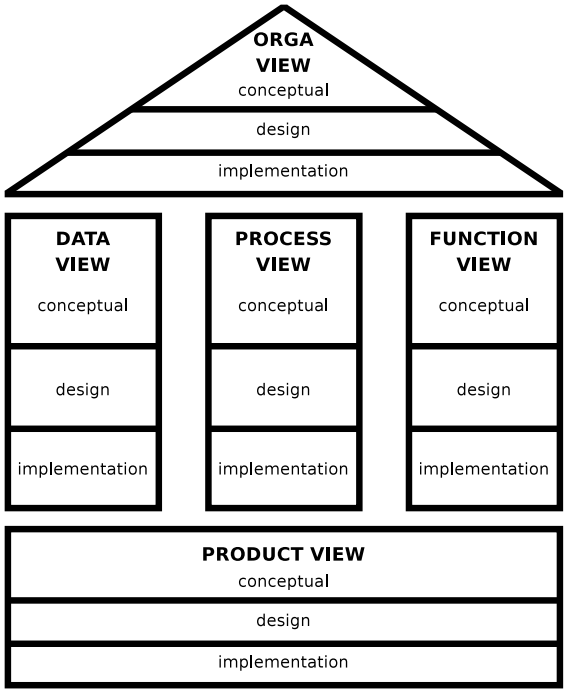

Figure 2.1: Dimensions and abstraction levels defined by ARIS

tional charts, product portfolios, balanced scorecards, UML, etc. for modelling all aspects of an enterprise model. In many cases, a user is free to choose between different alternatives like either using BPMN or EPC for business process modelling.

ARIS distinguishes five dimensions each having three abstraction levels as shown in figure 2.1. The process dimension in the centre of the figure links the elements of the other dimensions together. For example, a process like a business process consists of functions from the functional dimension, which are executed either by organisational elements from the organisational dimension or by IT systems from the functional dimension. Each function of the business process consumes and produces data elements defined in the data dimension. A business process also produces products, which are defined in the product dimension.

The ARIS meta model [STA05] is flexible enough to represent those different modelling languages and to integrate them. The ARIS meta model predefines a set of object types, connection types, model types (diagram types), and symbol types[5]. Examples for an object type are the “organisational unit” object type, the “UML interface” object type or the “(business) function” object type. Symbol types are used to describe subtypes of object types. For example, a function has the subtype “system function”, which is an automated function. Different object types are connected with each other through connection types (i. e. relations). For example, a function and an organisational unit can be connected with each other using the “carries out” connection type, meaning that the organisational unit executes the function. Model types define diagrams like “organisational chart” or “UML use-case diagram”. The ARIS meta model specifies, which object types can be related to

[5]There are more artefacts like attribute groups, but the overall principle remains.

each other using which connection type. In addition, it specifies which object types and connection types can be used in which model type. A user instantiates the pre-defined object, model, and connection types like creating an organisational unit named “sales department” in an organisational chart model type.

The object types can be reused between the various models. For example, a specific instance of an “organisational unit” object type can be used in an organisational chart as well as in a business process model. Model instances can be linked to object instances. For example, in a business process model a “function allocation diagram” can be assigned to a function to describe the function in more detail. A common application of ARIS is using it for business process modelling. Business process modelling is described in the following subsection.

2.1.4 Business Process Modelling

Business processes are captured visually to document and communicate them using a modelling language. There are different graphical modelling languages available like EPC [KNS92, STA05], BPMN [OMG06], and UML activity diagrams [OWS+03] to name a few. In addition, products like workflow environments often provide their own specific modelling language.

The EPC modelling language as originally described by Keller et al. [KNS92] is a directed graph consisting of arcs and nodes. Arcs connect nodes to define the control flow of the process. Nodes are either functions (i. e. activities), events (i. e. pre- and post-conditions) or operators. An event with no incoming arc is called a start event, an event with no outgoing arc is a called an end event. Events and functions alternate. Two consecutive functions are not allowed. Each function and event has maximum only one incoming and outgoing arc. Exceptions from this rule are operators. They might have either several incoming or several outgoing arcs. Operators can be either “OR”, “AND” or “XOR”.

The original EPC modelling language has limited applicability, because it only uses elements from the functional and process dimension of the enterprise model. For example, it is impossible to specify who is executing a function and which input and outputs are needed. Therefore, the EPC modelling language used in ARIS today (shown in figure 2.2) is an extended version including elements from other dimensions of the enterprise model. For example, it is possible to relate organisational elements to a function to specify who is executing a function. Also, elements representing IT systems can be added for the same purpose. Functions can be related to different kinds of data elements to model the data consumed and the data produced by a function. It is also possible to model the products created by the process. There are many additional relations to other elements, which are needed outside of pure process modelling. For example, for controlling purposes, functions can be related to KPI instances to define concrete metrics. Also, all elements have a variety of attributes like those needed for process simulation.

Following the classification by Mendling et al. [MLZ06], the EPC modelling language has a graph-structured representation scheme, i. e. the control flow is specified by arcs representing the execution logic between nodes. Still, it is possible to model different workflow patterns [vdAtHKB03] like sequence, concurrency, alternatives, and loops which

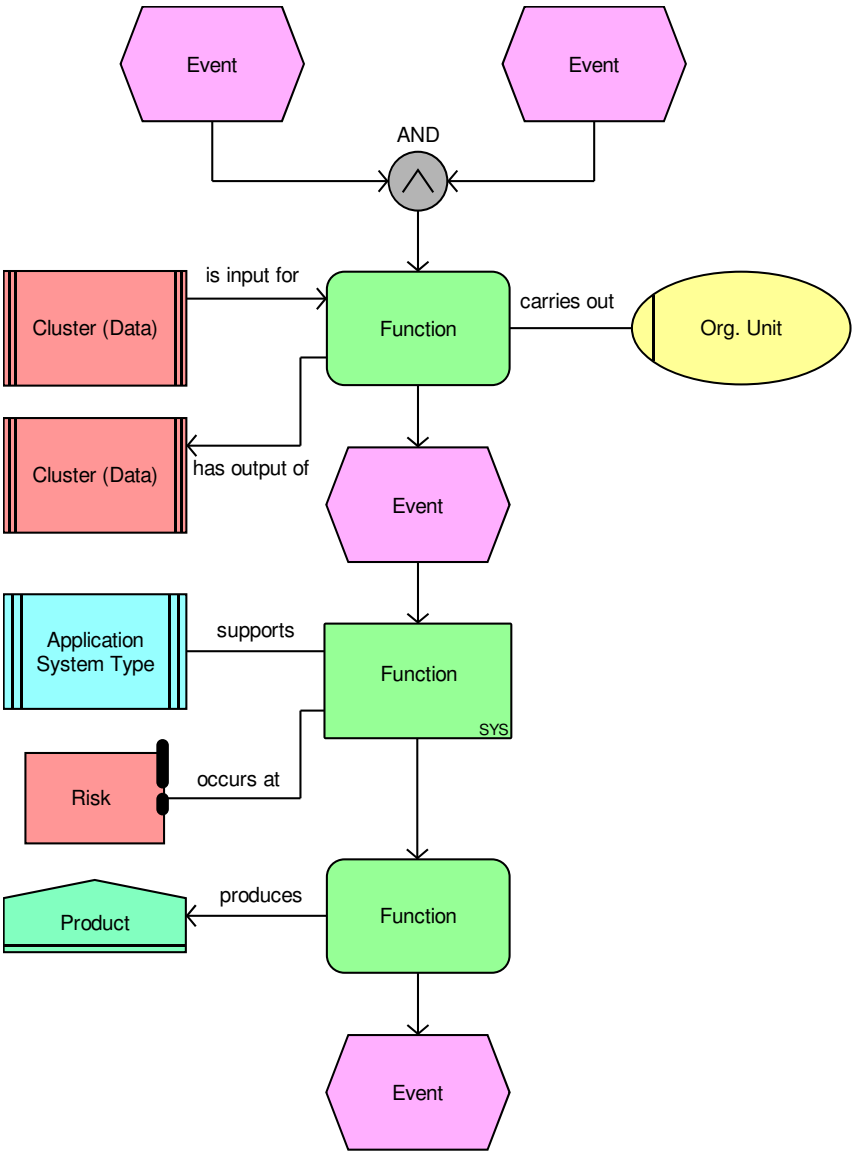

Figure 2.2: Current EPC notation used in ARIS software tools

can be found in block-oriented modelling languages. However, it is impossible to represent all workflow patterns as Mendling et al. [MNN05] show.

Currently, there are two exchange formats for EPC models available. The ARIS software tools support the proprietary AML format, which also includes diagram information. EPML [MN06] is an alternative exchange format, which is used by many OpenSource EPC modelling tools. A transformation from AML to EPML exists as well [MN04].

The EPC modelling language is criticised, because of its unclear semantics [Weh07]. There are different suggestions to overcome this issue like informal semantics [NR02], local semantics (e. g. free choice semantics [vdA99], boolean semantics [Weh07]), and non-local semantics [Kin06]. Despite those problems, EPCs are used in many real-world projects and the EPC modelling language is also used to document the SAP reference processes.

Similar observations about unclear semantics can be made for BPMN and UML activity diagrams [KHSW05]. Still, there is a growing popularity of BPMN despite those problems. It is expected that BPMN 2[6] provides a better definition of the execution semantics. The BPMN modelling language [OMG06] mainly focuses on the functional and process dimension of the enterprise model, but it is possible to relate elements from other dimensions to BPMN elements as well. Similar to EPC, BPMN is a structured graph consisting of flow objects, connecting objects, swimlanes, and artefacts. Flow objects are events, activities, and gateways. Possible connecting objects are sequence flows, message flows, and associations. Swimlanes consist of pools, which may contain lanes. Artefacts are data objects, groups, and annotations. Besides those elements, BPMN provides a variety of attributes, which have no visual representation. Business process models like BPMN and EPC are often the base for business process automation. This is discussed in the following subsection.

2.1.5 Business Process Automation

Even though the usage of IT in companies might not create a competitive advantage anymore [Car04], having no IT support at all will not work for most companies. Therefore, companies try to automate or support their business processes using IT. This can be done either by introducing standard software and customising it or by creating IT applications specific for the business processes. According to Leymann [Ley04], the combination of web services to implement a business need is currently a promising way to create such a business process specific application. In this scenario, web services are combined by an orchestration language such as BPEL [BPE03, BPE07a, JMS06]. The orchestration language is executed on an execution engine like Oracle BPEL Process Server or IBM Websphere.

BPEL is a XML based language to describe an orchestration of web services as a process. For an overview of web service technology see the next section 2.2. BPEL supports the graph-oriented as well as the block-oriented paradigm as defined by Mendling et al. [MLZ06], because its ancestors are XLANG, a block-oriented language, and WSFL, a graph-oriented language [WvdADtH02]. Each element of a BPEL process is an activity, which is either a primitive or a structured one. Primitive activities are invoke, receive, reply, assign, throw, wait, terminate, and empty. To construct more complex control flows, structured activities like sequence, while, switch, flow, pick, and scope can be used. BPEL allows nesting those structured activities. In addition, links can be used to define processes in a graph-oriented manner. Links can be also used to influence the execution order of parallel activities. Besides the different activities, BPEL also provides elements for compensation handling and exception handling.

BPEL allows the usage of elements from the process, function, and data dimension. The organisational dimension is not addressed by BPEL directly. For example, it is not possible to specify a human activity. However, with BPEL4People [BPE07b] a standardised extension exists for this as well.

[6] The author of this thesis participates in the BPMN 2 standardisation effort at OMG (http://www.omg.org/).

Even though BPEL can be used to automate business processes, it is not meant to be used by business experts for business process modelling. Instead, it is a language meant to be compiled and executed. Therefore, it is necessary to derive a BPEL description from a business process model. This can be done either automatically through model transformation as discussed in section 2.4 or manually as e. g. described by Allweyer [All06, All08].

The refinement of a business process into an executable one is hindered by the so called business-IT divide [SF03, DvdA04, KHSW05]. The knowledge and background of business experts and IT experts is different and therefore a seamless communication between both groups is not always possible. For example, business experts might use a different terminology than IT experts. Also, a business expert designs a process from a business perspective trying to optimise the way the business is executed. In contrast, an IT expert designs a process from an implementation perspective trying to optimise the execution on IT systems. Even though both worlds share some concepts, there is also information needed in only one of the worlds. For example, information for simulating a business process is not needed by an IT expert. On the contrary, a business expert is not interested in exception and transaction handling.

Different strategies exist trying to overcome the business-IT divide. Smith and Fingar [SF03] try to obliterate the business-IT divide. Instead of transforming the business process model into an executable one, the business process model is directly executed. So far, this vision of obliterating the business-IT divide is not achieved. Another approach is using semantic technologies like reasoners, ontologies, and semantic web services. This approach is known as semantic business process management and is described in section 2.6. Besides manual approaches like [All06, All08], many authors suggest using automated model transformations to derive an executable process model out of a business process model. The state of the art on model transformations for business process automation is presented in section 2.4. Many of the approaches discussed in section 2.4 apply the concept of service-oriented architecture (SOA), which is introduced in the following section.

2.2 Service-Oriented Architecture

2.2.1 Overview

The concept of service-oriented architecture (SOA) is a hot topic discussed in research as well as in industry. Most discussions focus on using SOA as a technological concept for designing an IT infrastructure. This view on SOA is discussed in the following subsection. There are other publications about a broader more abstract view of SOA. Here, SOA is perceived as a management concept shaping the overall structure of an enterprise. This understanding of SOA is discussed in subsection 2.2.3.

2.2.2 Technological SOA

In one of the first publications on SOA [SN96], SOA is described as an IT architecture. Common business logic and data input/output functionality is moved into services so that

it can be shared between different applications. A service is mainly seen as a component providing access to a data store like a database. It is the idea to make the access to data transparent by hiding the way data is distributed among different databases. In contrast to ordinary software architecture, the service is not part of one application's architecture, but it is outside of the different applications using it. The authors [SN96] do not give any details how a service should be implemented or which technologies should be used to implement the interaction between service and service consumers (i. e. the applications using the shared functionality).

This view on SOA is supported by service-oriented computing [PG03]. Services are seen as "self-describing, open components that support rapid, low-cost composition of distributed applications" [PG03, p. 26]. However, in contrast to the previous work on SOA [SN96], services are said to communicate over the Internet, which is leading to the term "web service". This documents a step away from using the service concept for software architecture as it was done in component oriented software design [Szy97, HS00]. Even though Papazoglou and Georgakopoulos [PG03] demand web services, they do not specify which specific communication protocol to use. They only point out that web services must be described and advertised, but they do not define which technologies to use. A similar definition of web services can be found in [Kre01]. Here, web services are "a collection of operations that are network-accessible through standardized XML messaging" [Kre01, p. 6].

Today, SOA standard literature [KBS05, Erl05, MSJL06] introduces web services as a technology based on the so called "W3C stack". The term W3C stack refers to a set of standards published by the World Wide Web Consortium (W3C) like WSDL [WSD01], SOAP [GHM+03], HTTP [FGM+99], and the many different web service security standards (see [MSJL06] for an overview). WSDL [WSD01] is a XML based language describing the interface of a web service. Operations with their parameters can be defined and grouped into interfaces (which are called porttypes in WSDL version 1.1). Furthermore, bindings can be defined specifying by which transport mechanism the operations of a porttype can be accessed. Messages send between service consumer and service provider are encapsulated using SOAP [GHM+03]. SOAP messages are transported by a transport mechanism like HTTP [FGM+99], SMTP or FTP.

In order to allow potential service consumers to find a web service, the web service description (i. e. WSDL) is published in a service registry by the service provider as shown in figure 2.3 [Kre01]. The service consumer retrieves the web service description and binds the web service. The message exchange between service consumer and service provider is based on SOAP messages, which are transported with protocols like HTTP. There are different efforts for providing a web service registry standard like UDDI [CHvRR04] or ebXML [FNS05]. Even though the W3C stack is adopted in industry, web service registries have not seen the same adoption rate.

Web service orchestration languages like BPEL [BPE03, BPE07a, JMS06] and XPDL [XPD08] combine several web services based on a process definition. This executable process definition itself is a web service, which can be invoked by another web service orchestration. This allows building a recursive hierarchy of web services. In complex IT environments, the BPEL execution engine does not directly invoke the web services as

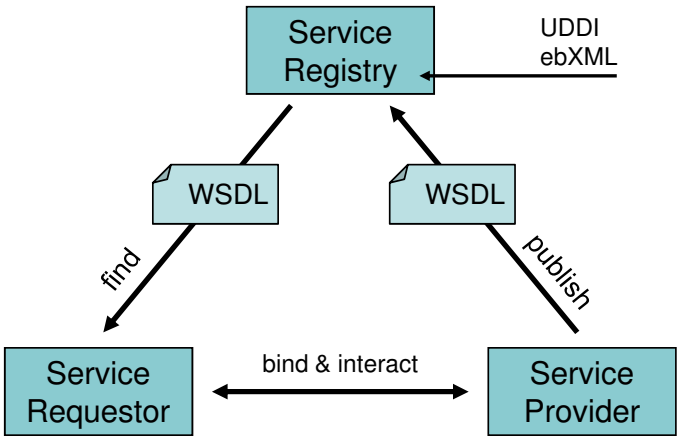

Figure 2.3: SOA pyramid [Kre01]

specified by the BPEL process, but forwards each invocation request to an ESB software. Among other things, an ESB software supports load balancing so that execution of a web service can be spread over several machines. The use of ESB software allows introducing cloud computing [Ley08], because the computation resources are not visible anymore to the consumer.

Different authors [HHV06, ST07, KBS05, Erl05, MSJL06] provide collections of best practices supporting the design of reusable web services. There are also specific software development processes and methodologies available for web service development (see e. g. [PvdH06]).

Even though there seems to be a wide adoption of the W3C stack in enterprise computing, some authors point to the fact that using the W3C stack is one possible way to implement a SOA. Recently, the concept of using standard Internet protocols like HTTP without adding new layers like SOAP and WSDL on top of it received some traction [Alg07, Vin08]. This approach is known as REST [Fie00]. At the current point, it is unclear if the W3C stack will be adopted all over industry or if alternative concepts like REST will succeed. However, this shows that a SOA must be designed in a way to be as independent of any specific technology as possible, so that the implementation technology can be changed if new products and technologies become available. Besides discussing the specific technology to base a SOA on, there are also broader and more abstract definitions of SOA as discussed in the following subsection.

2.2.3 Service-Orientation as Management Concept

There is some confusion about SOA, because SOA is often mixed with service science, even though both topics are not directly related. Service science [Teb06] deals with designing products, which are offered as service. Such services have some unique features compared to ordinary manufactured products like that a closer customer interaction is needed while delivering the service. For example, in a consulting engagement the customer is working together with the contractor to come up with a possible solution. According to Teboul [Teb06], today every product sold also contains to some degree a service. In

his view, service is “front-stage” meaning it is the visible part of the customer interaction, whereas the product is “back-stage” and its preparation is not directly visible to the customer. Even though such a view on service can be adopted in SOA as well, it is not the primary concern of service science. Service science focuses on different topics like how a customer interaction must be designed to have a satisfying service experience. Therefore, service science literature is not further investigated, because it does not directly contribute to the topic of this thesis.

Extensive work on SOA is available as the OASIS SOA Reference Model [MLM+06]. The SOA Reference Model defines the main terms used in this context like service, service consumer, service provider, service description, service interface, etc. It defines SOA as “a paradigm for organizing and utilizing distributed capabilities” [MLM+06, p. 8]. According to this definition, services are “the mechanism by which needs and capabilities are brought together” [MLM+06, p. 9]. A service must have the capability to perform work for someone with a need, it must specify, which work it can perform for others, and it must offer to actually do the work if requested [MLM+06]. Therefore, a service needs “visibility” to be used. Visibility consists of “awareness”, “willingness”, and “reachability” [MLM+06]. Awareness means that service consumer and service provider must be aware of each other. Willingness refers to the fact that both parties intent to work together. Finally, reachability means that service consumer and service provider can communicate. If one of the three components is missing, no interaction is possible. For example, if both parties are willing to work together and if they are both reachable, but if they do not know of each other (i. e. no awareness), an interaction is impossible. The SOA Reference Model defines additional terms like real world effects, contract and policy, and execution context. The model can be used to standardise the terminology used in a SOA project. It also helps to visualise different conceptions of SOA, e. g. if SOA is solely seen as a technological concept. The SOA Reference Model is independent of any technology. In fact, it does not even demand an IT implementation of services.

Several researchers and standardisation bodies are working on different parts of a SOA modelling method. For example, there are various efforts trying to define the semantics of such a modelling language by creating SOA meta models. CBDI Service Architecture and Engineering (CBDI-SAE) [CBD07] is a SOA meta model including a taxonomy of all terms used. According to [CBD07], CBDI-SAE was made available to OMG for public standardisation. Even though CBDI-SAE provides concepts for describing a business-oriented SOA independent of technology, it still focuses on services implemented as software. For example, it provides detailed models to describe the deployment of software services. CBDI-SAE is not integrated with any existing enterprise architecture framework or business process management modelling method. This makes it hard for potential users to adopt CBDI-SAE, because they first have to integrate it in the framework they use. Also, CBDI-SAE does not provide a modelling language, but focuses on defining a SOA meta model. Again, potential users have to define their own modelling language in order to capture the content specified by CBDI-SAE.

Jones and Morris [JM05] propose in their submission paper to OASIS a SOA methodology. This SOA methodology is a modelling procedure with an implicitly defined modelling language. However, an unambiguous definition of the modelling language is missing.

Jones extends this work in [Jon06] giving some examples how such a modelling language might look like. The work [JM05, Jon06] focuses on a business architecture. In an initial step, it captures the main services offered by a company. While building a service architecture, it should be first described what needs to be done, instead of describing how it is achieved. The authors reject the usage of processes on top level of an enterprise model and instead propose the usage of business services. Only few of them are defined on the top level. Those business service are further refined. Supporting services are added for capabilities not directly contributing to the overall business goal, but still needed for operation. Besides the modelling procedure, the work [JM05, Jon06] also provides details how to conduct a SOA project. For example, it defines roles needed in such a project.

Different enterprise architecture frameworks with a strong focus on service orientation exist. The ArchiMate framework [ARC04, ARC06] uses service entities on different abstraction levels and views within the framework. A service is always connecting the different abstraction levels. For example, the more generic application layer can use technology services provided by the technology layer. However, this approach of a fixed set of service categories is not very flexible. The enterprise architecture framework MODAF [MOD07] has a strong notion of defining capabilities. It provides different views to describe capabilities and their relationships. ArchiMate and MODAF are missing a graphical modelling language. Potential users have to define their own modelling language.

The ARIS [Sch02, STA05] modelling method for enterprise architecture and business process management provides extensive support for modelling service-oriented IT architectures. An IT system can be described on different abstraction levels using different modelling languages like UML. WSDL web services can be modelled in ARIS, too. However, a notion of a service independent of IT is missing in ARIS.

2.3 Model-Driven Integration Engineering

2.3.1 Overview

Parts of the research presented in this thesis were conducted within the German research project OrViA[7]. The OrViA project investigates the application of model-driven techniques for integration engineering. Integration engineering [Thr05, Thr08] aims at establishing engineering methods for conducting integration projects. A typical example of such an integration project is combining different information systems in a heterogeneous IT landscape.

The OrViA research project [FKSW06, FKT08] establishes a framework to use model-driven techniques like model transformations to support integration projects. The OrViA framework, which is presented in subsection 2.3.3, builds on top of model-driven architecture (MDA) [MM03]. An introduction to MDA is provided in subsection 2.3.2. An important part of the OrViA framework is using validation techniques to ensure that the models created correctly reflect the perceived reality. An overview of the applied model checking techniques is provided in subsection 2.3.4.

[7] http://www.orvia.de/

2.3.2 Model-Driven Architecture

Model-driven architecture (MDA) [MM03] is a paradigm promoted by OMG. In most cases, MDA is used within model-driven software development. However, MDA is not limited to this domain, because it focuses on model-driven system development. A system may include "anything: a program, a single computer system, [...], people, an enterprise, a federation of enterprises" [MM03, p. 2]. This is a broad definition, which is similar to the one found in general system theory [vB76]. However, it is admitted that in most cases MDA is used for model-driven software engineering [MM03, p. 2].

MDA introduces three major viewpoints as a mean for abstraction: computation independent viewpoint, platform independent viewpoint, and platform specific viewpoint. Models are used to capture the view on a system from one of the three viewpoints. Consequently, there are three different model levels:

- The computation independent model (CIM) is an outside view on the system not revealing any internal details [MM03]. A CIM is used by practitioners, who are dealing with the system, but who usually do not have the necessary knowledge to understand the internal realisation details of the system. Therefore, the CIM is used to bridge the gap between those making use of a system and those implementing it.
- The platform independent model (PIM) describes internal details of a system, but abstracts from a specific platform [MM03]. Therefore, the description provided is suitable for different platforms.
- The platform specific model (PSM) is the most concrete level in MDA [MM03]. It provides all details required to execute the system on a specific platform. Therefore, the PSM is not portable, because it makes use of the specific concepts of a platform.

MDA envisions using model transformations to derive a PSM out of a PIM [MM03]. Such a transformation uses the information available in the PIM and other information like specific transformation rules for a target platform. It is expected that a PIM can be transformed into different PSMs using different transformations. MDA does not define how such a transformation is implemented, but it only outlines the overall concepts. Therefore, a transformation does not have to be (completely) automated. This is a main difference to computer-aided software engineering (CASE), where all parts of a software should be generated out of models.

MDA is too abstract to be applied directly. For example, MDA does not define which models are needed, which aspects of a system must be described, and how to execute a transformation. Therefore, specific frameworks are needed guiding an user in applying model-driven engineering. The OrViA framework, described in the following subsection, is such a framework.

2.3.3 OrViA Framework

The OrViA framework [KTS05, FKSW06, FKT08] is illustrated in figure 2.4. It can be seen that the OrViA framework uses models on different abstraction levels. The OrViA framework provides a top-down approach to model-driven integration engineering.

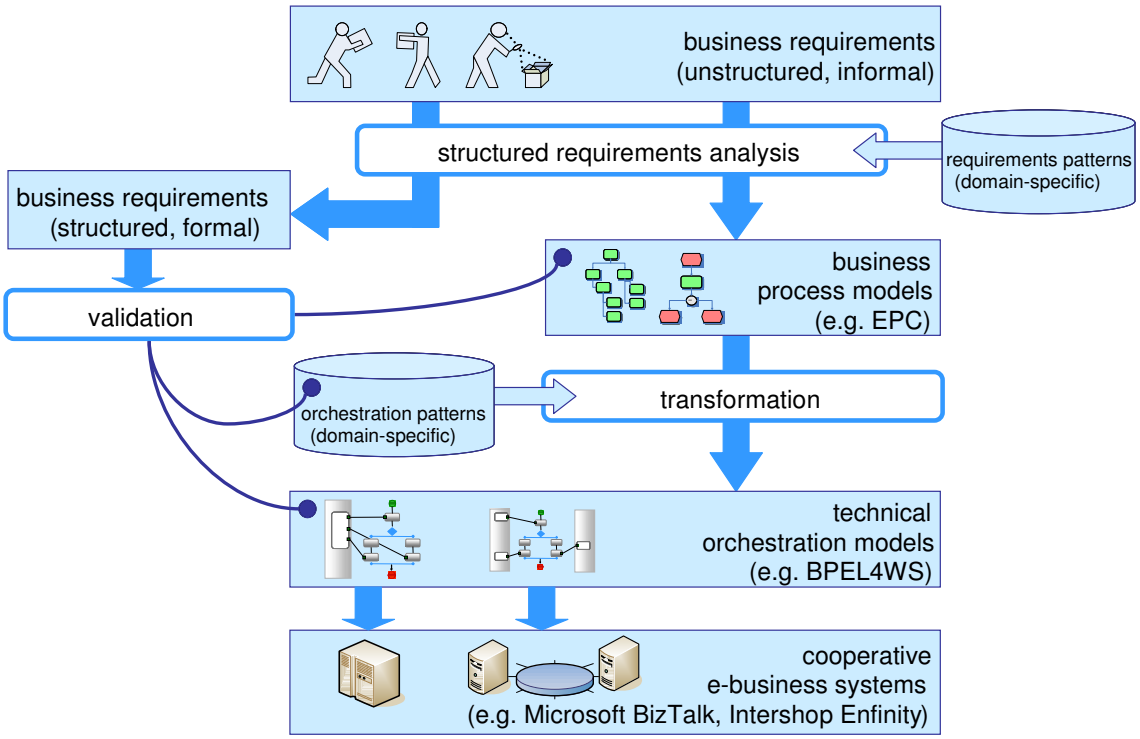

Figure 2.4: OrViA framework [KTS05]

At the beginning, only informal and unstructured requirements exist. This tacit knowledge is turned into structured and formal models through structured requirements analysis. The OrViA framework does not specify how to perform structured requirements analysis. For example, in case of software engineering, requirements engineering techniques [KS04] are used. One application area of the OrViA framework is business process automation (see subsection 2.1.5). Therefore, techniques taken from business process management can be used as well. The OrViA framework envisions that domain specific patterns are used to facilitate this step. For example, domain specific reference models can be used to make models more expressive.

After structured requirements analysis, formal and structured models exist. Those models are viewed as a platform independent description of the integration scenario. The OrViA framework applies model transformations to those models to turn them into technical platform specific models. For example, in case of business process automation, business process models (e. g. EPC or BPMN models) are transformed into executable models like BPEL. This transformation uses platform specific transformation rules. The generated platform specific models are used for execution, which is shown at the bottom of figure 2.4. The transformations for business process automation discussed in section 2.4 are a possible implementation of the transformation step within the OrViA framework.

Figure 2.4 shows validation as another main element of the OrViA framework. Validation is used at various places during the integration project. It ensures that the informal requirements existing at the beginning are correctly implemented at the end of the integration project. For example, besides formalising the requirements, also rules to be

implemented are extracted. The rules are used to check that the platform independent as well as the platform specific models comply to the rules established. Within the OrViA research project, a specific technology is used to implement validation. This technology is presented in the following subsection.

2.3.4 Model Checking

A modelling language consists of syntax, notation, and semantic. Whereas the syntax specifies how to use the different elements of the modelling language, the semantic specifies the meaning of the different elements. Many works like [KKGL08] exist, which are validating the syntax of a given model. Such works are important, because they ensure that the modelling language is used correctly. However, syntax validation cannot ensure that a model is meaningful, because it does not analyses the content of the model. Therefore, the OrViA research project investigates more advanced technologies enabling content validation. Specifically, the OrViA research project investigates techniques for temporal process validation. Such a validation approach consists of two main parts:

- models to be validated
- rules, which the models must comply with

For example, in case of model-driven business process automation, informal requirements are captured as business process models using e. g. the EPC modelling language. The created EPC models are one input for the validation. In addition, rules must be defined so that the (EPC) models can be validated against the rules. Such rules are often expressed using logical expressions. There exist different formalisms to define such logical expressions and [PRS04] give an overview of some of them. The OrViA research project uses computation tree logic (CTL) [CD88] as underlying formalism to define temporal logic statements. However, such logical expressions are not easy to read for an user with no background in mathematics and are therefore not suitable for business experts. Therefore, the OrViA research project introduces a graphical notation [FF08, FFS08] enabling the graphical definition of CTL expressions. Figure 2.5 shows the different graphical elements of G-CTL.

Rules as well as models to be validated are the input of a model checker. In the OrViA research project, the model checker SMV [McM93] is used. SMV performs its computation on a finite state machine given as Kripke structure [CGP00]. The OrViA research project uses the ARIS software to model the G-CTL rules and the models to be checked (i. e. EPC). The ARIS software exports those models in a XML dialect called AML. Therefore, a transformation of AML to the necessary input format of SMV is needed. The transformation is implemented using the operator hierarchy concept [FSH05, FP07], which provides more advanced transformation operators in contrast to ordinary XML transformation languages like XSLT. Based on the input, SMV computes the validity of the model. If the model is not valid, a counter example is given showing how at least one rule is broken by the given model.

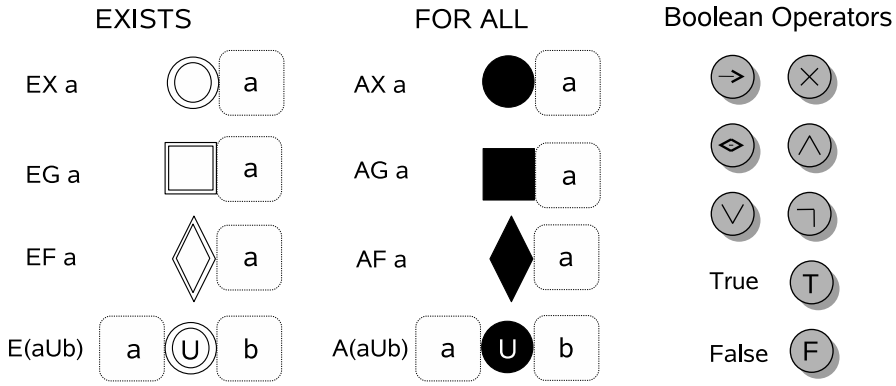

Figure 2.5: Elements of G-CTL [FF08]

2.4 Business Process Transformation

2.4.1 Overview

It is the aim of business process automation (see subsection 2.1.5) to provide an IT implementation for business processes to automate and support them by IT. Today, BPEL is used to define an executable orchestration of web services (for a description of web service technology see section 2.2) to implement business processes. A BPEL representation can be derived manually or through automated model transformation from an existing business process model. This section provides a detailed overview of automated model transformation approaches for business process automation. Descriptions of manual approaches can be found in literature [All06, All08]. No matter if following a manual or automated approach, in both cases the business-IT divide as discussed in subsection 2.1.5 must be bridged.

This section first discusses in subsection 2.4.2 the business to IT transformation process, revealing requirements for such transformations. In addition, there are some variation points and issues influencing the design of an automated business to IT transformation as discussed in subsection 2.4.3. Different approaches for such transformations exist and are presented in subsection 2.4.4, subsection 2.4.5, and subsection 2.4.6. The literature review on business to IT transformations is evaluated in subsection 2.4.7.

2.4.2 Business to IT Transformation Process

Figure 2.6 shows the business to IT transformation process to clarify the concepts and issues related to it. The involved artefacts are outlined as boxes of different sizes representing information sets. They are arranged according to their level of abstraction and the point of their use.

At first, a business expert describes a business process at an abstract (i. e. business-oriented) level according to functional and control flow related aspects in a business process modelling language like BPMN or EPC. The abstract business process is enriched with additional information concerning data flows, service interactions, and other perspectives. These modelling activities are supported by a modelling tool, which for instance highlights syntax errors and inconsistencies or retrieves applicable entities. Then, the refined

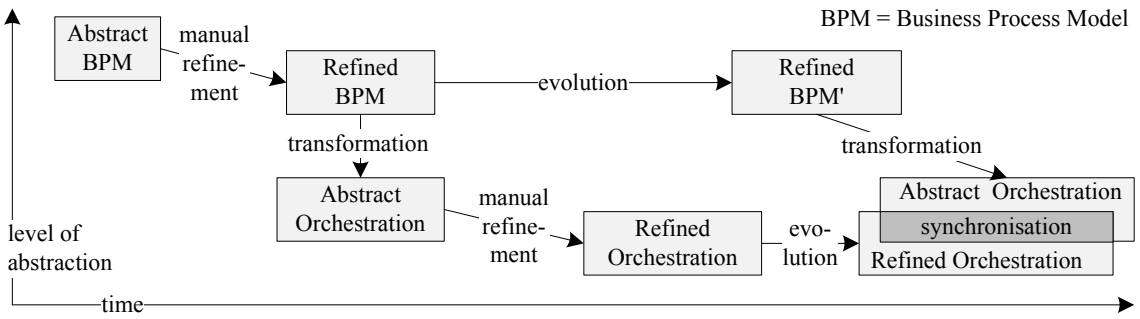

Figure 2.6: Artefacts in business to IT transformation process

business process model is translated into an abstract orchestration model (i. e. BPEL). Before executing it, the model is extended by an IT expert adding technology-related aspects such as service bindings or exception handling.

The artefacts created evolve over time. The IT expert may change the refined orchestration. At the same time, the business expert may modify the business process model, because of changing business requirements, incremental development cycles or inappropriate test results. The changed business process model again is translated in an abstract orchestration model. Since the transformation result conflicts with the evolved refined orchestration, the models must be synchronised.

2.4.3 Variation Points and Transformation Issues

The outlined transformation process consists of several variation points and issues. A variation point is a case where a decision between rather equivalent alternatives for instantiating the transformation process in a specific environment must be taken. For example, a transformation process might either support BPMN or EPC to define the business process model. An issue is an inherent difficulty for which no obvious solution exists. For example, transforming a graph-structured process graph into block-oriented BPEL is not possible under all circumstances.

Variation Point: Business Process Modelling Language
The modelling language defines modelling concepts used to describe certain aspects of business processes and the level of abstraction at which these aspects are considered. EPC and BPMN provide accepted notations for this purpose as discussed in subsection 2.1.4. BPMN provides more technical-oriented concepts, but is also considered as an adequate notation for business experts. Another relevant notation often used in context of object-oriented design is UML activity diagrams.

Issue: Complexity Reduction Strategy
Business processes are manifold entities. An adequate description of them for the purpose of transformation introduces a significant cognitive complexity. Different mechanisms may be used for complexity reduction, e. g. introduction of new language constructs, introduction of domain-specific patterns or multi-perspective modelling.

Variation Point: SOA Implementation Technology
During a business process automation project a decision must be taken which implementation technology to use. The solution space ranges from workflow related languages XPDL and web service oriented languages like BPEL to domain-specific execution languages used in business domains such as e-government and e-commerce. Besides, there are many proprietary languages offered by different middleware vendors. For more details see subsection 2.1.5.

Issue: Transformation Power
The degree of how the divide between business and IT can be bridged through model transformations depends on various aspects like: the process perspectives used, the granularity of entities, the tolerated extent of ambiguity, the incompleteness and inconsistency of input models, the readability of output models, and others.

With respect to granularity and amount of data of input and output models, Mens et al. [MCVG05] distinguish horizontal and vertical transformations. A horizontal transformation translates input and output models on the same level of abstraction, whereas a vertical transformation refines the input model to a more detailed level. In context of business to IT transformations, a horizontal transformation uses a business process model annotated with technical details. In case of a vertical transformation, only an orchestration stub is created. In both cases, the transformation value is rather limited. Vertical transformations try to bridge the gap between business and IT by adding additional information, which is rather difficult to accomplish and has impacts in synchronisation of co-evolved artefacts.

The semantics of business process modelling languages is often not precisely defined as discussed in subsection 2.1.4 [KHSW05, Weh07]. A transformation approach has to handle this issue, e. g. by restricting the input language to an unambiguous subset.

Variation Point: Representation Scheme
Concerning readability, Mendling et al. [MLZ06] classify process modelling languages according to two different representation schemes. EPC and BPMN are mostly graph-structured, i. e. the control flow is specified by arcs representing the execution logic between nodes (see subsection 2.1.4). In opposite, block-oriented languages define the control flow by nested elements representing sequences, concurrency, alternatives, and loops. BPEL provides concepts for both paradigms in a redundant manner [WvdADtH02]. Consequently, transformation approaches between graph-structured business process modelling languages and BPEL have a certain freedom of choice.

Issue: Adaptability and Extensibility of the Transformation
Customisation of the execution environment or the modelling language may cause adaptation or extensions of the transformation as well. Declarative implementations expressed in relational or graph based transformation languages tend to provide more adaptable and understandable transformation rules than operational implementations expressed in imperative transformation languages. Functional and template based approaches may be used for both paradigms.

Issue: Iterative Development Support
The co-evolution of process models at different levels of abstraction causes divergent artefacts. As outlined in figure 2.6, conflicting models overlap to some extent. Changed abstract orchestration elements resulting from changed business process models have to be merged with changed elements of the refined orchestration. A comprehensive transformation approach defines mechanisms for model synchronisation, which is also known as reconciliation [KHSW05].

Variation Point: Level of Automation
Even though this section describes work on automated business to IT transformations, the degree of automation might vary. For example, a transformation approach might require additional manual steps before the actual transformation is started. Therefore, there is certain freedom in choosing a level of automation.

Considering these issues and variation points, existing transformation approaches can be grouped into two main classes:

1. Approaches focusing on control flow transformation (presented in subsection 2.4.4).
2. Approaches trying to provide more comprehensive support by transforming additional artefacts besides the control flow. This class can be further divided into the following two subclasses:
   a) Approaches using domain-specific language extensions (presented in subsection 2.4.5).
   b) Approaches using frameworks (presented in subsection 2.4.6).

The literature overview focuses only on works transforming business process models (EPC, BPMN or UML activity diagrams) into the orchestration language BPEL, because BPEL has the widest industrial adoption.

2.4.4 Control Flow Centred Approaches

Basic considerations about transformations of business process models into executable ones (and vice versa) are described by Hauser and Koehler [HK04]. They use compiler theory techniques to transform process graphs (a subset of UML activity diagrams) into BPEL. Their approach translates sequential process graphs (without OR-splits and OR-joins) into goto-programs, which are further processed by a goto-elimination algorithm. Concurrent parts are translated by identification of sequential “single entry single exit” (SESE) fragments.

In a similar approach, van der Aalst and Lassen [vdAL05] abstract from concrete modelling languages (like EPC, BPMN or UML activity diagrams) and use workflow nets instead. Similar to [HK04], they identify and translate structured SESE fragments.

A pragmatic transformation approach from EPC to BPEL is proposed by Ziemann and Mendling [ZM05], who restrict their approach to acyclic EPCs. Depending on connected data elements, EPC functions are mapped to basic BPEL activities (like invoke, receive or reply). The EPC is translated to graph-structured BPEL constructs on the basis of split and

join patterns. The approach is implemented as XML transformation working on the EPML format (for information on EPML see [MN06, MN04]).

Kopp et al. [KUL06] describe a similar approach. The authors map acyclic extended Nautilus EPCs, which allow in contrast to standard EPCs [KNS92] multiple events between functions, to graph-structured BPEL. EPC events are interpreted as transition conditions of BPEL links.

In contrast to graph-structured transformations, the approach described by Specht et al. [SDTK05] relies on the identification of workflow patterns [vdAtHKB03] to transform structured EPCs to block-structured BPEL. Inner EPC events are interpreted as business-oriented documentation and are ignored in the transformation process. The algorithm searches for corresponding split and join patterns and transforms such pairs to BPEL switch and flow activities.

Similar transformation approaches exist for other graph based business process modelling languages. White [Whi05] proposes a mapping between BPMN and BPEL on the base of templates and relations between syntax elements. Yet, this description is considered as informal and incomplete [MLZ06, ODBtH06]. The BPMN specification [OMG06] does not provide significant improvements with respect to this. A more formalised approach is given by Ouyang et al. [ODBtH06]. For each BPMN activity so-called precondition sets are identified, which are mapped to event condition actions in BPEL (onEvent handlers). The authors emphasise that their approach can be improved according to compactness and comprehensibility aspects. Following this intention, Ouyang et al. [OvdADtH06] combine this approach with a structure-identification mechanism based on components [vdAL05].

2.4.5 Approaches Based on Domain-Specific Language Extensions

Several approaches rely on UML activity diagrams extended by self-defined or standardised UML profiles. Mantell [Man03] proposes a BPEL-specific profile to model and transform UML activity diagrams. Activities are marked with stereotypes such as «process», «receive» or «invoke».

Bordbar and Staikopoulos [BS04] use a specific UML profile, too. UML activity diagrams modelled by business experts are refined and enriched with stereotypes such as «CallOperationAction» or «datastore». The enriched process models are transformed to BPEL for which the authors outline a few transformation rules.

Skogan et al. [SGS04] take existing web services in terms of WSDL descriptions into account. WSDL descriptions are transformed to UML class diagrams, which are referenced by UML activity diagrams. The transformation to BPEL relies on technical information in the business process model. For example, activities are stereotyped with «WebServiceCall» or «DataTransformation».

Heckel and Voigt [HV04] define a BPEL-specific UML profile for UML component, class, and activity diagrams. They call for synchronism between the UML source and the BPEL target, which allows iterative incremental development. For this purpose, a declarative transformation approach using graph transformation techniques is proposed.

Another BPEL generation approach is given by Yu et al. [YZZ+07]. This approach differs

from other UML profile based approaches by relying on a standard UML profile (EDOC CCA[8]) and using OMG's standard transformation technique QVT.

Following the intention of EPC transformation approaches like [ZM05, KUL06, SDTK05], Schmelzle [Sch07] transforms acyclic EPCs to graph-structured BPEL processes. He explicitly differentiates between business oriented and implementation oriented levels of abstractions. The author declines using default values for aspects not expressed in the business process models but required on the implementation level. Instead, an intermediate level is introduced, where additional information can be supplemented using annotations.

2.4.6 Approaches Based on Frameworks

Another class of approaches proposes extensible frameworks, which have to be adapted to the specific requirements of an application domain. Allweyer [All07] applies model-driven architecture (MDA) [MM03] concepts to the area of business process management and outlines an EPC framework based on high-level process patterns. The conceptual framework requires the specification of validation rules for source models, templates for target models, and transformation rules defining the execution logic of templates. The proposed approach should be applicable for recurring business related as well as implementation related problems. The outlined approach is presented as a control flow centred non-iterative one-step refinement, whereas extensions according to a multi-perspective and multi-step approach are possible.

Considering multi language support as an essential feature in design and implementation phases of process-driven SOAs, Zdun and Dustdar [ZD06] propose a language engineering framework based on model-driven development concepts. The framework allows the recursive specification of modelling patterns based on pattern primitives. For example, to decompose the control flow perspective the concepts "Macroflow", "Macroflowsteps", "Microflow", and "Microflowsteps" are introduced. The authors refer to code generation capabilities of the framework but without any detailed explanation.

Instead of a pattern based approach, Roser et al. [RLB07] propose a language oriented framework. It allows the extension of standard workflow models with pre-defined domain concepts, such as "application", "service" or "offer". Additionally, to support different expectations, experiences, and skills of business experts and IT experts, language concepts can be visualised differently using customised concrete syntaxes. For the purpose of adaptability and reuse, the framework aims at the separation of domain specific and domain independent concerns. It is therefore structured into domain specific language adapters and code generation templates and domain independent process transformation and rephrasing components.

2.4.7 Evaluation

The variation points and issues discussed in subsection 2.4.3 are used to evaluate the transformation approaches. Because of the different intentions, control flow centred and holistic approaches are discussed separately.

[8] http://www.omg.org/technology/documents/formal/edoc.htm

Table 2.2: Control flow centred transformation approaches

HT/VT = Horizontal/Vertical trans., EP/EM = Element Preservation/ Minimisation, SI/SM = Structure Identification/Maximisation	Input/Output		Abstr.		Strategy				Paradigm					Descr.	
	Source	Target	HT	VT	EP	EM	SI	SM	Imperative	Functional	Relational	Graph	Template	Formal	Semif.
[ZM05]	EPK	BPEL	×		×				×					×	
[ODBtH06]	BPMN	BPEL	×		×					×				×	
[KUL06]	N-eEPK	BPEL	×			×			×					×	
[SDTK05]	EPK	BPEL	×				×						×		×
[Whi05]	BPMN	BPEL	×				×				×		×		×
[HK04]	UML AD	BPEL	×					×	×					×	
[vdAL05]	WFN	BPEL	×					×	×					×	
[OvdADtH06]	BPMN	BPEL	×					×	×	×				×	

Control flow centred transformation approaches focus on the process perspective, which according to [KtHvdA03] is the most essential dimension of process modelling languages and engines. The mapping of the functional perspective is basically restricted to specialisation refinements. Abstract concepts (e. g. EPC functions) are replaced by more concrete concepts with reduced extension and increased intension (e. g. BPEL receive, reply, invoke activities). Other refinement strategies (see [CE00, GS04]) like functional or aspectual decomposition, choice of representation, choice of algorithm, concretisation, elaboration, and realisation are only partly used. Therefore, these approaches are horizontal transformations (see table 2.2).

Control flow centred approaches deal with different process representation schemes (graph-oriented, block-oriented) and can therefore be classified according to the four transformation strategies for business process models described in [MLZ06]: “Element-Preservation” (EP), “Element-Minimisation” (EM), “Structure-Identification” (SI), and “Structure-Maximisation” (SM). Since most approaches are formally described, the underlying programming paradigm used to implement the approach can be determined. The spectrum ranges from imperative and functional implementations to relational, graph based, and template based approaches [MCVG05]. The evaluation for control flow centred approaches is shown in table 2.2.

Table 2.3 summarises domain specific and framework based approaches. Most of them extend control flow centred process transformations to holistic approaches in terms of usability and code generation capabilities. The described UML based transformations use the UML language extension mechanisms to introduce new technical concepts. The level of abstraction between the UML source and the BPEL target are rather close to one another. In many cases this is due to the motivation of introducing a modelling notation for BPEL (see [HV04]). Other approaches aim at complexity reduction in business process modelling by providing a design method, which is aligned to business needs. They increase the level of abstraction by introducing business related concepts expressed as new

Table 2.3: Domain specific and framework based transformation approaches SA/AU = (Semi) automated, NLC= New language concepts, MM= Multi- perspective modelling, ER= Extension required, ID=Iterative development

	Input/Output		Abstr.		Auto.		Compl.				Usab.	
	Source	Target	HT	VT	SA	AU	NLC	Patterns	MM	Adaptability	ER	ID
[Man03]	UML profile	BPEL	×			×	+	−	+	±	−	−
[BS04]	UML profile	BPEL	×		×		+	−	+	±	−	−
[SGS04]	UML profile	BPEL	×			×	+	−	+	±	−	−
[HV04]	UML profile	BPEL	×			×	+	−	+	±	−	+
[YZZ+07]	UML CCA	BPEL	×			×	+	−	+	±	−	−
[Sch07]	annot. EPC	BPEL		×	×		+	−	−	−	−	−
[All07]	annot. EPC	BPEL		×	×		−	+	+	+	+	−
[ZD06]	DSL	BPEL		×	n/a		−	+	+	+	+	−
[RLB07]	DSL	BPEL		×		×	+	−	−	+	+	−

language concepts or patterns, which enable reuse of recurring problem solutions. The cognitive complexity is also reduced by multi perspective modelling.

Depending on the considered transformation input, some approaches offer a completely automated transformation process. Adaptability is mainly addressed by framework based approaches. A practical usage of them requires extensions. In general, the need for incremental development is basically not considered in all reviewed works. Only one approach (the horizontal UML to BPEL transformation given in [HV04]) addresses this issue by taking advantage of bidirectional unambiguous pair grammar.

2.5 Service Discovery

2.5.1 Overview

In context of business process automation as described in subsection 2.1.5, business processes are automated by an IT implementation. Today, such an IT implementation is often based on web services (see subsection 2.2.2). To support business process automation, appropriate web services must be selected. A web service must be able to support the capabilities requested by a function in a business process. Hence, web service discovery supports the potential service consumer in finding a web service by analysing the available service description and comparing them to the service request issued by the service consumer (see also figure 2.3). A concrete service discovery implementation is influenced by different aspects:

- Different approaches for matching exist like structural, lexical, and semantic matching. The approaches are discussed in subsection 2.5.2.

- Different strategies can be applied to combine the results generated by the matching algorithm. The strategies are discussed in subsection 2.5.3.
- Service discovery differs if done during design-time or runtime (see subsection 2.5.3).
- Service discovery is influenced by how the service discovery process described in subsection 2.5.4 is implemented.

This section is structured according to the different aspects. It uses literature to illustrate the different aspects.

2.5.2 Approaches for Service Discovery

There are no specific approaches dealing with web service discovery for business process automation. However, there is an enormous number of publications about web service discovery in various domains. For example, many publications are targeting web service discovery in context of grid computing [FK99, FKT01]. Here, services are bound during execution often based on quality of service (QoS) parameters. A related domain is agent systems [Wei99] trying to identify a communication partner with a set of defined capabilities. Other publications deal with identifying web services in context of software engineering. The idea is to construct complex (software) systems by combining basic web services. Public market places are created so that service providers can advertise their offerings and service consumers can evaluate and bind them. Today, public standards for such service registries are available like UDDI [CHvRR04] and ebXML Registry Information Model [FNS05] as discussed in subsection 2.2.2.

Even though web service discovery is used in various domains, there are basically three different approaches applied:

1. *Structural* discovery approaches use syntactical information available like the interface description and the definition of the data messages exchanged between the communication partners. This kind of matching is very technical, as it requires the service requester to specify structural requirements like a certain operation signature or data type. A typical example of such a discovery approach is given by Ramasamy [Ram06]. Ramasamy compares operation names and operation parameters to the service request to discover web services.

2. *Lexical* discovery approaches use natural language descriptions. For example, web service operation names usually contain some terms describing their functionality. Also, WSDL and other standards allow embedding natural language descriptions. The lexical algorithms remove stop words from those descriptions, find synonyms using lexical databases like WordNet [Fel98] and compute similarity coefficients. For example, Zhuang et al. [ZMJ05] present an algorithm to compute the similarity of two web services. Their approach uses the information given in the WSDL files and does not require any additional annotations. They do manual pre-processing of the WSDL files to remove abbreviations, but it should be possible to use lexical databases like WordNet to automate this task in the future.

3. *Semantic* descriptions often based on ontologies are another major approach for web service discovery. They use formal methods to describe web service capabilities and properties so that machine reasoning can be used to identify possible candidates for a service request. There are competing formalisms for describing this semantic information like the Web Service Modeling Ontology (WSMO) [FLP+06] or OWL-S [MBH+04]. The proposed standard WSDL-S [AFM+05] provides some extensions for WSDL so that semantic descriptions in any formalism can be referenced from a WSDL file and so semantic annotation of existing web services becomes possible. An early example for semantic matching is Paolucci et al. [PKPS02]. They use DAML-S to describe the capabilities of a web service as well as the service request. In a more recent example Kritikos and Plexousakis [KP06] describe quality of service (QoS) parameters using OWL-S allowing matching on non-functional web service properties.

2.5.3 Strategies for Discovery Result Combination

Most discovery algorithms combine different approaches to achieve a better result. For example, Wang and Stroulia [WS03] combine structural and lexical analysis. Kokash et al. [KvdHD06, p. 526] describe three strategies for the combination of the discovery results generated by different discovery approaches:

- The *mixed* strategy uses different discovery approaches and matching algorithms in parallel and unites the returned result sets into one final result set. Normally, duplicates are removed from the final result set.
- The *cascading* strategy applies different discovery approaches and matching algorithms in sequence. A matching is only performed on the result set returned by the previous algorithm. This helps to reduce the amount of processing needed and it can increase the overall result quality. This can be seen as a stepwise refinement.
- The *switching* strategy selects between different discovery strategies and matching algorithms based on predefined criteria. For example, if the results returned by a first algorithm are not satisfactory, a second algorithm is used. The cascading and switching strategy can be combined to create more complex strategies.

Besides those strategies, there is another important characteristic to correctly classify web service discovery approaches and matching algorithms. One has to distinguish between discovery during design-time and runtime. The former is normally initiated by a user designing a web service composition for example to create a custom software application or to automate a business process. This is also sometimes referred to as early binding. The latter one is used during execution of a service composition. In this case, the composition only contains a requirements definition for a service call, but it does not specify which specific web service to use. At runtime, discovery is done to find all web services matching the requirements specification and the best fitting web service is used. This is sometimes referred to as late binding.

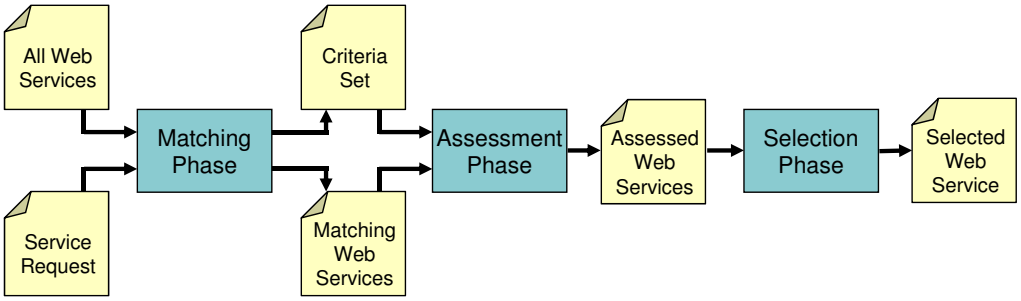

Figure 2.7: Service discovery process [KvdHD06]

2.5.4 Service Discovery Process

According to Kokash et al. [KvdHD06, p. 522] web service discovery consists of three major phases, also illustrated in figure 2.7:

1. During the *matching* phase, matching algorithms belonging to the different discovery approaches are applied. The results are combined according to the chosen strategy. The result set might solely consist of all web services matching the request or they might be ranked according to their fitness.

2. During the *assessment* phase, the matching results are further refined by a set of criteria. Where matching is normally done automatically, the assessment is often done manually, especially if web service discovery is done during design-time.

3. In the final *selection* phase a web service is chosen and used in the composition as intended. This might also mean to adapt either the web service or the consuming process or application.

2.6 Semantic Business Process Management

2.6.1 Motivation

Enterprise models captured with one of the popular modelling methods (see subsection 2.1.2) often contain valuable content. This content is communicated, analysed, and changed by human users to continuously improve the way the business is conducted. However, the content of an enterprise model is not directly machine processable, even though in most cases enterprise models are documented with some kind of software product like ARIS. A software might be able to list all diagrams of a certain type or to search for text in the different diagrams, but more complex analysis is not directly available. For example, if a manager is interested in a detailed analysis of the documented as-is processes, either a human user must perform this analysis using the content available in the enterprise model or a new analysis report must be implemented.

A similar situation exists in context of the Internet. The Internet contains an enormous amount of information, distributed among a high number of web pages, databases, web services, etc. Still, machines are not able to understand this information. To overcome

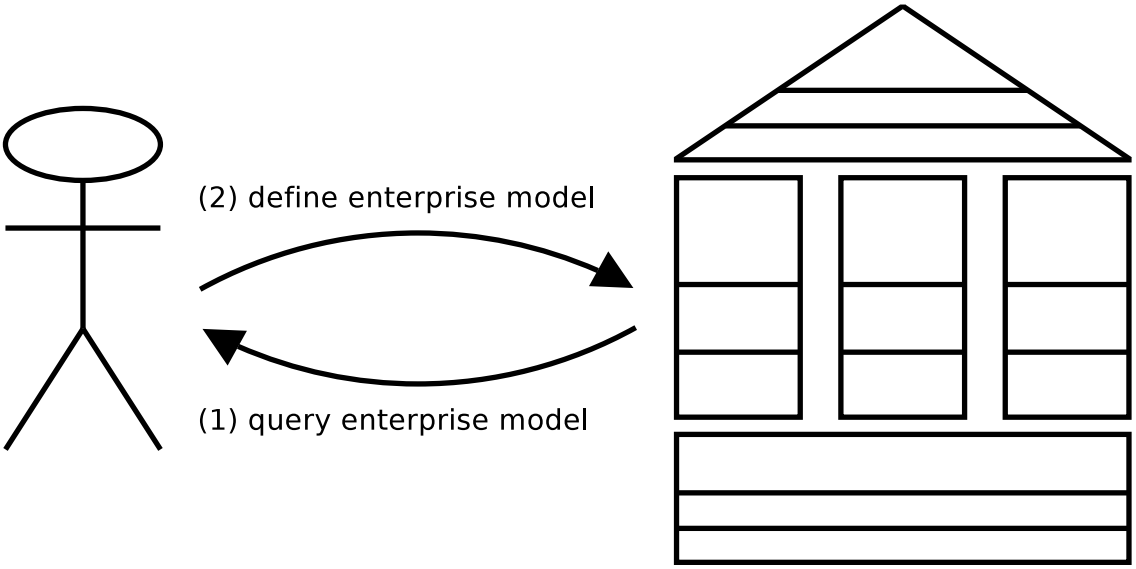

Figure 2.8: Use-cases of semantic technologies in business process management

this issue, Berners-Lee et al. [BLHL01] introduce the semantic web, which is powered by semantic technologies like ontologies, reasoners, and semantic web services to make the content of the Internet accessible to machines. Ontologies are used to capture the semantics of web content using taxonomies and logical expressions. Reasoners are specific software programs able to process those semantic descriptions to provide analysis capabilities. Web services described with ontologies lead to semantic web services. The introduction of semantic technologies to the Internet allows intelligent queries. For example, an agent program could be asked to organise a trip with some defined parameters. The software agent can now look up all necessary information on the Internet, because it understands the query issued by the human as well as the web services and their capabilities available on the Internet [BLHL01].

Hepp et al. [HLD+05] take the idea of the semantic web and apply to business process management. Business process management enabled by semantic technologies like ontologies, reasoners, and semantic web services leads to a new discipline known as semantic business process management. This section provides an overview of the current state in semantic business process management research.

2.6.2 Use-Cases of Semantics in Business Process Management

Figure 2.8 visualises the two main use-cases of semantic technologies applied to business process management (see also [HLD+05]):

1. Semantic technologies (especially reasoning) can be used to analyse semantic enterprise models.

2. Semantic technologies can be used to derive new parts of semantic enterprise models.

In the first use-case, semantic technologies are used to discover so far unknown facts from the enterprise model. This is possible, because the content of the enterprise model is processable by machines. For example, a food trading company is influenced by a new regulation released by the EU Commission or some other government organisation. The new regulation defines that certain kinds of vegetables must be sold within a shorter time period than before. The food trading company has to implement this regulation. Otherwise, it will loose its licence. A business expert has to analyse all as-is processes to find out those processes affected by the new regulation. This manual analysis work can be simplified if semantic technologies are used to query a semantically enabled enterprise model. The business expert defines a query based on the domain ontology used in the company. A reasoner uses the query as well as the content of the enterprise model to discover all business processes and other elements of the enterprise affected by the new regulation.

In the second use-case, semantic technologies are used to generate or at least partially define new elements of the enterprise model. For example, a business process is annotated with semantic descriptions specifying what capabilities are expected by each function so that they can be automated. Semantic web service discovery (see subsection 2.5.2) is used to find matching web services and reasoning techniques are used to derive an executable process model. If two interacting web services use different data structures, a mediator can be used to moderate between the different data formats. This is possible, because the data formats of the web services are specified semantically, too.

Semantic business process management also promises bridging the business-IT divide [HLD+05] as described in subsection 2.1.5. Instead of forcing business experts to apply IT concepts in business process modelling or asking IT experts to get familiar with business administration knowledge, an ontological mapping is established between both domains. Business experts and IT experts still use their own specific languages and methods, but they are integrated on a more abstract level through ontological mapping.

Semantic business process management is based on the assumption that all aspects of an enterprise model can be captured semantically. Therefore, ontologies must exist for the different aspects of an enterprise model. The following subsection introduces the currently available ontologies for semantic business process management.

2.6.3 Ontologies for Semantic Business Process Management

Ontologies are the backbone of semantic business process management, because they capture the content in a machine processable way. “An ontology is an explicit specification of a conceptualization” [Gru93, p. 1]. Ontologies are created in a development process called ontological engineering [MI96, GPFLC04]. This development process involves capturing the knowledge to be formalised. The knowledge is turned into competence questions. The competence questions are used to later check if the formalised ontology is able to answer the questions. Similar to software development, designing an ontology is an iterative approach, because the domain described is evolving while designing the ontology [Hep07].

An early version of an ontology for semantic business process management is pre-

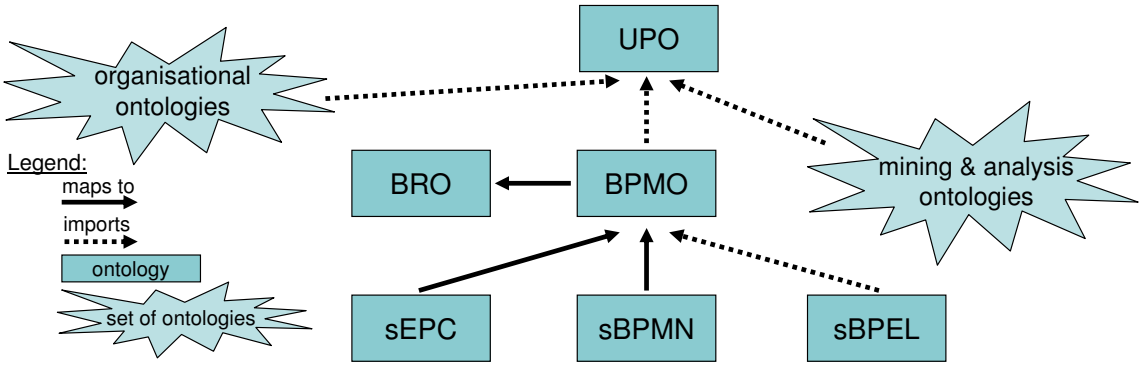

Figure 2.9: Ontology stack for semantic business process management [HR07, PBvL+08, SUP07]

sented by Uschold et al. [UKMZ98] as "Enterprise Ontology". The enterprise ontology provides concepts for organisational modelling, strategy modelling, process modelling, and resource modelling. Even though the enterprise ontology is quite comprehensive, it does not provide support for current modelling languages used in industry. Furthermore, the enterprise ontology is provided as one single definition not allowing replacing or extending parts of it with other ontologies.

The SUPER[9] research project, a public integrated project financed by the EU Commission, proposes a stack of ontologies to represent the various aspects of an enterprise model [HR07, PBvL+08, SUP07]. A simplified version of the SUPER ontology stack for semantic business process management is shown in figure 2.9. All ontologies are defined with WSML [FLP+06]. In contrast to the enterprise ontology [UKMZ98], there is not a single ontology representing all aspects of an enterprise model. The upper process ontology (UPO) [HR07, PBvL+08, SUP07] is used to integrated the other ontologies in the stack by defining common concepts like process, agent or role.

A successful adoption of semantic business process management in industry is very unlikely if existing enterprise models cannot be reused. To enable reuse, the ontology stack provides support for existing modelling languages like EPC and BPMN (see figure 2.9). The sEPC ontology [FKS08] is introduced for semantic business process modelling using the EPC modelling language. Besides the elements defined by the EPC modelling language, processes can be described semantically like annotating a function with a WSMO goal [FLP+06]. Similar work is available for BPMN with the sBPMN ontology [AFKK07]. The sEPC and sBPMN ontologies provide similar concepts, which are mapped to the business process modelling ontology (BPMO) [PBvL+08, SUP07]. In addition, BPMO is used to link the two process modelling ontologies to other ontologies like the set of organisational ontologies. The organisational ontologies are not depicted in detail, but a typical example is an ontology for defining business goals [MK08]. Additional ontologies are available or prepared, e. g. to model organisational structures or strategies.

BPMO facilitates the transformation of business process models into executable ones [PBvL+08], because only a single transformation is required instead of providing a trans-

[9] The author of this thesis is a member of the SUPER research project (http://www.ip-super.org/).

formation for every business process modelling language to be supported. The ontology stack also provides an ontologised version of BPEL, namely the sBPEL ontology [NWvL07]. In contrast to SA-WSDL [FL07], the sBPEL ontology does not only add links pointing to semantic descriptions to BPEL, but embeds the semantic descriptions directly in BPEL. For example, a new activity to invoke a semantic goal is provided. The semantic descriptions are based on WSMO [FLP+06] and are not solely used to support semantic web service discovery during runtime, but also to enable process mediation to integrate heterogeneous processes and data definitions.

The behavioural reasoning ontology (BRO) [Nor08] enables temporal reasoning. An additional set of ontologies is used to enable semantic process mining and analysis of executed business processes [PDAdM08]. Besides those core ontologies of the stack, it is possible to add domain specific ontologies. For example, Frankowski et al. [FRS07] define an ontology for the telecommunication company Telekomunikacja Polska.

At different points in the ontology stack, semantic descriptions of web services are used. Those semantic descriptions are expected to be based on WSMO [FLP+06]. WSMO proposes four main components to describe semantic web services, namely: ontologies, web services, goals, and mediators [FLP+06]. Ontologies are used to define terminology specific for the semantic web service. Web services provide the actual implementations of the semantic web service. Goals are used to describe the capabilities of the semantic web service from a user perspective. Here, the ontologies defined are used. Finally, mediators describe mechanisms to resolve interoperability problems. There are other ontologies available for describing semantic web services, e. g. OWL-S [MBH+04]. OWL-S is meant to support discovery, invocation, composition, and monitoring of semantic web services.

2.6.4 Methodology and Applications

Besides providing the necessary ontologies, it is also important to provide guidance how to apply them. The SUPER methodology [WMF+07, dFMM+08, WHMN07] defines a lifecycle consisting of four phases:

- In the modelling phase, business processes are captured using the sEPC or sBPMN modelling language. Besides defining the business process, semantic annotations are added to the models so that they are machine processable. Those semantic annotations allow auto-completion of processes, because similar already existing process models can be discovered while modelling.

- In the implementation phase, business processes are transformed into executable ones. For example, sBPEL representations are generated. Also, semantic web services, which are able to support the semantically annotated functions in the business process model, are discovered. If no semantic web service exists, service composition is used. During the implementation phase, also the necessary deployment information is generated.

- In the execution phase, the semantic business process is executed on a semantic execution engine. This execution engine supports the discovery of semantic web services during runtime if requested. Also, semantic web services can be invoked.

- In the analysis phase, executed business processes are analysed so that they can be optimised [CAdMZ+07]. Also semantic process mining techniques are used to extract so far unknown process models from execution logs [AdMPvdA+07] or to extract ontologies.

This short overview of the SUPER methodology for semantic business process management shows that it mainly focuses on business process automation. At the current point, semantic business process management focuses on a technological definition of business process management, similar to what Weske [Wes07] presents for non-semantic business process management (see also subsection 2.1.1). However, preliminary work exists focusing on other elements of business process management like compliance management. For example, El Kharbili et al. [EKSMP08b] introduce a framework for semantic compliance management. In this framework, regulations and laws are broken down into rules, which are semantically defined. Those semantic rules are used to check if a semantic enterprise model complies with those rules. The approach taken is similar to the validation techniques applied within the OrViA research project (see subsection 2.3.4), but here a different set of technologies is used.

2.6.5 Tools for Semantic Business Process Management

Semantic business process management can be understood as a modelling method as defined by Karagiannis and Kühn [KK02] and as discussed in section 1.1. Therefore, semantic business process management is not yet a complete modelling method by providing a modelling technique consisting of a modelling language (i. e. the ontologies introduced in subsection 2.6.3) and a modelling procedure (i. e. the methodology discussed in subsection 2.6.4). Semantic business process management must also provide tools to leverage the content defined with the modelling technique. As semantic business process management is still a young research discipline, only few tools are available.

Dimitrov et al. [DSSK07] provide an extension for the ontology development software WSMO Studio [DSKM07], which allows modelling of BPMO processes in a visual manner using the BPMN notation. In a similar effort, Born et al. [BHK+08] provide an extension of SAP's BPMN modelling tool Maestro, which allows creating sBPMN processes and initiating semantic process composition. Maestro is not a product available to ordinary users, but instead it is a research prototype used at SAP to evaluate new technologies.

Preliminary work on a semantic BPEL execution engine exists as well. In [vLND+07], the authors describe an architecture of such an execution environment. One important component is the semantic service bus, which provides access to reasoning services and functionality to invoke semantic web services. Here, existing environments like WSMX [HCM+05] and IRS III [DCH+04] can be reused. Currently, the authors of those two environments are collaborating to create a reference architecture of a semantic execution environment as a public OASIS standard[10].

[10] See the homepage of the belonging technical committee at OASIS: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=semantic-ex

In another effort focusing on semantic process mining, Alves de Medeiros et al. [AdMvdAP08] describe an architecture of tools based on the ProM platform[11]. First semantic plugins for ProM are already available on ProM's homepage.

2.6.6 Evaluation

This overview shows that semantic business process management is an actively researched area. So far, most efforts are fundamental research either:

- defining the necessary languages [HR07, PBvL+08, UKMZ98, SUP07, FLP+06, FKS08, AFKK07, MK08, NWvL07, Nor08, PDAdM08, FRS07] or
- providing some preliminary tools [DSKM07, DSSK07, BHK+08, vLND+07, HCM+05, DCH+04, AdMvdAP08] or
- outlining overall approaches and methodologies [HLD+05, WMF+07, dFMM+08, WHMN07, CAdMZ+07, AdMPvdA+07, EKSMP08b].

However, empirical evaluation of the proposed technologies to validate the practical relevance of semantic business process management is still missing. Also, integration of semantic business process management with current modelling methods is not investigated yet. Such work would provide feedback, which could be incorporated in research agendas to realign the research efforts with requirements from the industrial field. It could also demonstrate the relevance of semantic business process management to practitioners, fostering the adoption of it in industry.

[11] http://prom.sourceforge.net/

Chapter

3 Modelling Language for Service-Oriented Business Process Management

This chapter describes one contribution of this thesis (see figure 3.1): the extension of the ARIS modelling language to enable support for service-oriented business process management. Section 3.1 first refreshes the motivation for such an extension, discusses some constraints, and describes the process followed while developing the extension. The presentation of the extension can be divided in three major parts: First, the requirements to be supported are described (section 3.2) and formalised as the SOA meta model (section 3.3 and section 3.4). All elements of the SOA meta model are described in detail (section 3.5). Second, the ARIS extension developed is explained in section 3.6. Finally, the usage of the extended ARIS modelling language is illustrated by an example (section 3.7), which concludes this chapter.

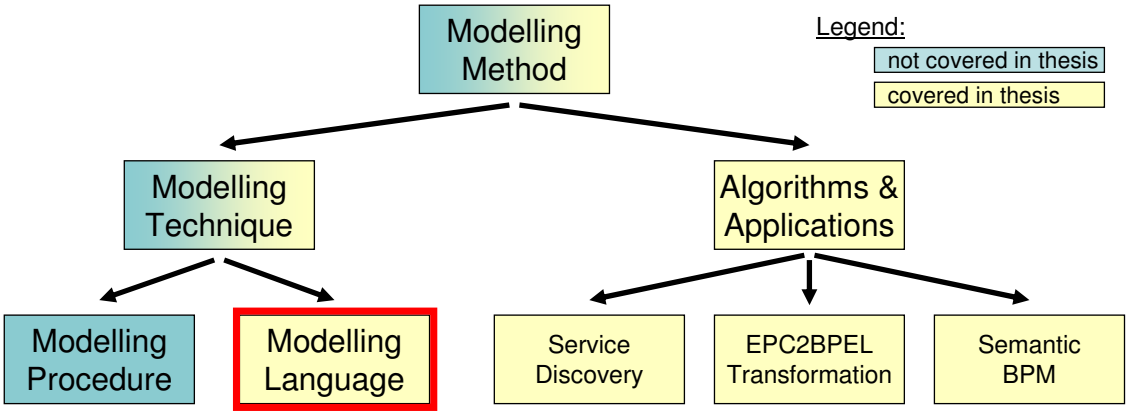

Figure 3.1: Contribution ARIS modelling language extension

3.1 Overview

3.1.1 Motivation

In literature (see section 2.1), two different understandings of business process management exist: a business-oriented and a technical understanding. Even though the second one can be seen as a part of the first one, an integration between both understandings is only hardly achieved. A similar situation exists for works on SOA as shown in the literature review in section 2.2. However, the technical understanding of SOA has far more support and is clearly leading.

In fact, there are only few attempts trying to integrate those different understandings. A notably exception is the OASIS SOA Reference Model [MLM+06], because it abstracts from concrete usages. Besides, other works exist as shown in subsection 2.2.3. However, those works offer only single point solutions by addressing only parts of a modelling method. As a matter of fact, there is no complete modelling method covering all parts. For example, CBDI-SAE [CBD07] defines a comprehensive meta model, but it misses a modelling language and modelling procedure.

To overcome this problem, this thesis extends the existing modelling method ARIS to cover all relevant aspects of a modelling method for service-oriented business process management. In this thesis, this is called “ARIS extension”. The ARIS extension was designed in a structured development process as described in subsection 3.1.3. The design of the ARIS extension was also constrained by several points, which are discussed in the following subsection.

3.1.2 Constraints

The design and implementation of the ARIS extension was constrained by the following points discussed below. The points are general rules by IDS Scheer applying to all ARIS extensions done. They try to prevent ARIS modelling method getting too complex.

1. The ARIS extension must enable reuse of as many existing ARIS models as possible to preserve users’ investments. Introducing a completely new modelling language is not a valid solution.

2. The ARIS extension must be fully integrated with the existing ARIS modelling language. This includes reusing as many existing objects and diagrams as possible.

3. Relying on pilot users is not possible, because users expect to receive a solution rather than standardising their competitive advantage.

4. The skill sets of the typical ARIS user must be taken into account. Internal user analysis at IDS Scheer shows that most ARIS users have no natural science nor mathematical background but instead studied business administration and related social science subjects.

5. There are many software modelling languages available like UML, which must be integrate if possible.

Those points constrain how a possible solution might look like. For example, the SOA meta model proposed by CBDI-SAE [CBD07] is too complex. Also, the non-availability of pilot users means that gathering requirements for such an extension through user involvement is not possible. The following subsection describes how the work on the ARIS extension was organised.

3.1.3 ARIS Extension Development Process

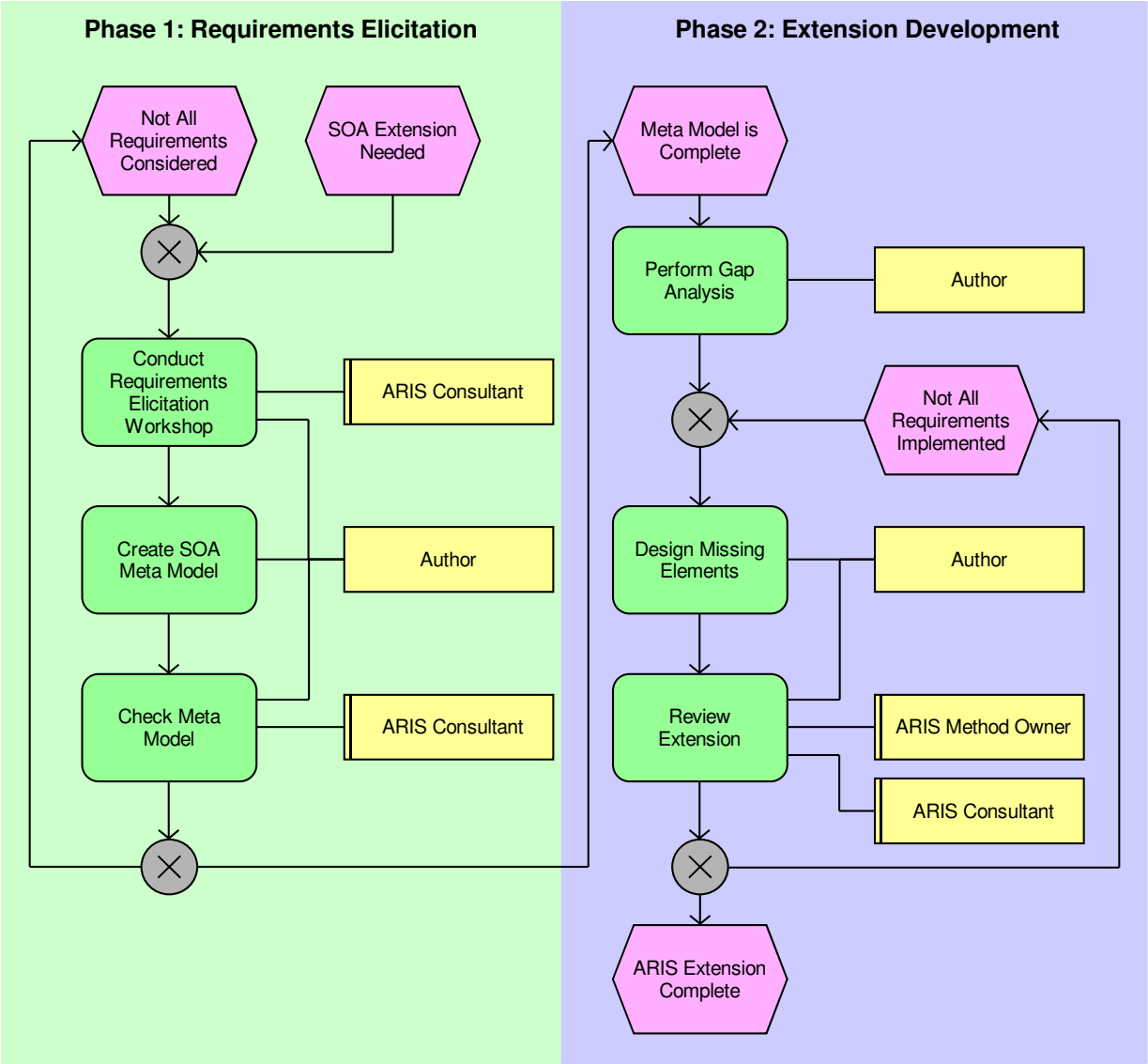

Figure 3.2: ARIS extension development process

The development process applied for creating the ARIS extension consists of two major phases (see also figure 3.2):

- In the first phase, the requirements for such an extension are gathered. This is shown on the left side of figure 3.2.

- In the second phase, the ARIS extension is developed based on the requirements gathered before. This is shown on the right side of figure 3.2.

Figure 3.2 shows the different steps of the development process. To gather requirements, a workshop with IDS Scheer consultants working in the domains enterprise architecture management, business process automation, IT architecture management, and general business process management was conducted. Afterwards, the requirements were turned into the SOA meta model presented in section 3.4, also using other existing work on this topic (see subsection 2.2.3 for related work). This SOA meta model was reviewed by the consultants. Those three steps were repeated several times to ensure all requirements are reflected in the SOA meta model. After finishing the SOA meta model, the second phase of the development process was started.

During the second phase, first a gap analysis was performed. The existing ARIS modelling language was compared with the SOA meta model to find those parts of the SOA meta model not covered yet by the ARIS modelling language. The gap analysis is presented in subsection 3.6.2. If a gap was identified, the gap was closed by designing the missing parts. Those extensions were reviewed by the ARIS method owners as well as the consultants, who were involved in requirements elicitation. The method owners provided feedback whether the proposed extension follows the guidelines of the ARIS modelling technique. The consultants checked that all their requirements are fulfilled in the design of the ARIS method extension. After no further gaps could be identified, the design of the ARIS extension was considered to be complete.

The following sections first present the identified use cases to be supported by the ARIS extension (section 3.2), considerations about a service description (section 3.3), an overview of the SOA meta model (section 3.4), a detailed look at the elements of the SOA meta model (section 3.5), the ARIS modelling language extension (section 3.6), and an example illustrating the ARIS extension (section 3.7).

3.2 Use Cases to be Supported

3.2.1 Overview

From an abstract point of view, it is the aim of the ARIS extension to support business experts in creating a business-oriented service description. Such a service description helps business experts to evaluate the capabilities provided by a service and to decide if a service should be used. A comprehensive service description does not solely detail the capabilities of a service, but also who implements the service, who is responsible for the service description, and how the service can be used. The main use cases are (partly based on [OEtH02]):

- Service discovery: A potential user defines his need and all existing and available services having the necessary capabilities are suggested.
- Service composition: Several services are combined to provide a more complex capability.

- Service substitution: If a currently used service becomes unavailable, the service is replaced with another service with similar capabilities.
- Service governance and management: Creating and maintaining a service architecture without redundancies and with clearly defined responsibilities.

Those use cases are further described in the following subsections. In reality, those use cases are interrelated. For example, service discovery requires up-to-date service descriptions, which is the task of service governance. Also, service substitution might depend on service discovery so that a replacement can be identified quickly.

3.2.2 Service Discovery

The service description defines which capabilities are provided by a service. Therefore, the service description is a central information object for potential service consumers trying to identify a service for their needs. By comparing the offered capabilities with the own needs, a service consumer decides whether the usage of the service is possible and beneficial.

This comparison of need and capability is similar to general market mechanisms relying on supply and demand. Services are suppliers of capabilities and service consumers are demanders of needs. The evaluation of a service is not limited to the capabilities, but it must also include the conditions under which the capabilities are provided. For example, the service usage creates costs, which must be covered by the service consumer. If the service consumer (e. g. a company department) does not pay them directly, it will pay them indirectly by creating enough business value to cover the costs of an IT department. Besides financial conditions, there might be other conditions like the temporal or spatial availability of the service.

It is also possible that the service itself defines conditions to be met by the service consumer. For example, a service might define that all communication must be encrypted. A service consumer issuing a non-encrypted request will be rejected in such a scenario.

The service description is an offer by the service provider. The offer is binding meaning that the service must provide the advertised capabilities if requested. The OASIS SOA Reference Model [MLM+06] characterises this as “willingness”.

Besides providing the necessary ARIS extension to describe a service so that it can be discovered, this thesis also contributes a service discovery application supporting a service consumer in discovering a matching service for a function in a business process. Here, service discovery is done during design-time while the business expert is creating a business process model. This application is described in chapter 4. The application semantic business process management uses a formalised service description so that the service description is machine processable. A prototype is provided demonstrating service discovery during process execution. This application is described in chapter 7.

3.2.3 Service Composition

It is common to not only invoke a single service, but instead to use several services to consume a more complex capability. Such a service composition can be implemented

differently. For example, based on a business process an executable service orchestration is created. This service orchestration itself is provided as a service. For a service consumer, it does not make a difference if a single service or a service composition is invoked. Another possibility for service composition is the combination of services in a software architecture. Such an approach is based on previous work done in the area of component-oriented software engineering like [Szy97, HS00].

Having standardised interfaces is a fundamental requirement of service composition. In addition, the interfaces must be documented in the service description. Only if services provide their functionality through standardised interfaces, which can be integrated easily, service composition can be done. However, having standardised interfaces does not mean services must be implemented with the same technology. The service composition is based on the service interface and the service description, but the technology behind is not visible.

This thesis extends the modelling language ARIS to cover all aspects necessary for service composition. In addition, an exemplarily application is developed demonstrating how to turn a business process modelled with the EPC notation into a service orchestration based on BPEL. The belonging application generates a service interface for the BPEL process as well, so that it can be consumed like any other service. This application is described in chapter 5.

3.2.4 Service Substitution

The service substitution use case adds details to a service description so that a service with similar capabilities can replace a currently used one. A substitution is only possible if the substitute has no incompatible capabilities or conditions. For example, if a currently used service supports encrypted as well as non-encrypted invocation, a substitute can only constrain the communication to encrypted if the service consumer also supports an encrypted request.

Service substitution is important in failover scenarios. Here, a service becomes unavailable, e. g. because of hardware failure. The service must be replaced immediately by a substitute to ensure the operation of the business process or company. Therefore, service substitution requires that services with similar or equal capabilities are described in a similar way so that a substitute can be found easily. Also, the ability to explicitly define a substitute might be needed in some cases.

3.2.5 Service Governance and Management

Services and their development must be planned, maintained, and continuously improved. The belonging activities are summarised as service governance or service management. Activities can be grouped in operational, tactical, and strategical tasks.

Operational activities focus on a short timeframe. They ensure the daily availability of deployed services. They also monitor and control the current execution of services. In case of breakdown, it is an operational task to provide a substitute if the failed service is critical for the operation of the company. Tactical tasks are focused on a midterm timeframe.

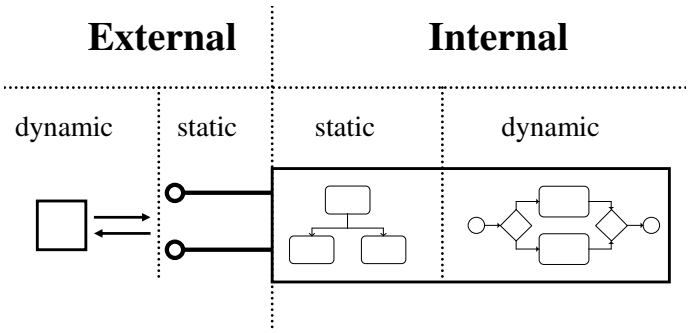

Figure 3.3: Views and aspects of service description

Here, the deployment or replacement of services is planned and implemented. Strategical tasks have a significant longer timeframe. For example, they focus on the development of the overall service architecture like defining the necessary service categories to group services. They also establish the necessary means to enable service usage like training potential service consumers in evaluating service descriptions. A service description must enable service governance and management for operational, tactical, and strategical tasks.

The following section introduces some basic considerations about a service description as a systematic. It details which views, aspects, and levels must be represented by a service description.

3.3 Service Description Systematic

3.3.1 Aspects and Views

This section introduces a systematic, which defines the basic structure of a service description. This systematic is mainly based on general system theory [vB76], because a service can be viewed as a system. In that sense, a service consists of elements and relations among those elements. There are static as well as dynamic relations possible between different elements. Therefore, a service description must distinguish between dynamic and static aspects as shown in figure 3.3. For example, a company might define a service architecture decomposing some high-level services into detailed ones. Here, static relations are used to build a hierarchy of services. A service might not be always available, for example depending on location and time. Here, a dynamic relation is established between service, location, and time.

As outlined in the previous section, there are different use cases, which a service description has to support. Service governance and management is mainly an internal activity not completely visible to service consumers. In contrast, elements of a service description used during service discovery must be available externally to enable the evaluation of the capabilities by a potential service consumer. Therefore, a service description must distinguish between an internal and external view as shown in figure 3.3. The external view contains all elements relevant for service consumers and which should be visible externally, whereas the internal view contains all remaining elements. External and internal

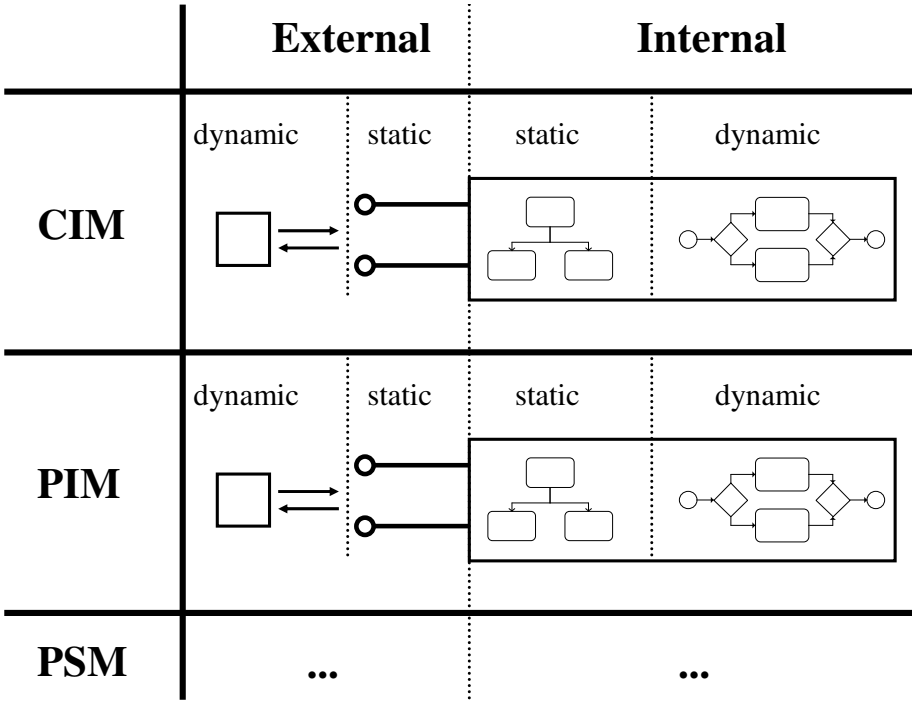

Figure 3.4: Levels of service description

view represent dynamic as well as static information. For example, the structure of the service interface is a static external information. An interaction protocol constraining the invocation order of service operations is a dynamic external information, because it is important for service consumers to know in which order to invoke certain operations of the service. In contrast, an information describing the physical deployment of the service is an internal information, which is not relevant for an outside viewer.

Even though it is clear that a service description must distinguish between internal and external view, it is impossible to define which elements belong to which view. Each company has to define on its own, which information must be provided to service consumers and which information is only available internally.

3.3.2 Levels

The service description provides various information of different detail. Also, the service description is used by different users like business experts, IT architects, and system administrators. Therefore, the service description must distinguish different levels of detail as shown in figure 3.4. This thesis uses the levels defined by model driven architecture (MDA) [MM03] (see subsection 2.3.2), namely:

- The computation independent model (CIM) level describes the service from a business point of view. A service described on this level can be implemented in many different ways like contracting an external party or providing an IT implementation.
- The platform independent model (PIM) level describes services provided as soft-

ware. However, this level is not bound to any specific technology, so a service can be implemented e. g. using the W3C stack or based on REST.

- The platform specific model (PSM) level describes services based on a specific technology or framework like web services. A web service description provided in the WSDL format is such a platform specific model.

In this model, a level is always independent from the levels below. For example, a service might be described on CIM level, but the service might be not available as an IT implementation. In that case, the PIM and PSM levels are missing. In another case, a service might feature several IT implementations and so several PIM and PSM levels are available.

Most current work on SOA focuses on the PIM and PSM level by providing standards to describe technical service artefacts. A service description independent of IT is ignored. This hinders an integration with business process management, because business processes are described independent of IT by business experts. By adding a computation independent level to the service description, an integration is enabled. However, only having a computation independent service description is not enough either, because more technical descriptions are required for business process automation.

3.3.3 Service Categories

In business process management it is common to group business processes in different categories like core processes and supporting processes [SS08a]. The same is useful for services. This requirement can also be found in SOA literature (e. g. [Jon06]). Applied to the service description systematic, it must be possible to represent different service categories as shown in figure 3.5.

Each service is assigned to exactly one category. To allow an unambiguous assignment, clear criteria must be defined for each category. Those criteria must be reworked if a service cannot be clearly assigned to a category. Besides defining criteria, it is also important to define the viewpoint from which values for each criterion for a given service are defined. For example, a company may classify a service to send short text message to mobile devices as infrastructural service. However, a company, which is operating a mobile network, may classify such a service as a value adding service, because it is probably one of their cash cows. That shows that defining the viewpoint is important for service categorisation. It is a task of service governance to define categories and the belonging criteria. Also, the categorisation of each service must be checked from time to time to see if a service still belongs to a category. There is no guarantee that a service belongs always to the same category for its whole lifetime.

Categorisation of services can be used to define specific management and governance processes. For example, services directly impacting the overall performance of the company may need a more careful management than a service, which is very seldom used. Because of limited resources, it can be decided to only focus on services of a specific category ignoring all other services. Maintenance of services in different categories might

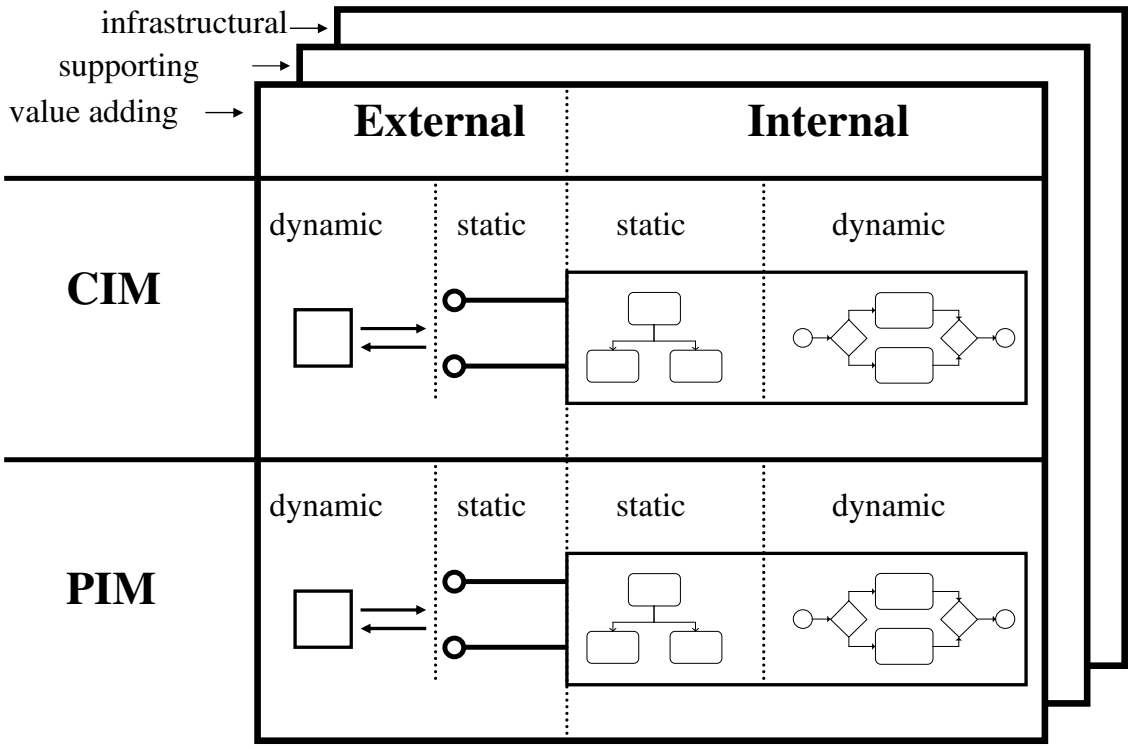

Figure 3.5: Service categories

also require different skills. For example, an infrastructural service might require comprehensive IT knowledge whereas maintenance of a value adding service relies on business administration knowledge.

The number of categories and the category names are company specific. It is impossible to provide a pre-defined set. Therefore, a modelling language for service description must be customisable. However, this thesis suggests introducing the following three categories with the following criteria (see also figure 3.5):

- Value adding services are the core of each company. According to their name, they directly contribute to the success of the company by generating business value. Value adding services are provided to customers and are therefore visible externally. It can be expected that only few value adding services exist. Jones [Jon06] calls this category business services and estimates their number to less than 10. It is unlikely that a value adding service can be implemented completely through IT. Probably, a value adding service is a complex composition of many services, which are not all implemented through IT. Possible criteria fulfilled by a value adding service are:
  - directly contributes to the value creation of the company
  - cannot be substituted
  - is externally visible
  - contains business logic
  - cannot be implemented completely by IT
  - there are less than 10 services of this kind

  - often offered to customers

- Supporting services are the building blocks of value adding services. They provide common business logic, which is reused in many places. Therefore, supporting services have a high degree of reuse. Typical examples of supporting services are accounting, human resource, and payroll management. There is usually a good software coverage for supporting services so that they can be implemented using standard software packages. It is possible to substitute a supporting service. However, such a substitution must be carefully planned and managed as a change project. Possible criteria fulfilled by a supporting service are:
  - service supports value creation
  - contains business logic
  - can be substituted, but not on short notice
  - is not offered to customers
  - can be implemented through standard software
  - high degree of reuse
  - there are several hundred services of this kind

- Infrastructural services represent the many individual IT functions needed to operate a company. Those services do not support value creation, even though they can be used in a service composition creating value. However, infrastructural services only provide technical capabilities, which can be substituted easily and on short notice. Such services are not externally visible and a very high degree of reuse can be expected. Such services are normally provided by application systems, standard software, middleware products, and external service providers. Possible criteria fulfilled by an infrastructural service are:
  - does not support value creation
  - does not contain business logic
  - can be substituted on short notice
  - not externally visible
  - probably completely automated execution
  - very high degree of reuse
  - there can be several thousand of this kind

Most SOA literature and approaches focus only on infrastructural services, ignoring the more abstract service categories. The following section presents the SOA meta model, which relies on the systematic described in this section. The SOA meta model supports dynamic as well as static aspects, which can be distributed between an internal and external view. The SOA meta model describes a service on different levels of abstraction (i. e. CIM, PIM, PSM) and enables the definition of company specific service categories.

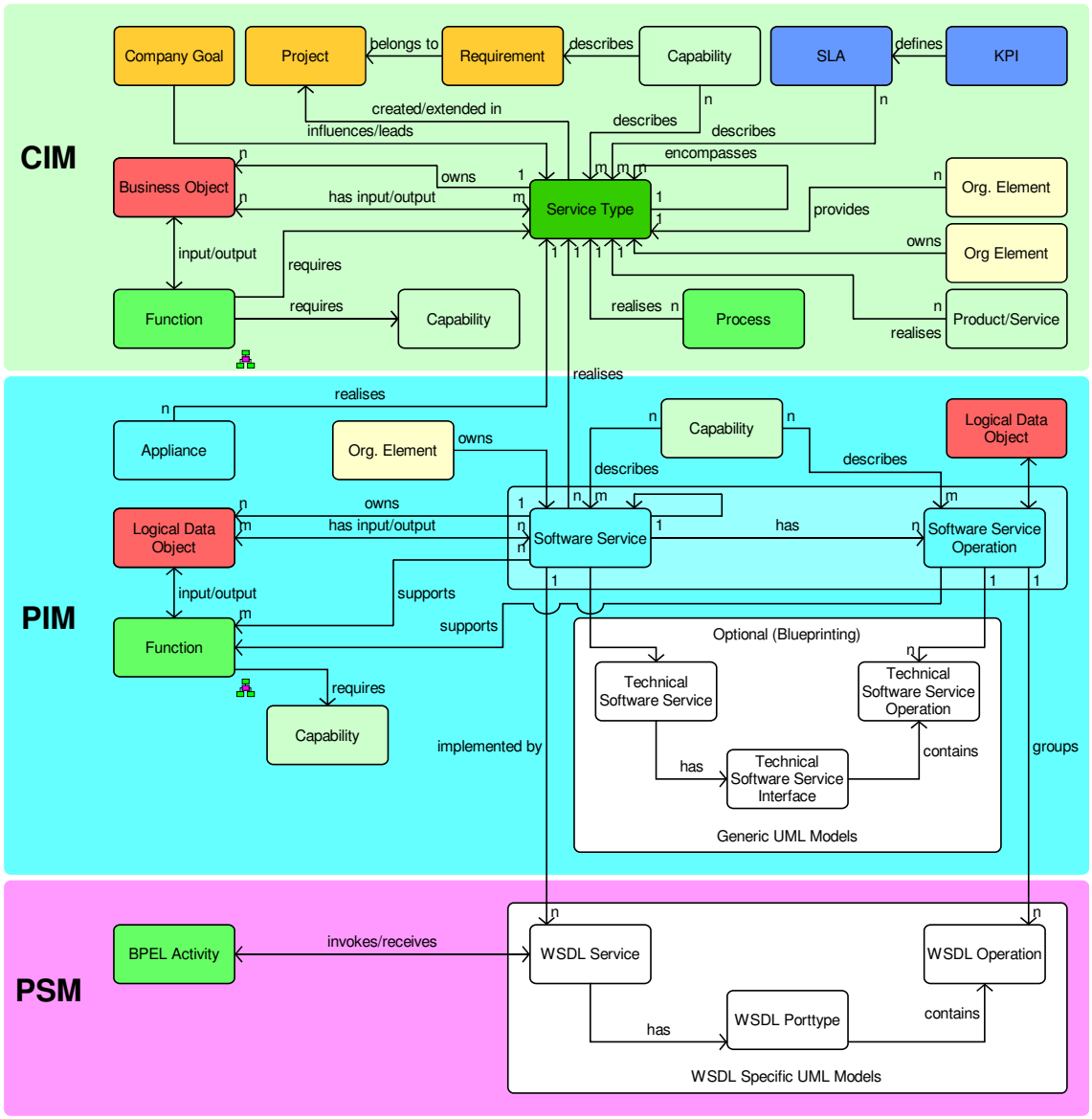

Figure 3.6: SOA meta model

3.4 SOA Meta Model

Figure 3.6 shows the complete SOA meta model. The SOA meta model was created by gathering requirements from consultants. Those requirements were formalised as the SOA meta model (see subsection 3.1.3 for details on the development process). Figure 3.6 uses an informal modelling language to represent the SOA meta model. Rectangles represent concepts of the SOA meta model. Some objects are grouped like the generic UML models used on the PIM level to represent a platform independent software service design. This grouping carries no further semantics, but is introduced to make the model more readable. Concepts as well as relations represent elements of a service description.

Using an informal modelling language might not be a good practice. Therefore, part of the SOA meta model is available as UML class diagram in figure 3.7 to demonstrate that

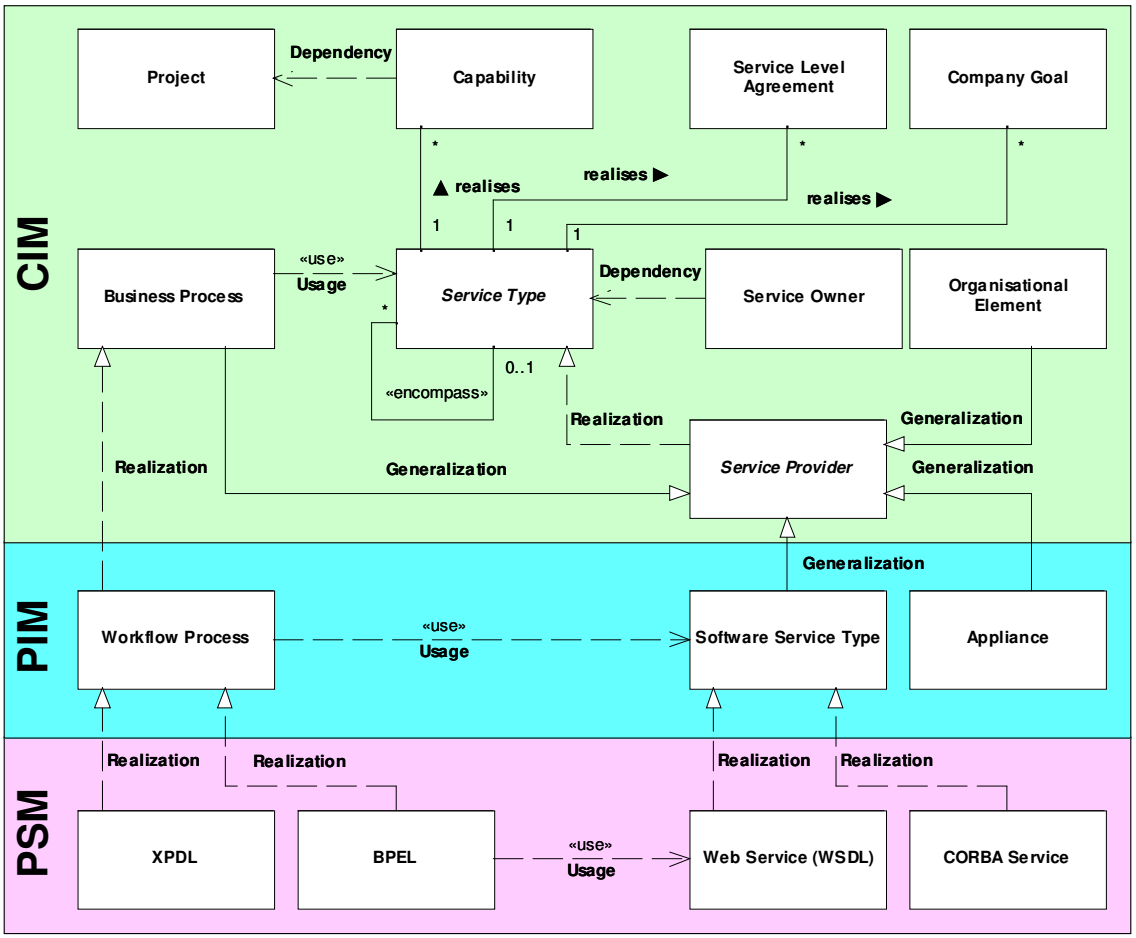

Figure 3.7: Subset of SOA meta model represented as UML class diagram

the informal description can be formalised. Using UML for meta modelling is an accepted approach according to Karagiannis and Kühn [KK02]. The UML class diagram only shows a subset of the SOA meta model, because the notation requires more space. Therefore, in the following sections the informal representation shown in figure 3.6 is used while describing the different concepts of the SOA meta model.

The structure of the SOA meta model follows what is described in the systematic in section 3.3. The SOA meta model is structured into the three MDA levels CIM, PIM, and PSM. Dynamic as well as static aspects are not distinguished in the SOA meta model, because the SOA meta model describes all relations between the concepts. However, dynamic as well as static aspects are later distinguished in the modelling language developed by providing different diagrams (see subsection 3.6.4). As discussed in the systematic in subsection 3.3.1, it is impossible to define, which elements of a service description belong to the internal and external view. Therefore, the SOA meta model does not assign concepts to specific views.

The different elements of a service description, represented as concepts and relations in the SOA meta model, are described in the following section.

3.5 Service Description Elements

3.5.1 Overview

The SOA meta model shown in figure 3.6 consists of several concepts and relations between the concepts. Each concept represents an element of a service description. It can be seen that most concepts focus on the CIM level and fewer concepts are provided for the PIM and PSM level. Of course, there are important aspects on those levels as well, but those levels are already covered well by the ARIS modelling language and are therefore not included in the SOA meta model. For example, it is possible to model on which kind of hardware a software service is deployed. This is already possible prior the modelling language extension contributed in this thesis. The following subsections describe the different elements, thereby specifying the scope of the modelling language to be developed.

3.5.2 Capability

Entities like organisations, IT systems, and business processes have a set of capabilities. Their capabilities enable them "to solve or support a solution for the problems they face in the course of their business" [MLM+06, p. 8]. A capability is a descriptive element documenting a system. In the specific case of the SOA meta model (see figure 3.6), capabilities are used to describe service types and software service types. Capabilities can be used to describe functional and non-functional properties. However, in case quantitative description is possible, defining service level agreements (SLA) is recommended, because they are more expressive than capabilities. For a discussion on non-functional properties see subsection 3.5.5.

The SOA meta model does not limit the way capabilities are defined. The reviewed works use different formalisms for describing capabilities and there seems to be no consensus among them. For example, in WSMO [FLP+06] a formal language based on logical expressions is used. On the other hand, MODAF [MOD07] suggests defining a taxonomy. The SOA meta model allows linking capabilities to more detailed and maybe even formalised descriptions like WSMO goals, but such a description is not mandatory. Chapter 7 introduces an application using such formalised descriptions.

A capability is always global and can be reused to describe different services. Therefore, a capability must clearly define the context it can be used in. Reusing capabilities among services allows identifying services with similar capabilities (see use case service substitution in subsection 3.2.4), which is important e. g. in case of service failure or while harmonising a service architecture.

To ensure reuse of capabilities, the capability architecture must be actively managed. Unused capabilities must be reviewed and newly created capabilities must be integrated to prevent redundancy. The vocabulary used for naming capabilities must be standardised to prevent competing definitions and misunderstandings. If the capabilities of an entity like an IT system are changed, this must be manually reflected in the belonging models. This is a management issue and cannot be automated. Specific methods like ontological engineering [MI96, GPFLC04] must be applied for creating a sound capability architecture. The

modelling procedure ARIS Value Engineering (AVE) provides specific guidelines. However, AVE is not part of this thesis.

3.5.3 Service Type

The service concept presented in MacKenzie et al. is generic enough to not only cover software or web services, but instead any "mechanism enabling access to one or more capabilities" [MLM+06, p. 12]. A similar view can be found in service science [Teb06], where a service is also not limited to technology. This abstract notation of a service is also present in the SOA meta model by introducing the service type concept as a mechanism enabling access to a set of capabilities. A similar construct called "notional service" can be found in CBDI-SAE [CBD07].

Service types themselves can be grouped into different categories like business services, decision services, supporting services, infrastructure services, etc. (see subsection 3.3.3). There is no consensus on useful service categories, because the categorisation depends on the company's context. Therefore, the SOA meta model does not distinguish between different service types. However, it allows the definition of different subtypes so that it is possible to group services into categories.

The functionality made available by the service type is described by relating capabilities to it. A capability can be reused by several service types to describe similar service types. Besides, the service type is related to much more concepts in the SOA meta model. The purpose of those relations is described in the following subsections.

3.5.4 Service Owner

Each service type has a service owner. The service owner is responsible for maintaining the service type's description and advertising the service offering. In addition, the service owner is responsible for reviewing change requests like newly requested or changed capabilities. It might be also the task of the service owner to manage the development of the service. If the tasks are too complex to be handled by a single person, the role can be shared. Knowing who is responsible for a service is important for external service consumers as well as internally. Therefore, this information should be available in the external view of the service description.

3.5.5 Non-Functional Description

Describing a service using capabilities has the disadvantage that it is only an unstructured definition. For example, if a capability defines that a service must be reacting quickly to a service request, it is unclear what "react quickly" means. In such cases, it is better to use concrete quantitative definitions instead of relying on "soft" specifications. Usually, such a quantitative definition describes the circumstances under which a service is provided. Such descriptions can be seen as non-functional properties of the service. However, while developing the SOA meta model there was no consensus among the involved consultants on a clear definition for functional and non-functional properties. Therefore, capabilities are

not divided in those two categories. Instead, the SOA meta model shown in figure 3.6 only distinguishes between capabilities and service level agreements (SLA), which are defined through key performance indicators. It is up to the users of the ARIS extension to define the SLAs needed to correctly describe their services.

According to O'Sullivan et al. [OEtH02], non-functional properties can be further divided:

- Availability refers to the temporal and spatial constraints applied to a service [OEtH02]. This requirement is detailed in subsection 3.5.12.
- Channels constrain the way a service is accessed [OEtH02]. This is reflected in the SOA meta model by allowing different realisations of a service type as discussed in subsection 3.5.8.
- Charging styles define how service usage must be paid. Typical examples are charging per request, charging per measured unit, and charging based on a ratio of the service effect (e. g. commission) [OEtH02]. It is expected that capabilities are used to model the different charging styles.
- Settlements define the obligations to be expected from service usage. For example, a service might be rented for a longer time or a new contract is established each time the service is consumed. Additional terms and conditions might be defined for service usage like a description what happens if something goes wrong during service delivery [OEtH02]. Again, capabilities are used to model this information unless quantitative descriptions can be provided. In that case, SLAs are used.
- Payment obligations define how the service usage must be paid. For example, it must be defined if the service usage is charged immediately upon request or only after the service request was answered completely [OEtH02]. It is expected that capabilities are used to model this information.
- Service quality defines how the service is delivered like reliability and responsiveness [OEtH02]. That are quantitative descriptions and therefore SLAs are used.
- Security and trust must be defined as well, for example how data send to the service is protected [OEtH02]. It is expected that capabilities are used to express this kind of information.
- Ownership and rights received by service usage must be specified as well. For example, it must be defined if the service provider is allowed to offer the service to other service requestors as well [OEtH02]. Again, capabilities are used to model this information.

This list of non-functional properties shows that a service description must contain many more details besides the functional description. The details are necessary so that a valid legal contract is established by requesting a service.

3.5.6 Data Description

The SOA meta model shown in figure 3.6 relates the different service concepts on the CIM, PIM, and PSM level to data objects. It can be seen that each service concept can consume, produce, and own data. It is important to note that different kinds of data are used on the different levels in the SOA meta model. For example, data descriptions on the CIM level must be independent of computational aspects (i. e. conceptual data models like business objects or a shared business vocabulary). Here, a business vocabulary or a taxonomy of the main business terms might be used. In contrast, XML schema definitions (XSDs) may be used on the PSM level.

Consuming and producing data establishes an important link between services and business processes, because functions also consume and produce data. For example, if a service should support a function, it must be able to consume and produce the same data as the function does.

Services can also own data. This is an important modelling concept in context of component oriented software design [Szy97, HS00]. For example, the methodology established by Herzum and Sims [HS00] makes heavy use of designing components around central data objects. Such components can be represented with the service type. Therefore, a service type can own data. Such an ownership specifies that all access to the data must be made through the service. This is a conceptual requirement, which can be relaxed during implementation, for example to ensure high performance.

3.5.7 Service Architecture

It might be useful to nest services to build a service architecture. Therefore, the SOA meta model shown in figure 3.6 allows a recursive relation to the service type concept itself. There is no general rule whether the more fine grained services are visible externally or internally. Also, it might be possible that only fine grained services can be consumed, but that those services belong to a more abstract service in the service architecture. Even though services can be refined into more fine grained services, it is not defined that those fine grained services must belong to the same service category as the parent service.

The nesting of services to build a hierarchy of services is a common concept, which can also be found in component oriented software design [Szy97, HS00]. Also, it might be a use case to model the overall structure of an enterprise using services as shown in [JM05, Jon06].

3.5.8 Available Realisations

Figure 3.6 shows that a service type can be realised or provided in different ways, namely:

- as a software service type
- by an organisation
- as an appliance (combination of hardware and software)

- as a business process
- by a product

Each service provider must have all capabilities specified by the service type. Otherwise, the service provider does not fulfil the service offering as described by the service type. It is important to note that the service type itself is not a realisation, which can be consumed. The service type is only a mechanism to provide access to such realisations and it is described using capabilities offered by all realisations.

The real-world effects resulting from contracting a service provider must be the same for all service providers. However, the service realisations provided might differ in the way they achieve the real-world effects. Therefore, realisations can have additional capabilities as long as they do not provide fewer or conflicting capabilities as defined by the service type.

3.5.9 Strategy Alignment

In order to align the service architecture with the company strategy, a service type can be related to company goals. In that way it is possible to model, which company goals are supported by a service type. The ARIS modelling method supports different methods and techniques to strategy definition like balanced scorecards. If SOA is not solely understood as an implementation technology, it is important to align the service architecture with the overall company goals. The service landscape is informed by the company goals providing support in implementing it. Therefore, relating company goals and other strategical modelling concepts to services is important to support strategic business-IT alignment as defined by Henderson and Venkatraman [HV93].

3.5.10 Project Management Alignment

Similar to the alignment of the service architecture with the overall company strategy, it must be possible to relate services to projects and programs. Projects and programs are established as concrete actions to implement a company strategy. Such actions might define new services or alter existing ones.

In general, the service owner must decide whether the service is changed in a project. Besides modelling that a service belongs to a certain project, it is also important to specify which capabilities of the service are added or changed during the project. If a capability should be added to a service in a project, a requirement is created, which also belongs to the project. After completing the project, the requirement is removed and the new capability is added to the service description.

3.5.11 Usage in Processes

Services are introduced to support and structure business activities. Business activities are usually described using business process models. Therefore, it must be also possible to use services in business process models to model that a function in a business process

is supported or even automated by a service. It can be seen in figure 3.6 that the different services on the three levels can all be used in process models. However, a service should only be used in those processes, which are on the same abstraction level as the service. For example, it would be wrong to use a service type in a BPEL process. Here, only web services are used, because BPEL processes and web services are both platform specific.

It is not a trivial task to identify services to be used in a process. One possible approach is analysing the data processed and produced by a function. A service able to support this function must also process and produce the same data objects. Another approach is relying on functional descriptions using capabilities. Here, a function specifies the capabilities to be possessed by a service, which should support the function. By analysing the service descriptions, it is possible to identify services offering the requested set of capabilities. Here, user support is needed, because this is a complex task. Especially in case of big service landscapes it cannot be done manually. Therefore, the application service discovery is contributed by this thesis as discussed in chapter 4.

Services are not only used in processes, but the SOA meta model also shows that a process is a valid realisation of a service. For example, on the PSM level a BPEL process realises a web service.

3.5.12 Service Availability

Besides those service description elements represented as concepts in the SOA meta model shown in figure 3.6, the requirement to model the spatial as well as temporal availability of a service is not represented there.

Spatial availability describes at which (physical or virtual) location a service is provided. This is especially interesting for service realisations. For example, a web service realising a service type might be available in all offices, but a certain department handling the same request might be only available in the headquarter of a company.

Temporal availability describes at which time a service is provided. For example, a web service realising a service type might be available all the day, but a certain department handling the same request might only be available on work days during working hours. A service description must be able to represent such availability information.

3.6 ARIS Modelling Method Extension

3.6.1 Overview

Subsection 3.1.3 described the development process of the ARIS extension, which is shown in figure 3.2. First, the requirements for such an extension were gathered and formalised as the SOA meta model discussed in the previous sections. This part of the development process is shown on the left side of figure 3.2. The SOA meta model (see section 3.4) and the belonging systematic (see section 3.3) can be viewed as a requirements specification defining what should be possible to model with the ARIS modelling language. This requirement specification was input for the development process of the ARIS extension, which is shown on the right side of figure 3.2. As a first step, a gap

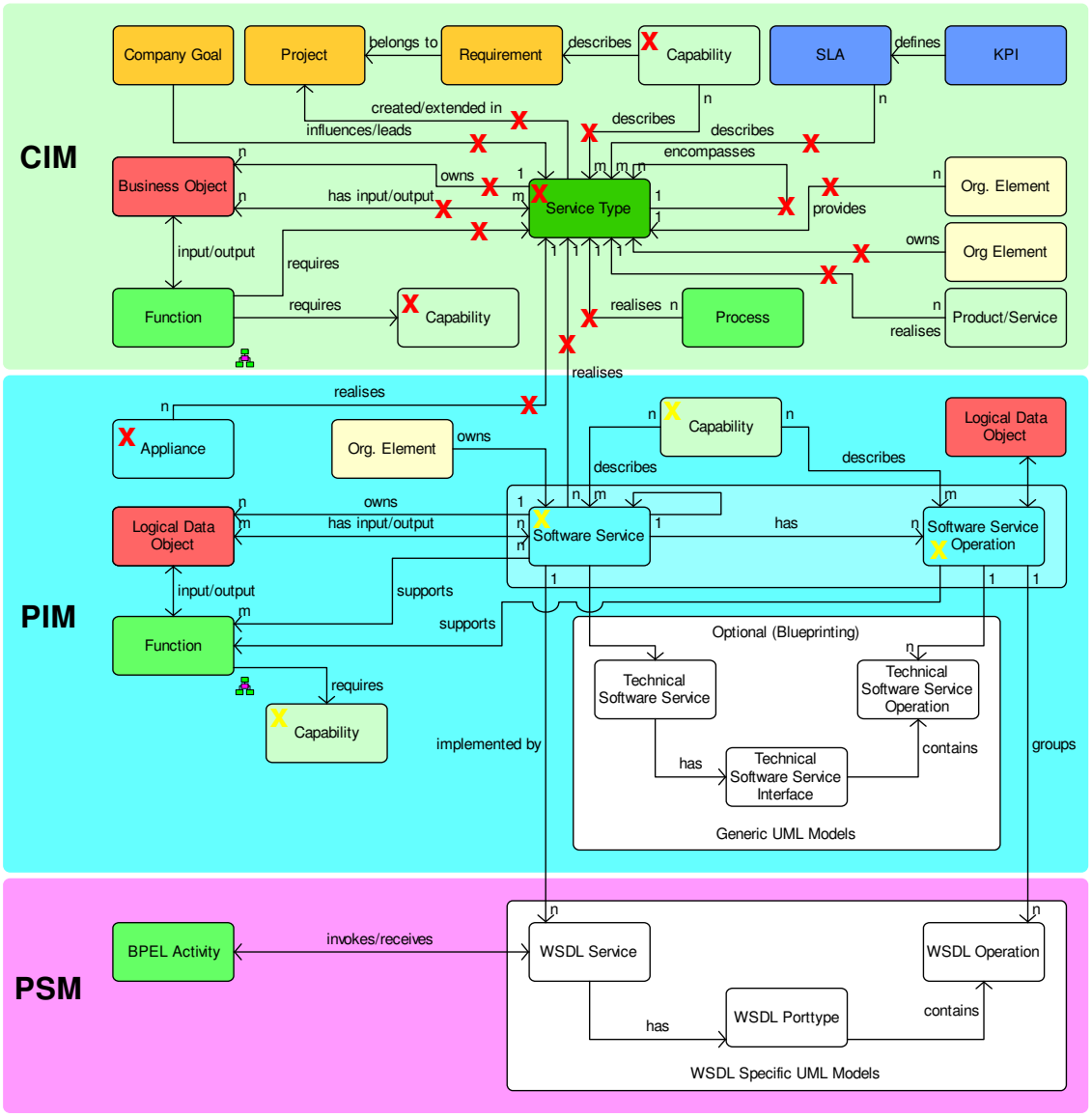

Figure 3.8: Parts of SOA meta model not supported by ARIS

analysis was performed to identify those parts of the SOA meta model, which cannot be represented with the ARIS modelling language already today. This gap analysis is presented in subsection 3.6.2. In a second step, the missing elements like new object types and model types were designed. The newly introduced or changed elements of the ARIS modelling language are described in subsection 3.6.3 and subsection 3.6.4.

3.6.2 Gap Analysis

Figure 3.8 shows, which concepts of the SOA meta model were not represented completely before developing the ARIS extension. A red cross denotes that the concept was completely missing. A yellow cross denotes that the concept was present, but used with a different or narrower meaning. For example, the ARIS modelling language allowed de-

scribing application systems independent from the technology they are implemented with using the “application system type” object type. However, there was no subtype to mark application system types representing a software service type. This also applies to operations of an application system type.

Prior extending ARIS, there was no object type representing a capability. Instead, an object type called “information system function” existed, which was used to model the capabilities of information systems. However, the “information system function” object type was mainly used to describe IT capabilities and therefore has a narrower meaning compared to a general capability able to describe any functionality and not only IT based ones.

There is no object type in the ARIS modelling language to represent an appliance. Instead, “application system type” object types are used. In a model type called “access diagram” it is possible to model that the application system type is bound to a specific kind of “hardware”.

Business objects (i. e. conceptual data objects) are represented in the ARIS modelling language by two object types, namely: “technical term” and “cluster”. Logical data objects (i. e. data objects on the platform independent level) are represented with four different object types, namely: “technical term”, “cluster”, “entity type”, and “class”. Those different object types are reused and no additional object types are added to the ARIS modelling language for data modelling.

The gap analysis revealed that a service concept independent of IT was almost completely missing in the ARIS modelling language. An object type called “functional cluster” existed, which was part of the proprietary modelling technique “IT city planning” used for IT architecture management. A functional cluster groups IT systems with similar capabilities. However, as the functional cluster is used in IT architecture management, it is not considered to be computation independent. Therefore, figure 3.8 shows that no equivalent for the service type existed in the ARIS modelling language before extending it. As the service type concept was missing, it was also not possible to represent the different relations to it as shown in figure 3.8.

The gap analysis revealed the parts of the SOA meta model not yet supported by the ARIS modelling language. This information was used to design the ARIS extension. The added or changed object and symbol types of the ARIS modelling language are described in the following subsection.

3.6.3 New or Changed Object and Symbol Types

Figure 3.9 gives an overview of the main changes done to the ARIS modelling language by the ARIS extension described in this thesis. Changed or new object/symbol types are: “capability”, “service type”, and “software service type”.

Capability

As discussed in the previous subsection, the object type “information system function” was used to describe capabilities of IT systems. This object type was renamed to “capability” lifting its meaning to a computation independent level. This is illustrated in figure 3.9. It was decided not to introduce a completely new object type, because no easy to understand

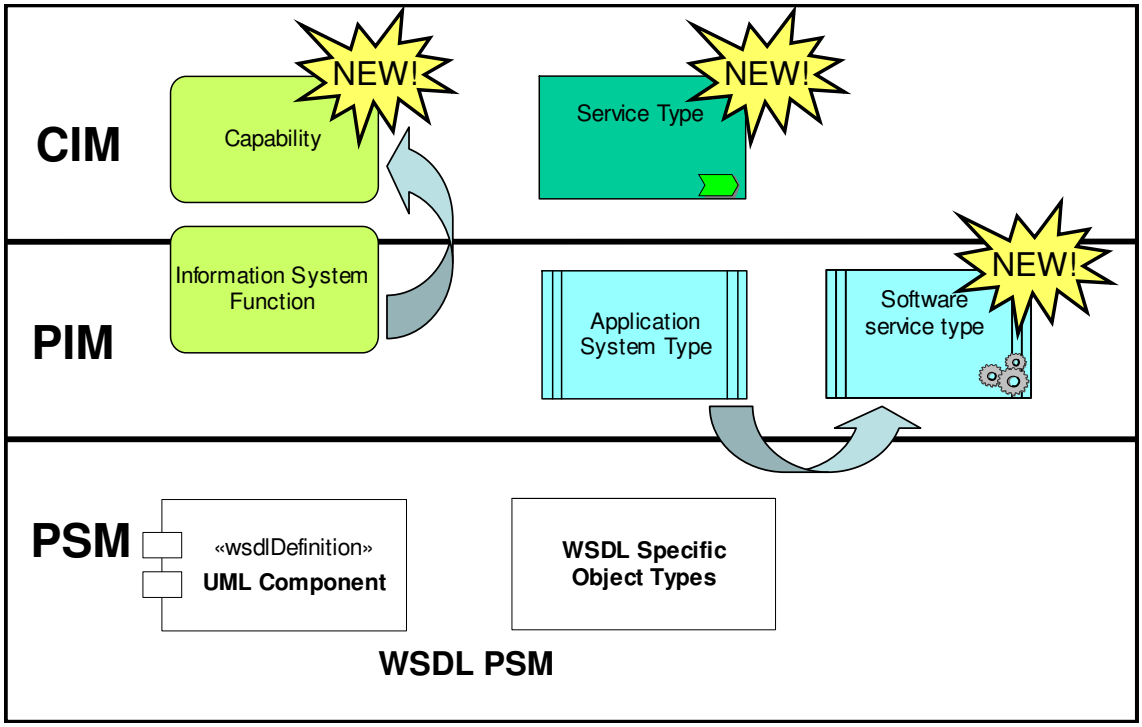

Figure 3.9: ARIS extension grouped according to MDA levels

description of the differences between capability and information system function could have been provided. In addition, it is a general aim to make the ARIS modelling language not too complex, e. g. by removing unused concepts. The “capability” object type has only one symbol type, which was also renamed to “capability”.

The “capability” object type is now used in the ARIS modelling language to describe any functional or non-functional properties of conceptual services (i. e. service types), logical services (i. e. software service types), and general IT systems (i. e. application system types). In addition, capabilities are used to describe the requirements of a function. By comparing capabilities required by a function and capabilities provided access to by services, it is possible to identify services able to support functions.

Service Type

The gap analysis revealed that there was no element in the ARIS modelling language to represent a service independent of IT. The only object type with a similar but not identical meaning was the “functional cluster” object type used to group IT systems in IT architecture management. The “functional cluster” object type was renamed to “service type” to broaden its meaning and to lift it to a computation independent level. To provide backward compatibility, the existing symbol types of the “functional cluster” object type like “zone”, “district”, and “functional block” were not removed and not renamed. However, a new symbol type “business service” was added, which should be used in case of service-oriented modelling. The symbol type “business service” is shown in figure 3.9.

The ARIS modelling language as well as the ARIS software tools support defining new symbol types for an existing object type. Therefore, it is possible that users define their

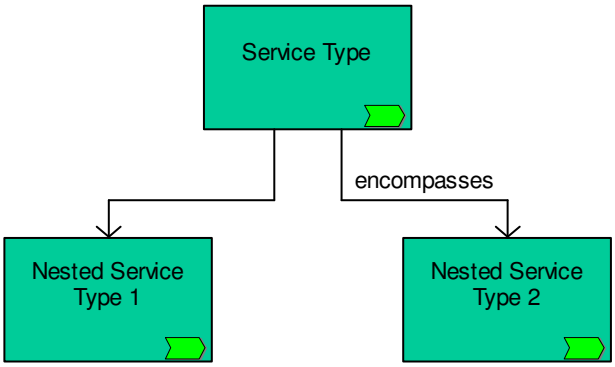

Figure 3.10: Service architecture diagram used for modelling service architecture

own categories of service types by introducing new symbol types for the “service type” object type. Supporting an arbitrary number of service categories was a requirement by the involved consultants and is discussed in subsection 3.3.3.

Software Service Type

Even prior the ARIS extension, it was possible to model application systems and their parts with the ARIS modelling language. It was requested by the involved consultants to make architectural models more expressive by marking parts of an application system, which are exposed as services. Therefore, a new symbol type called “software service type” was added to the “application system type” object type. In addition, a new symbol type “software service operation type” was added to the object type “IT function type”. Using those new symbol types, it is now possible to mark parts of an application system, which are exposed as software services. This is illustrated in figure 3.9.

It is important to note that a “software service type” is not limited to any specific technology. For example, a software service type can be implemented with web services based on the W3C stack but also by REST or any other suitable technology. Existing implementations are represented in the ARIS modelling language through UML models. In case of W3C web services, a special UML profile is provided and the content of WSDL files is mapped to UML class and component diagrams. A relation between “software service type” and implemented web service is established in an “access diagram” model type drawing an “encompasses” connection type from the “software service type” to the “UML component” object type representing the web service.

3.6.4 Changed Model Types

The previous subsection introduced the new object and symbol types, which were added to the ARIS modelling language. Besides, also model types (i. e. diagrams) are needed to do the actual modelling. This subsection describes the model types changed by the ARIS extension.

The model types do not distinguish internal and external views on the service description (see subsection 3.3.1), because there is no general applicable rule defining which information must belong to which view. Instead, it is expected that users separate the

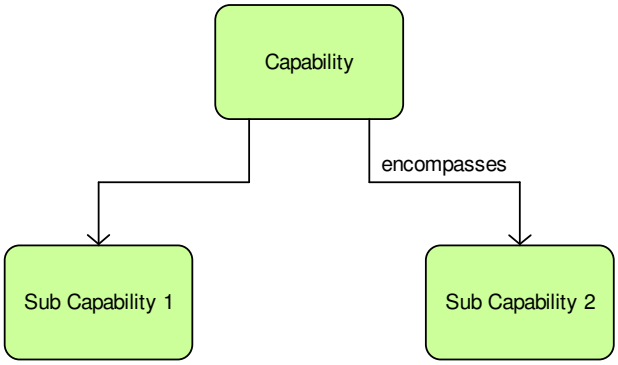

Figure 3.11: Service architecture diagram used for modelling capability architecture

views on their own and distributed the information between different model type instances of the same type.

Service Architecture Diagram

The SOA meta model shown in figure 3.6 specifies that it must be possible to build a hierarchy of service types. For this purpose, the model type “service architecture diagram” is used. As shown in figure 3.10, an “encompasses” connection type can be established between two “service type” object types. The usage of the “encompasses” connection type is not constrained by the symbol type of the object types. Therefore, it is possible to do arbitrary nesting between different service types. This is needed, because the number and naming of symbol types must be customisable so that users can introduce their own symbol types (i. e. categories) of service types.

The model type “service architecture diagram” existed before, but was named “enterprise architecture model” before. It was previously used to model a hierarchy of “functional cluster” object types. As the belonging proprietary modelling technique “IT city planning” is not used anymore, the model types used before were renamed and reused for service-oriented modelling on a computational independent level.

The model type “service architecture diagram” can also be used to describe a hierarchy of “capability” object types. This is not shown in the SOA meta model, because it was not a requirement of the involved consultants. Instead, it was already possible to model a hierarchy of “information system function” object types in the past. As the “capability” object type is replacing the “information system function” object type, this model concept must be preserved in the ARIS modelling language. Similar to the hierarchy of service types, also an “encompasses” connection type can be modelled between “capability” object types in the “service architecture diagram” model type as shown in figure 3.11.

A service architecture diagram can also be used to relate capabilities to service types. Here, the connection type “encompasses” is established pointing from a “service type” object type to a “capability” object type. However, modelling the capabilities of a service type in the service architecture diagram is not recommended, because the service architecture diagram should contain overall information on the service architecture but not specific information only influencing a single service type. For the latter case, the model type “service allocation diagram” should be used, which is introduced in the following paragraph.

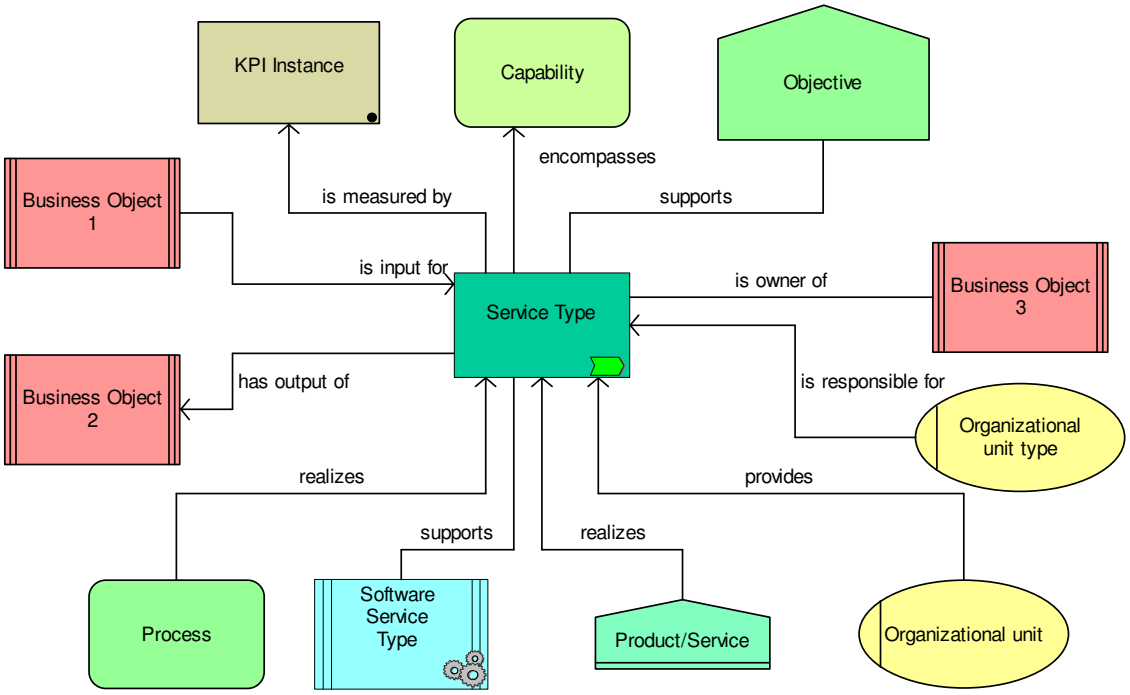

Figure 3.12: Service allocation diagram

Service Allocation Diagram

Figure 3.12 shows the model type “service allocation diagram”. A service allocation diagram is used to describe an individual service by collecting all necessary information. For example, capabilities can be related to the service type and a service owner can be defined. The “service allocation diagram” model type existed prior extending ARIS, but the model type was called “IE context model”. This model type was also used in the now obsolete modelling technique “IT city planning”.

The connection types used in the “service allocation diagram” often do not correspond to the relations defined in the SOA meta model. For example, the SOA meta model defines a connection type “describes” pointing from a capability to a service type. However, in the service allocation diagram a connection type “encompasses” pointing in the opposite direction is used. The reason for this discrepancy is that existing connection types were reused while designing the ARIS extension. Even though this is not an optimal solution from a scientific point of view, it is a pragmatic solution enforced by the ARIS method owners.

As shown in figure 3.12, a service type can own, process, and produce business objects. Such business objects are represented in the ARIS modelling language by the object types “technical term” and “cluster”. The shown connection types can be defined between technical terms and service types as well as between clusters and service types.

There are different object types possible to represent an organisational object. Figure 3.12 only shows the object type “organisational unit”. However, the connection types relating the object types “service type” and “organisational unit” shown in figure 3.12 can

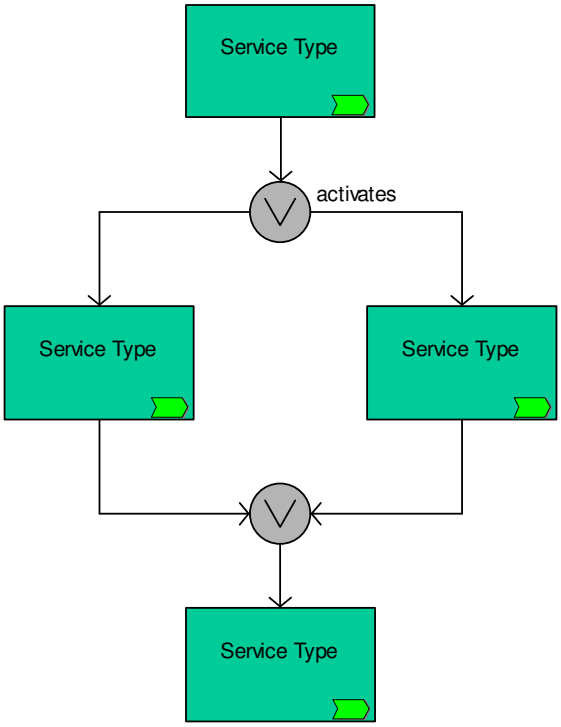

Figure 3.13: Service collaboration diagram

be also modelled between the object type “service type” and the following object types, which all represent an organisational object in the ARIS modelling language:

- organisational unit type
- organisational unit
- person
- group
- position
- location
- person type

The service allocation diagram documents static aspects of the service description. It can be used as a view on internal as well as external description elements. Dynamic descriptions are modelled in the service collaboration diagram, which is discussed in the following subsection.

Service Collaboration Diagram

Figure 3.13 shows a “service collaboration diagram” model type. It is used to model dynamic aspects of the computation independent service description. Prior the ARIS extension, a model type “IE activation model” existed. This model was renamed by the ARIS

extension. The service collaboration diagram provides various constructs to model the dynamic relationships between services and their environment. However, those modelling capabilities already existed before and were not changed by the ARIS extension. The model type is still affected by the ARIS extension, because the “functional cluster” object type was renamed to “service type” and a new symbol type “business service” was introduced.

Service Support Matrix
A service support matrix is used to model the spatial availability of a service. However, the service support matrix is not an own model type in the ARIS modelling language but instead a specific feature of the ARIS software tools. The ARIS software tools allow creating a matrix by assigning different object types as column and row headers. At the end of the chapter in section 3.7, an example of such a matrix is provided in figure 3.16. It can be seen that the service type defines the column header and the different locations define the row headers. The cells contain all those realisations of the service type, which are available at the location specified in the row header.

Time Based Modelling
Similar to the service support matrix, time based modelling is a specific feature of the ARIS software tools, but it is not an own model type in the ARIS modelling language. Time based modelling looks similar to a service support matrix. The column header is defined by the service type and each row represents a point in time or a time range. Each cell contains all realisations of the service type, which are available at the time specified in the row header.

Event-driven Process Chain (EPC)
Besides services using the different model types described in the previous paragraphs, services are also used in business process modelling. Therefore, it was necessary to extend the EPC notation. Figure 3.14 shows the possible connection types. It can be seen that “function” object types can be related to “service type” object types using the “supports” connection type. This modelling construct describes that the execution of the function is supported by the service type. An employee executing the business process can look up the description of the service type to identify a service realisation or service provider.

Another use case of service-oriented business process management aims at automating all functions of a business process so that the business process itself can be executed on a process engine. Therefore, it is possible to relate “software service type” object types and their operations (i. e. “software service operation type” object types) to functions in the business process using the “supports” connection type. Here, it is important to note that the software service types represent services implemented through technology, but not specifying which technology is used. This ensures that the business process models do not contain any platform specific details. This is important, because if technology changes, the business process models can be reused without changing them.

The relations shown in the EPC model in figure 3.14 can also be modelled in a “function allocation diagram” model type. A function allocation diagram is assigned to a function in a business process model to move information in a separated view. This reduces the overall

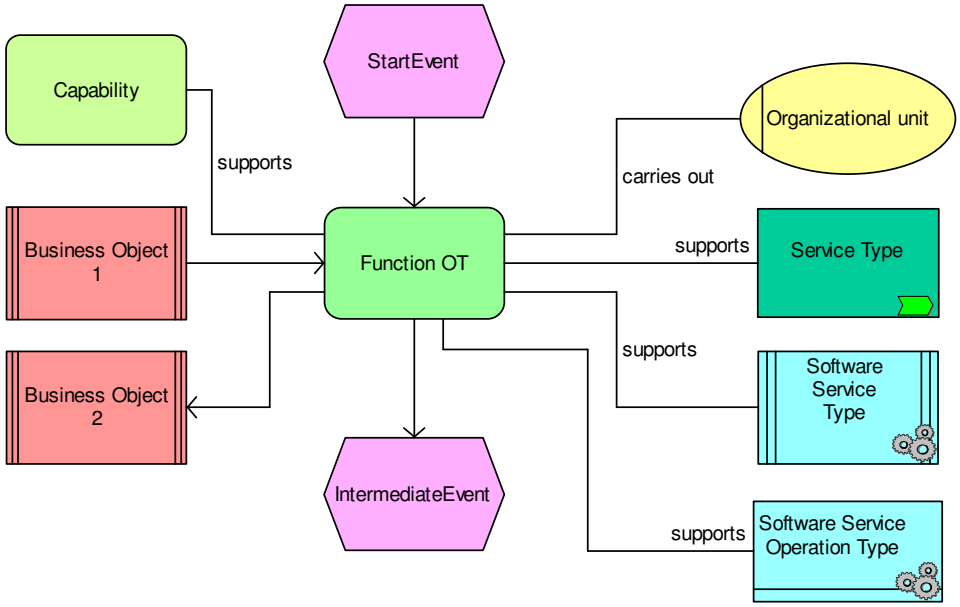

Figure 3.14: Extended EPC model

complexity of the business process model, because not all information is made available in one diagram. The “function allocation diagram” model type cannot be only assigned to a function of an EPC model, but to any “function” object type. This object type is also used in BPMN models. Therefore, the ARIS extension also supports service-oriented business process management using BPMN for business process modelling.

3.7 Example

The ARIS extension described in the previous sections enables several use cases like planning and describing a service architecture, describing services, linking service change requests to projects, orchestrating services to implement business processes, discovering services, and analysing the service repository. The following example focuses mainly on describing a service, discovering a service, and creating a service orchestration. Where appropriate, pointers to the applications developed are provided.

In a fictitious company, employees are allowed to rent a car for a business trip if no company car is available and if the destination cannot be reached by public transport. There are several alternatives available to support an employee in renting a car. For example, an employee can ask a secretary, contact the official partner travel agency of the company, use a special web service provided by the main car rental partner of the company or use a booking terminal available at the airport near the company’s headquarter. Even though the alternatives have different properties, they all provide the same type of service to the employee – renting a car. Therefore, a service type “Car Rental Service” is defined and the existing service providers are linked to it as shown in figure 3.15. The service type is described using capabilities and the most important business objects from the company’s information architecture are linked to the service type, too. The company’s “Travel & Fleet

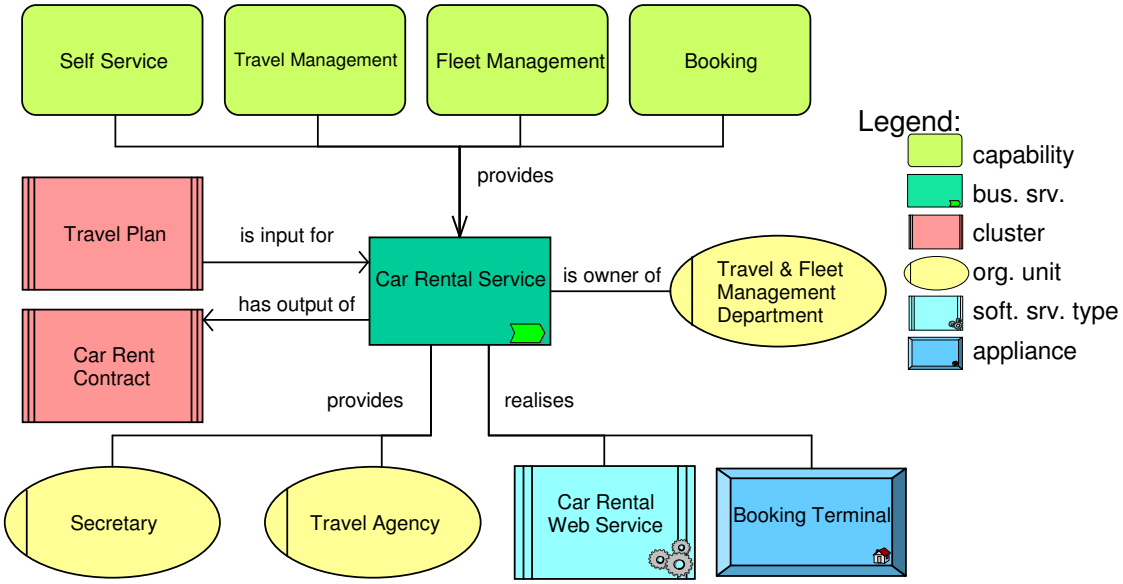

Figure 3.15: Example service allocation diagram

Management Department" owns the service and is responsible for maintaining and developing it. All this information is documented in a service allocation diagram as shown in figure 3.15.

The service support matrix shown in figure 3.16 models the spatial availability of the service providers. The column headers contain the service types and the row headers the locations. In the example, only the availability of the car rental service is modelled. A service realisation or provider is assigned to a cell if the service provider offers the specified service type at the given location. Figure 3.16 shows that the "Car Rental Service" is available at the locations "Paris" and "London". It also shows that in "London" the service is provided by a "Secretary" and a web service. In addition to that, a "Booking Terminal" and a "Travel Agency" is available in "Paris". The "Booking Terminal" denotes an appliance, which consists of hardware and software.

Service support matrix as well as service allocation diagram can be used for analysis. For example, redundancies in the service landscape can be identified or a different provider can be looked up in case a contracted service provider becomes unavailable. The service support matrix can be used to evaluate different scenarios like introducing a new realisation at a location or de-supporting a realisation in the future. In chapter 4, an application is introduced, which provides automated support for service discovery based on the content of the service allocation diagram.

Services can be used in business process models to document, which service supports a function. Figure 3.17 shows a part of a business process model using the EPC notation. The function "Rent a Car" consumes the business object "Travel Plan" and produces the business object "Car Rent Contract". It is documented that a service is needed providing the capabilities "Self Service" and "Travel Management". Based on this information and the information given in the service allocation diagram, it is possible to discover a service as shown in chapter 4. The service description becomes machine processable if the

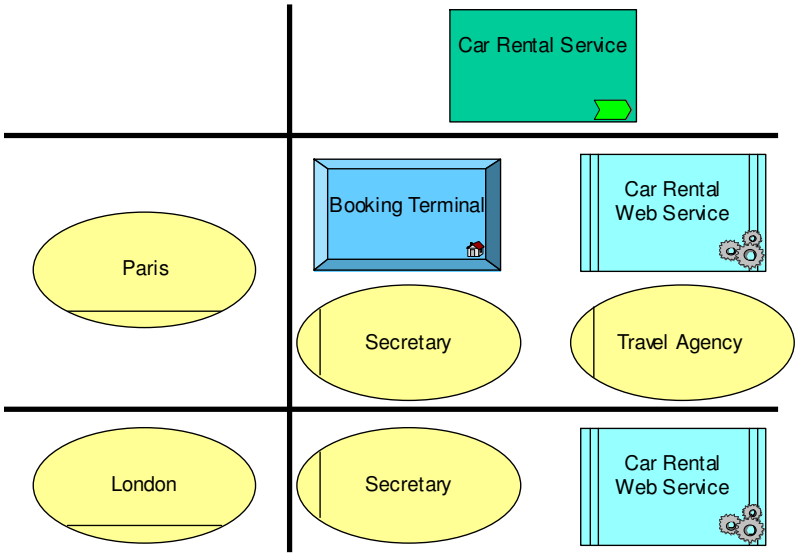

Figure 3.16: Example service support matrix (spatial availability)

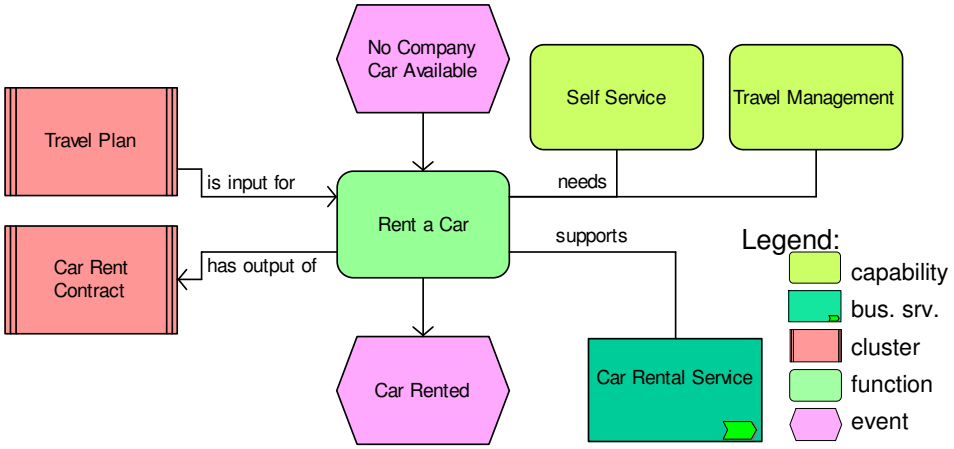

Figure 3.17: Snippet of example EPC process

service description is further formalised. An application using such a formalised service description is described in chapter 7.

The business process model shown in figure 3.17 is used by an end user to decide, which service to use. If the business process is not automated but meant as a working instruction, a user can use the information given in the service allocation diagram to evaluate the different service providers. If all functions are supported by a software service implemented as a web service, the business process can be transformed into an executable model like BPEL. The automated EPC to BPEL transformation application is described in chapter 5. Another analysis is looking up all usages of a service type. This helps to estimate the impact of a proposed change and who is affected by a breakdown of a service realisation.

Chapter

4 Service Discovery

This chapter describes the first out of three applications based on the ARIS extension (see figure 4.1). Service discovery aims at providing support in identifying services to be used for business process modelling. A motivation for developing this application is given in section 4.1. Section 4.2 describes the overall approach to service discovery and relates the approach to the state of the art presentation in section 2.5. The following sections present the algorithms used in detail. Section 4.3 describes the structural matching algorithm and section 4.4 introduces the semantic matching algorithm. Service assessment is supported by a graphical user interface, which is presented in section 4.5. To illustrate the different algorithms, an example is provided in section 4.6.

4.1 Motivation

As it was shown in the state of the art presentation about business process automation in subsection 2.1.5, it is a common approach to use an orchestration of web services to

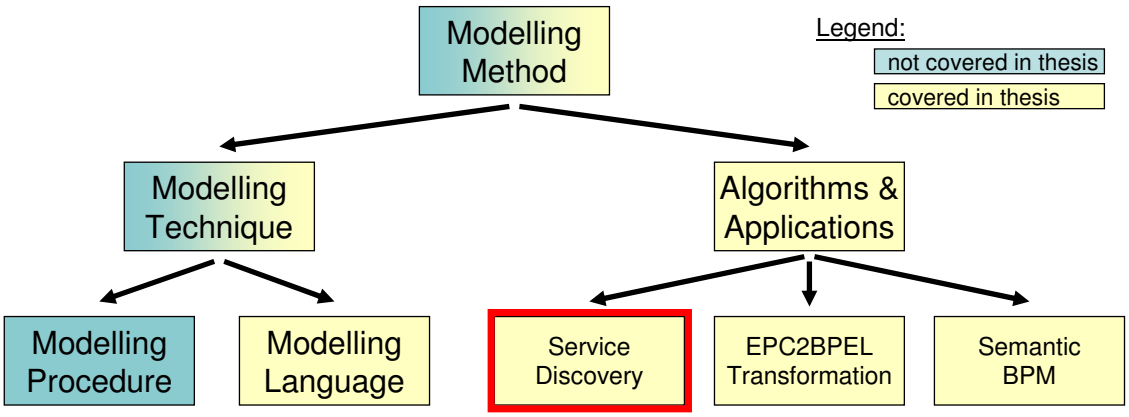

Figure 4.1: Contribution service discovery application

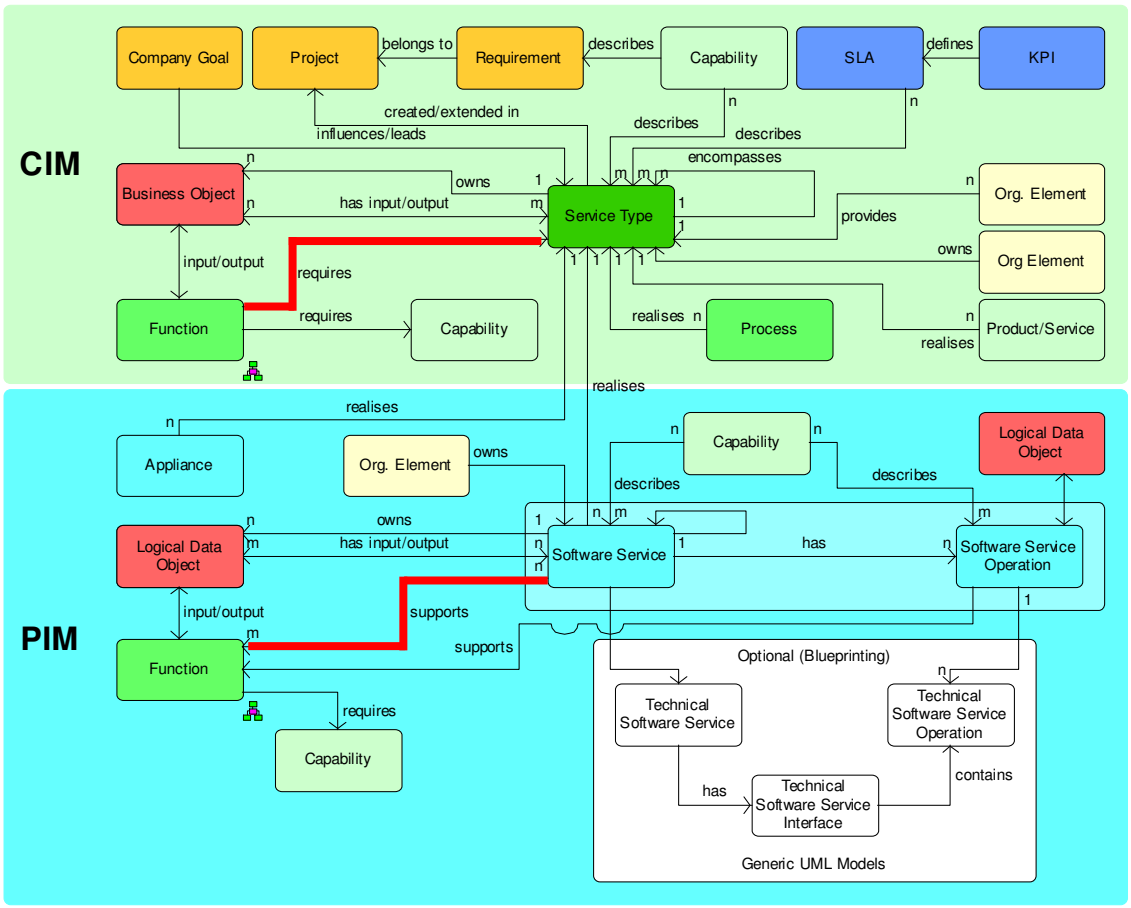

Figure 4.2: Relations established by service discovery application

automate a business process. A web service orchestration language like BPEL is often used for this purpose. However, languages like BPEL and XPDL are technical artefacts, which are hardly usable by business experts. Therefore, it is expected that such executable orchestrations are derived through model transformation from a business process model.

To enable such a transformation, business experts must be able to select appropriate web services and add them to their business process model. However, adding a WSDL web service to a business process model is not a very good practice, because business process models should be described on a platform independent level so that technology changes do not affect it. Therefore, an additional abstraction is needed. The ARIS modelling language uses the concept of “software service types” to abstract from any concrete implementation. Software service types are added to the business process model. However, the overall problem remains. Business experts have to select appropriate software service types.

Selecting a software service type able to automate a function is not a trivial task to do. In a real-world environment, there might be several hundred software services, many of them with similar functionality. Therefore, the user must be supported in evaluating service offerings. Such a support functionality is introduced by the application described in this

chapter. The service discovery application helps to connect functions to software service types or service types as visualised in figure 4.2.

The application provided aims at service discovery during design-time. The business expert is supported in selecting an appropriate service while modelling a business process. Service selection can also be done during runtime as shown in the state of the art presentation on service discovery in section 2.5. An application supporting service discovery during runtime is presented in chapter 7. However, the case study presented in chapter 8 shows that currently business experts are not able to clearly motivate service discovery during runtime.

Even though the service discovery application supports the discovery of service types as well as software service types, this chapter mainly focuses on the discovery of software service types, because that is the more complex problem. Software service types are internally mapped to implementations like WSDL web services and a discovery algorithm must use the information provided by the WSDL description as well. This is not the case for service types, because they are only described through capabilities and the business objects they consume and produce. The following section gives more details on the overall approach taken.

4.2 Overall Approach to Service Discovery

The state of the art presentation in section 2.5 provided the following criteria to classify work on service discovery:

- There are different approaches for service matching like structural, lexical, and semantic matching (see subsection 2.5.2).
- There are different strategies to combine the results of matching (see subsection 2.5.3).
- Service discovery can be done during design-time and runtime (see subsection 2.5.3).
- Service discovery applications differ in the way they implement the general service discovery process consisting of the main phases matching, assessment, and selection (see subsection 2.5.4).

The previous section already discussed that the service discovery application focuses on service discovery during design-time. It is the aim of the application to support business experts in selecting appropriate services while they design business process models.

The service discovery application uses a structural and a lightweight semantic matching algorithm. A lexical matching algorithm is not used, even though the user can refine the matching results during the assessment phase using ordinary string search. This is discussed in detail in section 4.5.

The structural matching algorithm discovers matching software service types by analysing the data processed in the business process and the message formats supported

by the web services implementing the software service types. The structural matching algorithm is described in section 4.3.

The service discovery application also uses a lightweight semantic matching algorithm. It analyses the capabilities used in the service description and compares the available capabilities to the capabilities needed to automate the selected function. This algorithm is called only a lightweight semantic matching algorithm, because it does not use semantic web technologies like reasoners and ontologies, but it still analyses the semantics of a service. The algorithm is described in section 4.4.

The results of the structural and semantic matching algorithm are united following a mixed strategy. A service is considered to fulfil the service discovery request if it is either discovered by structural or semantic matching or by both algorithms. Duplicates are removed from the final result set before presenting it to the user for assessment.

The service discovery application is structured according to the three phases of the service discovery process. The user initiates the service discovery by selecting the function to be related to a service (i. e. either to a service type or to a software service type). During the first phase, the context of the selected function is analysed and the service request is derived. All services available in the ARIS software tools are compared against the service discovery request using the structural and lightweight semantic matching algorithms. In the second phase, the result set is presented to the user for assessment. In the third phase, the user selects the service to use and the service is added to the business process model relating it to the selected function. The overall behaviour of the service discovery application does not differ for service types and software service types. The first phase of the service discovery process is automated, the other two phases are manual steps done by the user.

The following subsections describe the different parts of the application in detail. During the description, no specific user roles are mentioned to make the description not too complicated. An example is given in section 4.6. This example provides a walk-through also describing the involved user roles.

4.3 Structural Matching Algorithm

The ARIS software tools support the WSDL standard version 1.1 [WSD01]. The content of the WSDL file is represented in the ARIS modelling language using UML diagrams. For example, a WSDL porttype is mapped to an “UML interface” and the belonging operations to “UML operations”. XSD files can be imported in the same way. The content of those files is also represented using UML models. Those UML models are not meant to be used by business experts. Therefore, an abstraction is introduced by using software service types and software service operation types. Also, a business expert is not expected to deal with XSD data definitions. These data definitions are mapped to a logical data model, which consists of the object types “technical term”, “cluster”, “entity type” or “class”. The mapping between the different abstraction levels is established by relating the UML objects to the more abstract object types in different model types. The structural matching algorithm uses this mapping information to discover services by navigating from the logical data constructs used in the business process model to the technical constructs provided by

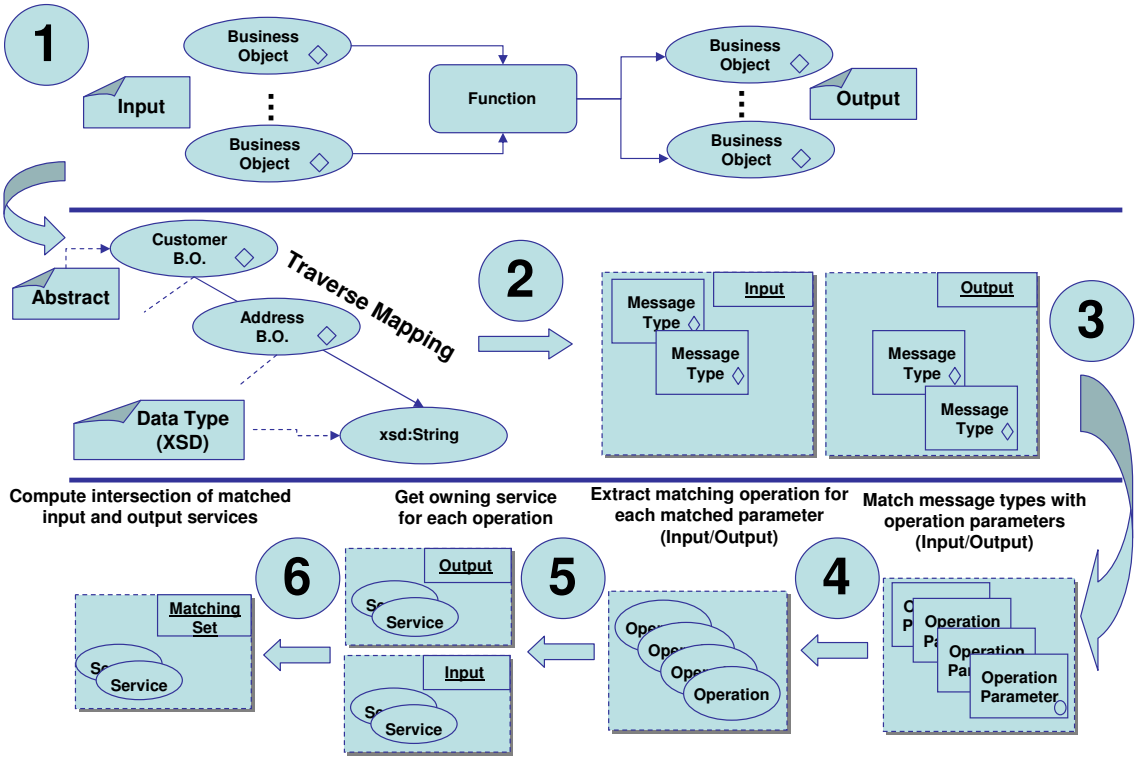

Figure 4.3: Structural service matching algorithm

the web service. The structural matching algorithm does not work if the mapping is not provided by the user. Establishing this mapping is the task of creating an information architecture[1].

The structural matching algorithm for discovering software service types works as follows: First, it extracts all business objects (i. e. either technical terms, clusters, entity types or classes) modelled as input and output of the function (see step 1 in figure 4.3). After this first step, there are two sets, one containing all business objects required as input and the other one containing all business objects required as output. For each of those two sets, the algorithm navigates through the data mapping to identify all message types. Implementing this navigation is not trivial, because the ARIS modelling language allows arbitrary abstraction levels between business object and message type including cyclic dependencies. For example, the business object customer used in a business process is further decomposed into an address. This address can be represented using different message types. The algorithm has to identify all message types mapped to the business object. This is illustrated between step 1 and 2 in figure 4.3. After this step, there are two sets of message types, one containing message types required as input and one containing message types required as output. In step 3, the algorithm checks to which operation parameters those message types belong and if the parameters have the same direction as the message types (i. e. input or output). Extracting this information is possible, because the complete WSDL content is available in the ARIS software tools and mapped to UML

[1] Other terms used for such an initiative are shared business vocabulary, business taxonomy or data architecture.

models. If the message type is the type of a parameter with the correct direction, the algorithm extracts the belonging operation as shown in step 4. Afterwards, the web service owning the operation is extracted in step 5. At the end, there are two sets of web services: one set supporting all input business objects and the other set containing all web services supporting the output business objects. In the final step 6, both result sets are intersected. The final result set of the structural matching algorithm contains only those web services, which are part of both preliminary result sets and are therefore able to support all input as well as all output business objects. The ARIS modelling language requires that each web service is mapped to exactly one software service type. Therefore, in a final step (not shown in the figure), the software service types implemented by the discovered web services are extracted.

The algorithm does not check that a web service has at least one operation able to support all business objects in the parameter list. While automating business processes, this is considered to be no problem, because during transformation of a business process into an executable process model (e. g. BPEL), a function can be split up into several technical steps. Also, adding another operation to a web service able to handle all business objects in one request is often possible if the web service is owned by the company.

It can be seen that the structural matching algorithm is quite complex in case software service types must be discovered. The complexity is reduced while discovering service types. Service types also use business objects in their description. Therefore, no mapping between different abstraction levels is needed. In addition, service types do not have operations with input and output parameters.

The biggest disadvantage of the structural matching algorithm is the effort required for mapping business objects to technical data structures. Often, users of the ARIS modelling method already invested in creating a comprehensive information architecture consisting of the most important business objects. However, mapping those business objects to technical data structures requires effort. Each customer must decide if this investment can be justified. As an alternative, a more lightweight algorithm is provided, requiring less effort. The algorithm is described in the following section.

4.4 Semantic Matching Algorithm

Not all users of the ARIS modelling method are interested in creating and managing an information architecture. Therefore, a second more lightweight approach for service discovery is provided. The algorithm relies on capabilities. Capabilities are used to describe the service (i. e. service type and software service type). Capabilities are also used to describe the functionality expected by a function (see subsection 3.5.2). There are no differences in the algorithm for discovering service types and software service types.

The semantic matching algorithm first extracts all capabilities assigned to the function in the business process model. The extracted capabilities describe the service request. Afterwards, the algorithm extracts the capabilities assigned to each service and compares this list to the service request. All services providing access to all capabilities in the service request are added to the result set. The semantic matching algorithm does not support navigating a hierarchy of capabilities. For example, capabilities can be refined into a

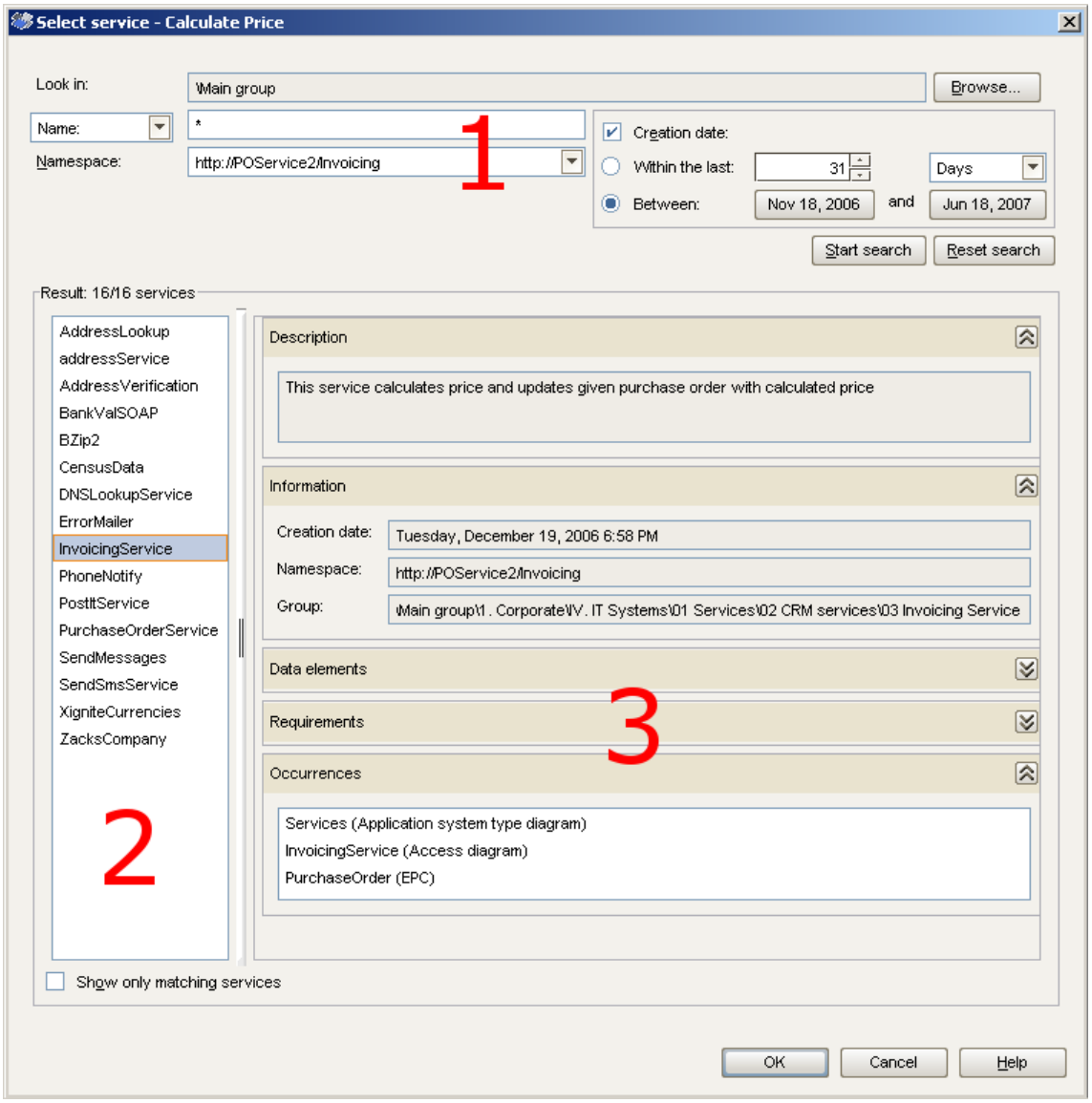

Figure 4.4: Graphical user interface for service assessment

capability architecture using the model type “service architecture diagram” (see subsection 3.6.4). This hierarchy is not considered in the semantic matching algorithm. A service is only discovered if it provides access to exactly those capabilities, which are specified in the service discovery request.

4.5 Service Assessment and Refinement

The final result set consists of all services discovered either by the structural matching algorithm or by the semantic matching algorithm. The matching results are presented to the user in a graphical user interface. A screenshot can be seen in figure 4.4. The screen design consists of three parts, which are marked in the screenshot with the numbers 1–3.

During the assessment phase, the user further refines the result set. For example,

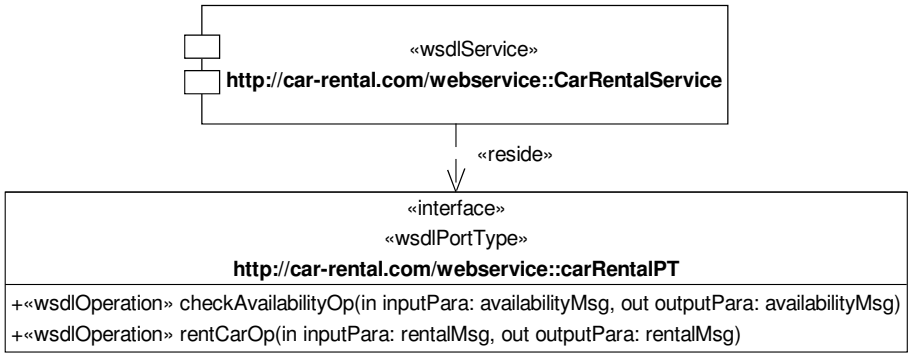

Figure 4.5: WSDL web service as UML component diagram

the user can search the descriptions and names of the services with a string search or he can filter the list of services according to their namespace (only possible for software service types). He can also filter the list according to the date the services were created or imported into the ARIS software tools. Those refinement settings are done in part 1 of the screen design.

The current result set can be seen in part 2 of the screen design. This part also allows switching between a list of all services and the list with matching services. This is useful in case the matching algorithms did not return a satisfying discovery result.

The user has to assess if a service fulfils the service request. This assessment cannot be done based on the name of a service. Therefore, additional information is shown in part 3 of the screen design for the currently selected service. For example, all business objects supported by the service are shown as well as the textual description. It is also possible to see in which other contexts the service is used.

Finally, the user selects a service and confirms this selection. The dialog closes and the service is automatically related to the function in the business process model. At this point, the application described in this chapter is complete.

4.6 Example

This section provides an example to better illustrate the service discovery application. The example mentions two different roles – a business expert and an IT architect. The business expert has no IT background, but instead experience in business process modelling. The IT architect has SOA know-how and is able to use typical SOA middleware products and standards. In reality, there are usually more roles involved. The example is aligned with the example used in the previous chapter, which illustrated the use of the ARIS extension (see section 3.7).

A fictitious company defines an internal business process for organising business trips. If such a business trip has to be done by car, the employee has to use a company car if available. Only if no company car is available, the employee is allowed to rent a car from a defined car rental company.

The car rental company provides a web service for this purpose. In order to be able to use this web service in business process modelling, the web service is imported in the

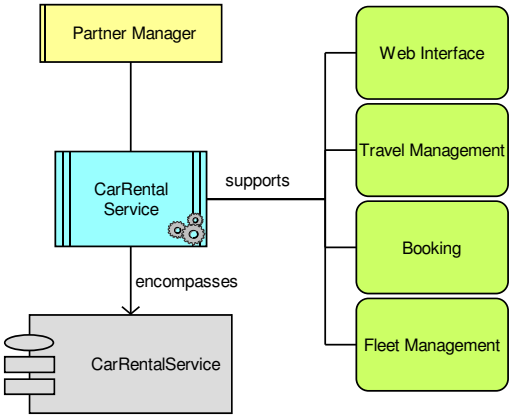

Figure 4.6: Description of software service type (access diagram)

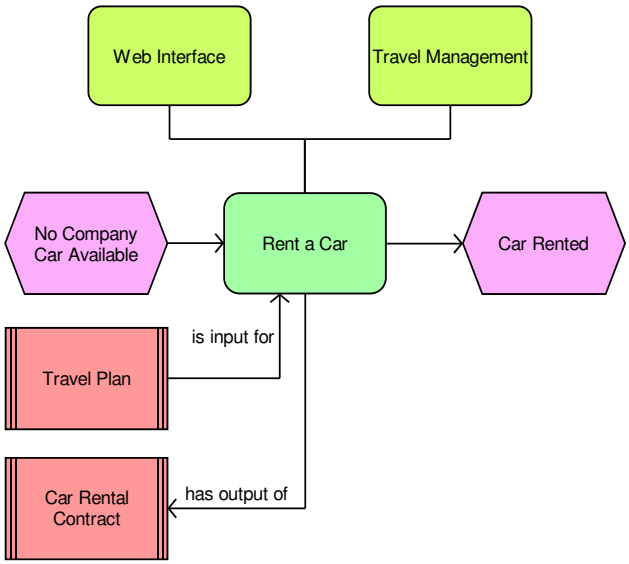

Figure 4.7: Business process without software service type

ARIS software tools. An IT architect imports the web service. The content of the WSDL file is visualised as an UML component diagram as shown in figure 4.5.

Besides using this technical information, the IT architect abstracts from the web service by introducing a software service type for it. The software service type is described by the IT architect using capabilities. The IT architect chooses between existing capabilities. The set of capabilities was defined in the past during an enterprise architecture modelling effort. In addition, the IT architect might add who is responsible for this software service type. Figure 4.6 shows the description of the software service type. It has four capabilities and one owner.

If the company has an information architecture, the IT architect maps the message types used by the web service to the belonging business objects. This can be a complex task and he might have to consult the business expert to identify the correct business objects. The mapping is done in different diagrams, which are not shown.

A business expert models the business process described at the beginning of this sec-

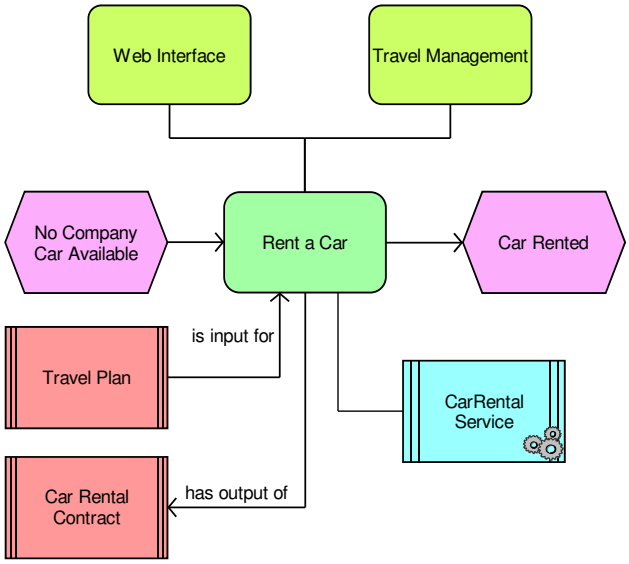

Figure 4.8: Business process with software service type

tion. Figure 4.7 shows a small part of the business process using the EPC notation. The business expert creates a function and connects it with the input and output business objects. In addition, the business expert specifies expected functionality by relating the function to capabilities. In reality, companies have either an information architecture or a capability architecture, but probably not both.

The business expert wants to automate the function using a software service type. He selects the function and starts the service discovery application. The service discovery application evaluates the content of the business process by extracting all input and output business objects and extracting the capabilities connected to the function. This information is the input for the semantic and structural matching algorithms as described in the previous sections. The results are shown to the business expert in the graphical user interface discussed in section 4.5. The business expert selects a software service type after assessing the different choices. The software service type is added to the business process as shown in figure 4.8. The symbol of the function is changed as well to visualise that this function is now automated by a software service type.

The resulting business process cannot be executed directly, because different technical information is still missing. An IT analyst uses the EPC to BPEL transformation application described in the next chapter to generate a corresponding BPEL model. This BPEL model must be further refined, for example selecting correct web service operations or defining technical exception handling.

The example given in this section shows that a business expert is able to automate business processes by discovering matching software service types. In order to select a software service type, the business expert does not need IT knowledge. In addition, an IT expert implementing a business process receives a detailed specification for his work.

Chapter

5 Automated EPC to BPEL Transformation

In order to bridge the business-IT divide, automated model transformations can be used. In case of business process automation, business process models are transformed into executable models like EPC to BPEL. This thesis contributes such a transformation. After introducing the transformation in section 5.1, industrial requirements are extracted and condensed in a general business to IT transformation framework (see section 5.2). Afterwards, the chapter presents the different parts of the transformation application like the control flow transformation in section 5.3, the data transformation in section 5.4, the service transformation in section 5.5, and the merge support in section 5.6. The EPC to BPEL transformation is one of three applications developed within this thesis (see figure 5.1).

5.1 Introduction

Business process automation, as described in subsection 2.1.5, aims as implementing business processes through IT. In general, business process automation does not spec-

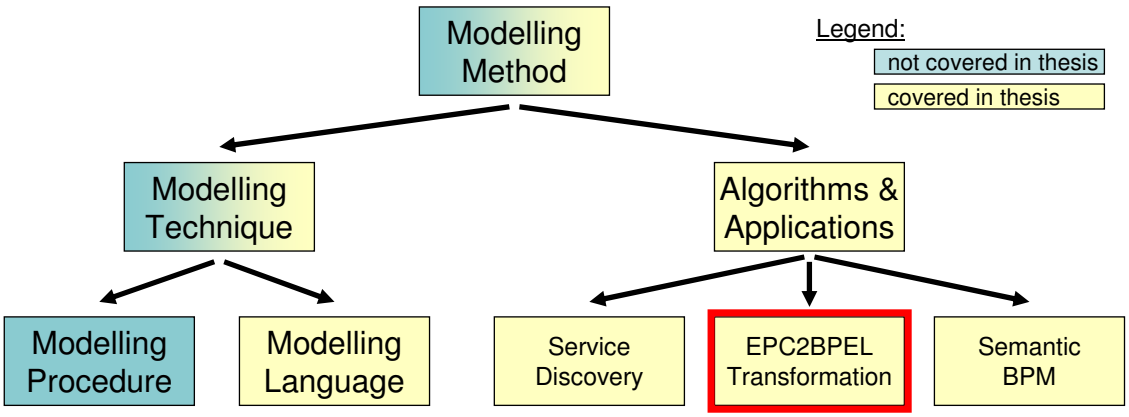

Figure 5.1: Contribution EPC to BPEL process transformation application

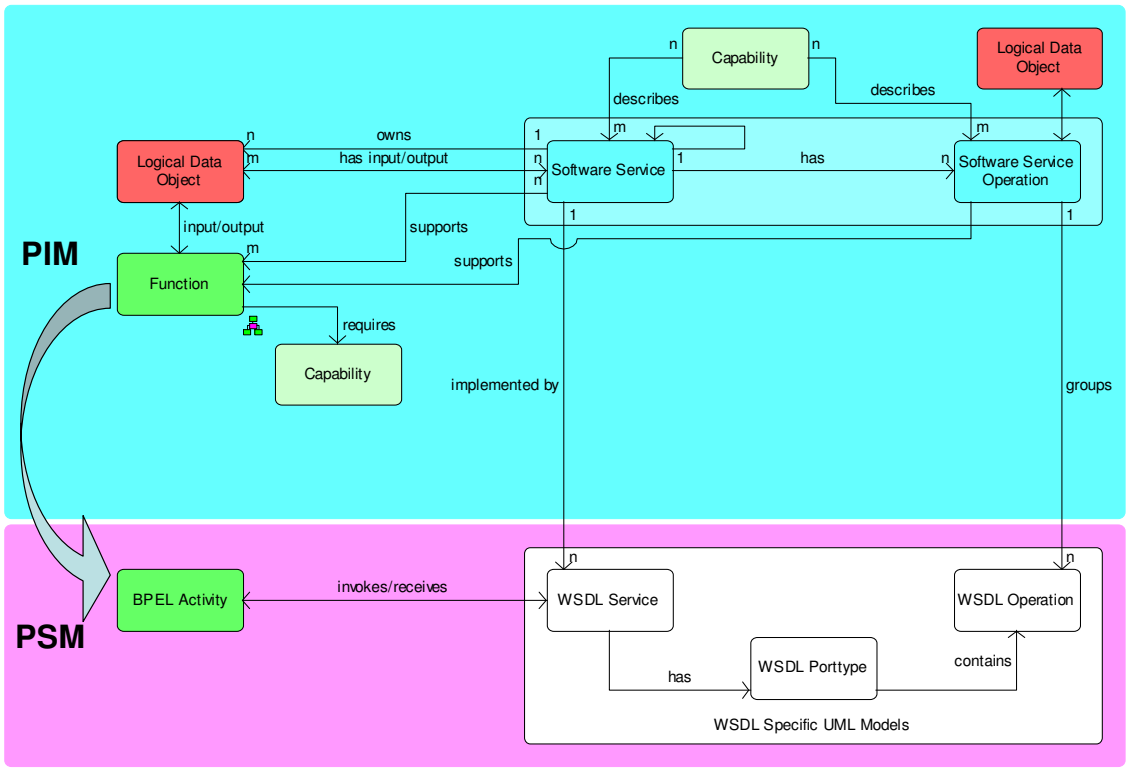

Figure 5.2: ARIS extension and EPC to BPEL transformation application

ify whether the implementation must be programmed manually or if it is generated, e. g. through model transformations. However, section 2.4 discusses various works by other researchers, who are aiming at automated generation of a process implementation based on BPEL. Typical business process modelling languages supported by such transformations are BPMN, EPC, and UML activity diagrams.

This chapter describes the development of the EPC to BPEL transformation application. As shown in figure 5.2, the transformation consumes platform independent business process models and generates platform specific executable models. The transformation application supports the EPC notation for business process modelling and BPEL as executable orchestration language.

The state of the art presentation in section 2.4 shows that most researchers focus on horizontal control flow transformations, but only few of them consider additional perspectives like data transformation. This is surprising, because BPEL does not only have a control flow perspective, but also a data and functional perspective. Currently, there is no single transformation available, which also takes into account the data perspective.

The research work presented in [Sch07, KUL06, ZM05] shows at least rudimental transformations for the functional perspective. However, those transformations do not correctly separate the different artefacts between the different modelling levels. For example, [Sch07, KUL06, ZM05] suggest to directly add web services to the business process model, so that the corresponding partnerlinks can be generated during the transformation. However, adding a platform specific concept like web service descriptions (i. e. WSDL) to a

business process model (e. g. BPMN, EPC) also means that the business process model is bound to a specific technology and therefore it is not platform independent anymore. Thus, it is impossible to transform the business process model to any other platform if the implementation technology changes.

In the following section, requirements for such a transformation are extracted, mainly based on industrial experience and industrial case studies. The requirements are condensed as a general business to IT transformation framework. This framework is based on an axiom, which is introduced in the following.

5.2 Business to IT Transformation Framework

5.2.1 Axiom

Most of the approaches discussed in section 2.4 follow a horizontal transformation strategy. Source models are translated into another format without further refinement. A horizontal transformation starting with an abstract business process model results in an abstract orchestration model requiring significant refinement efforts to make it executable. An alternative is to start from a business process model augmented with technical details. Using business process models augmented with technical details is addressed by several research approaches and can be found in commercial products, e. g. Intalio BPMS[1] and Lombardi Blueprint[2]. The horizontal strategy is adequate for providing a modelling language that is to some extend independent of the orchestration language, because minor changes in the execution platform may be reflected in the transformation definition. However, this approach forces business experts to think in terms of executable business processes and to get aware of the underlying technology and its constraints.

According to experience gathered by interviewing consultants of IDS Scheer but also by studying industrial case studies (e. g. [All08, MGH07]), a rather tight relationship between business process model and its implementation causes problems in heterogeneous IT landscapes. Such landscapes usually consist of different middleware products provided by vendors like SAP, Oracle, IBM, and Microsoft. For example, in [MGH07] a business process is executed on IBM and SAP middleware simultaneously. Therefore, a clear separation between business process and executable model is needed.

Taking these arguments into account, the following axiom is postulated, which forms the basic assumption for the business to IT transformation framework: “Business process models (e. g. BPMN, EPC) must be platform independent and a platform specific IT implementation (e. g. BPEL) must be derived through a vertical transformation strategy”. This axiom has some consequences, which are discussed in the following subsection.

5.2.2 Axiom’s Consequences and Requirements from the Field

If the axiom described in the previous subsection is accepted as basic assumption for business to IT transformations, a business process model shall not contain any platform

[1] http://www.intalio.com/
[2] http://www.lombardisoftware.com/

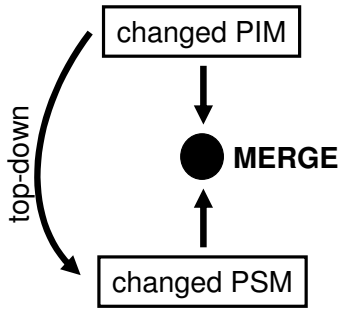

Figure 5.3: Concurrent changes resolved by means of merge functionality

specific details. This means, WSDL or SOAP descriptions shall not be added to the business process model, because even though WSDL and SOAP are standards supported by different middleware products, they are still only one way to implement a SOA (see also subsection 2.2.2).

As platform specific details should not be added to the business process model, the executable model should be separated from it. This means, there is only one business process model, but there might be several implementations ($1 : n$). Each implementation can follow another paradigm. For example, a business process model can be implemented as a web service orchestration, but also as an ordinary software application. This clear separation allows having a customised view on the business process – one view for business experts and one view for implementation experts like software engineers.

Besides those consequences, there are also additional requirements from the field. According to the experience gathered, it is possible and allowed to restrict the expressiveness of the source model language to a subset, which can be unambiguously transformed to an executable model. Obviously, this restriction should at least allow well-structured input models including cycles. Instead of supporting exotic control flow transformations, the business expert should be guided according to clear modelling guidelines and supported by validation functionality provided by the modelling tool.

In a vertical transformation strategy, it is necessary that the generated executable model is further refined. Therefore, the generated executable model should be comprehensible for human users. For example, in case of BPEL only block elements should be used ignoring BPEL's flow character. This is also demanded by [MLZ06, vdAL05].

As the generated model is usually changed manually after transformation, it must be possible to have a link between business process model and executable model so that users can navigate easily between both worlds. If source and target model are available in one tool, this is fairly easy to achieve, but if different tools are involved, some integration between the tools is needed.

Transforming a business process model into an executable one is not a task done only once [All08, see e. g.]. In contrast, business process models as well as BPEL models are changed several times, also concurrent changes might happen. To resolve such concurrent changes, a merge functionality is needed. A transformation algorithm must provide support for merging concurrent changes like automatically resolving all conflicts or presenting conflicts to the user for manual resolution. This is visualised in figure 5.3.

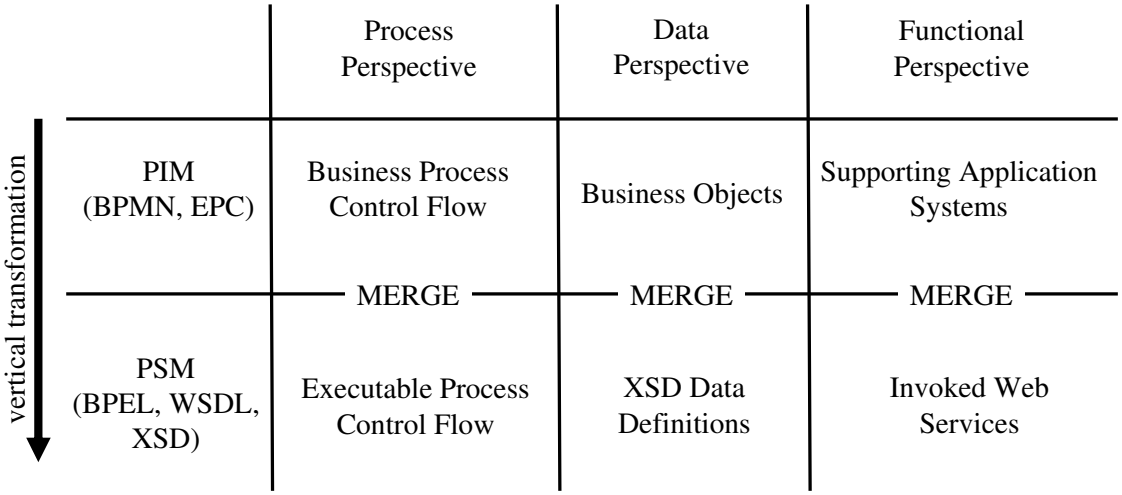

Figure 5.4: Business to IT transformation framework

Finally, it is not enough to solely focus on a transformation of the control flow. As pointed out before, a transformation must take BPEL's ability into account to represent data as well as functions (services). In fact, transformation of the control flow is not the biggest challenge from an industrial point of view [All08] and it is solved in many commercial software products already.

The next subsection develops a general framework for business to IT transformations taking into account the requirements and the axiom's consequences discussed in this section.

5.2.3 Framework

Based on the requirements stated in the previous section and the axiom's consequences, it is possible to formulate a general framework for business to IT transformations (see figure 5.4). The framework refers to different modelling perspectives – process, data, and interaction perspective – and thus reduces cognitive complexity at design-time. Furthermore, figure 5.4 exemplifies some entities on two different levels of abstraction as well as their relationships, which are operationalised by a vertical transformation. Manual refinements in iterative development cycles are explicitly addressed by perspective-specific merge functionalities.

The business to IT transformation framework does not designate a certain way of how the transformation itself is implemented. For example, the control flow transformation might be completely automated whereas the data transformation might involve manual user interaction.

A transformation application not supporting all perspectives has only limited practical applicability. Therefore, the framework shown in figure 5.4 highlights that the intended transformation does not solely consist of a control flow transformation, but that also data and interaction related aspects must be transformed. For example, a business process usually has a notion of data like business objects consumed and produced by functions. Those business objects must be present in the generated BPEL model as well, e. g. as

input and output variables. The same is true for functional aspects. A business process might specify which application system type supports which function. This information must be present in the generated BPEL model, e. g. as partnerlinks invoked by a BPEL activity.

The framework does not propose a particular way to implement the merge of different model versions. At least, it must support model matching, change detection, change visualisation, and merge capabilities. Implementation details, e. g. the type of merge (2-way, 3-way, 4-way), type of match (identity based, heuristic) or time of synchronisation, depend on the requirements of the specific usage scenario. Also, a transformation might use different approaches for perspective specific merge functionality. There seems to be no need at this point to have complete roundtrip support, so changes done in the generated BPEL model should not be automatically propagated back to the business process model. Therefore, a business to IT transformation does not have to be bidirectional.

As the presented business to IT transformation framework incorporates real-world requirements, it ensures that transformations implementing it have a bigger practical impact. In the following sections, the EPC to BPEL transformation application is described in detail. The EPC to BPEL transformation application implements the framework developed in this section.

5.3 Control Flow Transformation

The EPC to BPEL transformation application implements the business to IT transformation framework described in section 5.2. The belonging control flow transformation is described in this section. The transformation application also provides support for transforming the data perspective (see section 5.4) and the functional perspective (see section 5.5). It also supports merging as discussed in section 5.6.

The control flow transformation is based on a mapping of workflow patterns between EPC and BPEL. The 20 workflow patterns documented by Aalst et al. [vdAtHKB03] describe common constructs like sequences, decisions, and loops found in many workflow languages. Those constructs can be combined with each other to build more complex control flows. The analysis done by Wohed et al. [WvdADtH02] shows that BPEL does not support all workflow patterns. Another analysis by Mendling et al. [MNN05] shows similar results for EPC. The compilation of both studies is shown in table 5.1.

Table 5.1 shows that BPEL supports all workflow patterns also supported by EPC except for workflow pattern number 10. Workflow pattern number 10 describes arbitrary cycles. Aalst et al. [vdAtHKB03] explain that arbitrary cycles can be converted into structured cycles if there is no advanced workflow pattern used in the cycle itself. As EPC do not support the advanced workflow patterns like workflow pattern number 14, this requirement is fulfilled and the cycles can be mapped in most cases to a BPEL while activity. The transformation restricts the EPC notation to further prevent modelling such constructs. For example, the transformation only supports EPC models using the XOR or AND operator. In addition, a branch must be joined with the same operator it was opened with and the nesting of the operators must be correct without intersections. Those restrictions do not allow the modelling of workflow pattern 6 and 7. The restrictions lead to structured EPC

Table 5.1: Workflow patterns supported by EPC and BPEL [MNN05, WvdADtH02]

No.	Name Workflow Pattern	EPC	BPEL
1	Sequence	yes	yes
2	Parallel Split	yes	yes
3	Synchronization	yes	yes
4	Exclusive Choice	yes	yes
5	Simple Merge	yes	yes
6	Multi-Choice	(yes)	yes
7	Synchronizing Merge	(yes)	yes
8	Multi-Merge	no	no
9	Discriminator	no	no
10	Arbitrary Cycles	yes	no
11	Implicit Termination	yes	yes
12	Multiple Instances Without Synchronization	no	yes
13	Multiple Instances With a Priori Design-Time Knowledge	no	yes
14	Multiple Instances With a Priori Runtime Knowledge	no	no
15	Multiple Instances Without a Priori Runtime Knowledge	no	no
16	Deferred Choice	no	yes
17	Interleaved Parallel Routing	no	partly
18	Milestone	no	no
19	Cancel Activity	no	yes
20	Cancel Case	no	yes

models. Being able to only create structured EPC models also helps business experts to define easier to understand models. Restricting the expressiveness of the language used for business process modelling is an accepted approach in industry and also supported by the business to IT transformation framework introduced in the previous section. The control flow transformation also supports implicit closing of a branch at the end of a process. Cyclic EPCs are supported as well, because that is a core requirement of the framework.

From a more abstract point of view, the control flow transformation uses the “structure-identification” strategy as described by Mendling et al. [MLZ06]. The transformation is similar to Specht et al. [SDTK05], but it is not limited to the control flow perspective only.

As the modelling language is restricted, the business expert must be supported in modelling EPCs according to those restrictions. Therefore, validation functionality is provided, which checks an EPC model for possible violations. This validation functionality also provides a description of the problem found and how it might be solved. The validation functionality can be executed on user request. In addition, the validation functionality is executed each time before an EPC gets transformed into BPEL. According to the ARIS software tools’ product philosophy, the restrictions are not enforced while modelling.

The validation functionality does not simply ensure that the restrictions to the control flow are correctly implemented, but it also checks other aspects of the EPC model. For example, a warning is issued if business objects used in the EPC are not correctly mapped to technical data definitions (i. e. XSD). This information is needed during data transforma-

tion to correctly generate the necessary BPEL variables and message types. The data transformation is described in the following section.

5.4 Data Transformation

A business process model usually contains data elements detailing the data flow within the process. The ARIS modelling language allows relating a function to business objects in an EPC. The business object is either consumed or produced by the function. Business objects are mapped to a logical data model like an entity relationship model or an UML class diagram. There are usually several implementations of a logical data model like specific database schemas or XSDs in case of web services.

XML schema definitions (XSD) are represented in the ARIS modelling language using UML models with a specific UML profile. The user has to map the business objects to the logical data model, which itself is mapped to different implementations. This mapping is done before the transformation and is usually part of a different effort focusing on establishing an information architecture. By using business objects in the business process model instead of using technical data descriptions, the EPC remains technology neutral and is not bound to any specific implementation platform. The mapping is evaluated during the transformation to identify for each business object the belonging data implementation. BPEL variables are created for each business object and the variables are added as input and output to the generated BPEL activities like invoke, receive, and reply. The visualisation of generated variables is shown in figure 5.5. Data transformation also occurs while generating a service interface for the BPEL process itself. This is discussed in subsection 5.5.3.

5.5 Functional Transformation

5.5.1 Transformation of Service Information

In the ARIS modelling language, a “function” object type of a business process can be related to an “application system type” object type to model that the function is supported or even automated by an IT functionality. An application system type is a logical representation, which can be mapped to different implementations like legacy software or web services. In case it is mapped to a web service, the symbol type “software service type” is used for the “application system type” object type. This mapping between software service type and implementing web service is also visualised in the SOA meta model like the snippet shown in figure 5.2 at the beginning of this chapter.

As in case of XSDs, WSDL web services are represented in the ARIS modelling language by mapping the WSDL description to UML models using a specific UML profile. The mapping between application system types and their implementations is done before the transformation, e. g. while importing a WSDL file into the ARIS software tools. By using application system types in the EPC and mapping them to web services outside of the business process, the EPC does not contain any platform specific details.

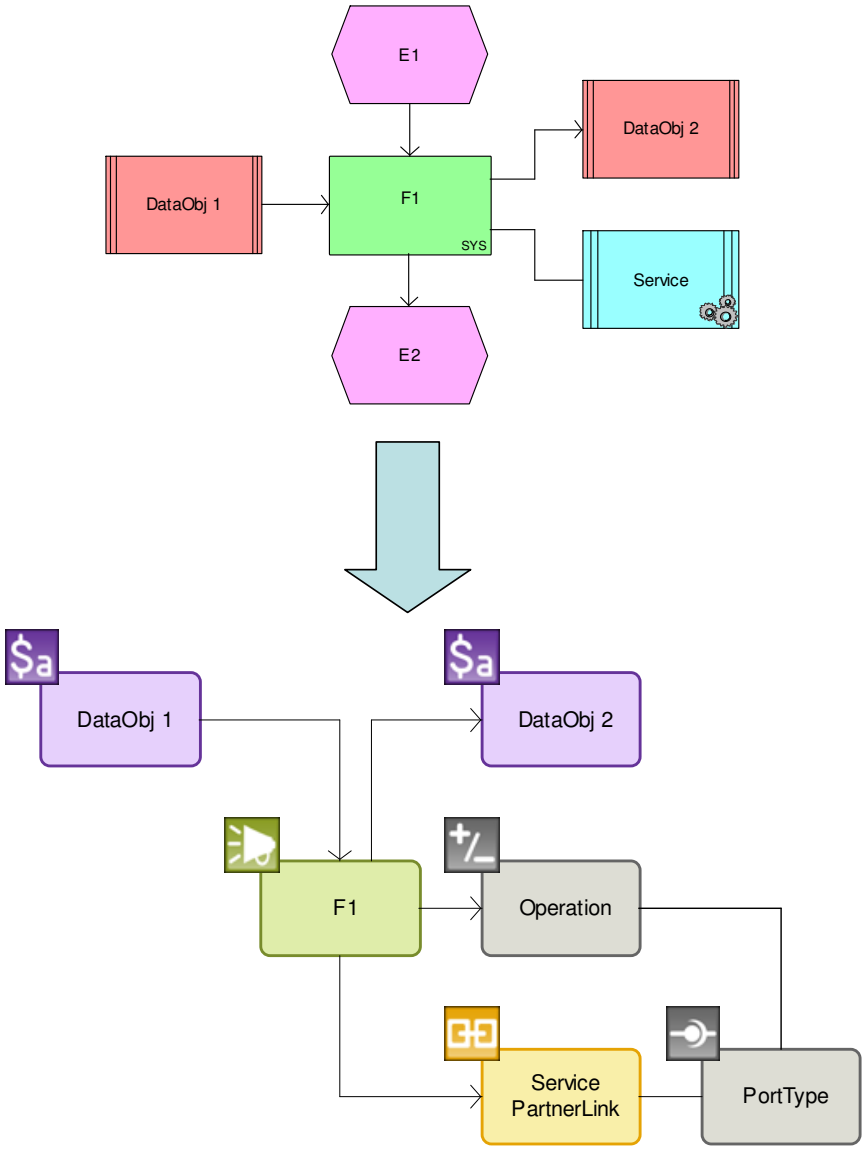

Figure 5.5: Transformation of service information from EPC to BPEL

The mapping between application system types and web services is evaluated during the transformation to identify for each function the belonging web service. For each function, a BPEL invoke or receive activity is generated. The BPEL activity generated depends on the data objects related to the function in the EPC. The default situation is generating a BPEL invoke activity. If the function produces only an output data object, it is mapped to a BPEL receive activity.

Details of the BPEL activity are not presented in the main BPEL diagram generated. Instead, a “BPEL allocation diagram” model type is assigned to each activity in the BPEL process. In this model type, the correct porttype and partnerlink are generated. It is also possible that the user selects a web service operation. In that case, the operation is modelled in the BPEL allocation diagram, too. In addition, the partnerlink types are also generated and modelled in a BPEL allocation diagram connected to the start element of

the BPEL process. Those additional details fulfil the aim of generating an executable BPEL model out of a business process description. An example of a BPEL allocation diagram is shown at the bottom of figure 5.5. This figure also shows how variables are visualised in the generated BPEL model.

BPEL supports synchronous as well as asynchronous process design. The following subsection describes how the transformation application supports those two different flavors of BPEL.

5.5.2 Synchronous and Asynchronous BPEL Processes

BPEL supports synchronous and asynchronous processes. If an external entity calls a synchronous BPEL process, the external entity waits for the answer of the BPEL process and is blocked for this period of time. In an asynchronous call, the external entity does not wait but instead the BPEL process invokes the external entity to call back. A synchronous BPEL process starts with a BPEL receive activity and ends with a BPEL reply activity. An asynchronous BPEL process starts with a BPEL receive activity, but ends with a BPEL invoke activity.

EPCs can be transformed to synchronous as well as asynchronous processes. The user of the transformation application specifies if he wants to generate a synchronous or asynchronous process. However, in some cases the structure of the EPC already determines what kind of BPEL process must be generated. For example, if an EPC has several start events merged with an AND or XOR operator, the EPC can be only transformed to an asynchronous BPEL process.

5.5.3 Service Interface Generation

Besides creating the BPEL constructs for the web services consumed, the transformation application creates a WSDL description for the generated BPEL process itself. This web service description is needed to invoke the BPEL process. The web service description includes a complete partnerlink type with the belonging message types for the public message exchange with the BPEL process.

Each BPEL process itself is also a web service, which can be invoked by other processes. By transforming a business process, the EPC can be reused as a web service. In order to reuse the business process as a web service, it must have an interface description in WSDL. Therefore, the user can specify the target namespace and the name of the web service to be generated. The transformation application generates the WSDL description, which can be exported as a file along with the BPEL file.

A service interface description does not only specify the operations of the service, but also the input and output messages. Specifying those message types must also be possible in EPC, otherwise reuse of the EPC as a web service is not possible. The transformation application supports two different ways of modelling such a process interfaces. Both ways are illustrated in figure 5.6. The preferable way can be seen on the left side of the figure. Here, the business objects are directly connected to the start and end event of the process. In the second, system functions named “Process Messaging Activity” are added

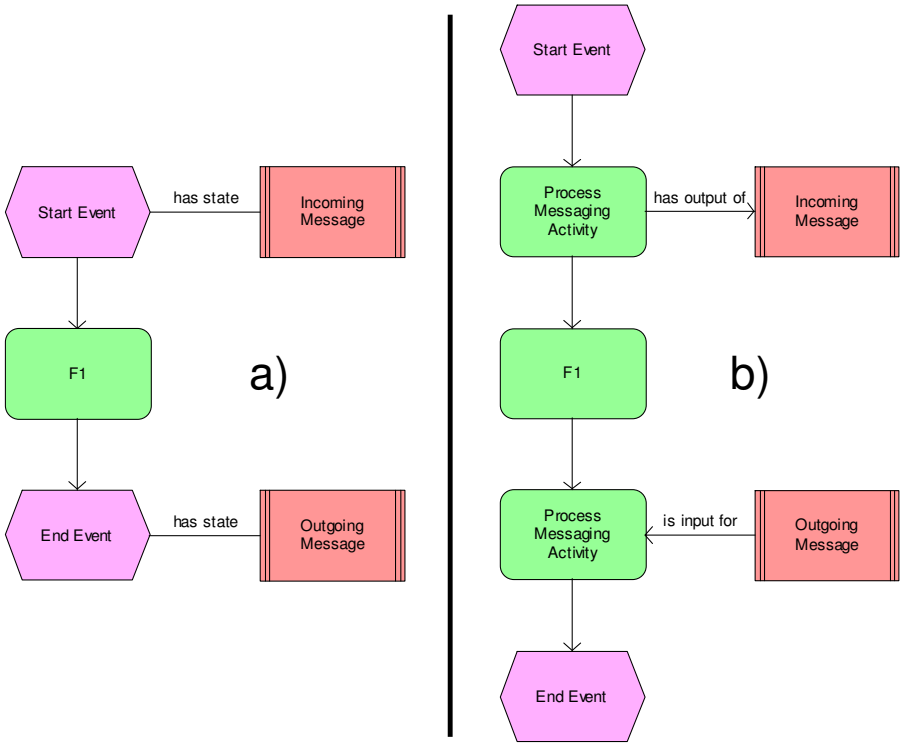

Figure 5.6: Implicit (a) and explicit (b) modelling of external message interface

directly after the start event and directly before the end event. The business objects are connected to these system functions as shown on the right side of figure 5.6. The business objects are again mapped to technical data definitions (i. e. XSD). The transformation application evaluates this information to create the necessary data definitions for the web service interface of the BPEL process.

5.5.4 Support for Proprietary BPEL Extensions

Even though BPEL is a public standard and supported by many different vendors, almost all vendors providing a BPEL execution engine have their own proprietary extensions. Typical examples are extensions for human tasks and business rules. The transformation application is meant to be vendor neutral by only creating standard BPEL and WSDL. It does not support such product specific BPEL extensions out of the box. However, in customer projects support is needed. Therefore, the transformation can be extended.

If an EPC function is connected to an organisational element like an employee or organisational role, a so called “extension activity” is added to the BPEL process as a placeholder. The user can extend this to export the human task in a format specific for the execution engine used.

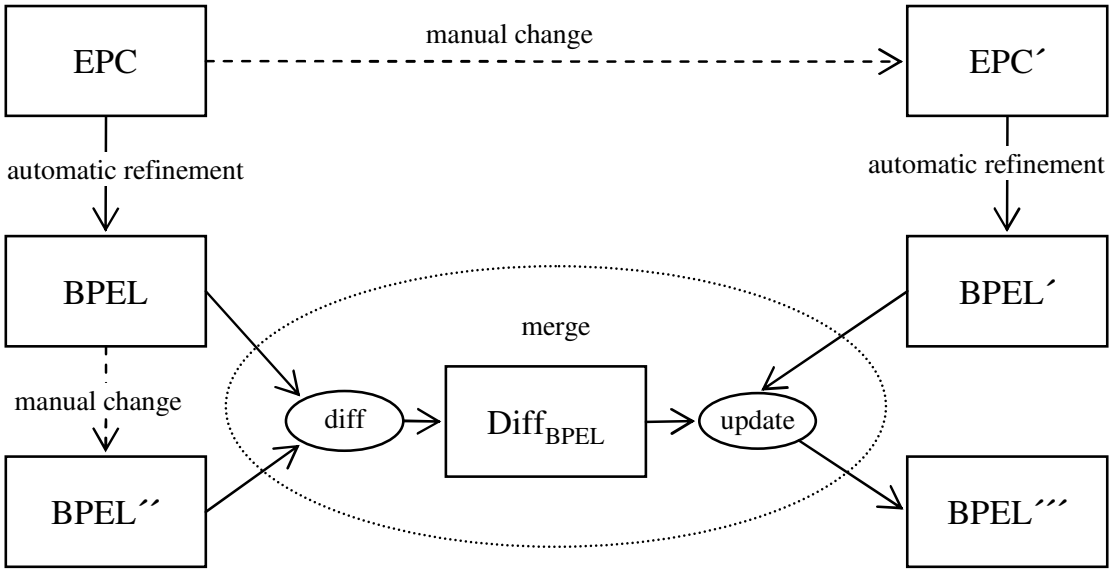

Figure 5.7: Merge support for EPC to BPEL transformation

5.6 Merge Support

The practical usage of a business to IT transformation requires that the user can run the transformation several times without loosing manual changes already done on the target IT model. This is a main requirement of the business to IT transformation framework introduced in section 5.2.

The EPC to BPEL transformation application protects changes as depicted in figure 5.7. It is assumed that the user changes the business process model (e. g. adding a new function) as well as the implementation model (e. g. some manual refinements). The manual changes of the implementation model are computed on the base of a model difference operation that takes the original implementation model (BPEL) and the changed model (BPEL´´) as input. The difference model contains model changes and stores which elements, attributes or relations are added, removed, and changed. The model differences are added to the new implementation model (BPEL´), which results from the same automatic refinement operation as the original implementation model (BPEL). There can be conflicts arising from concurrent changes. For example, a function is removed while its corresponding implementation function was changed or refined. In such cases, the merge function resolves conflicts according to a priority mode which determines whether changes in the business model dominate over changes in the implementation model or vice versa. The result of the merge operation is a new implementation model preserving most of the manual changes. The previously changed implementation model (BPEL´´) is not overwritten by the merged implementation model (BPEL´´´) so that the user can review the merge result.

Chapter

6 Case Study: Model-Driven Business Process Automation

This thesis contains two case studies, which were conducted to evaluate the ARIS extension and the applications developed. This chapter presents the first case study, which evaluated the applications developed for business process automation. The introduction in section 6.1 specifies, which parts of the thesis' contribution are evaluated and which previous work is used. Section 6.2 provides some additional ideas about the research question tackled by this case study. Afterwards, section 6.3 presents the real-world scenario used in this case study and section 6.4 discusses how the case study was implemented. Finally, the results are presented in section 6.5. The case study was conducted within the public research project OrViA[1].

6.1 Introduction

The case study presented in this chapter was conducted to evaluate the following contributions made in this thesis:

- The ARIS modelling language together with the ARIS extension developed are covered in this case study. The case study mainly focuses on the PIM and PSM level of the SOA meta model (see section 3.4).
- The service discovery application described in chapter 4 is evaluated, too. Service discovery is used to add software service types to a platform independent business process model.
- The EPC to BPEL transformation application described in chapter 5 is used in the case study to transform the annotated business process into an executable service orchestration.

[1] http://www.orvia.de/

Such an evaluation is necessary, because according to guidelines for design science research it is not enough to solely design new artefacts like the ARIS extension, but the artefacts must also be carefully evaluated [HMPR04]. See section 1.4 for a complete discussion on that topic.

The case study presented does not focus on a single artefact, but instead combines several of them to see how the artefacts interact and work together. For example, the service discovery application is needed to model a service-oriented business process model. The annotation of the business process model is done based on the ARIS extension developed in this thesis. The service-oriented business process model is input for the EPC to BPEL transformation application to generate an executable business process model.

Besides artefacts developed in this thesis, the case study incorporates other works and techniques, which were introduced in the state of the art chapter. The included techniques are:

- The case study operates in the field of business process automation as discussed in subsection 2.1.5.
- The case study uses model transformations (see section 2.4) to automate a business process.
- More specifically, the case study follows model-driven integration engineering (see section 2.3) for business process automation.
- The OrViA framework (see subsection 2.3.3) is used as guiding principle for model-driven integration engineering.
- Model checking techniques as discussed in subsection 2.3.4 are used in the case study, too.

This short overview of the included artefacts and techniques shows that the case study is a complex one trying to explore the combination of a set of artefacts. The following section elaborates in detail on the research aim of this case study by formulating a hypothesis.

The case study presented in this chapter was done within the German research project OrViA[2]. As such a research project is a collaborative effort, other researchers participated in it. The example business process was provided by the German company "Datenverarbeitungszentrum Mecklenburg-Vorpommern GmbH" [LRK06, KTRL07]. The domain specific extensions used to capture the example process are documented in [SDH06, DHW08]. The techniques for model validation are described in subsection 2.3.4 and published in [FF08]. This collaborative effort is documented in several joined publications [SKD+08, FFS08, SKI08]. The author of this thesis put together the different techniques and methods to form an integrated case study and the author performed the analysis of the case study results.

[2] http://www.orvia.de/

6.2 Research Aim

The case study was conducted to investigate if model-driven integration engineering following the OrViA framework is possible and what problems exist. In order to prevent favouring the OrViA framework by ignoring criticism, critical rationalism by Popper [Pop34] is applied and the research question is turned around into the following hypothesis:

Hypothesis It is impossible to successfully use the OrViA framework as a guiding principle for model-driven SOA implementations based on the ARIS modelling method.

To make the hypothesis operational, the application is considered to be "successful" if fewer budget is required, quality is improved, project length is shortened or a combination of those three factors. Based on that hypothesis, the research question can be further refined asking what is missing that it does not work, what shortcomings exist, where is tool support missing, which parts of the OrViA framework are not integrated, what parts of the ARIS modelling language are not useful, and where is future research needed? This shows that the evaluation is explorative trying to draw overall conclusions, but not focusing on a single problem. The application of the different artefacts is explored based on an industrial use case, which is described in the following section.

6.3 Scenario: Electronic Access to Register of Residents

The case study is conducted in the e-government domain, whereas "the term e-government focuses on the use of information and communication technologies (ICT) by governments applied to the full range of government functions" [Lau01, p. 2].

The real-world e-government scenario used in the case study is provided by the German company "Datenverarbeitungszentrum Mecklenburg-Vorpommern GmbH" (DVZ M-V[3]) [LRK06, KTRL07]. The company is based in Schwerin, Germany. DVZ M-V maintains the IT systems of the public administration in the German region Mecklenburg-Vorpommern. They also support and implement new administrational processes.

DVZ M-V is responsible for the technical infrastructure and operation of the register of residents in Mecklenburg-Vorpommern. A German citizen must be registered at the place where he lives. Information about all residents of Mecklenburg-Vorpommern is stored in the register of residents maintained by DVZ M-V.

The register of residents is used by public administration, companies, and individuals. For example, a company can use the register of residents to validate an address (e. g. address of an invoice to be issued). The register of residents is not a directory. For example, it is not possible to query all residents living in a certain street. The register of residents can only be used to validate an address. The access to the register of residents is regulated by the law "Landesmeldegesetz Mecklenburg-Vorpommern".

DVZ M-V provides electronic access to the register of residents. This service is offered to interested parties through their web portal as a web service. The service is called "simple electronic access to the register of residents" (EARR). A visualisation of EARR

[3] http://www.dvz-mv.de/

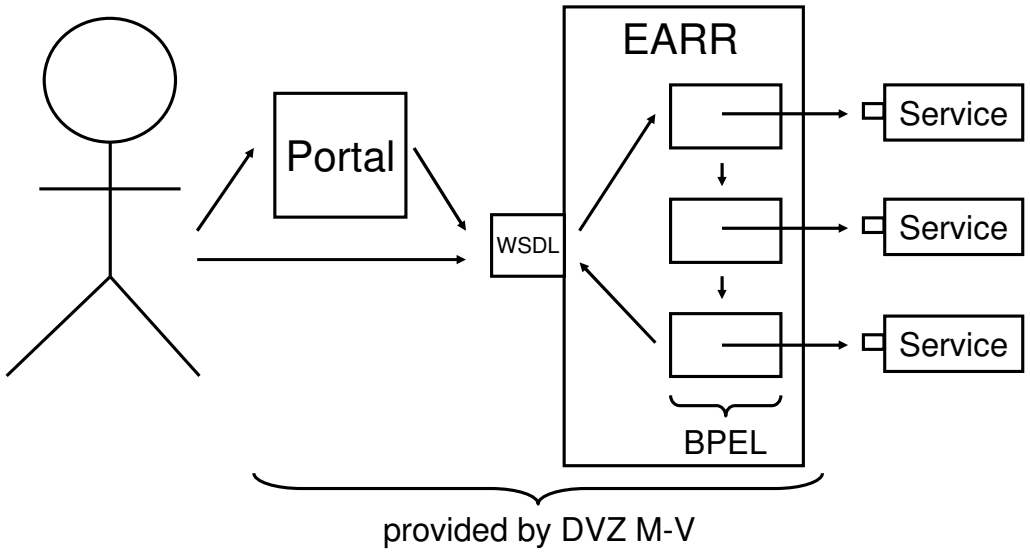

Figure 6.1: Simple electronic access to register of residents (EARR) service

is shown in figure 6.1. The prerequisites and conditions under which EARR takes place are defined by several legal regulations like the law “Landesmeldegesetz Mecklenburg-Vorpommern”.

Different public authorities take part in this service as service providers and service consumers, using different IT systems. For example, there are 100 different registration applications by six different vendors used in the region Mecklenburg-Vorpommern. To ensure interoperability between these systems, standardisation is needed. In the field of German governmental registration services, the public “XMeld” standard[4] defines the messages to be exchanged between the different systems. The implementation of EARR must comply with those legal regulations and standards. Unfortunately, several versions of the XMeld standard are available and must be supported by the EARR service simultaneously.

The process of validating an address and name of a single person with the register of residents starts with an incoming XMeld message describing the validation request. Next, it is checked if the request can be answered complying with legal regulations like data protection. If this is the case, access is granted and the responsible IT system is determined, i. e. the register containing the requested data. For example, if the person is not living in the German region Mecklenburg-Vorpommern, the request is forwarded to IT systems of the corresponding region. This is done through a so called intermediary service. There are several IT systems in Mecklenburg-Vorpommern, because the register of residents is organised peripherally. The request is forwarded to the designated system and the system answers with another XMeld message. This answer is passed back to the requestor. Each request is documented.

Following the idea of a technical SOA as described in subsection 2.2.2, all described tasks are executed by a business process execution engine invoking several WSDL web services. As the tasks are orchestrated in the shape of a BPEL process, the entire process can easily be reused as a service in more complex settings. For example, the EARR

[4] http://www.osci.de/xmeld132a/xmeld-132a.zipversion1.3.2a

service can be used to validate a list of persons and addresses instead of creating a single request for each person address pair.

Before this case study was conducted, DVZ M-V already implemented the EARR service manually. The implementation used the BPEL execution engine Microsoft BizTalk Server and the BPEL orchestration was created with Microsoft Orchestration Designer. This previous experience allows comparing the usage of the OrViA framework and the ARIS modelling language against a manual approach.

The following section describes what was done during the case study and how the different elements of the OrViA were instantiated. Also, it is shown how the ARIS modelling language and the two evaluated applications were used.

6.4 Case Study

6.4.1 Overview

According to the research question described in section 6.2, it is the aim to evaluate if and how the OrViA framework can be used successfully as a guiding principle for a model-driven SOA implementation based on the ARIS modelling language. The case study implements the electronic access to the register of residents service as described in the previous section. This section describes the case study and is structured according to the three main building blocks of the OrViA framework, namely:

- structured requirements analysis (including the service discovery application)
- validation
- transformation (including the EPC to BPEL transformation application)

6.4.2 Structured Requirements Analysis

The previous manual implementation of the EARR service was directly modelled in BPEL by DVZ M-V. This modelling resulted in a very complex model, because business details as well as technical details were mixed in one model. This model was platform specific (PSM) not allowing exchanging the underlying technology. The OrViA framework instead recommends to first do structured requirements analysis on a platform independent level (PIM) and to derive the platform specific level through model transformation and stepwise refinement. This allows separating business and technical details.

To elicit the requirements, the domain experts of DVZ M-V were interviewed and the relevant laws and regulations were studied. After, a first version of the process model was created and reworked together with the domain experts of DVZ M-V in several workshops. Also, the lead developer, who created the previous implementation of EARR, was interviewed and the BPEL model he created was studied. The existing implementation gave additional insights so that doing the same mistakes again could be prevented. For example, in the manually created version the handling for different XMeld versions was directly included in the BPEL process.

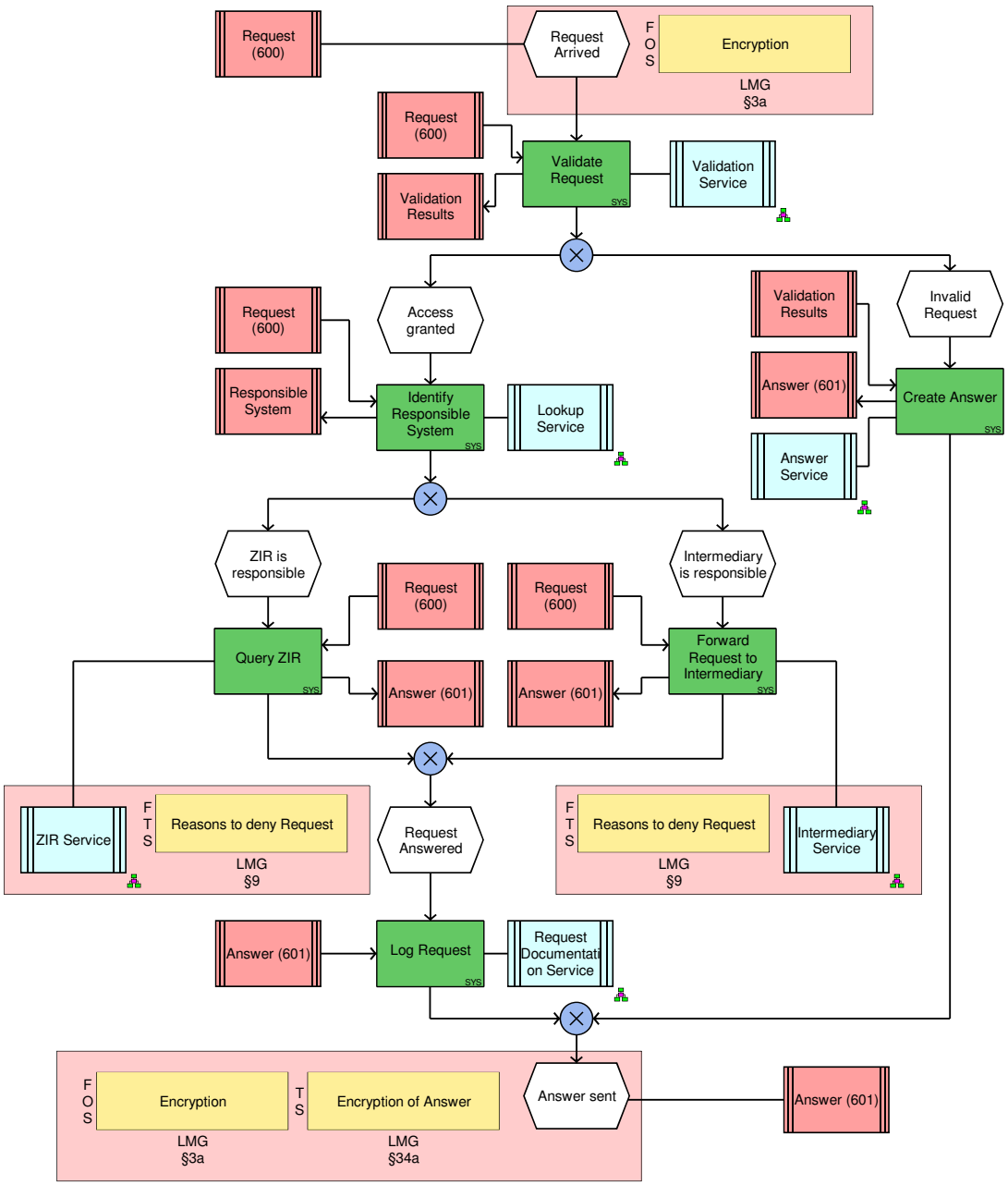

Figure 6.2: Business process model of EARR in EPC notation

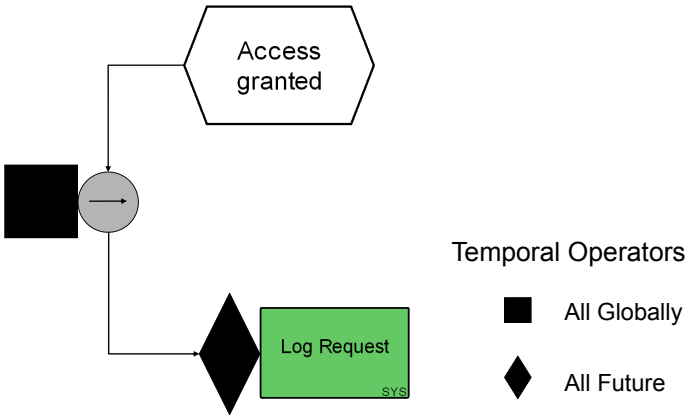

Figure 6.3: Example graphical validation rule for EARR process

To formally define the requirements, the ARIS modelling language was used. A service oriented business process model was created using the EPC notation. The service discovery application was used to assign a software service type to each function. The resulting model is shown in figure 6.2. The EPC notation was extended to also cover domain-specific details like annotations with the corresponding laws and regulations [SDH06, DHW08]. This helps domain experts to easily navigate from the business process model to the belonging laws or regulations. The laws and regulations describe each step in detail.

The resulting business process model is detailed enough to be transformed to BPEL using the EPC to BPEL transformation application. However, before the transformation was done, first the model was validated. The OrViA framework demands the usage of model validation to ensure that all informal requirements are correctly covered. The informal and unstructured requirements are defined in the laws and regulations. The following subsection describes how validation was done.

6.4.3 Validation

The case study applied the model checking techniques introduced in subsection 2.3.4. First, the laws and regulations were analysed for possible rules. Afterwards, the rules were modelled using G-CTL. For example, it is defined that each access to the register of residents must be documented. This rule is shown in figure 6.3. In plain CTL, this rule is represented by the follwoing expression: “AG(E_Access_granted -> AF(F_Log_Request))”. It means that on every process path (“AG”) beginning from the event “Access granted” the function “Log Request” has to exist on all paths in the future (“AF”).

The G-CTL rules as well as the EPC model were exported in the ARIS software tools’ proprietary AML format. To enable model checking, the content of the AML files was converted to the required format of the model checker SMV. The conversion was done using the operator hierarchy concept described in subsection 2.3.4. Computation of the

validation result by the model checker SMV took only a few microseconds. If a rule could not be validated, a counter example was given by the model checker.

To evaluate if G-CTL can be used by business experts, a modelling workshop with business experts was conducted. At the beginning, the G-CTL notation was introduced based on some examples. Afterwards, the business experts were asked to formalise rules given as textual description. In most cases, the business experts were able to create the correct G-CTL diagram.

6.4.4 Transformation and Execution

The validation showed that all requirements were formalised correctly. Also, the validation functionality of the EPC to BPEL transformation application issued no warnings or errors. Therefore, the EPC model was transformed into BPEL using the transformation application described in chapter 5.

It was not possible to deploy directly the generated BPEL model, but instead some manual refinements were needed like fixing the variable definitions and adding additional namespaces to the BPEL header. Still, no significant reworks were needed. The BPEL process was deployed on the Oracle SOA Suite[5]. It was not possible to deploy the generated model on Microsoft BizTalk, because BizTalk is not standard compliant. The web services were not hosted on the Oracle server, because in reality they are not hosted on the same machine either. The web services were deployed on the Java servlet container Apache Tomcat[6]. It was not possible to directly use the web services provided by DVZ M-V, because of data protection reasons. Therefore, a dummy implementation was created for the web services as well as for the end user portal[7]. This implementation followed the implementation done by DVZ M-V.

6.5 Results

6.5.1 Case Study Domain

The previous section described how the OrViA framework was instantiated in the case study, where the two applications developed in this thesis were used, and where the ARIS modelling language extended in this thesis was applied. The case study was conducted to check if the hypothesis formulated in section 6.2 holds. The hypothesis postulates that the OrViA framework cannot be used as a guiding principle to do a model-driven SOA implementation based on the ARIS modelling language. After conducting the case study, the author tends to slightly reject the hypothesis under the conditions discussed in this section.

The case study was done with a use case from the e-government domain. Therefore, the hypothesis can only be rejected for the e-government domain if the selected use case

[5] http://www.oracle.com/technologies/soa/soa-suite.html
[6] http://tomcat.apache.org/
[7] https://service.mv-regierung.de/web/emrauser/emra

is representative for this domain. The author is confident that the use case is representative for the e-government domain, because it was selected by a well-established use case partner working in this domain. The use case partner tried in the past to implement the same use case, but in contrast to the case study, the use case partner followed a manual approach without using structured requirements analysis. The use case is directly implementing a public law and it involves different actors like the register of residents, gateways to other German regions, and an end user portal. As the use case implements the business process as it is defined by the law, the use case is not too simplistic but instead a realistic one. Even though only one case study was done in the e-government domain, the currently most important technologies were covered, because the used technologies were selected by the use case partner already before the case study was conducted.

As the case study was only done for a use case in the e-government domain, additional use cases in other domains are required to see if the OrViA framework based on the ARIS modelling language can be applied there as well. Because of the limited time available to prepare this thesis, conducting additional case studies was not possible.

6.5.2 Structured Requirements Analysis

One important building block of the OrViA framework is structured requirements analysis. In order to be able to reject the hypothesis, the usefulness and applicability of it must be shown in case of the use case. The ARIS modelling language was used as a base for structured requirements analysis. The process defined by the law was formalised using the EPC notation. To better represent the requirements, the standard EPC notation was extended to cover domain-specific elements like clauses from the law. This approach of combining the advantages of a standard notation with a domain-specific language proved to be very successful. Using a standard notation has the advantage that it is supported by commercial tools, which also provide the necessary transformations to follow a model-driven SOA implementation approach. On the other hand, extending such a standard notation by domain-specific elements allows to better capture the domain and to adapt the used language to the language used by the domain experts. This increases the comprehensibility of the models for the domain experts and results in a higher acceptance by them.

It was possible to model all aspects important for the SOA implementation and to later derive the implementation through automated model transformation. During structured requirements analysis, the usage of the service discovery application was evaluated as well. The service discovery application was able to suggest the correct services for each process step. As no capabilities were used, only the structural matching algorithm was used.

Additional benefits of doing structured requirements analysis were identified. For example, as the SOA implementation is directly derived from the business process model, the use case partner can use the business process model for proving the compliance of the implementation with the law. This is possible, because structured requirements specification and implementation are not mixed in a single BPEL model, but instead the implementation is derived out of the structured requirements specification through an automated transfor-

mation. Therefore, it is enough to check the structured requirements specification during an audit in contrast to also checking the actual implementation.

6.5.3 Transformation and Execution

Another important element of the OrViA framework is the transformation used to derive the IT implementation out of the structured requirements model. Here, the EPC to BPEL transformation application contributed by this thesis was used.

The author tends to reject the hypothesis in case of the transformation as well, but there are a few more concerns to be discussed. It was possible to use the transformation to create the BPEL model. The structure of the business process model as well as the selected software services were correctly transformed into the corresponding BPEL constructs. This helps to speed up the implementation step, because creating all those constructs manually requires much more effort and is an error-prone task. On the other hand, the current transformation has some shortcomings, which result from bugs in the transformation as well as conceptual problems. For example, transforming business object descriptions given in the business process model into data definitions (given as XSDs) is not working as expected. However, this was due to a bug in the transformation implementation and not related to any fundamental conceptual problems in the approach. It was possible to provide small scripts fixing the wrongly created data constructs.

A more pressing issue is related to the transformation of conditions in the control flow of the business process model. As BPEL is an executable language, the conditions for loops and branches must be defined with a strict syntax like XPath expressions. However, a business process model usually does not contain such formal expressions nor can it be expected that a business expert is able or willing to create such expressions. This would be wrong from a conceptual view point, as well. XPath expressions are a concrete technology and should therefore not be added to a platform independent model like the business process model. Therefore, it must be investigated, how such conditions can be expressed in a technology independent way. A possible solution might be adding business rules to the EPC, but this needs further investigations. Besides those problems discussed, it was possible to deploy and execute the generated BPEL models without having to change a lot.

6.5.4 Validation

The third core element of the OrViA framework is validation. According to the OrViA framework, validation should be applied to different artefacts like the business process model, the transformation rules, and the generated executable process model. During the case study, validation was not done on the transformation rules, because they are part of the transformation application. Validation was used for the business process model. Based on the law, a set of rules was created, which must be enforced like that each access to the register of residents must be documented. Afterwards, the business process model was checked to see if it complies with the rules using model checking techniques. From an algorithmic and technological standpoint, the author can confirm that validation works.

However, a more interesting question is whether the approach is feasible in real-world projects. The workshop with the business experts showed that they are able to formulate correct CTL expressions using the G-CTL notation.

Another important question is how the rules can be integrated with the process models. For example, it must be possible to reference a process model element like an event or a function in the rules. The current solution is not satisfying, because the dependency between rule and process model is too tight. If a rule specifies that a function must occur after a certain event, the model checker will be only able to validate this rule if the function and the event in the process model are named exactly as in the rule. If business experts are allowed to freely name model elements while creating a business process model, this is very unlikely to happen. Therefore, the vocabulary used for the model elements must be defined and naming conventions must be enforced. At the current point, the author sees this as a major problem.

The OrViA framework suggests validating the generated executable process model in order to ensure that manual changes have not changed the semantics and that the executable model is still implementing all business requirements. This validation step was not done in the case study, because the approach would have been similar to validating the business process model and the same limitations would apply. In summary, the author rejects the hypothesis in case of validation, even though this is the most problematic part.

6.5.5 Tool Chain

The author does not consider it to be a threat to the validity of the study that only one specific set of tools was used (mainly ARIS software tools, Oracle BPEL Process Server, and Apache Tomcat), because the focus was on evaluating if the OrViA framework can be applied for such an implementation and not if it works with any kind of tool combination. However, it will be interesting to see if such a tool chain can also be built using Open Source software or with other commercial products.

In general, the author found it challenging to integrate the different tools to form a complete tool chain, even though there are public standards like BPEL and WSDL. Making the top-down approach work is possible, but implementing a roundtrip scenario is almost impossible. For example, if the BPEL model is changed in the ARIS software tools as well as in Oracle JDeveloper, it is hard to merge those changes. The OrViA framework only provides a top-down path with no backward links, because this makes the OrViA framework simple and easy to understand. On the other hand, it might be a too simplistic view for real-world projects, which is another point why the hypothesis is only slightly rejected.

6.5.6 Concluding Remarks

Besides the problems and limitations discussed above, there are also some clear advantages of applying the OrViA framework together with the ARIS modelling language. The OrViA framework clearly divides the necessary tasks into packages. Each package requires specific skills like having profound business knowledge for structured requirements analysis or having software engineering skills for deployment and execution of the gener-

ated executable process model. This clear separation helps to reduce the overall complexity, because people only need a part of the overall required skill set to handle the part they are assigned to. The complexity is further reduced by step-wise refinement. Each step only adds few aspects to the models and is therefore easier to handle. For example, during business process modelling, software services are discovered and selected but providing the correct binding information is done at a later step. This confirms the advantage of having different abstraction levels as the ARIS modelling language extended in this thesis does. It also shows that a vertical transformation is needed to convert between the different abstraction levels.

The OrViA framework supports in providing different perspectives on the overall solution, which is another advantage and success factor for real-world projects. Another advantage lies in the fact that the OrViA framework is agnostic of the software engineering methodology used. It does not matter if the project is done following the Waterfall model or using an agile approach. This is an important fact, because companies usually have their own methodologies, which often cannot and should not be replaced. Therefore, being independent of a concrete methodology supports the adoption of the OrViA framework. On the other hand, the OrViA framework is conceptual and therefore it cannot be used out of the box. Companies wishing to use the OrViA framework have to conduct a pilot project to see how the framework must be tailored for their needs.

In summary, the author slightly rejects the hypothesis that the OrViA framework cannot be used successfully as a guiding principle for model-driven SOA implementations based on the ARIS modelling language. The two applications evaluated proved to be useful as well.

Chapter

7 Semantic Business Process Management

The previous chapters introduced the extension added to the ARIS modelling language and described two applications analysing the content captured with the ARIS modelling language. This chapter introduces the third and final application (see figure 7.1) – semantic business process management. This application combines the ARIS modelling language with semantic technologies to make the models machine processable. After motivating the use of semantics in section 7.1, the overall technical solution is outlined in section 7.2. The solution consists of two main parts: a semantically extended version of the ARIS software tools (see section 7.3) and a semantic execution environment (see section 7.4). The semantic business process management application is evaluated in the case study presented in chapter 8.

The author of this thesis defined the overall architecture of the semantic business process management application and how the different components of it work together. The actual software implementation was done in the diploma thesis [Sta08] supervised by the author.

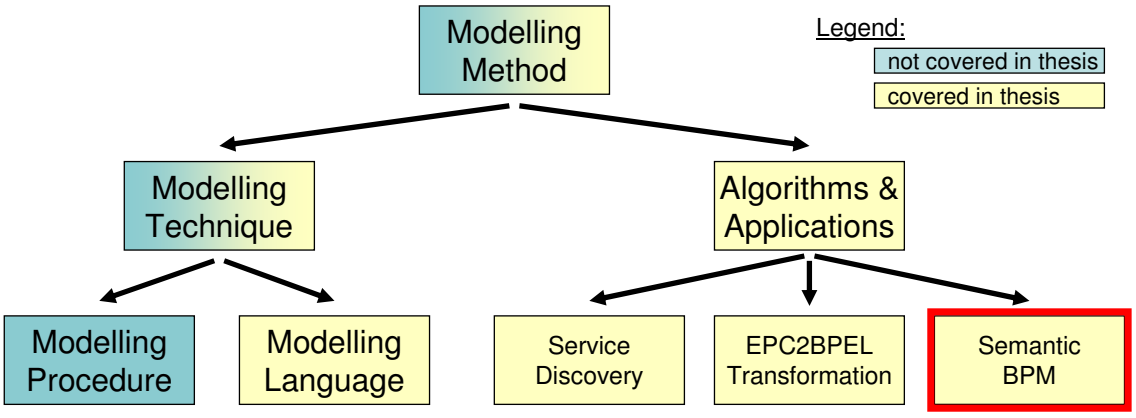

Figure 7.1: Contribution semantic business process management application

7.1 Motivation

The use of semantic technologies in business process management promises many advantages. For example, semantic technologies like ontologies, reasoners, and mediators can be used to analyse an enterprise model like checking if the model complies with a law or regulation. Additional advantages are discussed in the state of the art presentation in section 2.6. In general, it is the aim of semantic business process management to make the content of enterprise models accessible for machines. As discussed in subsection 2.6.2, there are two main use cases supported by semantic business process management:

1. Semantic technologies are used to analyse the content of an enterprise model.
2. Semantic technologies are used to derive new parts of an enterprise model.

So far, there is no empirical evaluation if the promised advantages can be achieved. In addition, it is not clear if semantic business process management can be integrated with existing methods like the ARIS method. Therefore, this chapter presents a prototypical application integrating semantic business process management with the ARIS method. The application only supports the first use case – analysing the enterprise model through machine reasoning. The application is used in the case study presented in chapter 8, where the actual empirical evaluation is done.

The application does not only enable evaluation of semantic business process management, but it also shows that the ARIS extension developed is flexible enough to be integrated with other approaches. For example, the ARIS extension introduces capabilities, but only textual descriptions are used to define their meaning. Semantic business process management uses formal logical definitions to capture the capabilities of a system. It is the challenge of this application to integrate both approaches.

The application also focuses on supporting business process automation like the EPC to BPEL transformation application described in chapter 5. In contrast to the EPC to BPEL transformation application, services used for automating the functions of the business process are not selected while modelling. Instead, only a semantic description is specified and service selection is done automatically during process execution. Therefore, the application described in this chapter consists of a semantic modelling environment and a semantic execution environment. The overall technical solution is described in the following section.

7.2 Solution Overview

The application contributes to the domain of business process automation. Semantic descriptions are used during process execution to discover and bind web services. The semantic descriptions are defined by business experts while modelling the business process. Therefore, the application consists of two parts:

1. The ARIS software tools are extended to allow semantic annotations of EPC process models. In addition, the EPC to BPEL transformation application is extended so that

the semantic annotations are preserved while generating the BPEL model. This part of the application is described in section 7.3.

2. Current BPEL execution engines are not able to evaluate semantic annotations. Therefore, a proxy service is provided, which takes care of semantic service discovery during process execution. It is described in section 7.4.

The following section describes in detail the changes done to the ARIS software tools and the EPC to BPEL transformation application to enable semantic annotations.

7.3 Semantic ARIS Software Tools

7.3.1 Overview

The semantic modelling environment is based on the ARIS software tools. The following parts are added to the ARIS software tools to enable support for semantic business process management:

- Semantic descriptions like WSMO goals can be loaded into the ARIS software tools. Therefore, the semantic descriptions must be represented by the ARIS modelling language. This is described in subsection 7.3.2.
- Business experts must be able to annotate an EPC business process model with the semantic descriptions. The belonging user interface concept is described in subsection 7.3.3.
- Semantic annotations also add ontological input and output instances to the business process model. Those instances must be combined to define a complete data flow. The belonging tool is described in subsection 7.3.4.
- The EPC to BPEL transformation application must be extended to preserve the semantic annotations during process transformation. The implemented solution is described in subsection 7.3.5.

7.3.2 Representation of Semantics in ARIS

The semantic web community develops and supports different formalisms like OWL-S [MBH+04] and WSMO [FLP+06] (see also section 2.6). Therefore, the semantically extended ARIS software tools must not be bound to a specific semantic formalism, but instead be as independent as possible. This means, the way semantic descriptions are represented by the ARIS modelling language must be the same for different semantic formalisms if possible. In an ideal case, this allows exchanging the semantic formalism without having to change the semantically annotated process models. Also, the user interface to select or create a semantic description must be identical, because the semantic formalism used does not matter for a business expert. The semantic business process management application developed supports semantic annotations based on WSMO (see

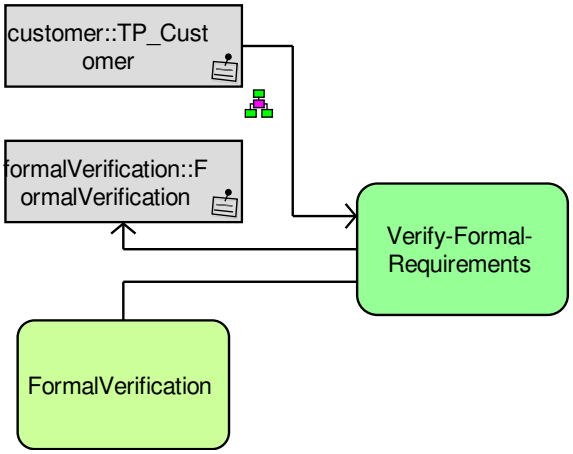

Figure 7.2: Function annotated with WSMO goal

subsection 2.6.3). Nevertheless, the general modelling principle used in the application can be straightforwardly transferred to other formalisms like OWL-S.

Describing the functionality of (IT) systems is not a completely new approach and is used in IT architecture management since several years. However, the semantic community proposes a much more formal approach, which is harder to understand by people with no background in logic, mathematics or computer science, but which, on the other hand, is expressive enough to use automated reasoning. The ARIS modelling language uses the “capability” object type to describe the functionality of systems. This object is used to represent the semantic descriptions as well.

Figure 7.2 shows a part of an EPC process model with the function “Verify-Formal-Requirements”. In a complete EPC process model, this function would be related to other functions or events (see subsection 2.1.4). The semantic description “Formal Verification” is represented by a “capability”, which is connected to the function. The “capability” was created by importing a WSMO goal description as described in the next subsection. As the semantic description is represented by a separate modelling object and not stored as an attribute value of the function, it can be reused to annotate different functions or IT systems. For example, if the semantic description is changed, only this object and its attributes must be updated. All other objects like functions related to it will have the updated semantic description as well. This ensures that semantic descriptions stay consistent and it prevents redundancy, because each semantic description is only stored once in the ARIS software tools. Figure 7.2 also shows the ontological input/output instances defined by the WSMO goal. They are represented by a separate object and connected the function as well. The following subsection shows the graphical user interface used to add semantic descriptions to business process models.

7.3.3 Selecting a WSMO Goal in ARIS

Functions of an EPC process model can be annotated with semantic descriptions to describe their functionality. In the semantic business process management application, the

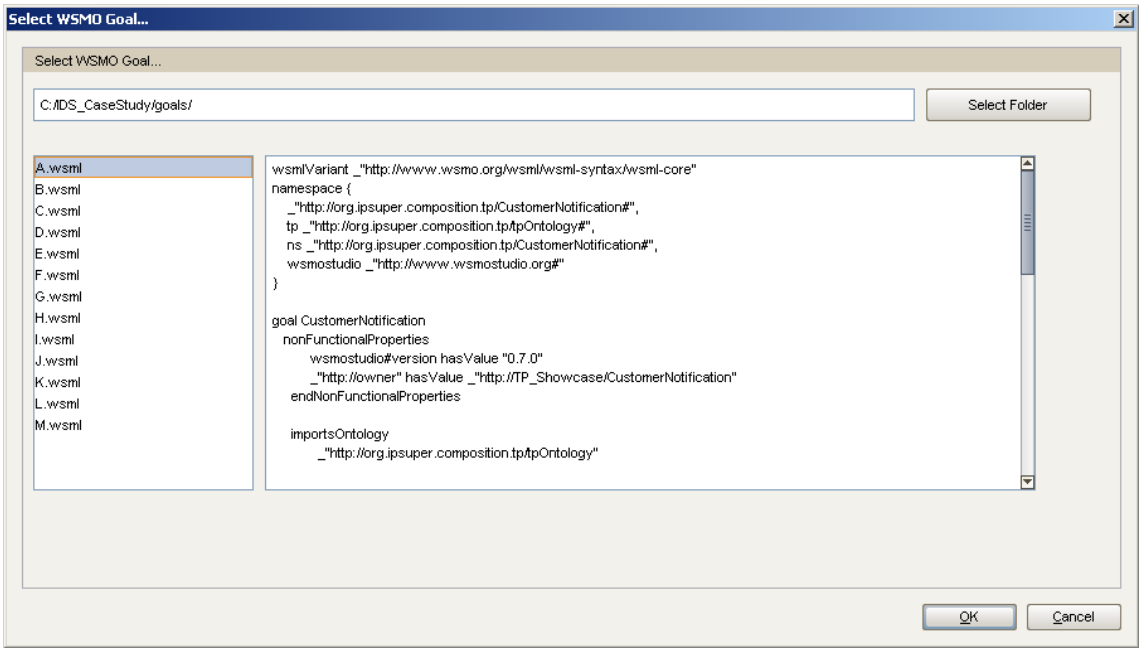

Figure 7.3: GUI to select WSMO goal in ARIS software tools

annotation is supported by the graphical user interface shown in figure 7.3. The dialog is executed after selecting the function in the EPC, which should be annotated.

First, the user selects a folder on the local hard drive, where WSMO goal descriptions as WSML files are stored. The dialog lists all files found in the selected folder on the left side of the dialog. If the user clicks on an entry in the list, the content of the file is shown on the right side of the dialog. At this stage, the content of the WSML file is shown without any syntax highlighting or other visual support.

The user evaluates the applicability of a WSMO goal based on the WSML code. Finally, the user confirms the selection of a WSMO goal by clicking the “OK” button. A “capability” object type is created and automatically related to the function as shown in figure 7.2 and discussed before.

Another small support functionality is provided allowing to remove a semantic description from a function, too. The dialog to select a WSMO goal and the support functionality to remove a WSMO goal are implemented by the scripting language embedded in the ARIS software tools. The scripts are available to the user and can be changed by them if necessary.

7.3.4 Completing the Data Flow

By selecting a WSMO goal to annotate a function, the WSMO goal and the ontological input/output instances are added to the EPC. The ontological input/output instances must be mapped to define a complete data flow. This is illustrated in figure 7.4. A support functionality is provided by the semantic business process management application to define this mapping between ontological input/output instances. The user first selects an output instance and afterwards the input instance. Now, the support functionality is

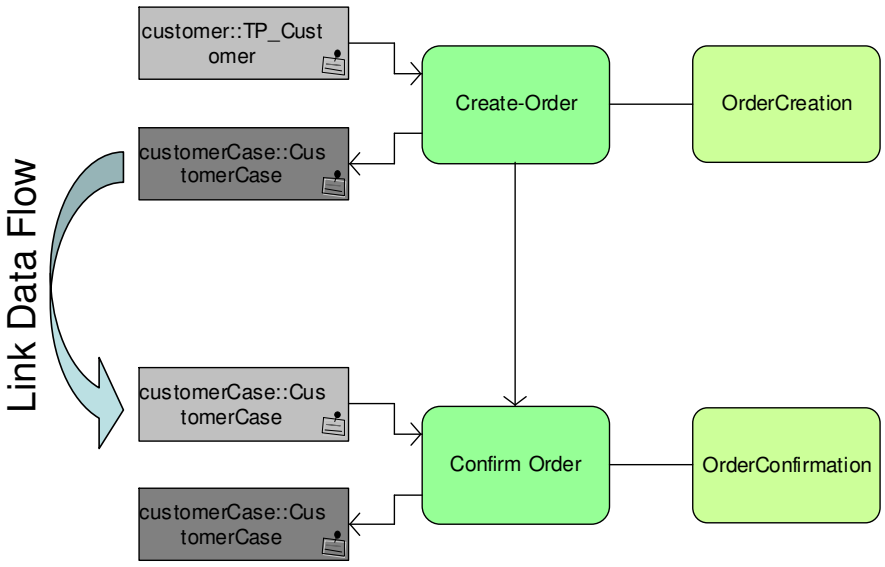

Figure 7.4: Completing data flow in EPC

executed by the user. The functionality creates a new diagram where the mapping is documented. The semantic business process management application also includes a functionality to remove such mappings.

Another small functionality is provided to define condition expressions at process branches. The user selects the belonging operator and calls the functionality. The functionality adds a function in front of the operator. This function is used to maintain the condition expression. As it is only a minor functionality, it is not visualised here.

7.3.5 Injecting Semantic Annotations in BPEL

After all functions of the EPC are semantically annotated, the business process model is transformed into an executable one. The EPC to BPEL transformation application described in chapter 5 does not preserve those semantic descriptions, because the application is not aware of this concept. Therefore, the semantic business process management application extends the transformation. Each time a function is semantically annotated, a special call to a proxy web service able to handle the semantic description is generated in the BPEL process. The generated BPEL process is standard conform and can be executed on an ordinary BPEL execution engine. The discovery of web services based on the semantic descriptions provided are handled by the proxy service, which is described in detail in the following section.

7.4 Semantic Execution Environment

7.4.1 Overview

The BPEL language version 1.1 is used as format for executable business processes. BPEL allows orchestrating a set of web services and there are many middleware products by various vendors supporting this standard. BPEL itself has a mechanism to support dynamic binding during runtime. Each web service is represented as a partnerlink in BPEL. The partnerlinks are a special kind of variable specifying the web service to be called. It is possible to exchange the content of a partnerlink during runtime by assigning a new value to it. However, the value can only be exchanged with content of the same partnerlink type. Because of this limitation, this mechanism is not used in the semantic business process management application.

A possible solution is generating an instance of the sBPEL ontology [NWvL07] (see also subsection 2.6.3). However, current BPEL execution engines are not able to process this format and a newly developed semantic BPEL execution engine was not ready at the point this application was developed. Therefore, the author decided to implement parts of the application outside the ontology stack for semantic business process management (see subsection 2.6.3) and provide an alternative execution approach.

In the semantic business process management application, all web service discovery requests are handled by a proxy service called “semantic invocation service” (SISi). Each time a web service should be discovered during runtime, the BPEL execution engine forwards the discovery request to SISi. The following subsection explains in detail the architecture and interface design of SISi.

7.4.2 Semantic Invocation Service

A central component of the solution to semantic web service discovery during process execution is the semantic invocation service (SISi). Figure 7.5 sketches the architecture. It can be seen that a classical layered software architecture (see e. g. [HNS00]) is used. The architecture consists of three layers. A reference implementation of SISi[1] is provided as OpenSource.

The top layer consists of the “External Interface Component” exposing SISi’s functionality to external users. It currently, only contains a “Web Service Interface Module”. This module is mostly generated code based on the Apache Axis2[2] web service framework. The web service interface is used by the BPEL process server to invoke SISi. SISi itself uses the web service interface module to invoke the discovered web service.

The actual application logic of SISi is available in the “Core Component”. A central “Controller Module” receives the semantic service discovery request and uses the “Semantic Discovery Module” to initiate the discovery. In a second step, the controller module uses the “Web Service Invocation Module” to call the discovered web service. Both modules used by the controller are rather small in the current implementation. They mainly forward

[1] http://code.google.com/p/semanticinvocationservice/
[2] http://ws.apache.org/axis2/

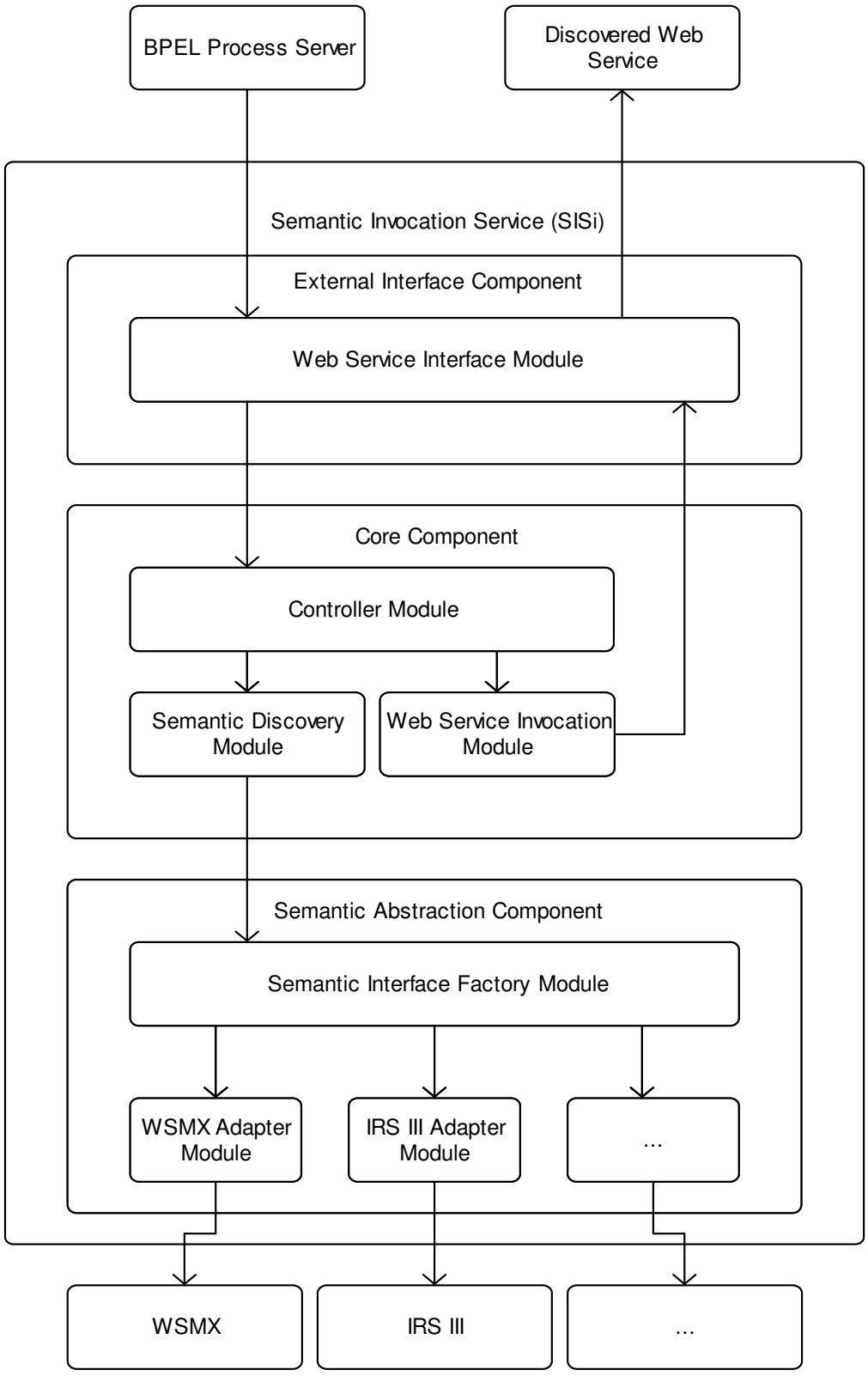

Figure 7.5: Software architecture of SISi

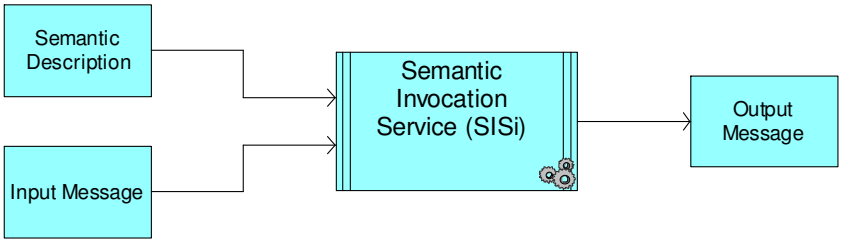

Figure 7.6: Input consumed and output produced by SISi

the requests to the other components of SISi. The modules are included to ensure extensibility to fulfil future requirements. For example, if data mediation is needed in a future version, this can be added to the semantic discovery module without having to change the controller.

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions name="WebServiceInterfaceWS"
 targetNamespace="http://sisi.externalInterface/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:tns="http://sisi.externalInterface/"
 xmlns:dataNs="http://sisi.externalInterface/dataTypes"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <wsdl:types>
  <schema xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://sisi.externalInterface/dataTypes"
    schemaLocation="SISi_WebServiceInterface_dataTypes.xsd"/>
  </schema>
 </wsdl:types>
 <wsdl:message name="invokeSemanticWebServiceRequest">
  <wsdl:part name="semanticDescription" element="xsd:string"/>
  <wsdl:part name="parameters" element="dataNs:hashMap"/>
 </wsdl:message>
 <wsdl:message name="invokeSemanticWebServiceResponse">
  <wsdl:part name="parameters" element="dataNs:hashMap"/>
 </wsdl:message>
 <wsdl:portType name="WebServiceInterfaceWS">
  <wsdl:operation name="invokeSemanticWebService">
   <wsdl:input message="invokeSemanticWebServiceRequest"/>
   <wsdl:output message="invokeSemanticWebServiceResponse"/>
  </wsdl:operation>
 </wsdl:portType>
</wsdl:definitions>
```

Figure 7.7: WSDL definition of SISi

The bottom layer called “Semantic Abstraction Component” provides access to the semantic discovery components. Even though there are preliminary efforts[3] to standardise

[3] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=semantic-ex

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://sisi.externalInterface/dataTypes">
 <xsd:complexType name="hashMap">
  <xsd:complexContent>
   <xsd:extension base="map">
    <xsd:sequence/>
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>
 <xsd:complexType name="map">
  <xsd:sequence>
   <xsd:element name="mapEntry" type="mapEntry" minOccurs="0"
    maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="mapEntry">
  <xsd:sequence>
   <xsd:element name="key" type="xsd:anyType"/>
   <xsd:element name="value" type="xsd:anyType"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:schema>
```

Figure 7.8: Data definition of SISi

such components in a reference architecture, the author is not aware of any widely accepted standard. Therefore, an individual adapter module is required for each semantic discovery component to be supported. The current implementation contains only an adapter module for WSMX [HCM+05]. All adapter modules must implement the same set of interface operations so that they can be used transparently through the "Semantic Interface Factory Module". This approach allows great flexibility, because any specifics of the different semantic discovery components to be supported are implemented in a single module.

Figure 7.6 shows the input consumed and the output produced by SISi. In order to perform its task, SISi needs the semantic description like a WSMO goal and it needs the input message for the web service to be invoked. As a result, SISi returns the message it received from the invoked web service. As SISi cannot foresee which web service will be found and invoked, it cannot provide an operation with parameters as the discovered web service has.

The code snippet in figure 7.7 shows the WSDL definition of SISi and figure 7.8 shows the belonging data definition of the different message parts. It can be seen that the input message consists of two parts – the semantic description and the message to be forwarded to the discovered web service. Currently, the type of the message part for the semantic description is only a plain string. It is unknown what type is needed for the input message of the discovered web service. Therefore, a hash map is used, which can contain objects of any type. The output message of SISi only consists of one message part. This message part transports the message received from the discovered and invoked web

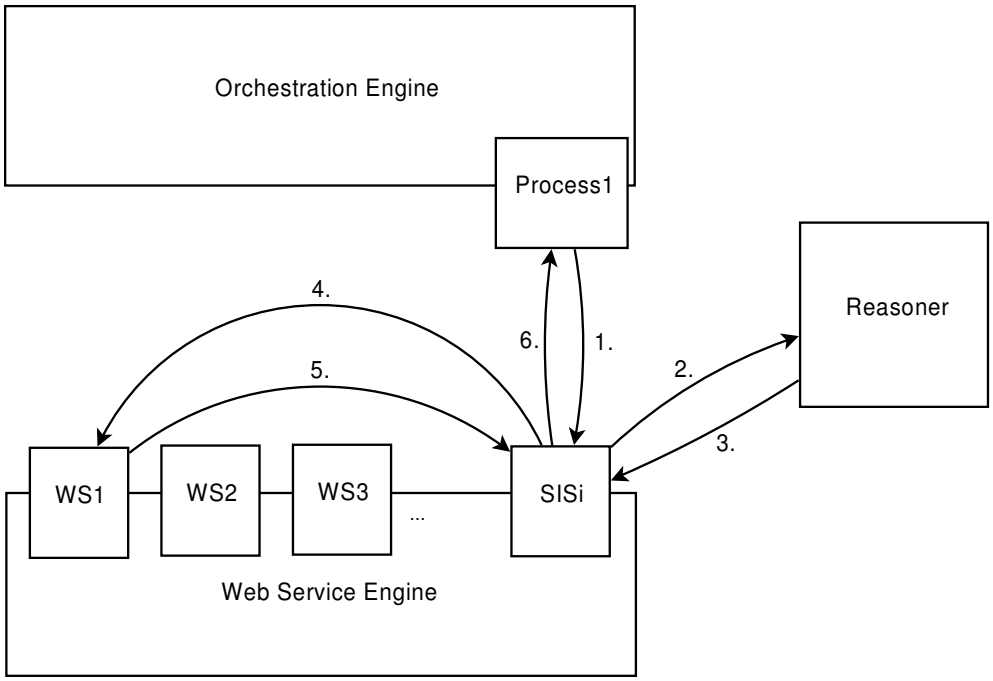

Figure 7.9: System architecture for semantic process execution

service back to the calling BPEL process. Again, as the format of this message is not known in advance, a hash map is used to store a collection of objects of any type.

7.4.3 Execution Principle

In order to execute the generated process, a special execution environment is needed, in which SISi is only one part. Additionally, the landscape includes an orchestration engine for executing the generated BPEL process as well as a web service engine hosting the involved web services. Furthermore, also a semantic reasoner is needed for discovering the web services based on the given semantic descriptions. The overall system landscape and the underlying execution principle can be found in figure 7.9. The execution of the BPEL process works as follows:

1. The BPEL process is executed on a standard orchestration engine. Whenever a semantically annotated function is found, the request is forwarded to SISi.

2. SISi receives the semantic discovery request. Besides the semantic description, SISi also takes the needed input parameters. SISi passes the semantic 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 the semantic web service description and finds through the included grounding information the underlying concrete web service implementation. Then, this web service is invoked with the input parameters received in the second step.

5. The web service is executed and the output data returned to SISi.

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

In the semantic business process management application, the system architecture is implemented with the following software components:

- The semantically annotated BPEL process is executed on the Oracle BPEL Process Server[4]. The BPEL process definition includes invocations of SISi each time a web service must be discovered during runtime.
- SISi itself is hosted on an instance of Apache Tomcat[5].
- WSMX[6] is used as semantic discovery environment. WSMX includes the necessary reasoning components. WSMX also supports invoking the discovered web services. Therefore, WSMX is doing the actual invocation of the web services and passes the output message back to SISi. Step 4 and 5 are handled by WSMX as shown in figure 7.9.
- The web services are hosted on another instance of Apache Tomcat.

[4] http://www.oracle.com/lang/de/technologies/soa/soa-suite.html
[5] http://tomcat.apache.org/
[6] http://www.wsmx.org/

Chapter

8 Case Study: Semantic Business Process Management

This thesis uses case studies to evaluate the artefacts created. This chapter presents the second case study, which mainly focuses on evaluating the ARIS extension (see chapter 3) and the semantic business process management application (see chapter 7). This chapter is structured as follows: First, the motivation for this case study is presented in section 8.1. Afterwards, the research aim is presented in section 8.2. The scenario used in the case study is presented in section 8.3. The case study done is described in section 8.4 and the results are presented and discussed at the end of this chapter in section 8.5.

8.1 Introduction

Semantic business process management as summarised in section 2.6 promises many advantages by enabling machines to process the content of enterprise models. Researchers aim at making semantic business process management a complete modelling method by working on specific modelling procedures, modelling languages, and applications. However, as the evaluation of semantic business process management literature in subsection 2.6.6 shows, today no empirical studies are available to validate the usefulness of it. In addition, semantic business process management is only expected to be successful if existing models (e. g. business process models) can be reused. So far, work on ontologies built on top of existing modelling languages is provided, but a more integrated approach is not available.

The present case study tackles both problems: providing empirical evaluation of semantic business process management and integrating it with an existing modelling method for business process management. In context of the ARIS extension developed, the case study evaluates if the ARIS extension is flexible enough to be combined with a more formalised approach. The case study presented is using the semantic business process management application described in chapter 7. Also the EPC to BPEL transformation application is reused and extended as described in subsection 7.3.5.

The case study presented in this chapter was done within the European research project

SUPER[1]. As such a research project is a collaborative effort, work done by the use-case partner Telekomunikacja Polska [ESF+08, pp. 19] was reused. The business process of the use-case partner TP was converted into an EPC model and adapted to the needs of the case study. In addition, WSMO goals as described in [FRS07] were reused without any changes, too. Reusing this existing work ensured that the example process used in the case study is a realistic one, which increases the validity of the case study. The collaboration between Telekomunikacja Polska and the author is documented in joined publications [SSEKR08, SSEKR09].

8.2 Research Aim

This case study aims at evaluating semantic business process management in an industrial setting. Therefore, the following research question is defined: "How does business process management benefit from introducing semantic technologies and how is the adoption in industry of semantic business process management hindered?"

The research interests are of explorative nature. According to Yin [Yin03], controlled experiments as well as case studies are research methods able to answer such "how" and "why" questions. Kitchenham et al. [KPP95] add that experiments are usually applied for "research-in-the-small" and case studies for "research-in-the-typical". The research aim clearly focuses on research-in-the-typical, because it intends to investigate the usage of semantic business process management in a real-world setting. Kitchenham et al. also state that case study research is often used to evaluate new technologies. This also applies for the research aim defined. To ensure that the research is conducted in an realistic environment, the existing ARIS method is used and extended to support semantic business process management as discussed in chapter 7. In addition, the case study is based on a real-world use case, which is described in section 8.3.

The case study research follows the methodology defined by Yin [Yin03]. Yin's methodology is augmented with ideas taken from Kitchenham et al. [KPP95], because they describe specific practices for case study research in software engineering, which are applicable here as well.

8.3 Scenario: VoIP Ordering Process of Telekomunikacja Polska

The business process used in the semantic tutorial was contributed by Telekomunikacja Polska[2] (TP) [ESF+08, pp. 19]. Telekomunikacja Polska Group is the dominant player in the Polish telecommunications market serving 10.6 million fixed-line subscribers and over 12 million mobile customers, as of Q1/2007, employing about 28.000 people.

The voice-over-IP (VoIP) ordering business process is illustrated in figure 8.1 using the EPC notation. Most events and all semantic annotations were removed from the process

[1] http://www.ip-super.org/
[2] http://www.tp.pl/

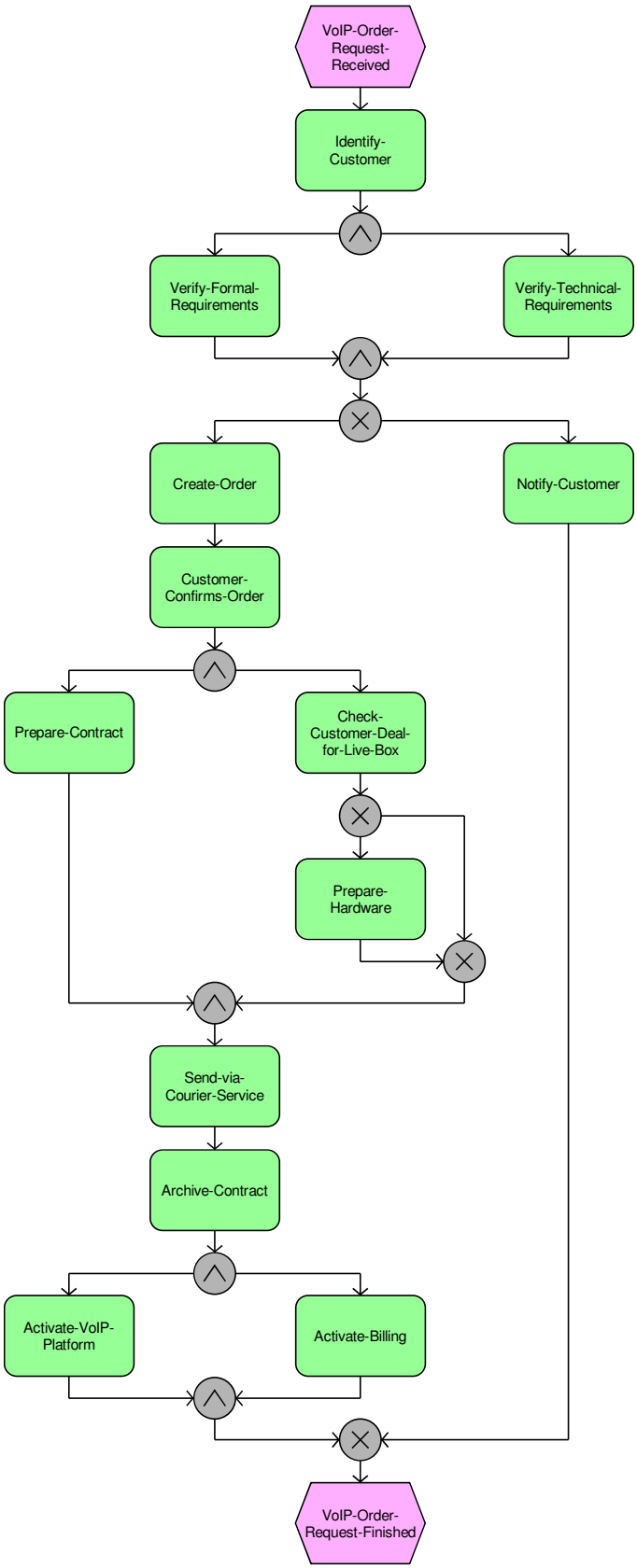

Figure 8.1: VoIP ordering process at Telekomunikacja Polska

model so that it fits on one page. The VoIP ordering process allows TP's customers to order the VoIP service for an existing contract. The ordering process is initiated by the customer through TP's web portal. After identifying the customer, the process first checks if all technical and formal requirements are fulfilled. A new order is created, which must be confirmed by the customer. A check is run to see if the customer already has the necessary hardware. If not, the hardware is sent together with the contract to the customer. After TP receives the signed contract, the contract is archived, the billing system is activated, and finally the VoIP service is activated. Frankowski et al. [FRS07] provide a set of semantic descriptions for the telecommunication domain, which were reused in the case study.

8.4 Case Study

8.4.1 Overview

The case study aims at evaluating semantic business process management in a real-world setting. The following elements contribute to this aim:

- The semantic business process management application is based on the ARIS method, which is an accepted modelling method for business process management.
- The real-world scenario described in the previous section is used.
- Participants of the case study have an industrial background (see subsection 8.4.3).

During the case study, a tutorial and the semantic business process management application were provided to participants. The tutorial describes in a step-by-step instruction how to use the application (see subsection 8.4.2). After conducting the tutorial, participants were interviewed. The questionnaire and details about the interviews can be found in subsection 8.4.4. The results of the case study are presented in section 8.5.

8.4.2 Semantic Tutorial

The tutorial describes the necessary steps to be conducted by the participants. This tutorial is a written document and a modelling database for the ARIS software tools. The tutorial starts with describing the domain ontology and the business process to be annotated semantically. Afterwards, it provides a step-by-step instruction to first annotate the business process with WSMO goals, second to complete the data flow, third to transform the business process into an executable one, and finally how the business process is executed. Each part of the tutorial is illustrated by one example. For example, it is explained how to annotate the first function in the business process. Annotating the remaining functions is the task of the participant and no further guidance is available in the tutorial. The tutorial does not include any descriptions about semantics.

Table 8.1: Participants of semantic business process management case study

Type of Organisation	Organisations	Interviews	Participants
Research Consulting Institute	2	2	3
University	1	2	2
University of Applied Sciences	2	4	4
Company	3	5	8
Sum:	8	13	17

8.4.3 Participants

There are participants from different organisations. Table 8.1 shows the different types of organisations, how many organisations of each type participated, how often the case study was done, and how many people took part. Research consulting institutes are research institutes, which are not solely financed through the public, but also offer commercial consulting services. A typical example is the German Fraunhofer institutes. The table distinguishes between university and university of applied sciences, because the latter one focuses on practical application in contrast to theoretical education. None of the participants was part of the research team and most of them had no prior knowledge of semantic technologies. No additional material for background reading was provided to the participants besides the semantic tutorial. The author paid special attention that the participants do not get aware of his own preconception of semantic business process management and the case study propositions. The questionnaire used is discussed in the following subsection.

8.4.4 Questionnaire

After the participants conducted the tutorial, their experience was gathered through semi-structured interviews with 18 open-ended questions. The interviewees were asked to describe what they have done, how non-semantic and semantic approach differ, and to reflect on the usage of semantics. At the beginning of each interview, the author elaborated on the background of the research effort and explained that the interview results are made anonymous and not publicly available assuring privacy. The author emphasised that he is not trying to prove or disprove semantics as beneficial. Participants were encouraged to ask questions. The interviews were not recorded but instead conducted by two researchers. One researcher led the interview and the other researcher focused on taking notes. Each interviewer wrote a small summary immediately after the interview and both summaries were then exchanged. Work artefacts were collected like the semantically annotated business process models. Some interviews were conducted as group interviews with two or three participants. Therefore, there are a higher number of participants than interviews.

The questionnaire is shown below. Questions were rephrased if necessary. Also, questions were skipped if the interviewee already provided an answer earlier during the interview.

1. Do you have any questions how the interview is conducted or about the background of the research?
2. To start, we like to ask you to describe briefly what you did in the tutorial.
3. You mentioned the term semantic description. How do you understand this term?
4. How do you define the term ontology?
5. During the tutorial a domain ontology is provided. Did you use the domain ontology while conducting the tutorial?
6. The domain ontology is available in the tutorial in different representations. Which representation did you use and why?
7. How does assigning a service to a function work in the tutorial?
8. How is a semantic description represented in the tutorial? (or: What is a goal and what does it comprises?)
9. During the tutorial you had to select a semantic description to annotate the functions in the business process. Were you confident to have selected the correct semantic description?
10. Did you made use of the pre-/postconditions defined while selecting a semantic description?
11. In a first step during the tutorial you did something about data. Can you explain what you have done there and why?
12. If you take a look at the overall semantic approach, what is in your opinion the main difference between the semantic approach and the non-semantic approach embedded in ARIS?
13. Can you motivate dynamic service binding? Is it relevant in industry?
14. Do you think that the semantic approach has any advantages compared to a non-semantic approach? Why?
15. Do you think the semantic approach is feasible for a business expert used to EPC modelling?
16. How long did you need to conduct the semantic tutorial?
17. Where you able to follow the descriptions in the tutorial or was something missing?
18. Thank you for participating in the tutorial. We would like to take a look at the modelling database used during the tutorial. Would you please be so kind to send us this database?

8.5 Results

8.5.1 Overview

This section presents the case study results and a discussion of the outcomes. The results are not simply observed facts, but also a summary of the discussions with the participants. This section is structured around the main interview points. The answers between participants from the different types of organisation were consistent if not discussed otherwise.

8.5.2 Understanding Semantics

At the beginning of each interview, participants were asked to summarise the different steps of the tutorial. Most participants were able to name the steps and their order. During this summary, most of them used the word “semantic”. They were asked how they understand semantics and how they define the term “ontology”.

All participants said a semantic description is not a technical one, but instead business oriented. Interestingly, some of the participants said that a semantic description defines only what needs to be done but not how to achieve it. None of the participants provided one of the popular definitions of ontology like “shared conceptualisation”. Instead, all participants tried to describe what an ontology is. Almost all participants pointed out that an ontology is a collection of terms, concepts or classes. Some of them used the term “glossary” or “taxonomy”, but only a few called an ontology a “namespace”, a “domain”, a “classification” or a “domain specific language”. Some participants pointed out that an ontology not only defines terms, but also relations between them. For example, one participant said an ontology describes “what exists and how everything is related to each other” and another said it is a “model of the world”. One participant pointed out that an ontology standardises the vocabulary used. Interestingly, some participants also talked about “business cases” while actually referring to concepts. However, only a few pointed out that an ontology is processable by machines.

None of the participants seemed to be comfortable with the term “ontology”, because the term is not known from daily language usage and seems artificial to them. The author concludes to not use the term ontology while talking to business experts, but instead talk about “semantics” or “semantic descriptions”. To give a more detailed definition, one should talk about a glossary of business terms, which also has detailed relations between terms in contrast to ordinary glossaries. One should also point out that semantic descriptions of services are business oriented, processable by machines, and used to describe what needs to be done, and not how it should be implemented.

8.5.3 Getting Familiar With the Domain

At the beginning of the semantic tutorial, the domain ontology developed by TP was presented to the participants. All participants confirmed to have studied the domain ontology at the beginning of the tutorial, but only a few of them used it later. The example process was still simple enough and the terminology used was also known to the participants, who proved to be experienced in business process management, because they were familiar

with similar business processes. Many participants pointed out that it is unclear where the domain ontology comes from and who creates it. The domain ontology was only available in the printed tutorial, but it was not available in the ARIS software tools. This was confusing for some participants, because they expected the ontology to be present in the tool, too. Many of them pointed out that for more complex ontologies an “ontology browser” is required to allow easy navigation between the different concepts.

The author concludes that having a domain ontology is useful even if no other semantic technologies are used. Such an ontology must be available in the business process modelling tool allowing easy usage and navigation.

8.5.4 Visualisation of Ontologies

The domain ontology was presented in three different ways to the participants: first a “star” of the main concepts generated by WSMO Studio, second a UML class diagram with a class for each concept plus the belonging attributes and the main relations, and third the WSML code.

If participants were familiar with UML modelling like the participants not working in a company, they found the UML class diagram most useful. Participants said that the UML class diagram contains far more information compared to the star diagram. If participants were not familiar with UML, they preferred the star, because it provides an easy to understand overview of the domain ontology. All participants said the WSML code is not useful and readable. Some of them noted that it might be possible to understand the WSML syntax after training, but that it is definitely not useful for business experts. One participating business expert confirmed that by saying he refuses to look at “something” like the WSML code.

The author concludes that a graphical representation along with a textual description of a domain ontology is required. Probably several graphical representations are necessary allowing the user to select the preferred one.

8.5.5 Selecting WSMO Goals

One important step of the tutorial was selecting a goal for each function using the graphical user interface described in subsection 7.3.3. It turned out that all participants identified the name of the WSMO goal in the WSML code and based their decision mostly on the name. Only a few of them looked at additional details of the goal description such as pre-/postconditions. However, most participants recognised they must use the pre-/postconditions when goal selection is ambiguous.

Most participants were not satisfied with goal selection. Many pointed out that browsing a list of goals does not scale and more advanced search mechanisms are needed. A participant suggested that it must be possible to filter the list of goals based on concepts taken from the ontology. Another participant proposed using the pre-/postconditions as filter criteria, e. g. only showing those goals able to produce a defined state. It was also suggested to add a graphical representation for each goal. One participant suggested the

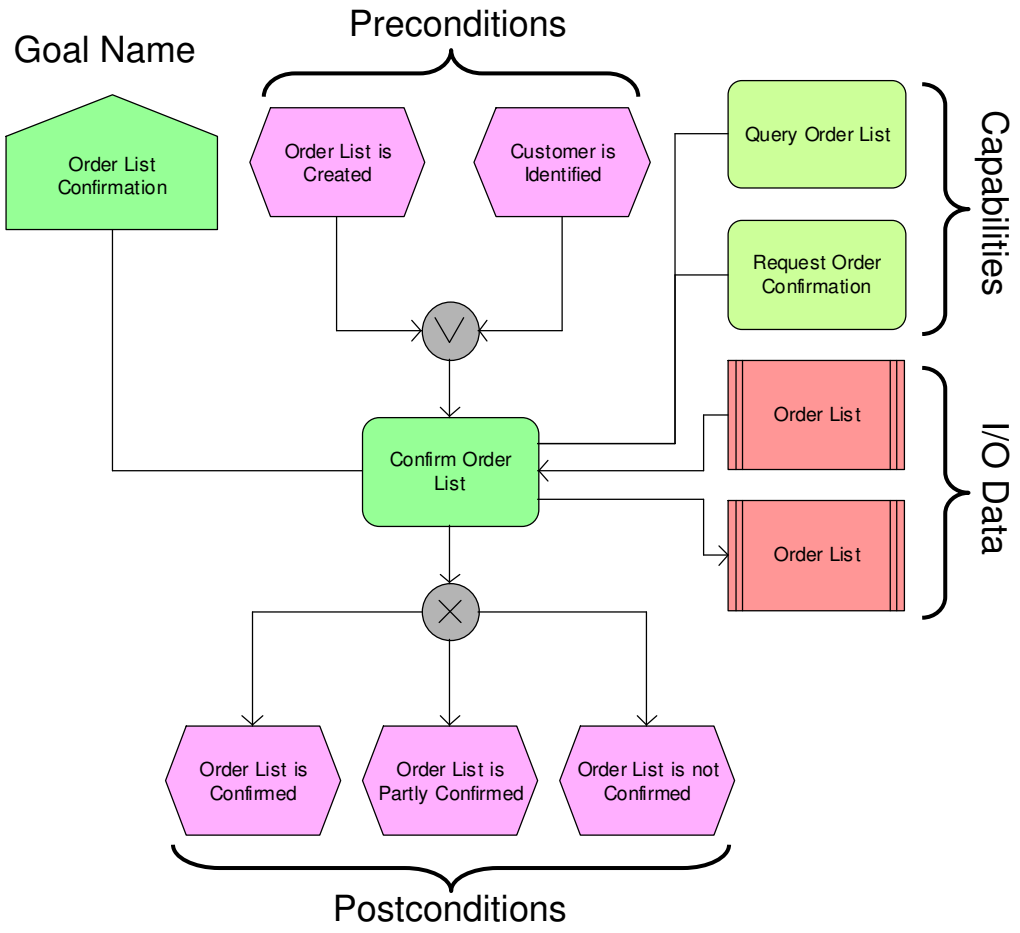

Figure 8.2: Participant contribution: graphical representation of WSMO goal

visualisation shown in figure 8.2. According to this participant, it is important to include the different visual elements in the same place so that information can be identified quickly.

The author concludes that goal selection is an important part and must be supported by a sophisticated tool. This requirement is amplified, because some participants pointed out that they cannot see any advantage compared to selecting a web service directly. Therefore, research should focus on ways to graphically visualise goals and semantic descriptions.

8.5.6 Completing the Data Flow

After selecting a goal, participants completed the data flow by mapping ontological input and output instances. Even though all participants were able to complete this step, some concerns were raised. Participants pointed out that input/output instances are similar to variables, whereas business objects are normally used in business process modelling. According to participants, those two concepts are not interchangeable, because a business object is always persistent whereas a variable must be stored in a data store explicitly. This is an interesting point, which must be further investigated. It seems that ontologi-

cal instances defined by WSMO goals are not abstract enough to be useful in business process modelling.

8.5.7 Motivating Service Binding During Runtime

Participants were asked to motivate dynamic service binding during process execution. There was a diverse set of answers with no clear conclusion. Participants were explicitly asked for an economic motivation. If they provided such a motivation, they often mentioned failover scenarios. In the author's opinion, this problem can be already solved today with enterprise service bus (ESB) platforms, but the participants were not confronted with this opinion. Participants said that instead of hard coding an endpoint URL into the executable process, only the service name is added to the process. This helps in case the service is moved to another server. Again, this seems to be a case where today's technologies such as service registries can be used.

Some participants noted dynamic service binding only makes sense if there are several services for each goal. If there is only a $1 : 1$ relation between service and goal, participants were not able to justify dynamic service binding. Participants with research background also pointed out that dynamic service binding in a company might not be as relevant as e. g. in ubiquitous computing, because a company is able to better control and govern the service architecture. Other participants mentioned dynamic service binding is dangerous, because it adds a new error source and increases the complexity of the enterprise computing stack. This shows there is no consensus whether dynamic service binding is necessary in business process automation.

8.5.8 Advantages and Disadvantages of Semantic Approach

Participants were asked about advantages and disadvantages of the semantic approach. Surprisingly, most of them mentioned a better separation of business and IT as the main advantage of the semantic approach, because the business process model does not contain technical details. Instead, the business expert only specifies the required capabilities. This helps business experts to concentrate on the business part of process modelling instead of dealing with implementation details. It also allows using not yet existing services. In addition, technical service descriptions do not have to be available in the business process modelling tool, which prevents redundancy. Participants characterised the semantic approach as creating a process template, which can be flexibly enacted, because the business process model only specifies what has to be done but not how to do it. Some participants mentioned the possibility of dynamic service binding as an advantage, but there was no consensus.

A significant problem is the conceptual mismatch between business objects and ontological input/output instances. Besides, several participants were not convinced that the investment in semantics can be justified economically, because ontologies must be defined and maintained. Some participants were reluctant about ontology modelling, because in their opinion similar efforts such as establishing an enterprise information architecture failed in the past. Such concerns are also raised by Hepp [Hep07] and must be taken

into account. All participants agreed that business experts are not able to create ontologies and goals. Graphical tools are required to overcome hurdles like WSML syntax and logical expressions. Using semantics might require having ontology engineers, which is a specific qualification. In general, the learning curve is increased, because semantics bring their own set of technologies, methods, and methodologies along. Also, the complexity of the enterprise computing stack is increased, which augments the probability of introducing errors and integration problems. Many participants pointed to the unbalanced distribution of efforts and benefits for using semantics as another major disadvantage. Ontologies, goals, and semantic descriptions must be defined by IT after consulting business experts, but those artefacts mainly help business experts. This discrepancy must be carefully managed to ensure close cooperation between all involved parties.

8.5.9 Feasibility of Semantic Approach for Business Experts

Participants were asked whether the semantic approach is feasible for business experts. Most of them agreed that it is feasible, but they also mentioned potential problems. Currently, technology is still too visible. For example, WSML code should not be shown and ontological input/output instances must be lifted to a more abstract level. Ontologies and goals used must exist upfront, because currently it is impossible for business experts to define and modify them on their own. Besides the tooling issues, a solution must be found to provide incentives to those who have to create the semantic descriptions.

Some participants were surprised that they were asked to annotate functions with semantic descriptions. They believed that a business process model already contains enough information. Those participants envisioned a more advanced way of using semantics. For example, one participant desired to have a repository of semantically described process fragments. Instead of defining the control flow of the business process, the participant expected to solely define the pre- and postconditions as well as constraints and the control flow would be automatically created. This vision seems to be similar with what van der Aalst and Pesic [vdAP06] propose as Declarative Service Flow language. Instead of defining a fixed control flow, the flow is declaratively defined allowing a more flexible enactment. It will be interesting to see if such visionary approaches will gain momentum.

Chapter

9 Conclusions

This chapter summarises the work presented (see section 9.1), compares the work to the initial theses (see section 9.2), and outlines future research activities (see section 9.3), which are planned to be done or which were already initiated.

9.1 Summary

This thesis aims at providing an integrated modelling method for service-oriented business process management. Instead of designing a completely new modelling method, the existing and established modelling method ARIS is reused. This thesis extends the ARIS modelling language so that services on different abstraction levels can be described. The ARIS extension is based on a comprehensive SOA meta model. In contrast to other works, the ARIS extension does not only cover the description of technical services, but also more abstract services can be described. In addition, it is fully integrated with all other parts of the ARIS modelling method. This shows that SOA and business process management can be integrated and are not contradicting concepts.

Based on the ARIS extension, three applications are developed to operate on the model content. The service discovery application helps business experts to evaluate service offerings and to select services for functions of a business process. The EPC to BPEL transformation application converts a business process annotated with software services into an executable BPEL orchestration following a vertical transformation strategy. The semantic business process management application uses a formalised service description and supports service discovery during process execution.

All artefacts created are evaluated in two empirical case studies to demonstrate their relevance and applicability. The first case study focuses on business process automation and covers the ARIS extension, the service discovery application, and the EPC to BPEL transformation application. The case study shows that business process automation is possible and beneficial based on those artefacts. The second case study evaluates the ARIS extension and the semantic business process management application. The results

show that semantic business process management is a promising approach, but that there are also several research challenges ahead.

The evaluation clearly demonstrates that the ARIS modelling method extension is able to mediate between the different perceptions of business process management and service-oriented architecture by providing an integrated modelling method.

9.2 Theses Fulfilment

Subsection 1.2.1 lists seven theses, which are expected to shape the work done in this thesis. The first thesis describes that an integrated modelling method is required, which does not solely focus on technical SOA artefacts. The modelling language introduced in chapter 3 and the applications and algorithms developed throughout this thesis confirm this thesis. An integrated modelling method is created, which covers business as well as technical aspects of a service-oriented enterprise architecture.

The structure of the modelling language introduced in chapter 3 also confirms the second thesis. The modelling language consists of several layers and features different notions of service concepts. For example, on the top layer a service type is introduced, which can be realised by a software service type on the platform independent layer.

Thesis three states that just relying on syntactical information like a programming interface description is not enough. This thesis is confirmed. The modelling language features different modelling constructs to semantically describe a service, for example using capabilities. As shown in the case studies in chapter 6 and chapter 8, such modelling constructs are important enabling business experts to assess service offerings.

Thesis four postulates that algorithms or applications are needed to transform content between different layers of the modelling language. Chapter 5 introduced one useful transformation, converting content on the platform independent layer to content on the platform specific layer. The usefulness of such a transformation is confirmed in the case study in chapter 6.

To support business experts in selecting appropriate services in a specific modelling situation, thesis five states that a supporting algorithm or application is needed. Such a service discovery algorithm is described in chapter 4 and proved to be useful in the case study in chapter 6.

Thesis six states that the modelling language must be extensible. The application semantic business process management described in chapter 7 demonstrates the extensibility of the modelling language.

Finally, thesis seven postulates that the modelling method must be integrated with existing modelling methods to ensure adoption in industry. The modelling method presented in this thesis is integrated with and based on the modelling method ARIS, which is one of the leading modelling methods for enterprise architecture and business process management in industry.

This overview shows that it was possible to confirm all theses stated at the beginning of this thesis in subsection 1.2.1.

9.3 Future Work

The thesis touches many different areas and therefore many opportunities for future research exist. For example, the EPC to BPEL transformation application described in chapter 5 needs a more advanced data transformation algorithm. The current version does not generate the correct variable definitions under some circumstances. To improve the situation, first work on this issue was already conducted in [CSR08]. In this work, a more advanced mapping between logical data objects and technical data definitions is used. A transformation application can use this mapping to generate the corresponding data flow.

The thesis developed one application using semantic technologies to support business process automation (see chapter 7). However, business process management is not limited to business process automation and therefore additional applications of semantic technologies in this domain are possible. In an early attempt, the author conducted a study to extract requirements for business process analysis. Those requirements document what practitioners try to achieve through business process analysis. One clear outcome is that practitioners are interested in compliance management, e. g. proving that they correctly implement a law or regulation. Therefore, in [EKSP08, EKSMP08b, EKSMP08a, EKS08] the usage of semantic technologies in compliance management is outlined. Laws and regulations are formalised as semantic policies, which must be enforced. On the other hand, enterprise models are defined semantically, too. Both artefacts, semantic policies and enterprise models, are consumed by a semantic inference engine to check whether the semantic enterprise model complies with the semantic policies. This approach is not limited to enterprise models, but it can also be used to enforce policies during process execution.

The service discovery application developed enables business experts to evaluate service offerings. However, the graphical user interface is still too complex and a much easier interface is needed. A new concept for such an user interface was developed by the author.

The ARIS extension developed is currently used in first end user projects. Here, new requirements emerge like providing a new model type to model value-chains consisting of service types. Those requirements are collected and carefully evaluated so that the ARIS modelling language covering SOA is further extended based on real-world requirements. In addition, the development of the belonging modelling procedure (see [Ric07, RP08]) might create additional requirements, which must be supported by a future version of the ARIS modelling language for service-oriented business process management.

This short overview of future work shows that the artefacts developed in this thesis are not final, but that they will evolve as soon as the conditions and environment they were built for changes. Also, new research findings will influence the design. The author will ensure that the artefacts are maintained, enhanced, and extended if needed.

Bibliography

[AdMPvdA+07] A. K. Alves de Medeiros, C. Pedrinaci, W. M. P. van der Aalst, J. Domingue, M. Song, A. Rozinat, B. Norton, and L. Cabral. An outlook on semantic business process mining and monitoring. In *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*, volume 4806 of *LNCS*, pages 1244–1255, Vilamoura, Portugal, November 2007.

[AdMvdAP08] A. K. Alves de Medeiros, W. M. P. van der Aalst, and C. Pedrinaci. Semantic process mining tools: Core building blocks. In *16th European Conference on Information Systems (ECIS)*, Galway, Ireland, 2008.

[AFKK07] W. Abramowicz, A. Filipowska, M. Kaczmarek, and T. Kaczmarek. Semantically enhanced business process modelling notation. In *Workshop on Semantic Business Process and Product Lifecycle Management (SBPM)*, volume 251 of *CEUR Workshop Proceedings*, pages 88–91, Innsbruck, Austria, June 2007.

[AFM+05] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M.-T. Schmidt, A. Sheth, and K. Verma. Web service semantics (wsdl-s) version 1.0. Technical report, W3 Consortium, November 2005.

[Alg07] J. Algermissen. Serviceorientierung mit rest. In G. Starke and S. Tilkov, editors, *SOA-Expertenwissen – Praxis, Methoden und Konzepte serviceorientierter Architekturen*, pages 777–804. dpunkt.verlag GmbH, Heidelberg, Germany, 2007.

[All06] T. Allweyer. *Geschäftsprozessmanagement – Strategie, Entwurf, Implementierung, Controlling*. IT lernen. W3L GmbH, 2006.

[All07] T. Allweyer. Erzeugung detaillierter und ausführbarer geschäftsprozessmodelle durch modell-zu-modell-transformationen. In *6th GI Workshop on Event-Driven Process Chains (EPK)*, volume 303 of *CEUR Workshop Proceedings*, 2007.

[All08] T. Allweyer. Vom fachlichen modell zum ausführbaren workflow: Am beispiel von aris und intalio bpms. Technical report, University of Applied Sciences Kaiserslautern, Kaiserslautern, Germany, February 2008.

[ARC04] Concepts for architectural description. Technical Report TI/RS/2003/007, Telematica Institute, December 2004. ArchiMate Deliverable 2.2.1 v4.0.

[ARC06] Architecture language reference manual. Technical Report TI/RS/2003/030, Telematica Institute, April 2006. ArchiMate Deliverable 2.2.2b v4.1.

[BHK+08] M. Born, J. Hoffmann, T. Kaczmarek, M. Kowalkiewicz, I. Markovic, J. Scicluna, I. Weber, and X. Zhou. Semantic annotation and composition of business processes with maestro. In *European Semantic Web Conference (ESWC) Demo Track*, June 2008.

[BKR05] J. Becker, M. Kugeler, and M. Rosemann, editors. *Prozessmanagement: Ein Leitfaden zur prozessorientierten Organisationsgestaltung*. Springer, Berlin, Germany, 5th edition, 2005.

[Ble07] M. Blechar. Magic quadrant for business process analysis market, 2h07. Technical report, Gartner, June 2007.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–44, 2001.

[BPE03] Business process execution language for web services (bpel4ws) 1.1. Technical report, OASIS, May 2003.

[BPE07a] Web services business process execution language (bpel) version 2.0. Technical report, OASIS, April 2007.

[BPE07b] Ws-bpel extension for people (bpel4people), version 1.0. Technical report, June 2007.

[BS04] B. Bordbar and A. Staikopoulos. On behavioural model transformation in web services. In *eCOMO – E-Business Processes and Infrastructure*, volume 3289 of *LNCS*, 2004.

[CAdMZ+07] I. Celino, A. K. Alves de Medeiros, G. Zeissler, M. Oppitz, F. Facca, and S. Zöller. Semantic business process analysis. In *Workshop on Semantic Business Process and Product Lifecycle Management (SBPM)*, volume 251 of *CEUR Workshop Proceedings*, pages 44–47, Innsbruck, Austria, June 2007.

[Car04] N. G. Carr. *Does IT matter? Information Technology and the Corrosion of Competitive Advantage*. Harvard Business School Press, 2004.

[CBD07] Cbdi service architecture and engineering (cbdi-sae). Technical report, Everware-CBDI Inc., 2007. Version 2.0.

[CD88] E. M. Clarke and I. A. Draghicescu. Expressibility results for linear-time and branching-time logics. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *LNCS*, pages 428–437, 1988.

[CE00] K. Czarnecki and U. W. Eisenecker. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, 2000.

[CGP00] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, London, UK, January 2000.

[CHvRR04] L. Clement, A. Hately, C. von Riegen, and T. Rogers. Uddi version 3.0.2. Technical report, OASIS, October 2004.

[Cre02] J. W. Creswell. *Research design: Qualitative, quantitative, and mixed method approaches*. Sage Publications, Thousand Oaks, USA, 2nd edition, 2002.

[CSR08] T. Conz, S. Stein, and C. Reck. Datentransformation im rahmen der geschäftsprozessautomatisierung. In K.-P. Fähnrich, S. Kühne, and M. Thränert, editors, *Model-Driven Integration Engineering*, volume XI of *Leipziger Beiträge zur Informatik*, pages 111–122. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, September 2008.

[Daw00] C. W. Dawson. *The essence of computing projects: A student's guide*. Pearson Education Limited, Essex, UK, 2000.

[DCH+04] J. Domingue, L. Cabral, F. Hakimpour, D. Sell, and E. Motta. Irs iii: A platform and infrastructure for creating wsmo based semantic web services. In *Workshop on WSMO Implementations (WIW2004)*, Frankfurt, Germany, 2004.

[Dem82] W. E. Deming. *Out of the Crisis*. MIT Press, 1982.

[DFK+07] J. Domingue, A. Filipowska, D. Karastoyanova, S. Stein, D. Roman, and M. Zaremba. *Semantic BPM – The Integration of Business Process Management and Semantic Web Services*. Innsbruck, Austria, June 2007. Tutorial at 4th European Semantic Web Conference (ESWC).

[dFMM+08] D. de Francisco, I. Markovic, J. Martinez, H. Munoz, and N. Perez. Methodological extensions for semantic business process modeling. In *10th International Conference on Enterprise Information Systems (ICEIS)*, Barcelona, Spain, June 2008.

[DHW08] J. Drawehn, O. Höß, and A. Weisbecker. Prozessbasierte anforderungsanalyse im kontext von serviceorientierten architekturen (soa). In K.-P. Fähnrich, S. Kühne, and M. Thränert, editors, *Model-Driven Integration*

Engineering, volume XI of *Leipziger Beiträge zur Informatik*, pages 51–60. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, September 2008.

[DSKM07] M. Dimitrov, A. Simov, M. Konstantinov, and V. Momtchev. Wsmo studio – a semantic web services modelling environment for wsmo (system description). In *4th European Semantic Web Conference (ESWC)*, number 4519 in LNCS, pages 749–758, Innsbruck, Austria, 2007.

[DSSK07] M. Dimitrov, A. Simov, S. Stein, and M. Konstantinov. A BPMO based semantic business process modelling environment. In M. Hepp, K. Hinkelmann, D. Karagiannis, R. Klein, and N. Stojanovic, editors, *Workshop on Semantic Business Process and Product Lifecycle Management (SBPM)*, volume 251 of *CEUR Workshop Proceedings*, pages 101–104, Innsbruck, Austria, June 2007.

[DvdA04] J. Dehnert and W. M. P. van der Aalst. Bridging the gap between business models and workflow specifications. *International Journal of Cooperative Information Systems*, 13(3):289–332, 2004.

[EKAdMSvdA08] M. El Kharbili, A. K. Alves de Medeiros, S. Stein, and W. M. P. van der Aalst. Business process compliance checking: Current state and future challenges. In *Modellierung betrieblicher Informationssysteme (MobIS)*, volume 141 of *LNI*, pages 107–113, Saarbrücken, Germany, November 2008.

[EKS08] M. El Kharbili and S. Stein. Compliance management auf basis von semantischen richtlinien. In K.-P. Fähnrich, S. Kühne, and M. Thränert, editors, *Model-Driven Integration Engineering*, volume XI of *Leipziger Beiträge zur Informatik*, pages 253–262. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, September 2008.

[EKSMP08a] M. El Kharbili, S. Stein, I. Markovic, and E. Pulvermüller. Towards a framework for semantic business process compliance management. In *The Impact of Governance, Risk, and Compliance on Information Systems (GRCIS)*, volume 339 of *CEUR Workshop Proceedings*, pages 1–15, Montpellier, France, June 2008.

[EKSMP08b] M. El Kharbili, S. Stein, I. Markovic, and E. Pulvermüller. Towards policy-powered semantic enterprise compliance management – discussion paper. In *3rd International Workshop on Semantic Business Process Management (SBPM)*, CEUR Workshop Proceedings, Tenerife, Spain, June 2008.

[EKSP08] M. El Kharbili, S. Stein, and E. Pulvermüller. Policy-based semantic compliance checking for business process management. In *Gemischter Workshop zu Referenzmodellierung und semantische Geschäft-*

sprozessmodellierung, volume 420 of *CEUR Workshop Proceedings*, pages 178–192, Saarbrücken, Germany, November 2008.

[Erl05] T. Erl. *Service-Oriented Architectures: Concepts, Technology, and Design*. Prentice Hall, London, UK, September 2005.

[ESF+08] M. Evenson, B. Schreder, J. Frankowski, P. Rubach, and M. Skorczyk. Crm and billing prototypes, version 1.2. Technical Report Deliverable 9.2, SUPER Research Project, November 2008.

[FBS+08] A. Filipowska, S. Bhiri, S. Stein, B. Norton, and M. Dimitrov. *Semantic Business Process Management*. Milan, Italy, September 2008. Tutorial at 6th International Conference on Business Process Management.

[Fel98] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[FF08] S. Feja and D. Fötsch. Model checking with graphical validation rules. In *15th Annual IEEE Conference and Workshop on the Engineering of Computer Based Systems (ECBS)*, pages 117–125. IEEE CS, 2008.

[FFS08] S. Feja, D. Fötsch, and S. Stein. Grafische validierungsregeln am beispiel von epks. In *Workshop Modellgetriebene Softwarearchitektur – Evolution, Integration und Migration (MSEIM)*, LNI, München, Germany, February 2008. GI.

[FGM+99] R. T. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1. Technical Report 2616, W3 Consortium, June 1999.

[Fie00] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, USA, 2000.

[FK99] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a new computing infrastructure*. Morgan Kaufman, San Francisco, CA, USA, 1999.

[FKS08] A. Filipowska, M. Kaczmarek, and S. Stein. Semantically annotated epc within semantic business process management. In *Workshop on Advances in Semantics for Web Services (semantics4ws)*, Milan, Italy, September 2008.

[FKSW06] K.-P. Fähnrich, S. Kühne, A. Speck, and J. Wagner, editors. *OrViA – Integration betrieblicher Informationssysteme: Problemanalysen und Lösungsansätze des Model-Driven Integration Engineering*, volume IV of *Leipziger Beiträge zur Informatik*. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, 2006.

[FKT01] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications*, 15(3):200–223, 2001.

[FKT08] K.-P. Fähnrich, S. Kühne, and M. Thränert, editors. *Model-Driven Integration Engineering*, volume XI of *Leipziger Beiträge zur Informatik*. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, September 2008.

[FL07] J. Farrell and H. Lausen. Semantic annotations for wsdl. Technical report, W3 Consortium, 2007.

[FLP+06] D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, and J. Domingue. *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Springer, 2006.

[FNS05] S. Fuger, F. Najmi, and N. Stojanovic. ebxml registry information model version 3.0. Technical report, OASIS, May 2005.

[FP07] D. Fötsch and E. Pulvermüller. Constructing higher-level transformation languages based on xml. In *5th International Conference on Software Methodologies, Tools and Techniques (SOMET)*, Rome, Italy, November 2007. IOS Press.

[FRS07] J. Frankowski, P. Rubach, and E. Szczekocka. Collaborative ontology development in real telecom environment. In *1st International Working Conference on Business Process and Services Computing (BPSC)*, volume 116 of *LNI*, pages 40–53, Leipzig, Germany, September 2007.

[FS09] T. Feld and S. Stein. Soa - lessons from the battlefield. In *International Symposium on Service Science (ISSS)*, Leipzig, Germany, March 2009.

[FSH05] D. Fötsch, A. Speck, and P. Hänsgen. The operator hierarchy concept for xml document transformation technologies. In *3. Berliner XML-Tage 2005 (BXML05)*, pages 59–70, Berlin, Germany, 2005.

[GHM+03] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, and H. Frystyk. Soap version 1.2 part 1: Messaging framework. Technical report, W3 Consortium, June 2003.

[GPFLC04] A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering*. Springer, London, UK, 2nd edition, 2004.

[Gru93] T. R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993.

[GS04] J. Greenfield and K. Short. *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. Wiley, 2004.

[Ham90] M. Hammer. Reengineering work: Don't automate, obliterate. *Harvard Business Review*, pages 104–112, July 1990.

[HCM+05] A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. Wsmx – a semantic service-oriented architecture. In *International Conference on Web Service (ICWS)*, Orlando, Florida, USA, 2005.

[Hep07] M. Hepp. Possible ontologies: How reality constrains the development of relevant ontologies. *IEEE Internet Computing*, 11(1):90–96, 2007.

[HHV06] A. Hess, B. Humm, and M. Voß. Regeln für serviceorientierte architekturen hoher qualität. *Informatik Spektrum*, 29(6):395–411, 2006.

[HK04] R. Hauser and J. Koehler. Compiling process graphs into executable code. In *3rd International Conference on Generative Programming and Component Engineering (GPCE)*, volume 3286 of *LNCS*, pages 317–336, Vancouver, Canada, 2004.

[HLD+05] M. Hepp, F. Leymann, J. Domingue, A. Wahler, and D. 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)*, pages 535–540, Beijing, China, 2005.

[HMPR04] A. R. Hevner, S. T. March, J. Park, and S. Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, March 2004.

[HN05] H. R. Hansen and G. Neumann. *Wirtschaftsinformatik 1*. Utb, 9th, revised edition, 2005.

[HNS00] C. Hofmeister, R. Nord, and D. Soni. *Applied software architecture*. Addison-Wesley, Massachusetts, USA, 2nd edition, 2000.

[HR07] M. Hepp and D. Roman. An ontology framework for semantic business process management. In *8th International Conference Wirtschaftsinformatik*, pages 423–440, Karlsruhe, 2007.

[HS00] P. Herzum and O. Sims. *Business Component Factory*. Wiley, 2000.

[HV93] J. C. Henderson and N. Venkatraman. Strategic alignment: Leveraging information technology for transforming organizations. *IBM Systems Journal*, 32(1):472–484, 1993.

[HV04] R. Heckel and H. Voigt. Model-based development of executable business processes for web services. In *Lectures on Concurrency and Petri Nets*, volume 3098 of *LNCS*, pages 559–584, 2004.

[JM05] S. Jones and M. Morris. A methodology for service architectures. Technical report, Capgemini UK plc, London, UK, October 2005. submitted to OASIS.

[JMS06] M. B. Juric, B. Mathew, and P. Sarang. *Business Process Execution Language for Web Services*. PACKT Publishing, Birmingham, UK, 2nd edition, 2006.

[Jon06] S. Jones. *Enterprise SOA Adoption Strategies*. Lulu, 2006.

[KBS05] D. Krafzig, K. Banke, and D. Slama. *Enterprise SOA: Service Oriented Architecture Best Practices*. Prentice Hall, 2005.

[KHSW05] J. Koehler, R. Hauser, S. Sendall, and M. Wahler. Declarative techniques for model-driven business process integration. *IBM Systems Journal*, 44(1):47–65, 2005.

[Kin06] E. Kindler. On the semantics of epcs: Resolving the vicious circle. *Data Knowledge Engineering*, 56(1):23–40, 2006.

[KK02] D. Karagiannis and H. Kühn. Metamodelling platforms. In *3rd International Conference EC-Web*, volume 2455 of *LNCS*, pages 182–195, Aix-en-Provence, France, 2002.

[KKGL08] S. Kühne, H. Kern, V. Gruhn, and R. Laue. Business process modelling with continuous validation. In *1st International Workshop on Model-Driven Engineering for Business Process Management (MDE4BPM)*, Milan, Italy, September 2008.

[KNS92] G. Keller, M. Nüttgens, and A.-W. Scheer. Semantische prozessmodellierung auf der grundlage ereignisgesteuerter prozessketten (epk). Technical Report Heft 89, Universität des Saarlandes, Saarbrücken, Germany, 1992.

[KP06] K. Kritikos and D. Plexousakis. Semantic qos metric matching. In *4th European Conference on Web Services (ECOWS)*, pages 265–274, December 2006.

[KPP95] B. Kitchenham, L. Pickard, and S. L. Pfleeger. Case studies for method and tool evaluation. *IEEE Software*, 12(4):52–62, 1995.

[Kre01] H. Kreger. Web services conceptual architecture (wsca 1.0). Technical report, IBM, May 2001.

[KS04] G. Kotonya and I. Sommerville. *Requirements Engineering: Processes and Techniques*. Wiley, Chichester, UK, 2004.

[KSI08] S. Kühne, S. Stein, and K. Ivanov. Abbildung fachlicher prozessmodelle auf bpel-basierte laufzeitumgebungen. In K.-P. Fähnrich, S. Kühne, and M. Thränert, editors, *Model-Driven Integration Engineering*, volume XI of *Leipziger Beiträge zur Informatik*, pages 93–110. Eigenverlag Leipziger Informatik-Verbund (LIV), September 2008.

[KtHvdA03] B. Kiepuszewski, A. ter Hofstede, and W. M. P. van der Aalst. Fundamentals of control flow in workflows. *Acta Informatica*, 39(3):143–209, 2003.

[KTRL07] S. Kühne, M. Thränert, W. Rotzoll, and J. Lehmann. Model-driven integration engineering in der e-government-domäne meldewesen. In *8th International Conference Wirtschaftsinformatik*, pages 109–126, Karlsruhe, Germany, 2007.

[KTS05] S. Kühne, M. Thränert, and A. Speck. Towards a methodology for orchestration and validation of cooperative e-business components. In M. J. Rutherford, editor, *7th GPCE Young Researcher Workshop*, pages 29–34, 2005.

[KUL06] O. Kopp, T. Unger, and F. Leymann. Nautilus event-driven process chains: Syntax, semantics, and their mapping to bpel. In *5th GI Workshop on Event-Driven Process Chains (EPK)*, pages 85–104, Vienna, Austria, 2006.

[KvdHD06] N. Kokash, W.-J. van den Heuvel, and V. D'Andrea. Leveraging web services discovery with customizable hybrid matching. In *4th International Conference on Service-Oriented Computing (ICSOC)*, number 4294 in LNCS, pages 522–528, Berlin, Germany, 2006.

[Lau01] E. Lau. E-government: Analysis framework and methodology. Technical Report PUMA(2001)16/ANN/REV1, OECD, December 2001.

[Ley04] F. Leymann. The influence of web services on software: Potentials and tasks. In *34th Annual Meeting of the German Computer Society*, Ulm, Germany, September 20-24 2004. Springer.

[Ley08] F. Leymann. Cloud computing (keynote). In *11th International Conference on Business Information Systems (BIS)*, volume 7 of *LNBIP*, Innsbruck, Austria, May 2008. Springer.

[LRK06] J. Lehmann, W. Rotzoll, and S. Kühne. Analyse der e-government-domäne meldewesen am beispiel des dienstes "einfache melderegisterauskunft". In K.-P. Fähnrich, S. Kühne, A. Speck, and J. Wagner, editors, *OrViA – Integration betrieblicher Informationssysteme*, volume IV of *Leipziger Beiträge zur Informatik*, pages 20–30. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, 2006.

[Man03] K. Mantell. From uml to bpel: Model driven architecture in a web services world. Technical report, IBM developerWorks, September 2003.

[MB97] J. Mingers and J. Brocklesby. Multimethodology: Towards a framework for mixing methodologies. *Omega: International Journal of Management Science*, 25(5):489–509, October 1997.

[MBH+04] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. Owl-s: Semantic markup for web services. Technical report, World Wide Web Consortium (W3C), 2004.

[McM93] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, August 1993.

[MCVG05] T. Mens, K. Czarnecki, and P. Van Gorp. A taxonomy of model transformations. In J. Bezivin and R. Heckel, editors, *Language Engineering for Model-Driven Software Development*, number 04101 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2005.

[MGH07] H. Moertl, H.-J. Groß, and M. Herrmann. Prototypische implementierung eines bestellanforderungsprozesses nach den prinzipien einer service-orientierten architektur (soa) bei der daimler ag. Technical report, Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V. (BITKOM), Berlin, Germany, 2007.

[MI96] R. Mizoguchi and M. Ikeda. Towards ontology engineering. Technical Report AI-TR-96-1, I.S.I.R., Osaka University, 1996.

[MK08] I. Markovic and M. Kowalkiewicz. Linking business goals to process models in semantic business process modeling. In *12th IEEE International EDOC Conference*, Munich, Germany, September 2008.

[MLM+06] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, and R. Metz. Reference model for service oriented architecture 1.0. Technical report, OASIS, July 2006.

[MLZ06] J. Mendling, K. B. Lassen, and U. Zdun. Transformation strategies between block-oriented and graph-oriented process modelling languages. In F. Lehner, H. Nösekabel, and P. Kleinschmidt, editors, *Multikonferenz Wirtschaftsinformatik (MKWI)*, volume 2, pages 297–312, Passau, Germany, 2006. GITO-Verlag Berlin.

[MM03] J. Miller and J. Mukerji. Mda guide. Technical Report omg/2003-06-01, Object Management Group (OMG), June 2003. Version 1.0.1.

[MN04] J. Mendling and M. Nüttgens. Transformation of aris markup language to epml. In M. Nüttgens and F. J. Rump, editors, *3rd GI Workshop on Event-Driven Process Chains (EPK)*, pages 61–79, Luxembourg, Luxembourg, 2004.

[MN06] J. Mendling and M. Nüttgens. Epc markup language (epml) – an xml-based interchange format for event-driven process chains (epc). *International Journal Information Systems and e-Business Management (ISeB)*, 4(3):245–263, July 2006.

[MNN05] J. Mendling, G. Neumann, and M. Nüttgens. Towards workflow pattern support of event-driven process chains (epc). In M. Nüttgens and J. Mendling, editors, *2nd Workshop XML4BPM*, pages 23–38, Karlsruhe, Germany, 2005.

[MOD07] Ministry of defence architectural framework (modaf). Technical report, UK Ministry of Defence, April 2007.

[MSJL06] J. McGovern, O. Sims, A. Jain, and M. Little. *Enterprise Service Oriented Architectures*. Springer, Dordrecht, The Netherlands, 2006.

[Nor08] B. Norton. Ontology-based behavioural reasoning for business processes. In *Workshop on Advances in Semantics for Web Services (semantics4ws)*, Milan, Italy, September 2008.

[NR02] M. Nüttgens and F. J. Rump. Syntax und semantik ereignisgesteuerter prozessketten (epk). In *Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen (Promise2002)*, Potsdam, Germany, 2002.

[NS07] B. Norton and S. Stein. *Semantics Applied to Business Process Management*. Vienna, Austria, May 2007. Tutorial at 1st European Semantic Technology Conference (ESTC).

[NWvL07] J. Nitzsche, D. Wutke, and T. van Lessen. An ontology for executable business processes. In *Workshop on Semantic Business Process and Product Lifecycle Management (SBPM)*, volume 251 of *CEUR Workshop Proceedings*, pages 52–63, Innsbruck, Austria, June 2007.

[ODBtH06] C. Ouyang, M. Dumas, S. Breutel, and A. H. M. ter Hofstede. Translating standard process models to bpel. In *18th International Conference on Advanced Information Systems Engineering (CAiSE)*, Luxembourg, Luxembourg, June 2006.

[OEtH02] J. O'Sullivan, D. Edmond, and A. H. M. ter Hofstede. What's in a service? *Distributed and Parallel Databases*, 12(2-3):117–133, 2002.

[OMG06] OMG. Business process modeling notation (bpmn) specification. Technical report, Object Management Group (OMG), February 2006.

[OvdADtH06] C. Ouyang, W. M. P. van der Aalst, M. Dumas, and A. H. M. ter Hofstede. Translating bpmn to bpel. Technical Report BPM-06-02, BPM Center, Eindhoven, Netherlands, 2006.

[OWS+03] B. Oestereich, C. Weiss, C. Schröder, T. Weilkiens, and A. Lenhard. *Objektorientierte Geschäftsprozessmodellierung mit der UML*. dpunkt.verlag, Heidelberg, Germany, 2003.

[PBvL+08] C. Pedrinaci, C. Brelage, T. van Lessen, J. Domingue, D. Karastoyanova, and F. Leymann. Semantic business process management: Scaling up the management of business processes. In *2nd IEEE International Conference on Semantic Computing (ICSC)*, Santa Clara, CA, USA, August 2008.

[PDAdM08] C. Pedrinaci, J. Domingue, and A. K. Alves de Medeiros. A core ontology for business process analysis. In *5th European Semantic Web Conference (ESWC)*, volume 5021 of *LNCS*, pages 49–64, Tenerife, Spain, June 2008.

[Pey07] H. Peyret. The forrester wave: Enterprise architecture tools, q2. Technical report, Forrester, April 2007.

[PG03] M. P. Papazoglou and D. Georgakopoulos. Service-oriented computing. *Communications of the ACM*, 46(10):25–28, October 2003.

[PKPS02] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara. Semantic matching of web services capabilities. In *1st International Semantic Web Conference (ISWC)*, number 2342 in LNCS, Sardinia, Italy, June 2002.

[Pop34] K. Popper. *Logik der Forschung*. Mohr Siebeck, Tübingen, Germany, 11th, revised edition, 1934.

[PRS04] J.-H. Pfeiffer, W. R. Rossak, and A. Speck. Applying model checking to workflow verification. In *11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS)*, pages 144–151, Washington, DC, USA, May 2004. IEEE CS.

[PSS07] C. Pedrinaci, M. Stollberg, and S. Stein. *Semantic Web Service for Business Process Management*. Mauritius, May 2007. Tutorial at 2nd International Conference on Internet and Web Applications and Services (ICIW).

[PvdH06] M. P. Papazoglou and W.-J. van den Heuvel. Service-oriented design and development methodology. *International Journal of Web Engineering and Technology (IJWET)*, 2(4), 2006.

[Ram06] V. Ramasamy. Syntactical & semantical web services discovery and composition. In *8th IEEE International Conference on E-Commerce Technology and 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE)*, 2006.

[Ric07] J. Ricken. Top-down modeling methodology for model-driven soa construction. In *On the Move to Meaningful Internet Systems*, volume 4805 of *LNCS*, pages 323–332, November 2007.

[RLB07] S. Roser, F. Lautenbacher, and B. Bauer. Generation of workflow code from dsms. In J. Sprinkle, J. Gray, M. Rossi, and J.-P. Tolvanen, editors, *7th OOPSLA Workshop on Domain-Specific Modeling (DSM2007)*, number TR-38 in Computer Science and Information System Reports, pages 149–159, Jyväskylä, Finland, 2007. Jyväskylä University Printing House.

[Rob04] C. Roberts. *The Dissertation Journey: A Practical and Comprehensive Guide to Planning, Writing, and Defending Your Dissertation*. Corwin Press, 2004.

[RP08] J. Ricken and M. Petit. Characterization of methods for process-oriented engineering of soa. In *2nd International Workshop on Collaborative Business Processes (CBP)*, Milan, Italy, September 2008.

[SBEK07] S. Stein, K. Barchewitz, and M. El Kharbili. Enabling business experts to discover web services for business process automation. In C. Pautasso and T. Gschwind, editors, *2nd Workshop on Emerging Web Services Technology*, pages 19–35, Halle, Germany, November 2007.

[Sch99] A.-W. Scheer. *ARIS – Business Process Frameworks*. Springer, Berlin, Germany, 3rd edition, 1999.

[Sch02] A.-W. Scheer. *ARIS – Vom Geschäftsprozeß zum Anwendungssystem*. Springer, Berlin, Germany, 4th edition, 2002.

[Sch07] O. Schmelzle. Transformation von annotierten geschäftsprozessen nach bpel. Master's thesis, Gottfried Wilhelm Leibniz Universität Hannover, Hannover, Germany, 2007.

[SDH06] T. Specht, J. Drawehn, and O. Höß. Domänspezifische modellierung von e-government-szenarien. In K.-P. Fähnrich, S. Kühne, A. Speck, and J. Wagner, editors, *OrViA – Integration betrieblicher Informationssysteme*, volume IV of *Leipziger Beiträge zur Informatik*, pages 80–93. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, 2006.

[SDS+09] K. Schneider, J. Dörr, S. Stein, S. Adam, and D. Lübke, editors. *Workshop on Requirements Engineering und Business Process Management - Konvergenz, Synonym oder doch so wie gehabt? (REBPM)*, Kaiserslautern, Germany, March 2009.

[SDTK05] T. Specht, J. Drawehn, M. Thränert, and S. Kühne. Modeling cooperative business processes and transformation to a service oriented architecture. In *7th International IEEE Conference on E-Commerce Technology*, 2005.

[SF03] H. Smith and P. Fingar. *Business Process Management: The Third Wave*. Meghan-Kiffer Press, Tampa, FL, USA, 1st edition, 2003.

[SGS04] D. Skogan, R. Grønmo, and I. Solheim. Web service composition in uml. In *8th IEEE International Enterprise Distributed Object Computing Conference (EDOC)*, 2004.

[SH05] P. Stahlknecht and U. Hasenkamp. *Einführung in die Wirtschaftsinformatik*. Springer, Berlin, Germany, 11th edition, 2005.

[SH09] S. Stein and Erik Hagen. Adopting bpmn with aris. Technical report, ARIS Expert Paper, Saarbrücken, Germany, January 2009.

[SI07a] S. Stein and K. Ivanov. Epk nach bpel transformation als voraussetzung für praktische umsetzung einer soa. In W.-G. Bleek, J. Raasch, and H. Züllighoven, editors, *Software Engineering 2007*, volume 105 of *LNI*, pages 75–80, Hamburg, Germany, March 2007. GI.

[SI07b] S. Stein and K. Ivanov. Fachliche beschreibung von services. In G. Starke and S. Tilkov, editors, *SOA-Expertenwissen - Praxis, Methoden und Konzepte serviceorientierter Architekturen*, pages 215–229. dpunkt.verlag GmbH, Heidelberg, Germany, 2007.

[SI07c] S. Stein and K. Ivanov. Fachlicher serviceentwurf im rahmen einer enterprise architecture. In S. Tilkov, editor, *Service-orientierte Architekturen (SOA)*. Euroforum Verlag, Düsseldorf, Germany, April 2007.

[SI07d] S. Stein and K. Ivanov. Vorgehensmodell zur entwicklung von geschäftsservicen. In K.-P. Fähnrich and M. Thränert, editors, *Integration Engineering – Motivation, Begriffe, Methoden und Anwendungsfälle*, volume VI of *Leipziger Beiträge zur Informatik*. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, 2007.

[SKD+08] S. Stein, S. Kühne, J. Drawehn, S. Feja, and W. Rotzoll. Evaluation of orvia framework for model-driven soa implementations: An industrial case study. In *6th International Conference on Business Process Management (BPM)*, volume 5240 of *LNCS*, pages 310–325, Milan, Italy, September 2008. Springer.

[SKI08] S. Stein, S. Kühne, and K. Ivanov. Business to it transformations revisited. In *1st International Workshop on Model-Driven Engineering for Business Process Management (MDE4BPM)*, Milan, Italy, September 2008.

[SKJK06] A.-W. Scheer, H. Kruppke, W. Jost, and H. Kindermann, editors. *Agilität durch ARIS Geschäftsprozessmanagement*. Springer, Berlin, Germany, 2006.

[SKSD08] A. Speck, S. Kühne, S. Stein, and J. Drawehn. Das projekt orvia – orchestrierung und validierung integrierter anwendungssysteme. In K.-P. Fähnrich, S. Kühne, and M. Thränert, editors, *Model-Driven Integration Engineering*, volume XI of *Leipziger Beiträge zur Informatik*, pages 35–47. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, September 2008.

[SKW06a] S. Stein, S. Kühne, and J. Wagner. Das forschungsprojekt orvia - orchestrierung und validierung integrierter anwendungssysteme. In K.-P. Fähnrich, S. Kühne, A. Speck, and J. Wagner, editors, *OrViA – Integration betrieblicher Informationssysteme*, volume IV of *Leipziger Beiträge zur Informatik*, pages 3–12. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, 2006.

[SKW06b] S. Stein, S. Kühne, and J. Wagner. Orvia - orchestrierung und validierung integrierter anwendungssysteme. In *BMBF Statuskonferenz Forschungsoffensive "Software Engineering 2006"*, Leipzig, Germany, 2006.

[SLEK09] S. Stein, Y. Lauer, and M. El Kharbili. Using template analysis as background reading technique for requirements elicitation. In *GI Software Engineering*, volume 143 of *LNI*, pages 127–138, Kaiserslautern, Germany, March 2009. GI.

[SLI08a] S. Stein, J. Lauer, and K. Ivanov. Aris method extension for business-driven soa. *Wirtschaftsinformatik*, 50(6):436–444, December 2008.

[SLI08b] S. Stein, J. Lauer, and K. Ivanov. Systematik einer fachlichen servicebeschreibung. In K.-P. Fähnrich, S. Kühne, and M. Thränert, editors, *Model-Driven Integration Engineering*, volume XI of *Leipziger Beiträge zur Informatik*, pages 61–65. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, September 2008.

[SN96] R. W. Schulte and Yefim V. Natis. Service oriented architectures, part 1. Technical report, Gartner, April 1996.

[SR08] S. Stein and U. Roediger. Fachliche servicebeschreibung am beispiel eines virtuellen autokonzerns. In T. Wieland, editor, *Service-orientierte Architekturen (SOA)*. OBJEKTspektrum, April 2008.

[SS08a] H. J. Schmelzer and W. Sesselmann. *Geschäftsprozessmanagement in der Praxis*. Carl Hanser Verlag, München, Germany, 6th revised edition, 2008.

[SS08b] S. Stein and C. Stamber. Semantic business process management. In D. Kuropka, P. Tröger, S. Staab, and M. Weske, editors, *Semantic Service Provisioning*, pages 127–143. Springer, Berlin, Germany, April 2008.

[SSEK08a] C. Stamber, S. Stein, and M. El Kharbili. Prototypical implementation of a pragmatic approach to semantic web service discovery during process execution. In W. Abramowicz and D. Fensel, editors, *11th International Conference on Business Information Systems (BIS)*, volume 7 of *LNBIP*, pages 201–212, Innsbruck, Austria, May 2008. Springer.

[SSEK08b] S. Stein, C. Stamber, and M. El Kharbili. Aris for semantic business process management. In *Workshop on Advances in Semantics for Web Services (semantics4ws)*, Milan, Italy, September 2008.

[SSEKR08] S. Stein, C. Stamber, M. El Kharbili, and P. Rubach. Semantic business process management: An empirical case study. In *Gemischter Workshop zu Referenzmodellierung und semantische Geschäftsprozessmodellierung*, volume 420 of *CEUR Workshop Proceedings*, pages 165–177, Saarbrücken, Germany, November 2008.

[SSEKR09] S. Stein, C. Stamber, M. El Kharbili, and P. Rubach. Semantic business process management: A case study. In G. Mentzas and A. Friesen, editors, *Semantic Enterprise Application Integration for Business Processes: Service-Oriented Frameworks*. Idea Group Reference, September 2009.

[ST07] G. Starke and S. Tilkov, editors. *SOA-Expertenwissen – Praxis, Methoden und Konzepte serviceorientierter Architekturen*. dpunkt.verlag GmbH, Heidelberg, Germany, 2007.

[STA05] A.-W. Scheer, O. Thomas, and O. Adam. Process modelling using event-driven process chains. In M. Dumas, W. M. P. van der Aalst, and A. H. M. ter Hofstede, editors, *Process-Aware Information Systems*, pages 119–146. Wiley, Hoboken, New Jersey, USA, 2005.

[Sta08] C. Stamber. Transformation of business processes onto soa platforms by means of semantic technologies, 2008. Diploma Thesis, Technische Universität Kaiserslautern, Germany.

[Ste06] S. Stein. Umsetzung des orvia-frameworks mit aris. In K.-P. Fähnrich, S. Kühne, A. Speck, and J. Wagner, editors, *OrViA – Integration betrieblicher Informationssysteme*, volume IV of *Leipziger Beiträge zur Informatik*, pages 73–79. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, 2006.

[Ste07a] S. Stein. Vom geschäftsprozess zum ausführbaren system. In D. Spath, A. Weisbecker, O. Höß, and J. Drawehn, editors, *Stuttgarter Softwaretechnik Forum 2007*, pages 73–81, Stuttgart, Germany, November 2007.

[Ste07b] S. Stein. *Von EPK nach BPEL – Technische Einblicke in ARIS SOA Architect*. Saarbrücken, Germany, January 2007. ARIS Online Tutorial.

[Ste08a] S. Stein. *ARIS SOA Architect in der Praxis*. Saarbrücken, Germany, February 2008. ARIS Webcasts.

[Ste08b] S. Stein. Business-oriented service description in a virtual automotive group. Technical report, ARIS Expert Paper, Saarbrücken, Germany, May 2008.

[Ste08c] S. Stein. Fachliche servicebeschreibung im rahmen des service engineerings. *Information Management & Consulting (IM)*, 23(3):36–41, 2008.

[Ste08d] S. Stein. *SOA Management mit ARIS*. Saarbrücken, Germany, April 2008. ARIS Webcasts.

[SUP07] Business process ontology framework, version 1.0. Technical Report Deliverable 1.1, SUPER Research Project, May 2007.

[Szy97] C. Szyperski. *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley, 1997.

[TB51] E. L. Trist and K. W. Bamforth. Some social and psychological consequences of the longwall method of coal-getting: An examination of the psychological situation and defences of a work group in relation to the social structure and technological content of the work system. *Human Relations*, 4(1):3–38, 1951.

[Teb06] J. Teboul. *Service Is Front Stage: Positioning Services for Value Advantage*. Palgrave Macmillan, New York, USA, 2006.

[Thr05] M. Thränert. Integration – eine begriffsbestimmung. In K.-P. Fähnrich, M. Thränert, and P. Wetzel, editors, *Umsetzung von kooperativen Geschäftsprozessen auf eine internetbasierte IT-Struktur*, volume III of *Leipziger Beiträge zur Informatik*, pages 11–22. Eigenverlag Leipziger Informatik-Verbund (LIV), Leipzig, Germany, 2005.

[Thr08] M. Thränert. *Integration-Engineering: Grundlagen, Vorgehen und Fallstudien*. PhD thesis, Universität Leipzig, Leipzig, Germany, 2008.

[TOG07] *TOGAF 2007 Edition (Incorporating 8.1.1)*. Van Haaren Publishing, October 2007. 978-9087530945.

[Tri63] E. L. Trist. *Organizational choice: Capabilities of groups at the coal face underr changing technoligies: the loss, rediscovery & transformation of a work tradition*. Tavistock Publications, London, UK, 1963.

[UKMZ98] M. Uschold, M. King, S. Moralee, and Y. Zorgios. The enterprise ontology. *The Knowledge Engineering Review*, 13, 1998.

[vB76] L. von Bertalanffy. *General System Theory*. Braziller Inc., New York, USA, 1976.

[vdA99] W. M. P. van der Aalst. Formalization and verification of event-driven process chains. *Information and Software Technology*, 41(10):639–650, 1999.

[vdAL05] W. M. P. van der Aalst and K. B. Lassen. Translating workflow nets to bpel4ws. Technical Report BPM-05-16, BPM Center, Eindhoven, Netherlands, 2005.

[vdAP06] W. M. P. van der Aalst and M. Pesic. Decserflow: Towards a truly declarative service flow language. Technical Report BPM-06-21, BPM Center, Eindhoven, Netherlands, 2006.

[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, July 2003.

[Vin08] S. Vinoski. Convenience over correctness. *IEEE Internet Computing*, 12(4):89–92, 2008.

[vLND+07] T. van Lessen, J. Nitzsche, M. Dimitrov, M. Konstantinov, D. Karastoyanova, and L. Cekov. An execution engine for semantic business processes. In *2nd International Workshop on Business Oriented Aspects concerning Semantics and Methodologies in Serviceoriented Computing (SeMSoC)*, 2007.

[Weh07] J. Wehler. Boolean and free-choice semantics of event-driven process chains. In *EPK 2007 – Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten*, volume 303 of *CEUR Workshop Proceedings*, 2007.

[Wei99] G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.

[Wes07] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, Berlin, Germany, 2007.

[WH07] T. Wilde and T. Hess. Forschungsmethoden der wirtschaftsinformatik – eine empirische untersuchung. *Wirtschaftsinformatik*, 49(4):280–287, 2007.

[Whi05] S. A. White. Using bpmn to model a bpel process. Technical report, BPTrends, March 2005.

[WHMN07] I. Weber, J. Hoffmann, J. Mendling, and J. Nitzsche. Towards a methodology for semantic business process modeling and configuration. In *2nd*

International Workshop on Business Oriented Aspects concerning Semantics and Methodologies in Service-oriented Computing (SeMSoC), 2007.

[WMF+07] B. Wetzstein, Z. Ma, A. Filipowska, M. Kaczmarek, S. Bhiri, S. Losada, J.-M. Lopez-Cob, and L. Cicurel. Semantic business process management: A lifecycle based requirements analysis. In *Workshop on Semantic Business Process and Product Lifecycle Management (SBPM)*, volume 251 of *CEUR Workshop Proceedings*, pages 1–11, Innsbruck, Austria, June 2007.

[WS03] Y. Wang and E. Stroulia. Flexible interface matching for web-service discovery. In *4th International Conference on Web Information Systems Engineering (WISE)*, pages 147–156, 2003.

[WSD01] Web service description language (wsdl) 1.1. Technical report, W3 Consortium, March 2001.

[WvdADtH02] P. Wohed, W. M. P. van der Aalst, M. Dumas, and A. H. M. ter Hofstede. Pattern-based analysis of bpel4ws. Technical Report FIT-TR-2002-04, Queensland University of Technology, Brisbane, Australia, 2002.

[XPD08] Process definition interface – xml process definition language (xpdl). Technical report, Workflow Management Coalition (WfMC), March 2008. version 2.1.

[Yin03] R. K. Yin. *Case Study Research: Design and Methods*, volume 5 of *Applied Social Research Methods Series*. Sage Publications, London, UK, 3rd edition, 2003.

[YZZ+07] X. Yu, Y. Zhang, T. Zhang, L. Wang, J. Zhao, G. Zheng, and X. Li. Towards a model driven approach to automatic bpel generation. In *3rd European Conference on Model Driven Architecture-Foundations and Applications (ECMDA)*, volume 4530 of *LNCS*, pages 204–218, Haifa, Israel, June 2007.

[ZD06] U. Zdun and S. Dustdar. Model-driven and pattern-based integration of process-driven soa models. In *The Role of Business Processes in Service Oriented Architectures*, number 06291 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2006.

[ZM05] J. Ziemann and J. Mendling. Epc-based modelling of bpel processes: a pragmatic transformation approach. In *MITIP2005*, Italy, 2005.

[ZMJ05] Z. Zhuang, P. Mitra, and A. Jaiswal. Corpus-based web services matchmaking. In *Workshop on Exploring Planning and Scheduling for Web Services, Grid and Autonomic Computing*, Pittsburgh, PA, USA, July 2005.

List of Figures

List of Tables

Glossary

AML	**A**RIS **M**arkup **L**anguage,
ARIS	**Ar**chitektur integrierter **I**nformations**s**ysteme (engl.: Architecture of Integrated Information Systems),
AVE	**A**RIS **V**alue **E**ngineering,
BPEL	**B**usiness **P**rocess **E**xecution **L**anguage,
BPM	**B**usiness **P**rocess **M**anagement,
BPMN	**B**usiness **P**rocess **M**odeling **N**otation,
BPMO	**B**usiness **P**rocess **M**odelling **O**ntology,
BRO	**B**ehavioural **R**easoning **O**ntology,
CASE	**C**omputer-**A**ided **S**oftware **E**ngineering,
CBDI-SAE	**CBDI S**oftware **A**rchitecture and **E**ngineering,
CIM	**C**omputation **I**ndependent **M**odel,
CTL	**C**omputation **T**ree **L**ogic,
DVZ M-V	**D**aten**v**erarbeitungs**z**entrum **M**ecklenburg-**V**orpommern,
EARR	**E**lectronic **A**ccess to **R**egister of **R**esidents,
ebXML	**E**lectronic **B**usiness using **XML**,
EPC	**E**vent-Driven **P**rocess **C**hain,
EPML	**EP**C **M**arkup **L**anguage,
ESB	**E**nterprise **S**ervice **B**us,
FTP	**F**ile **T**ransfer **P**rotocol,
G-CTL	**G**raphical-**CTL**,
IRS III	**I**nternet **R**easoning **S**ervice **III**,
IT	**I**nformation **T**echnology,
KPI	**K**ey **P**erformance **I**ndicator,
MDA	**M**odel **D**riven **A**rchitecture,
MODAF	UK **M**inistry **o**f **D**efence **A**rchitectural **F**ramework,
MXML	**M**ining **XML**,
OASIS	**O**rganization for the **A**dvancement of **S**tructured **In**formation,
OMG	**O**bject **M**anagement **G**roup,

OrViA ..... **Or**chestrierung und **V**alidierung **i**ntegrierter **A**nwendungssysteme (engl.: Orchestration and Validation of Integrated Application Systems),
PIM ..... **P**latform **I**ndependent **M**odel,
PSM ..... **P**latform **S**pecific **M**odel,
QoS ..... **Q**uality **o**f **S**ervice,
REST ..... **Re**presentational **S**tate **T**ransfer,
sBPEL ..... **s**emantic **BPEL**,
sBPMN ..... **s**emantic **BPMN**,
sEPC ..... **s**emantic **EPC**,
SESE ..... **S**ingle **E**ntry **S**ingle **E**xit,
SISi ..... **S**emantic **I**nvocation **S**erv**i**ce,
SLA ..... **S**ervice **L**evel **A**greement,
SMTP ..... **S**imple **M**ail **T**ransfer **P**rotocol,
SMV ..... **S**ymbolic **M**odel **V**erifier,
sMXML ..... **s**emantic **MXML**,
SOA ..... **S**ervice-**O**riented **A**rchitecture,
SUPER ..... **S**emantics **U**tilised for **P**rocess Management Within and Between **E**nte**r**prises,
TOGAF ..... **T**he **O**pen **G**roup **A**rchitecture **F**ramework,
TP ..... **T**elekomunikacja **P**olska,
UDDI ..... **U**niversal **D**escription, **D**iscovery and **I**ntegration,
UML ..... **U**nified **M**odeling **L**anguage,
UPO ..... **U**pper **P**rocess **O**ntology,
W3C ..... **W**orld **W**ide **W**eb **C**onsortium,
WSDL ..... **W**eb **S**ervice **D**escription **L**anguage,
WSFL ..... **W**eb **S**ervices **F**low **L**anguage,
WSML ..... **W**eb **S**ervice **M**odeling **L**anguage,
WSMO ..... **W**eb **S**ervice **M**odeling **O**ntology,
WSMX ..... **W**eb **S**ervice **M**odelling E**x**ecution Environment,
XML ..... E**x**tensible **M**arkup **L**anguage,
XPDL ..... **X**ML **P**rocess **D**efinition **L**anguage,
XSD ..... **X**ML **S**chema **D**efinition,
XSLT ..... E**x**tensible **S**tylesheet **L**anguage **T**ransformation,