SEMANTIC PROCESS MODELS

TRANSFORMATION, ADAPTATION, RESOURCE CONSIDERATION

Thomas Eisenbarth

Dissertation

for the degree of Doctor of Natural Sciences (Dr. rer. nat.)

University of Augsburg
Department of Computer Science
Software Methodologies for Distributed Systems

February 2013

First examiner: Prof. Dr. Bernhard Bauer
Examiner: Prof. Dr. Wolfgang Reif

Day of defense: 15.04.2013

Abstract

Use of information technology has constantly been growing over the past decades and has become an integral component in most enterprises today.

A particular technique, process modeling, aims — besides advantages such as documentation purposes — to increase quality and reproducibility of workflows such as business processes. From the intense use of this technique however, other problems emerge. As the amount of models grows and cohesion with other enterprise objects intensifies, conventional process modeling techniques expose some limitations caused by hidden interconnections between models and enterprise objects.

Additionally, the fact that specific resources are required for a successful implementation of a particular process is often neglected, because matching of required and existing resources is costly, and therefore rarely carried out. This information gap between *design* and *execution* can result in processes, that are impractical to implement.

Thus, the main objectives of this thesis are, on the one hand, to raise insights into the connections between process models and arbitrary enterprise resources. On the other hand, this thesis presents an approach to assess the feasibility of process models in consideration of process requirements. The presented work is evaluated in prototypical implementations as well as real world use cases and application scenarios.

Zusammenfassung

Die Nutzung von Informationstechnologie hat in der Vergangenheit stets zugenommen und wurde so zu einem integralen Bestandteil in vielen Unternehmen.

Die spezielle Technik der Prozessmodellierung wurde — nebst Vorteilen wie zu Dokumentationszwecken — hauptsächlich zur Steigerung der Qualität und Reproduzierbarkeit von Geschäftsabläufen eingeführt. Bei intensiver Nutzung dieser Technik zeigten sich jedoch Probleme. Gerade bei einer großen Anzahl von Prozessmodellen und mitunter nicht erkennbarer Verflechtung mit anderen Unternehmensbestandteilen werden Probleme mit herkömmlichen Prozessmodellierungstechniken und damit einhergehenden neue Herausforderungen erkennbar. Darüber hinaus wird häufig vernachlässigt, dass zur Ausführung der modellierten Prozesse bestimmte Ressourcen notwendig sind, weil kein Abgleich zwischen erforderlichen und tatsächlich zur Verfügung stehenden Ressourcen durchgeführt wird. Diese Informationslücke zwischen der *Modellierung* und der *Ausführung* führt zu potentiell nicht ausführbaren Prozessen.

In dieser Arbeit wird ein Ansatz vorgestellt, um zum einen die Verbindungen zwischen Prozessmodellen und beliebigen Unternehmensressourcen abzubilden. Dies dient unter anderem dazu, auf einer zentralen Wissensbasis Abhängigkeiten einfacher erkennen zu können. Zum anderen wird ein Ansatz vorgestellt, um die tatsächliche Ausführbarkeit von Prozessmodellen bereits zur Modellierungszeit beurteilen zu können, indem Ressourcenanforderungen geprüft werden. Der präsentierte Ansatz wird in prototypischen Implementierungen und Anwendungsszenarien evaluiert.

Acknowledgments

First and foremost I would like to thank my supervisor, Prof. Dr. Bernhard Bauer, for giving me the opportunity to conduct my research at the SMDS lab and his constant support and encouragement during the writing of this thesis. His comments and suggestions were a valuable input for me.

Special thanks to Prof. Dr. Wolfgang Reif, who also accepted to be an advisor for my thesis.

I am very grateful to the colleagues at the SMDS lab for a great atmosphere, many interesting discussions and lots of fun. I'd also like to thank all students that assisted me in a multitude of experiments and implementations.

Special thanks go to my parents Anita and Werner for their education and support during my schooldays and studies and to my brother Christian for all the support and amusement.

Last but not least I would like to especially thank my partner Alexandra for her understanding for my work and her enduring patience and support throughout the writing of this thesis.

Contents

Chapter 1

Introduction

In today's modern business world global competition and increasing customer expectations require that enterprises adapt to changing markets and other environmental changes frequently — and even more important — *quickly*. This in turn requires enterprises to define and continuously optimize organizational workflows. This is often achieved by defining operational sequences in form of graphical process models and make use of automated, Information Technology (IT)-based support whenever possible.

This lead to the widely known research areas around Business Process Management (BPM). BPM is an accepted method to encounter the challenges and problems described before and is increasingly popular within industry today. Besides modeling of so called *business* processes, such formally defined models are used in, e.g., hardware and software engineering to define phases, dependencies and requirements during the respective phases of hardware or software engineering, too.

Application of such process models strive for several goals such as

- simulate modeled processes before execution,
- complete iterating tasks (being defined in process models) in a reproducible way,
- minimize qualitative variation when executing processes,
- better planning and integration of IT support,
- support redesign of processes.

As mentioned before, "process modeling" by meaning modeling both *business* as well as *hardware or software* engineering processes, have both large as well as active research communities. Hence, there is plenty of literature on basic as well as advanced topics of process modeling available (van der Aalst et al., 2003b; van der Aalst and ter Hofstede, 2005; Papazoglou, 2003, 2008; Weske, 2007; Lautenbacher, 2010).

As of today, IT supports process modeling with a large amount of diverse tools ready for production use. Though, it is easy to lose track due to the ever-growing number of IT systems, processes and resources to handle within enterprises. For example, due to the large number of process models and the consequential problems of, e.g., duplicated models, approaches to detect clones and similar parts within process models were analyzed (van Dongen et al., 2008; Dijkman et al., 2011; Uba et al., 2011).

Another open challenge is the combination of process models with resources such as roles or even more specific requirements that include capabilities, skills and qualifications. Although a magnitude of resources, such as employees with specific skills, machines or IT services have to be available within enterprises when a process is executed, this is often neglected in process modeling today. Many of today's popular graphical modeling languages enable consideration of resources at a quite abstract level only. Often, resources are considered as human roles solely, neglecting the fact, that usually lots of other resources, such as manufacturing machines or IT systems in the case of *business* processes, climate-testing laboratories and testing equipment in the case of *hardware engineering* processes, or specific Unified Modeling Language (UML) tools and employees with specialized skills in the domain of *software engineering* processes are required.

Due to the absence of a central, intelligent knowledge base connecting all this information, it is difficult, if not impossible, to see relations between the involved elements today. As a result, interdependencies are hidden, or required resources unavailable at execution time. This information gap between a process *model* and its *realization* reduces the advantages of process modeling.

1.1 Challenges

Although business and engineering process modeling in general pursues the objective of improving process quality, this goal becomes more and more dif-

ficult, whenever a large number of diverging information has to be handled. (Bandara et al., 2006) names many issues in BPM seen by an organizational perspective but also mentions problems such as "structural differences" due to different technologies used at enterprises as well as "semantic differences". In the research domain covered in this thesis, this means process **models**, **stakeholders**, **IT services** or **resources** which are involved. Often a single knowledge base for all information is missing because of grown, heterogeneous IT systems, data silos in particular departments, or simply because units do not even know about the demand for information in other units. Aggregation of information that spans multiple of the mentioned elements is hard — if possible at all. In summary, there are several open challenges:

❶ **Models and model repositories constantly grow and get connected with further enterprise objects** Today, process models get more and more connected with arbitrary other enterprise objects such as IT services. These interconnections are usually hidden on the abstract modeling layer which is why analysis of that data is not possible.
Interconnections between process models and arbitrary enterprise objects are often implemented on low level programming code but unknown for analysts on more abstract levels within process modeling space. Thereby, it is impossible to find interdependencies between processes, and other involved business objects. Therefore, large process model repositories that are coupled with, e.g., IT services are difficult to handle.

❷ **Linking of resource requirements with process models is not possible** The definition and modeling of fine-grained resource requirements within process models is hardly possible today.

Today, popular graphical modeling languages do not support the definition of detailed resource requirements on an explicit level to enable the linkage of the requirements with a resource knowledge base. One possibility to describe that a set of tasks within a process model needs attention from someone with specific skills is by using so-called swim-lanes or roles. Though, this is not sufficient in terms of level of detail and leads to easily confusing models.

❸ **Process models are out of synchronization with existing resources** Process models used for business or engineering process modeling always require some resources to be executed. Without the respective resources, the

models cannot be executed successfully. There is hardly any possibility to check a set of existing process models for executability.

At runtime or even after execution of a process, it is obviously known, for example which person accomplished a certain task and is therefore required for the realization of a process. Though, currently, this information is not considered at design-time when process models are built, although this information is necessary to decide whether required resources are present at an enterprise, or must be obtained first. Otherwise, the designed model might be useless because it is not realizable.

Process models define which tasks are to be completed in which order but frequently get out of synchronization with existing resources within enterprises, because, e.g., people leave the company or software licenses expire.

❹ **Critical resources affect the criticality of processes** Resources and commodities in industrial processes have an enormous influence on the classification of criticality of the processes itself. As there is no solution for the description of resource requirements within process models as described in ❷, this classification is not possible, too.

The classification of criticality of commodities and especially non-renewable raw materials is a complex and time-consuming task. As the usage of such critical materials within processes lead to critical processes in turn, it is necessary to automatically

1. decide about the criticality of resources and
2. use this information to determine criticality of process models afterwards.

❺ **Modeling processes is labor-intensive and error-prone while the models need frequent adaptation** Designing process models turned out to be a labor-intensive, error-prone task. Additionally, process models have to be aligned to changing environments frequently because of, e.g., new regulatory measures, modifications of law, changes of available resources or simply because the demand of customers or suppliers requires changes within processes. Thus, process models need frequent adaptation. Therefore, an automated approach to design and adapt process models that is capable of, e.g., considering resource requirements is desirable. As automated planning

usually returns multiple possibilities, a way to quantify and sort these results to support the selection is also necessary.

The next section concludes the objectives this thesis should aim for to encounter aforementioned challenges.

1.2 Objectives

As described in the section before, process modeling aims to achieve benefits by structuring workflows in a formal way. Though, challenges remain when it comes to modeling large or lots of models or frequently adapting existing process models. Additionally, there are complex combinations and connections of processes with IT landscape such as services. Additionally, handling many resources that are necessary for enacting processes requires further research and new approaches. Another unsolved issue is consideration of resource constraints, that have to be matched with the resources existing in reality.

Thus, the main objectives of this thesis are the following.

① **Reference ontology and automated transformation method** In order to encounter challenges ❶ and ❷, the first objective is to develop a dense reference ontology for process models, including resource constraints using an up to date, well-known and standardized language on top of computer-readable, formal logics.

The ontology should include both the possibility to describe resources and respective constraints, as well as support modeling control flows within process models.

Furthermore, the goal of this thesis is to describe how existing models can be mapped into this reference ontology. This transformation should be achieved automatically by providing a formal transformation specification for such existing process models based on well-known graphical notation languages.

Additionally, the wider objective is to enable integration of such process models into existing semantic knowledge bases and to demonstrate benefits of such integration.

As a result of this mapping, we aim for enabling analysts with the possibility to discover interdependencies between processes and further enterprise

objects. This objective is encountered by querying techniques on top of the reference ontology. Thus, the objective is to discover a way to map and query process models, e.g., regarding utilized IT services.

② **Integration of process models with resource knowledge base and consideration of resource requirements** As lots of work has been done in the area of enterprise modeling in general, this thesis aims to make use of this work and consider existing approaches to model resources with ontologies. The objective is to show integration possibilities of process modeling with resource knowledge bases (we will describe and define this term in more detail later). The goal is to define resource constraints within process modeling space, and accomplish consistency checking with regard to the resource requirements of models at design-time. Furthermore, we aim at valuating models using resource properties, such as costs to compare different model alternatives. Therefore, this objective encounters the challenges described in ❷ and ❸.

③ **Resource (commodity) classification and integration into process space** As stated before, the goal is to model resources in a computer processable way using a formal description language. On top of this descriptions, the objective is to identify existing classification properties for commodities, and find a way to describe those properties within the formal description language together with commodities. Using those rules, classification of resources regarding criticality should be possible and therefore encounters challenge ❹.

④ **Automated model adaptation considering resources** The next objective of this thesis is to demonstrate capabilities of automated process modeling for the use case of adapting existing process models based upon semantic technologies and an existing planning approach. Additionally, a way to enhance automated modeling approaches by using before-mentioned resource consideration is demonstrated. This objective encounters challenge ❺.

A general objective in this thesis is to present approaches to handle the challenges by using formal logics in form of "semantic technologies", based upon Description Logic (DL) using well-known standards such as OWL 2. Those basics and related standards will be introduced and described in detail in Chapter 2.

A second general objective is the intention to demonstrate the possibilities

of enabling the objectives with the mentioned techniques in *prototypical implementations* as part of the evaluation.

1.3 Approach

In this section, a short description of the approach proposed in this thesis is given.

In order to face before-mentioned challenges and achieve the outlined objectives, the solutions presented in the following are build upon the following techniques.

We use a *formal*, *logic-based* representation of as much information as possible to enable usage of both *reasoning capabilities*, which permit retrieval of additional facts enabled by the underlying logics, as well as *standardized querying* on top of that knowledge base. By "objects" we mean all elements that interact with the objectives of this thesis, including process models, resources and criticality classification rules. "*PMon*" (**P**rocess **m**odeling **on**tology) is the name of the ontology developed throughout this thesis that enables modeling of processes within ontologies and resource consideration. We refer to *RESon* as the ontology developed to describe resources, skills and capabilities.

We demonstrate the applicability by showing prototypical implementations on top of Description Logic (DL) in form of Web Ontology Language (OWL) or rather OWL 2 as ontology languages (used for *PMon*), extended by using Semantic Web Rule Language (SWRL) (to describe classification properties and rules).

We use *model transformations* known from Model Driven Architecture (MDA) research (OMG, 2003; Kleppe et al., 2003) to transform existing models into this logic-based representation. Furthermore, we describe how to extend these process models with additional information to further capitalize the combination of the models with other information on the formal base. This includes interweaving of nearly arbitrary enterprise objects. We will focus on the IT domain and demonstrate applicability by defining interaction of process modeling with, e.g., software or services.

The resource requirement definition is accomplished by using an existing querying language on top of the utilized ontology language. In order to decide about executability of process models considering resource requirements, we match technology spaces of both process and resource descriptions. The

resource requirements are added to nodes within the process model. The resource checking algorithm which will be introduced in Chapter 5 segments the model into fragments and checks each fragment for executability by matching *RESon*, while special semantics of control-flow structures such as parallelism within models are respected.

As stated in the section before, we aim to demonstrate an approach to decide about criticality of resources (or rather commodities) and integrate it with *PMon*. We model and implement criticality properties into SWRL rules like reserves-to-production ratio, market power and country concentration or stock of inventory to name a few. These rules are evaluated and executed by specialized reasoners, that in turn classify resources regarding the criticality indicators. The integration with *PMon* enables us to detect critical process models, i.e., where critical commodities are required for processes, which in turn might render a process into a critical one because required commodities are classified to be critical.

Figure 1.1 gives an overview of all components and the collaboration. *PMon* is the ontology containing process models, while the resource ontology (we will also refer to this ontology as *RESon* in the following) contains all resources including detailed descriptions such as skills. Executability of processes is decided by matching resource requirements of process models (*PMon*) with the resource knowledge base (*RESon*). This information is also used within the automated adaptation approach which is based upon an existing semantic-based planner which we used for adapting models and exploiting the resource information. Classification of commodities is done within the resource ontology and integrates with *PMon*, which also holds use of commodities within process models. Finally, the overview shows the transformations of existing models into *PMon* using MDA techniques.

1.4 Outline

This thesis is structured as follows.

Chapter 1 describes challenges, objectives and a short preview of the approach envisioned within this thesis. Additionally, this outline as well as publications are presented.

Chapter 2 describes basic concepts, techniques and standards that are used within this thesis. The applicability of the different aspects and approaches in

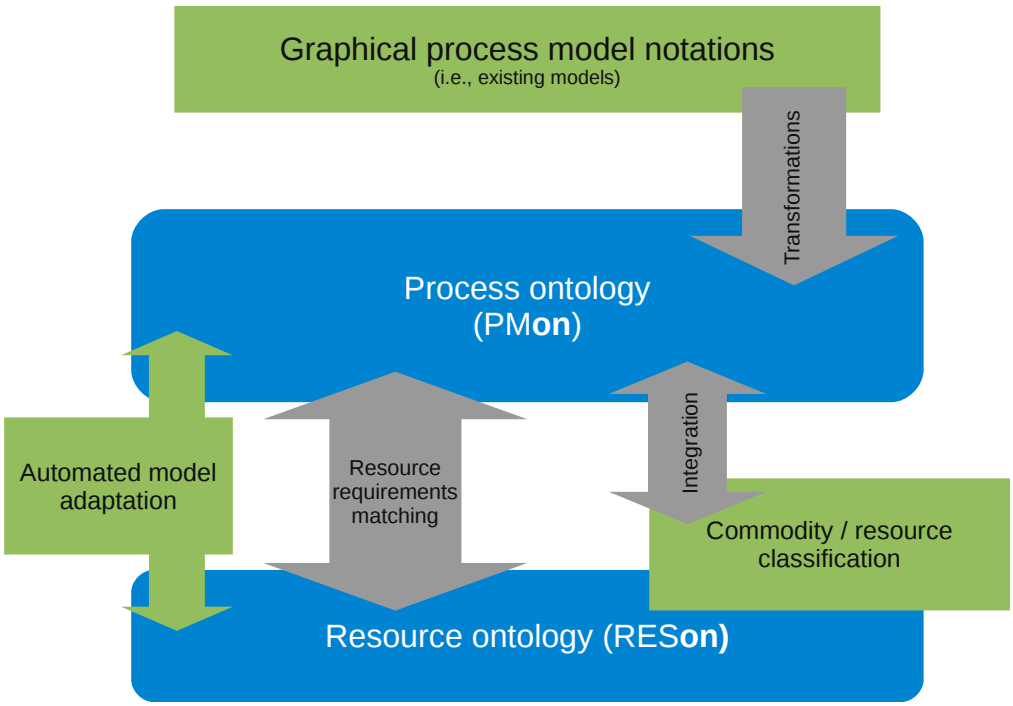

Figure 1.1: Overview of thesis' approach

this thesis are shown in the particular chapters in a running example, which is introduced in Chapter 2, too.

Chapter 3 presents related work and the research gap as well as how this thesis contributes to closing the gap.

In Chapter 4 an approach is presented to automatically transform existing process models into a generic, formal knowledge base (*PMon*) that allows combination of process modeling information with other enterprise architecture objects.

Chapter 5 introduces a novel way to combine the process and resource modeling space. We describe a way to use formal logics to describe resources and exploit inferencing techniques to enable powerful, yet simple definition of resources. Furthermore, we developed an approach to define resource requirements within process models, and verify feasibility of process models at design time. We show how this approach can be used within automatic

adaptation of process models.

In Chapter 6 we demonstrate how the formal definition of resources within a knowledge base can be exploited to classify resources (commodities) regarding their criticality. Additionally, we show how this classification colludes with the process models within *PMon*.

The theoretical parts are validated in Chapter 7. We show details of the prototypical implementation and demonstrate usage and benefits of the approaches presented within this thesis by two use cases and several scenarios, as well as a comparison of manual criticality classification with the automatic classification of commodities introduced in Chapter 6.

In Chapter 8 we will give a summary and discuss the presented approaches before giving an outlook for further research.

Figure 1.2 shows an overview of the chapters of this thesis. The chapters with grey background build the brackets around the work by motivating and concluding, respectively. Those are not necessarily essential to read, in case one knows the pros and cons about the technologies and techniques used throughout this thesis. Basically, the very same applies for the chapters marked with blue background which discuss used technologies in more detail, present similar approaches and an evaluation of the work in this thesis.

The chapters in green, at the center of the figure, mark the distinct main chapters. In case one knows about the basics those chapters can be read incoherently. Though, some references might exist between these chapters.

1.5 Publications

Some parts of this thesis appeared in previous publications.

Where multiple authors are listed, the respective paper has been written in collaboration with colleagues at the Software Methodologies for Distributed Systems lab of Prof. Dr. Bauer at the University of Augsburg or members of the Deutsche Forschungsgemeinschaft (Germany's funding organisation) (DFG) project SEMPRO^2 where the author of this thesis was involved in.

The semi-automatic planning of process models as introduced in Section 3.4 was developed as part of the research project SEMPRO.

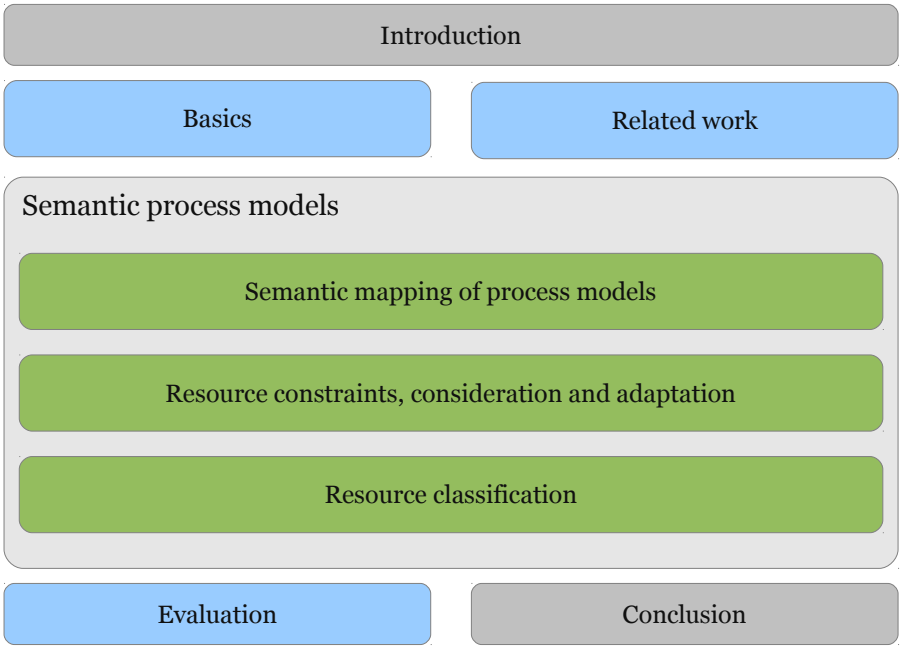

Figure 1.2: Overview of contents

Accepted publications

- Thomas Eisenbarth and Benedikt Gleich. *Using Semantic Technologies to Identify Critical Commodities to Enable Sustainable Process Improvement* Poster at *20th European Conference on Information Systems (ECIS)*, Barcelona, June 2012.

  The author of this thesis defined the technical parts (ontologies, rule design) and integration into process modeling and lead the prototypical implementation. The author acted as corresponding author and presented the work at the conference.

  This work is integrated in Chapter 6.

- Bernhard Bauer, Thomas Eisenbarth, Christoph Frenzel and Benjamin Honke. *Resource-Oriented Consistency Analysis of Engineering Processes*. In: *Proceedings of the 14th International Conference on Enterprise Information Systems (ICEIS)*, Wroclaw, Poland, July 2012. (Bauer et al., 2012)

The author of this thesis depicted the idea of combining the resource consideration approach with modeling in software engineering and defined technical parts (ontology, mapping, selection algorithm) in collaboration with the co-authors, lead the prototypical implementation and presented the work at the conference.

This work is integrated in Chapter 5.

- Thomas Eisenbarth, Florian Lautenbacher and Bernhard Bauer. *Adaptation of Process Models - A Semantic-Based Approach*. In: *Journal of Research and Practice in Information Technology*, Vol. 43, No. 1, February 2011. (Eisenbarth et al., 2011)

  The author of this thesis extended the preceding work (see next item) to fit the Journal requirements regarding extension of the work and acted as corresponding author.

  This work is integrated in Section 5.6.

- Florian Lautenbacher, Thomas Eisenbarth and Bernhard Bauer. *Process Model Adaptation using Semantic Technologies*. In: *The 4th International Workshop on Vocabularies, Ontologies and Rules for The Enterprise (VORTE 2009)*, Auckland, New Zealand, September 2009. (Lautenbacher et al., 2009)

  The author of this thesis worked in cooperation with the co-authors to enhance a first draft by Florian Lautenbacher for publishing.

  This work is integrated in Section 5.6.

Under review

Some publications are under review or planned to be submitted at the time this thesis was printed.

- Thomas Eisenbarth, Bernhard Bauer, Florian Lautenbacher, Julian Lambertz and Thomas Syldatke. *Semantic Technologies in Business Process and Enterprise Architecture Management*. In preparation for submission.

  The author of this thesis depicted the idea, defined the theoretical foundations such as the ontology and transformation rules and lead the prototypical implementation. The author acts as corresponding author.

  This work is integrated in Chapter 4.

- Marc-Andre Bewernik, Thomas Eisenbarth, Benjamin Mosig, Alexa Scheffler and Maximilian Röglinger. *Value-Based Selection of Process Models Considering Resource Restrictions*. Submitted to *Decision Support Systems*.

  The author of this thesis depicted the idea with Marc-Andre Bewernik, defined the theoretical foundations such as the theoretical, ontological description of resources with skills/capabilities, definition of resource requirements and the checking/matching algorithms as well as the prototypical implementation.

  This work is integrated in Chapter 5.

- Thomas Eisenbarth and Benedikt Gleich. *Identify Critical Commodities within Process Models using Semantic Technologies* Submitted to *Data & Knowledge Engineering*.

  The author of this thesis defined the technical parts (ontologies, rule design) and integration into process modeling.

  This work is integrated in Chapter 6.

- Mohsen Asadi, Benjamin Honke, Bardia Mohabbati, Thomas Eisenbarth, Dragan Gasevic and Bernhard Bauer. *A Process Line Approach for Situational Process Engineering*. Submitted to *Journal of Software: Evolution and Process*.

  This work is not been significantly integrated into a single part of this thesis.

Chapter 2

Basics

This chapter describes basic concepts and background knowledge necessary to understand further work in this thesis. In addition to basic definitions within the BPM and Software Engineering (SE) domain, the process management life-cycle will be described in Section 2.2 and concepts of semantic technologies in Section 2.3.

2.1 Basic concepts

Today, enterprises increase competitiveness and gain considerable advantages by optimizing operational procedures within business process models. Those (should) define precisely *which* tasks have to be executed *by whom* and *in which order*. Those process models also exist in disciplines like software, hardware and systems engineering where, e.g., requirements elicitation, development, and testing procedures are recorded.

The general term "process modeling" is used in different contexts. In information systems, typically modeling of workflows within enterprises is meant, also known as Business Process Management (BPM). In computer science disciplines such as software, hardware and systems engineering, the term is used to describe respective tasks, their ordering and sometimes associated resources such as which people are responsible for certain tasks, often denoted with roles.

BPM is a holistic management approach which aims to achieve both business effectiveness and efficiency. As such, BPM usually aims to help to improve processes continuously. While process modeling is used to define

the workflows, process optimization typically includes *simulation*, *analysis* and *re-engineering* of process models. We will describe these phases in more detail later in this section.

Generally speaking, models are *abstract, formal definitions* describing a system and its environment in a detailed manner. As such, *process models* should contain all possibilities the above mentioned *operational procedures* may take when being executed. I.e., the model should include all possible tasks in all possible execution paths. The latter is known as control-flow and will be discussed in more detail later on.

Thus, a process model is a description of a process at an abstract level. A process *instance* is an instantiation of such a model which typically takes place at runtime. Thus, the same process model can be used repeatedly and may have many instantiations.

2.2 Process models and process life-cycle

We already used the terms model, process model and some others in this thesis. We will use the following definitions throughout this thesis:

Definition 2.2.1 (Model) *"A model of a system is a description or specification of that system and its environment for some certain purpose. A model is often presented as a combination of drawings and text. The text may be in a modeling language or in a natural language." (OMG, 2003)*

Definition 2.2.2 (Task) *"A task defines some work to be done and can be specified in a number of ways, including a textual description in a file or an electronic mail message, a form, or a computer program." (Georgakopoulos et al., 1995)*

Please note that we use the terms "task" and "process action" synonymously.

One of the often cited definitions of business processes defines "business processes as a set of logically-related tasks performed to achieve a defined business outcome" (Davenport and Short, 1990). Strictly speaking, this describes the definition of a business process *model* in our understanding. "Logically-related" is what is also known as control-flow. Taking this into consideration, this leads to the following definition.

Definition 2.2.3 (Process Model) *A process model is an abstract set of process actions, being arranged with a number of patterns that define the control-flow of the process actions.*

Both systems engineering, as well as business processes fit to the above definition of a *process model* — in the respective domain. The mentioned "patterns" within the definition above refer to workflow patterns (van der Aalst et al., 2000; Russell et al., 2005b,a; Thom et al., 2007; Fortis and Fortis, 2009).

Although, minor differences exist in the focus of the domains, as the following definition of software processes shows:

Definition 2.2.4 (Software process) *"Software processes are human-oriented systems, i.e., systems in which humans and computerized tools cooperate in order to achieve a common goal. A process formalism must provide means to describe such interaction, by clearly defining, for instance, when and how a task is assigned to a tool or a human, and how to coordinate the operations of different human agents." (Bandinelli et al., 1993)*

Although such minor differences exist, the overall goal of process modeling within both disciplines is very similar. Therefore, we will demonstrate that approaches in this thesis are applicable for general process modeling independent from specific domains.

Process management life cycle Processes run through a life-cycle that typically starts with the identification of the necessity for formal process modeling. The phase of *analysis* of operational workflows and *design* of process models follows. Subsequently, *configuration* is carried out to prepare the phase of *implementation and enactment*. During the *evaluation* phase, runtime data is processed, aggregated and evaluated to gain knowledge about possibilities for further improvement which then are integrated in the following iteration. The life-cycle is shown in Figure 2.1 (cmp. Weske, 2007, p. 12).

There are several books about basic literature spanning multiple or all phases of process management (Becker et al., 2003; vom Brocke and Rosemann, 2010; Dumas et al., 2013). We will describe each phase in more detail in the following.

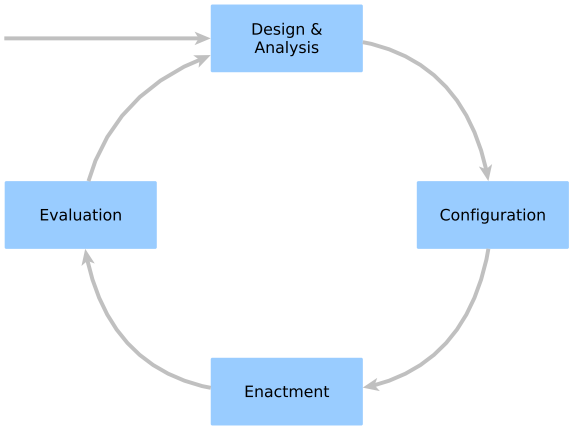

Figure 2.1: Process management life-cycle

2.2.1 Design and analysis

During analysis and design, information about requirements and possible workflows of process models is gathered and finally put into a (usually graphical) model. The analysis phase includes requirement gathering to find out which steps (tasks or process actions) are completed during the process, which are executed in parallel or have specific conditions and, e.g., which machine or person is necessary to successfully finish the task. Usually process actions have some kind of inputs and outputs that require the process actions to be executed in a specific order. This information can be gathered in surveys on the operational workflows as well as organizational and technical environment (Weske, 2007). Lots of basic literature on process model design exists for concrete modeling languages such as Business Modeling Notation (BPMN) (Silver, 2012; Debevoise et al., 2011) or Architecture of Integrated Information Systems (ARIS) Event-driven Process Chain (EPC) (Davis and Brabnder, 2007; Davis, 2008).

Subsequently, this information has to be documented in a model. According to Definition 2.2.1 this can be both in textual or graphical notation. Though, usually the latter variant of graphical modeling is used because it is easier to understand especially when models get more complex.

2.2.2 Configuration

After analyzing and modeling, the process needs to be adjusted to environmental specialties of an enterprise. This configuration typically includes attachment of technical information to the process so it is prepared to be executed in the next step (Rosa, 2009; La Rosa et al., 2008a; Gottschalk and Rosa, 2010).

Another task for this phase is the configuration of *reference processes* which capture common models and include numerous variations in a given domain. Due to those variations, the models are abstracted from concrete use cases and can be adopted and specialized by a large group of enterprises. There are several frameworks and approaches for this step (La Rosa et al., 2008b; Rosemann and van der Aalst, 2007; Hallerbach et al., 2008).

Additionally, definition of transactional behavior of processes or implementation of legacy systems necessary for the process is completed in this step.

2.2.3 Implementation and enactment

Implementation and enactment of a process typically comes along with the translation of a graphical process model into some execution language such as Business Process Execution Language (BPEL) or Extensible Markup Language (XML) Process Definition Language (XPDL). In order to successfully execute processes, workflow engines are utilized. BPM process runtime environments are offered by a large variety of vendors and Open Source projects. In the past, the phases of analysis, design, configuration and actual implementation have often been disconnected. While BPMN 1 was used for modeling, a transformation into BPEL was required as BPMN was lacking semantics necessary for execution. This lead to numerous work within research to describe these transformations (Ouyang et al., 2006; van der Aalst and Bisgaard Lassen, 2008; Ouyang et al., 2007), as well as its problems because of different concepts of graph and block-structure. The BPMN 1 specification includes a chapter regarding translation to BPEL (OMG, 2009b, Annex A), those limitations were even mentioned in the BPMN 1 FAQ nevertheless:

"By design there are some limitations on the process topologies that can be described in BPEL, so it is possible to represent processes in BPMN that cannot be mapped to BPEL" (OMG, 2012). These problems were widely discussed in research, too (Wohed et al., 2006b; Weidlich et al., 2008).

Those limitations and the effort for the transformations lead to more integrated solutions where models can be executed directly, without the need for transformations in other representations. This is supported by, e.g., Yet Another Workflow Language (YAWL) (van der Aalst and ter Hofstede, 2005) or BPMN 2 in the Activiti[1] project.

2.2.4 Evaluation

In the evaluation phase, amongst others, monitoring and log information is evaluated to analyze possibilities for improvement (van der Aalst, 2009). This includes analysis of, e.g., processes regarding their execution time to see if the average execution time is acceptable, or why peaks occur and how those could be mitigated and basically every analysis on all information that can be gathered from the execution phase (Vergidis et al., 2008; Mühlen and Shapiro, 2010). Research in this area includes process mining which can be situated at the crossing of data mining and process management. Artificial event generation is another discipline trying to induce knowledge that does not explicitly show up in log-files (van der Aalst, 2011). Event log merging tries to merge multiple sources of log information of process instances that occur in different, isolated information systems (Aalst et al., 2010).

This section introduced working definitions for processes and gave a short overview over the life cycle, processes usually pass through. We will continue the basics chapter with background information on the formal logics and ontologies used in the remaining of this thesis.

2.3 DL, ontologies and related standards

This section will give an overview of the formal logics used as foundation for other technologies that are utilized in this thesis.

We will describe basic principles of Description Logic and show the link to ontologies and the Semantic Web technologies such as OWL thereafter.

[1] http://www.activiti.org accessed as of 2012-06-17

2.3.1 Description Logic

Description Logics (DLs) are a well-investigated family of formal, logic-based knowledge representation languages and systems built by using those languages, respectively. DLs are used to describe conceptual knowledge of a specific application domain formally and infer new knowledge by using reasoning techniques (van Harmelen et al., 2007).

In DL systems, a Knowledge Base (KB) is composed of two components. The DL terminology, usually being referred to as Terminological box (TBox), contains vocabulary that in turn is built by using *concepts* and *roles*. Concepts define a set of individuals, while roles define relations between the individuals. The Assertional box (ABox) is built of assertions about individuals of the KB. Those assertions are built by using the vocabulary defined in the TBox. As both ABox and TBox are expressed and described by using DLs, we will describe those in the following (Baader et al., 2010).

As there is a magnitude of alternatives within DL, there is a convention how to describe expressivity by using operators.

Given A is used for atomic concepts, C and D for concept descriptions, R for roles, f and r for functional roles, n and m for non-negative integers and a and b as individuals, as known from the definitions in (Baader et al., 2003).

$\mathcal{AL}$ syntax $\mathcal{AL}$ is the basic DL and its syntax is formed as shown in Table 2.1.

C, D	$\longrightarrow$	$A\|$	atomic concept
		$\top\|$	universal concept
		$\bot\|$	bottom concept
		$\neg A\|$	atomic negation
		$C \sqcap D\|$	intersection
		$\forall R.C\|$	value restriction
		$\exists R.T$	limited existential quantification.

Table 2.1: Basic concept descriptions in DL $\mathcal{AL}$ language family

$$
\begin{aligned}
\top^{\mathcal{I}} &= \triangle^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
\neg A^{\mathcal{I}} &= \triangle^{\mathcal{I}} \setminus A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
\forall (R.C)^{\mathcal{I}} &= \{a \in \triangle^{\mathcal{I}} | \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.T)^{\mathcal{I}} &= \{a \in \triangle^{\mathcal{I}} | \exists b.(a,b) \in R^{\mathcal{I}}\}.
\end{aligned}
$$

Table 2.2: Extension of interpretation function to concept definitions (Baader et al., 2010)

$\mathcal{AL}$ semantics "An interpretation I consist of a non-empty set $\triangle I$ (the domain of the interpretation) and an interpretation function, which assigns to every atomic concept A a set $A^I \subseteq \triangle I$ and to every atomic role R a binary relation $R^I \subseteq \triangle^I \times \triangle^I$. The interpretation function is extended to concept descriptions by the following inductive definitions" (Baader et al., 2010).

For example, the concept definitions

- $Man \equiv Human \sqcap \neg Female$
- $Woman \equiv Human \sqcap Female$
- $Mother \equiv Female \sqcap \exists\, hasChild.\top$

define the atomic concepts "Man" and "Woman" to be "Humans". Additionally, it is stated, that men are *not* female but women are. Additionally, there is an atomic role (also referred to as relation or property) "hasChild". Finally, a mother is defined as a woman having a child. $\top$ is the universal concept, which is interpreted as all individuals in the application domain.

The basic $\mathcal{AL}$ language consists of several concepts. Though, the expressiveness of DL is extensible by using so-called constructors where the characters or symbols describe the respective expressiveness. E.g., $\mathcal{AL}$ extended with the concepts *complement*, *nominals* and *unqualified number restrictions* would result in $\mathcal{ALCON}$. We will show some of the extensions that will be relevant for upcoming chapters in Table 2.3.

As denoted in Table 2.3, $\mathcal{S}$ is an abbreviation and is build using $\mathcal{AL}$ extended by $\mathcal{CUE}$ and transitive roles. Please note, that although there are even more

Name	Syntax	Semantics	Symbol	
Top	$\top$	$\triangle^{\mathcal{I}}$	$\mathcal{AL}$	$\mathcal{S}$
Bottom	$\bot$	$\emptyset$		
Intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$		
Value restriction	$\forall R.C$	$\{a \in \triangle^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$		
Negation	$\neg C$	$\triangle^{\mathcal{I}} \setminus A^{\mathcal{I}}$	$\mathcal{C}$	
Union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$\mathcal{U}$	
Existential quantifier	$\exists R.C$	$\{a \in \triangle^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}}\} \wedge c \in C^{\mathcal{I}}$	$\mathcal{E}$	
Transitive roles	$R \in R_+$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^+$		
Role hierarchy	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$	$\mathcal{H}$	
Nominal	I	$I^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} with \lvert I^{\mathcal{I}}\rvert = 1$	$\mathcal{O}$	
Inverse role	R^-	$\{(d,e) \mid (e,d) \in R^{\mathcal{I}}\}$	$\mathcal{I}$	
Unqualified number restriction	$\geq nR$	$\{a \in \triangle^{\mathcal{I}} \mid \mid \{b \in \triangle^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\} \mid \geq n\}$	$\mathcal{N}$	
	$\leq nR$	$\{a \in \triangle^{\mathcal{I}} \mid \mid \{b \in \triangle^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\} \mid \leq n\}$		
	$= nR$	$\{a \in \triangle^{\mathcal{I}} \mid \mid \{b \in \triangle^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\} \mid = n\}$		
Qualified number restriction	$\geq nR.C$	$\{a \in \triangle^{\mathcal{I}} \mid \mid \{b \in \triangle^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \mid \geq n\}$	$\mathcal{Q}$	
	$\leq nR.C$	$\{a \in \triangle^{\mathcal{I}} \mid \mid \{b \in \triangle^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \mid \leq n\}$		
	$= nR.C$	$\{a \in \triangle^{\mathcal{I}} \mid \mid \{b \in \triangle^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \mid = n\}$		

Table 2.3: Overview of $\mathcal{AL}$ extensions relevant for this thesis, see (Baader et al., 2010) for a full list

extensions and abbreviations such as $\mathcal{FL}^-$ or $\mathcal{EL}^{++}$ we will not discuss those in detail, because Semantic Web technologies that will be presented in the upcoming sections rely on the presented concepts. For a complete list of constructors and a more in-depth discussion please refer to (Baader et al., 2010).

We will give some short examples for the DL concepts seen so far.

- *Unqualified number restrictions* allow the definition of sets of roles.

$$\text{DogLover} \equiv \text{Human} \sqcap \geq 3\ \text{hasDog}.$$

- *Qualified number restrictions* constrain the set by an additional concept.

$$\text{AlsatianDogLover} \equiv \text{Human} \sqcap \geq 3\ \text{hasDog.Alsatian}.$$

- *Nominal constructors* limit the set to a set of specific individuals.

$$\text{GermanDogBreeds} \equiv \{\ \text{GreatDane, Alsatian}\ \}.$$

Open and closed world assumption

As mentioned before, knowledge bases build with DL consist of two components, ABox and TBox. The ABox can be seen as a traditional, relational database that contains direct and binary relations. While classical database systems describe domains applying Closed World Assumption (CWA), an ABox describes the respective domain applying Open World Assumption (OWA).

To put it simply, CWA says that everything not being proofed to be true will be denoted to be false. OWA in turn is the opposite by stating that lack of knowledge does not imply falsity.

As known from classical database systems, in CWA systems a schema is necessary to describe all contents. This schema together with the content describes the knowledge within the database. Any information that is not modeled within the knowledge base does not exist.

In knowledge base systems based on OWA, one starts with an empty body of knowledge where everything is possible, and one iteratively restricts possibilities by constraints. In other words: No statement can be made about information that is not modeled within the knowledge base.

E.g., in order to decide whether a dog is able to fly in a CWA system, one would state that it is not possible as long as the fact is not present in the database. In OWA systems in contrast, the answer would be that it is unknown until the opposite is modeled.

Therefore, OWA is usually used when the knowledge about a domain is not supposed to be complete, while CWA is applied when the KB is assumed to be complete. The assumption about incomplete knowledge is especially useful when information should be re-used and extended by new knowledge.

Complexity

Reasoning on top of KBs is one of the main reasons for defining formal ontologies on top of logics. Usually reasoning includes several tasks that a reasoning software is expected to be capable of, such as:

- Deciding about satisfiability of concepts, i.e., whether individuals can exist at all, that are an instance of specific concepts.
- Subsumption (union) of concepts, i.e., if concept A subsumes concept B as A is more general than B.
- Consistency of ABox considering definitions of the TBox, i.e., individuals of the ABox do not violate definitions of the TBox.
- Test whether an individual a is an instance of a concept A.
- Find all individuals that are instances of a concept.
- Realization of a single individual a, i.e., definition of all concepts an individual belongs to.

The complexity of the basic DL $\mathcal{AL}$ and the introduced extensions is shown in Table 2.4.

Please refer to the DL complexity navigator (Zolin, 2011) for a more complete list of extensions, respective complexity calculation and more details as well as (Hitzler et al., 2009b) and (Baader et al., 2010) for detailed discussion of the various dialects and extended background on theory.

2.3.2 Ontologies

A lot of formal definitions for the term *ontology* exist in the literature. Especially since ontologies are discussed in philosophical research handling nature of being, existence, and reality, there are plenty of definitions that tend to this direction, too.

Symbol	Complexity
$\mathcal{AL}$	*PTime*
$\mathcal{ALC}$	*PSpace*
$\mathcal{SHIF}$	*ExpTime*
$\mathcal{SHOIN}(D)$	*NExpTime*
$\mathcal{SROIQ}$	2*NExpTime*

Table 2.4: Complexity of base DL $\mathcal{AL}$ and extensions

In the area of computer science, one of the often cited and well-known definitions is the following (Gruber, 1992).

Definition 2.3.1 (Ontology) *"An ontology is a formal specification of a shared conceptualization.".*

The World Wide Web Consortium (W3C) provides another short, yet precise definition of an ontology in its "OWL Web Ontology Language Use Cases and Requirements" document: "An ontology defines the terms used to describe and represent an area of knowledge." (Heflin, 2004) And further the description says: "Ontologies are used by people, databases, and applications that need to share domain information (a domain is just a specific subject area or area of knowledge, like medicine, tool manufacturing, real estate, automobile repair, financial management, etc.). Ontologies include computer-usable definitions of basic concepts in the domain and the relationships among them." (Heflin, 2004)

An ontology in the area of computer science (which is how it is understood throughout this thesis) consists of a set of *terms* and *relations* between them. The relations usually define a hierarchy using parent-child relationships. This definitions of terms and the hierarchy is known as *taxonomy*. Additionally, the terms are classes that group further information into those terms as instantiations that are known as *individuals*. Besides the taxonomy, an ontology additionally contains those individuals within the taxonomy as well as properties that enable further description of individuals, attributes, axioms, and rules.

In summary, an ontology is a machine-readable knowledge base system for a particular domain. In order to guarantee machine-readability, ontologies

are defined and serialized using *ontology languages*. A well-known, standardized language that is backed up with tool support is OWL, which will be introduced in the next section.

2.3.3 Related standards

As already mentioned in the section before, the vision of the Semantic Web aiming to build a global knowledge base by using formal logics on web sites is based upon a number of different standards. We will briefly describe those in the following.

Resource Description Framework (RDF) and RDF Schema (RDFs) In the late 1990s, the W3C Metadata Activity started working on a simple description language to characterize facts on resources. This language is formally defined, computer-readable and is known as RDF today.

RDF consists of a vocabulary containing classes and properties, and can be serialized, for example, into XML. See (Beckett and McBride, 2004) for details.

A first extension to RDF was a set of classes with certain properties which is known as RDFs (Brickley and Guha, 2004). RDFs enables definition of sub-classes and sub-properties for example.

An RDF statement consists of a triple (Subject, Predicate, Object) to formulate a statement on resources. A set of such triples forms a directed graph on which SPARQL Protocol And RDF Query Language (SPARQL) can be used for querying RDF. Specialized databases for storing RDF triple graphs are often called *triple stores*.

An example RDF graph containing hundreds of sample triples is, for example, available from The Semantic Web and Agent Technologies Lab at the Lehigh University. The Lehigh University Benchmark (LUBM) contains triples such as the following.

Triple 1 The following RDF statements defines a specific student to be a graduate student.

- **Subject** `<http://Department0.University0.edu/Student28>`

```
1  PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
2  PREFIX ub:<http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
3  PREFIX owl:<http://www.w3.org/2002/07/owl#>
4  PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
5
6  SELECT ?X WHERE
7  {
8   ?X rdf:type ub:GraduateStudent .
9   ?X ub:takesCourse <http://Department0.University0.edu/
       GraduateCourse2>
10 }
```

Listing 2.1: SPARQL query

- **Predicate** `rdf:type`
- **Object** `<http://Department0.University0.edu/GraduateStudent>`

Triple 2 This statements defines that the student takes a specific graduate course.

- **Subject** `<http://Department0.University0.edu/Student28>`
- **Predicate** `ub:takesCourse`
- **Object** `<http://Department0.University0.edu/GraduateCourse63>`

Please note that both `rdf:type` as well as `ub:takesCourse` are abbreviated and point to appropriate XML namespaces:

```
ub:<http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#>
rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

Both subject (`GraduateStudent28`) as well as object (`GraduateCourse63`) are Uniform Resource Identifiers (URIs) in the second triple. Additionally, RDFs enables usage of literals to, e.g., define the name of a student as plain string or the age as an integer value.

A SPARQL query asking for all undergraduate students that attend course `GraduateCourse63` is shown in Listing 2.1. The answer to the query could be, amongst others, `Student28` from the triples modeled above.

Today, formal description of content on the web is not widely adopted. Though, there are some examples that demonstrate the power of the idea. DBpedia is a project that regularly builds an RDF based knowledge base on top of Wikipedia that can be queried using SPARQL. An example query is to return all soccer players, who played as goalkeeper for a club that has a stadium with more than 40000 seats and who are born in a country with more than 10 million inhabitants[2]. Obviously, this information is difficult and very time-intensive to assemble manually, though by using automated querying on top of the knowledge of Wikipedia, the answer is given instantly.

The Metadata Activity was replaced by the W3C Semantic Web Activity in 2001. Efforts to design a more powerful web ontology language go back to the year 2000 when Defense Advanced Research Projects Agency (DARPA) started working on DARPA Agent Markup Language (DAML) which was merged into Ontology Interchange Language (OIL) later and finally led to OWL.

As we will use OWL throughout the remaining of this thesis, we will introduce this standard language in more detail in the following section.

OWL and OWL 2

OWL is the Web Ontology Language, a standard defined by the W3C that includes multiple languages to describe ontologies. OWL 2 is the subsequent standard definition (Hitzler et al., 2009a; Grau et al., 2008).

At the time of this writing, those two major versions of OWL exist. The first discussion about OWL 1 goes back to 2001 where a Web Ontology Working Group started at W3C. In 2004, OWL 1 achieved recommendation status at W3C while OWL 2 became a recommendation in 2009. As we do not use OWL 1 in the following, please refer to the W3C list of standards for the old, outdated recommendations.

The whole family of the OWL languages is build on Description Logics (DLs). Though, the respective sub-languages build on different extensions of DL and, therefore, offer varying functionality. See Table 2.5 for the different DL extensions, respective complexity and the OWL version build upon them.

The introduced DL syntax is mapped to OWL. E.g., the top concept $\top$ defining all individuals of a domain is `owl:Thing` in OWL. The bottom

[2] http://wiki.dbpedia.org/OnlineAccess#h28-5 accessed as of 2012-07-04

Symbol	Complexity	OWL version
$\mathcal{AL}$	*PTime*	/
$\mathcal{ALC}$	*PSpace*	/
$\mathcal{SHIF}$	*ExpTime*	OWL 1 Lite
$\mathcal{SHOIN}(D)$	*NExpTime*	OWL 1 DL
$\mathcal{SROIQ}(D)$	2*NExpTime*	OWL 2 DL

Table 2.5: Complexity of base DL $\mathcal{AL}$, extensions and corresponding OWL versions

OWL	**DL**
class	concept
property	role
object	individual

Table 2.6: Concepts in OWL and respective counterparts in DL

class $\bot$ containing no individuals maps to `owl:Nothing`.

Wording in OWL differs from what is known from DL. See Table 2.6 for an overview of terms in DL and their counterparts in OWL. We will refer to the OWL synonyms in the following.

OWL is part of the foundation for the vision of "Semantic Web" by Tim Berners-Lee which was initiated in a widely cited article called "The Semantic Web" in 2001 (Berners-Lee et al., 2001). When referring to "semantic technologies" we mean both the underlying formal logics, as well as standards being build on those such as RDF or OWL and OWL 2.

Besides usage in computer science, the technologies and standards presented before are used in medical science and bioinformatics, for example, where medical knowledge is represented. A well-known and large application of semantic technologies within medical science and healthcare is Systematized Nomenclature of Medicine – Clinical Terms (SNOMED CT) which is a comprehensive, multilingual clinical terminology for the healthcare domain and based upon DL $\mathcal{EL}^{++}$ (Spackman et al., 1997; Schulz et al., 2007).

As a detailed discussion of all aspects of all standards would go beyond the scope of this thesis, please refer to the respective standards by the W3C, or literature primarily discussing such basics (Hitzler et al., 2007; Allemang and Hendler, 2008; Hitzler et al., 2009b).

Semantic Web Rule Language

Additionally to OWL which we use for modeling ontologies we will apply SWRL as an advanced possibility to describe rules within ontologies. We will describe basics of SWRL in the following.

SWRL is an expressive OWL-based rule language having a clear specification of syntax and semantics. Additionally, it is standardized by the W3C (Horrocks et al., 2004). SWRL builds on the same logic foundation of DL as OWL does. It is supported by software tools today and compatible to OWL.

SWRL enables the specification of logical rules that define relations between OWL classes. Each rule consists of a list of required criteria (antecedent part, also being referred to as *body*), followed by an implication (consequent part, also being referred to as *head*), defining the result if all criteria are fulfilled. Therefore, a SWRL rule looks like this:

$rule : atom \wedge atom... \rightarrow atom \wedge atom$

An *atom* is an expression consisting of a predicate symbol that can be OWL classes, properties or data types with arguments that are OWL individuals, data values, or referring variables.

Built-in predicates include handling of numbers and strings such as `swrlb:lessThan`, `swrlb:add`, `swrlb:multiply` and many others (Horrocks et al., 2004).

Several reasoner implementations support only parts of the SWRL possibilities due to the problem of undecidability when considering the full specification. Though, it is a powerful extension of OWL which we will use in chapter Chapter 6.

2.4 Running example

Throughout this thesis a running example will be used to illustrate practical applicability of the theory and concepts of the presented parts. The

running example is a typical business process model that was introduced in a slightly modified version by zur Muehlen and Rosemann (Rosemann and zur Muehlen, 2005) which is shown in Figure 2.2. It is based upon an actual occurrence and was also used by (Churliov et al., 2006) and (Bolsinger et al., 2011).

Due to a problem with the process, more than 4.000 employees of a large educational institution were paid with one day delay. This caused bouncing checks, rejected automatic bill payments amongst further damage. The cause for the delay was a mistake made by an employee, who entered the wrong payroll date in one step of the payroll process. The fault was not recognized by two administrators signing off on the scheduled payroll run. So the erroneous payroll was transmitted to the university's bank for processing. When the error was discovered it was too late to re-schedule the payroll run.

We chose the process model because it is a typical real-world process that is not too complex, though. It contains enough tasks and some control-flow structures to demonstrate applicability of the approaches presented in the following chapters.

Additionally, the model is easily extensible as we will see later on.

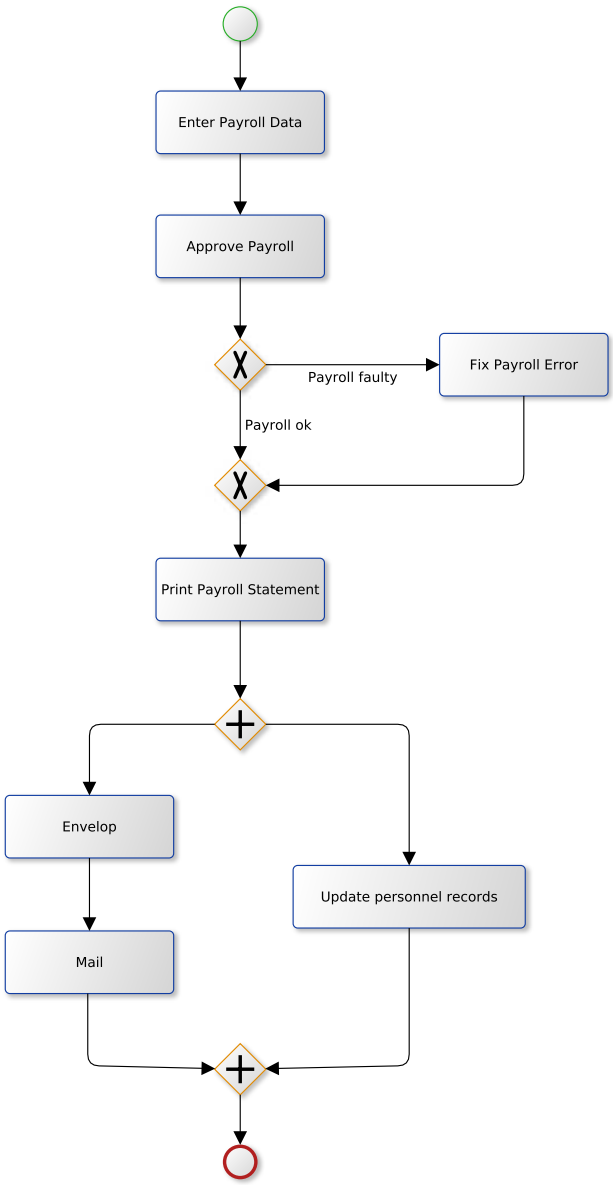

Figure 2.2: Running example: Payroll process model

Chapter 3

Related work

In this chapter, related research work to this thesis is discussed. This includes integration of Semantic Web technologies into business processes (Section 3.1), related research projects (Section 3.2) and ontology based process modeling (Section 3.3). Section 3.4 describes an semi-automated approach to generate process models using semantic technologies and Section 3.5 presents related work about resource consideration and classification.

In Section 3.6 we conclude with an analysis of the research gap aimed to close within this thesis.

3.1 Principles

As already mentioned in the introduction, BPM *does* help to handle some of the problems enterprises face today. According to Hepp (Hepp et al., 2005), companies have to deal with three dimensions when applying business process management strategies:

- Costs per process.
- Delay of process setup.
- Costs per process setup.

Companies need to be *efficient* which refers to low costs for designing and during runtime of process instances as well as *agile* which means short lag to setup new or modified processes. This must be ensured while enterprises

need to be able to evolve processes in a fine-granular way and keep costs small for setting up processes. Business processes depending on various information spread across enterprises as well as applications using this information need to be easily assembled in order to ensure that upcoming business ecosystems can be built shortly. These requirements describe the open issues that remain, e.g., in the implementation of processes, i.e., the enactment phase. This includes automation of service choreography and composition of web services that are modeled within process models.

Service Oriented Architectures (SOAs) enjoyed great popularity over the past few years. The approach promises to enable IT departments to decouple systems, making the whole IT infrastructure, software landscape a lot more agile and less fragile whenever the business side requests changes (Josuttis, 2007; Krafzig et al., 2005; Marks and Bell, 2006). Another pitfall many decision makers in IT departments increasingly try to avoid is known as *vendor trap*. Therefore, a often expressed goal is to stay vendor-independent whenever possible. This in turn comes along with Model Driven Engineering (MDE) approaches such as the MDA of the Object Management Group (OMG) that also tackles this problem (Kent, 2002; Kleppe et al., 2003). Though, SOA is residing on the enactment phase of the process model life-cycle and therefore does not further influence our work. We will use model-driven techniques for the transformations of processes described in Chapter 4.

Another idea to solve the challenge of fully automated processes at runtime was to combine semantic technologies and business process management (Hepp et al., 2005). This combination is often being referred to as Semantic Business Process Management. We will shorty introduce related work on Semantic Business Process Management (SBPM) in the following.

SBPM aims to improve Business Process Management by applying Semantic Web technologies.

As introduced in Chapter 2, Semantic Web technologies include ontology languages, reasoning software and query languages on a strong formal basis. Applied to BPM those techniques allow machine-accessible representation and manipulation of process models and model instances during runtime (Hepp et al., 2005).

Lots of research including major international research projects have been conducted on SBPM in the past. We will give an overview of those projects in the next section. As we mainly focus on the design-time of process modeling we will also focus on this life-cycle phase. For all other phases and general

work on SBPM there is plenty of general and overview literature (Hepp et al., 2005; Hepp and Roman, 2007; Wetzstein et al., 2007; Hoang et al., 2010).

3.2 Related research projects

As mentioned before, back in 2005 Hepp envisioned the combination of formal description and business process models by using emerging Semantic Web technologies (Hepp et al., 2005). Since then, a magnitude of research projects dealt with this vision and the steps necessary to gain the goals (Hoang et al., 2010). Additionally, companies — especially those offering business process consulting services or products — got involved in the research area since then.

Large-scale SBPM-related research projects are or were[1]:

- The objective of Semantics Utilised for Process Management within and between Enterprises (SUPER) project (financed from the European Union 6th Framework Programme) "was to raise Business Process Management (BPM) to the business level, where it belongs, from the IT level where it mostly resides now". It aims at providing a framework based on Semantic Web Service (SWS) technology in order to acquire, organize and share the knowledge embedded in technical representations such as business process models, systems and software on the one hand and human expertise on the other hand.

  As the original description says, the goals and resulted publications of SUPER were various. Presented results and approaches include semantic extensions to ARIS (Stein et al., 2008, 2009), compliance checking (El Kharbili et al., 2008a,b,c; El Kharbili and Stein, 2008; Weber et al., 2008) among others while most deliverables were built using Web Service Modeling Ontology (WSMO) as technological foundation.

- The research project FUSION focused on "development of an innovative approach, methodology and integration mechanism for the semantic integration of a heterogeneous set of business applications, platforms and languages within Small and Medium-sized Enterprises (SME)". Basically FUSION aimed at similar objectives as the SUPER project (Alexakis et al., 2007; Bouras et al., 2007; Kourtesis and Paraskakis, 2008):

[1] Some of the mentioned projects are completed at the moment of this writing.

  - FUSION aspires to demonstrate an innovative approach, methodology and integration mechanism for the semantic integration of a heterogeneous set of business applications.
  - Bring together Business Process Management, Semantic Web and Web Services.
  - Demonstrate and validate the results in use cases across semantically-enriched supply chains.

  FUSION was funded by the European Commission in the 6th Framework Programme, too.

- Semantic Technology Institute (STI) at the University of Innsbruck hosted and hosts several groups researching on in combination of semantic with business and e-commerce technologies. Semantics in Business Information Systems (SEBIS) for example "addresses research questions resulting from the use of computer systems for business purposes"[2]. The group was involved in the SUPER project.

- TU Wien lead a project called "Semantic Business Process Management for flexible dynamic value chains" that ended in 2008. Main focus in this project were management of business processes and a framework based on semantic technologies.

- Semantic based Modeling, Selfcomposition, and Selfconfiguration of Reference Processes (SEMPRO) and the successive SEMPRO^2 projects, both funded by DFG, were projects intended to explore Semantic Web technologies in conjunction with business process *modeling*. Both projects were realized at the University of Augsburg lasting from 2008 until 2012.

- THESEUS is a research project funded by the German Federal Ministry of Economics and Technology that was launched at the end of 2007. The projects' goal was to simplify access to information, link knowledge and to build a foundation for development of new services on the Internet. It aims at using semantic technologies to put several application scenarios into test: ALEXANDRIA is meant to be an information platform helping its users to make information public, edit and search for it using semantic technologies. CONTENTUS provides a multimedia platform for digital libraries and archives, MEDICO is an intelligent image search within medical databases, ORDO aims to support organization of digital information. PROCESSUS is the application scenario that shows

[2] http://sebis.deri.org/ as of 2011-03-15

how to compare products, solutions and business partners as well as knowledge on specific topics for skill-intensive work and TEXO finally is meant to build infrastructure basics for Internet-based services.

- Aletheia was a research project aiming at holistic view on product-related knowledge for producer, merchants and customers. Based on the expected expansion of product information available on the Internet, the project tries to explore strategies to return relevant information. The approach is addressed by a distributed information system based on a SOA. It was completed in July 2011 (Kunz et al., 2010).

Summing up the mentioned research projects, publications and efforts on SBPM we can state that there was a lot of effort put into the idea of combining Semantic Web technologies and business process management. SBPM evoked high expectations on improving modeling, management and monitoring in business process management. SBPM emerged as an independent research area pushing the combination of Semantic Web technologies with BPM to computationally exploit process models and achieve better automation based on formal models. We give an overview of the state of the art in SBPM in the following.

As described before, process modeling comprises different phases like modeling, analyzing or execution of business process management which all have research communities on these phases. In the meantime, efforts to realize the Semantic Web focuses on how large amounts of data present on the todays web can be enhanced using the appropriate technologies such as RDF, OWL and last but not least XML. The terms used to achieve the original goals changed over time, talking about "Linked Data" instead of "Semantic Web" today (Berners-Lee, 2006; Bizer et al., 2009).

We summed up our observations on this in Figure 3.1, which presents an overview of history and current development in the area of Linked Data and the concurrent progress on SBPM. As the figure shows, most research projects started around one year after the famous vision paper in 2005. The directions of SBPM research forked and different research groups and projects focused on those forks as, e.g., the SEMPRO and SEMPRO^2 projects did on modeling. Besides the pure support of BPM, Hepp worked on combining the central ideas of the Semantic Web Initiative to make data on the web computer processable. Therefore, Hepp developed *eClass* and *eClassOWL* build on top which is an ontology describing the types and properties of products and services ready to be crawled and processed by machines. These

efforts contribute the Linked Data initiative where web sites get enriched using semantic information.

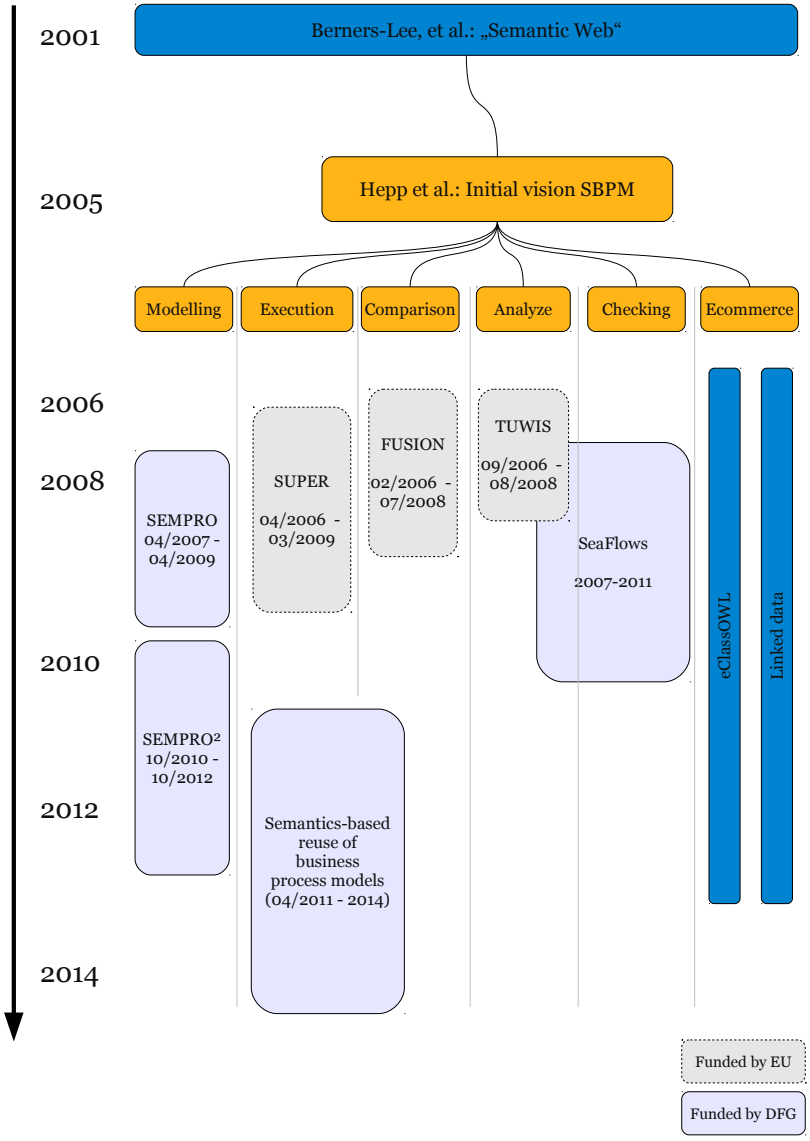

Figure 3.1: Historical and recent development in SBPM

Current state of SBPM When looking over relevant publications or research projects one can see that the majority of research was done at a very technical level (e.g., handling SWS or non-functional aspects of SWS as formal enhancements for SOA) which is why we state that the intended combination and convergence of IT and business is not yet achieved. This is underlined by the fact that current articles and books covering state of the art in BPM do not cover achievements in SBPM nor any other technique of the Semantic Web in detail. To be more concrete, (Harmon and Wolf, 2010) does not mention any technique at all, back in 2008 (Harmon and Wolf, 2008) at least mentioned that the OMG is working on business process meta-models and semantic models. Although, they state that "This is sophisticated and technical stuff, and it isn't surprising that only a few – predictably, large and technically sophisticated companies – are interested in these standards efforts." (Harmon and Wolf, 2008). A report called "State of the Business Process Management Market 2008" by Oracle claims to have "completed a thorough analysis of the business process management (BPM) market, drawing on more than 100 analyst reports, articles, and customer surveys." (Oracle Corp., 2008). The report does not mention efforts of the SBPM community which is another indication that — although having presented a magnitude of solutions to existing problems — SBPM has not arrived at industry yet.

We will describe more details of all areas this thesis contributes to in the following.

3.3 Ontology-based process modeling

Within the introduced research projects, some approaches have been developed, that focus on similar topics this thesis does. We will discuss those in the following and show differences to approaches presented in this thesis.

The focus of our research is on the *modeling* phase of process management where we aim to define models within ontology space and improve integration of processes with commodities as well as a resource knowledge base to decide about resource requirements necessary to execute processes. As there is a magnitude of projects on SBPM we focused on those projects and articles, that aimed at that phase, too.

The benefits of semantic enhancements within process modeling has been widely discussed (Hepp et al., 2005; Lin et al., 2006; Hepp and Roman, 2007; Heinrich et al., 2008; Lautenbacher, 2010). Many approaches work to-

wards automated configuration of process models, e.g., to detect services capable of handling specific tasks within processes that are searched for, selected and configured at runtime. Additionally, most approaches are specific to a certain modeling language, such as (Bögl et al., 2009; Filipowska et al., 2009b; Thomas and Fellmann, 2007) for EPCs, (Gasevic, 2004; Brockmans et al., 2006) for petri nets or (Abramowicz et al., 2007) for BPMN.

(Hepp and Roman, 2007) proposed ontologies including upper ontologies as well as formalisms for business ontologies. The group build the ontologies using Web Service Modeling Language (WSML). Although the paper suggest concrete details of ontologies to be build to represent process models, it lacks a detailed and formal specification how existing models can be transformed into ontological representations. Additionally, the specification of control-flow patterns such as parallelization are not discussed in detail. The article describes an *Upper Organizational Ontology* to describe concepts such as Organization, Role, Task, Division or Resource as well as some relations between them such as who reports to whom or is supervised by whom. Additionally, the *Business Organization Ontology* refines the concepts Role, Task, Employee and so on by common types like StaffMember or Manager. A *Business Resources Ontology* is meant to refine the concepts Resource by common concrete types that describe static resources. There are further ontologies mentioned, e.g., for rules and constraints, enterprise data, enterprise strategy, provisioning and consumption and business functions. Altogether the ontology framework is a modern interpretation of existing approaches such as The Enterprise Ontology (Uschold et al., 1995, 1998). The framework is powerful but obviously complex considering the multitude of ontologies. The authors argue that "a control flow-centric, procedural representation of business processes is often an over-specification of the actual process and should thus not be made the core of an ontological framework for Semantic Business Process Management." (Hepp and Roman, 2007). (Filipowska et al., 2009a) is an extended version of the work that discusses the differences of existing enterprise conceptualization frameworks such as Toronto Virtual Enterprise (TOVE) (Fox, 1992; Fadel et al., 1994b,a), Resources, events, agents (REA) (Geerts and McCarthy, 2000; Gailly et al., 2008) or The Enterprise Ontology that will be discussed later in the chapter. The main focus of the approach in the latter article is described as "functions, goals, organisation structure, roles, resources" which clearly differs from our work where we focus on a lightweight representation of process models within ontology space, an automated transformation for existing models as well as integration of resources to check models for executability at design-time.

(Markovic, 2008) shows an approach to query process models. Therefore, a model is proposed that is built combining five ontologies that handle different aspects such as business functions, resources, roles and goals on the one hand and a business process ontology on the other hand. A π-calculus is used to describe behavioral aspects that is able to express all workflow patterns. The work by Markovic was extended in his PhD thesis (Markovic, 2009) where he describes the work in more detail. Although the thesis describes ontologies and modeling in detail, an important question is left out. As usually enterprises do not employ semantic technologies together with process modeling but start using traditional process modeling first and maybe migrate to SBPM later, a migration path from traditional to semantic based modeling is desirable. A formal description of an automatic transformation into ontology space is not discussed by Markovic.

(Awad et al., 2008) argues that detecting similarity between process models or process model fragments is an important issue. The approach is based on a query language called BPMN Query (BPMN-Q) as well as enhanced Topic-based Vector Space Model (eTVSM) which detects similarities that are encoded within the eTVSM ontology or natural language plain text documents (Kuropka, 2004). Though, the approach does not handle ontological mapping of process models, as the title might hypothesize which is why the starting position for querying is apart to our work. A similar approach is presented in (Ehrig et al., 2007). Besides the mentioned differences, our approach is capable of exploiting reasoning capabilities that infers new knowledge within the semantic knowledge base.

Thomas and Fellmann presented an extension of process modeling languages using ontologies. Their approach was meant to represent labels of process models within ontologies (Thomas and Fellmann, 2009). It is aimed to "eliminate the scope for interpretation connected with the use of natural language". Subsequent articles of both authors mainly focus on *verification* of process models using semantic technologies.

As part of the SUPER project, a business process ontology has been built (Business Process Management Ontology (BPMO)). The ontology is aimed to translate from semantically enhanced EPC or BPMN process models to sBPEL, a semantically enhanced BPEL. As can be seen from the descriptions of the deliverables (SUPER project, 2011), the main focus was on the bridge from model to executable code whereas our approach focuses on the semantic-based representation of models to exploit querying and reasoning capabilities on the process models.

To sum up the existing work on process modeling using semantic technologies one has to admit that lots of work has been done. Though, there is a couple of missing building blocks to allow enterprises to make use of the benefits of this technology. Especially due to the assumption, that enterprises have process models (without semantic technologies) and would like to migrate to semantically enhanced ones without starting from scratch. This is where we focused on regarding ontological representation of process models.

3.4 Semi-automated modeling of process models

As stated in Chapter 2, traditional process modeling is a time-consuming and error-prone human task. Therefore, automatic modeling of process models is one pivotal goal in SBPM.

The (semi) automatic modeling approach (SEMPA) has been developed in the research project SEMPRO pursuing this goal (Heinrich et al., 2008; Lautenbacher, 2010). It is a significant enhancement compared to manual modeling as the developed algorithm especially improves modeling in terms of quality, reproducibility and modeling velocity (cf. Lautenbacher, 2010).

A complete description of all details of Semantic Based Planning Approach (SEMPA) is given as part of Lautenbachers' PhD thesis (Lautenbacher, 2010) and by Heinrich et al. in the original publication of the approach (Heinrich et al., 2008). Nevertheless, as we build on top of the concepts of SEMPA in the following, we will shortly describe the approach before showing up improvements and enhancements that we developed in the follow-up project SEMPRO^2.

SEMPA is based upon several requirements and assumptions that we will not describe in full detail here. Please refer to the before-mentioned literature for in-depth discussions. The remaining of this section is a short summary of the SEMPA approach (Heinrich et al., 2008; Lautenbacher, 2010).

Approach

SEMPA uses a process action library called lib_A in which process actions as well as their corresponding inputs and outputs are stored. Those inputs and outputs are annotated to the process actions, meaning the inputs and outputs are stored within an ontology in order so exploit reasoning capabilities.

The SEMPA algorithm is conceptually divided into three parts as shown in Figure 3.2.

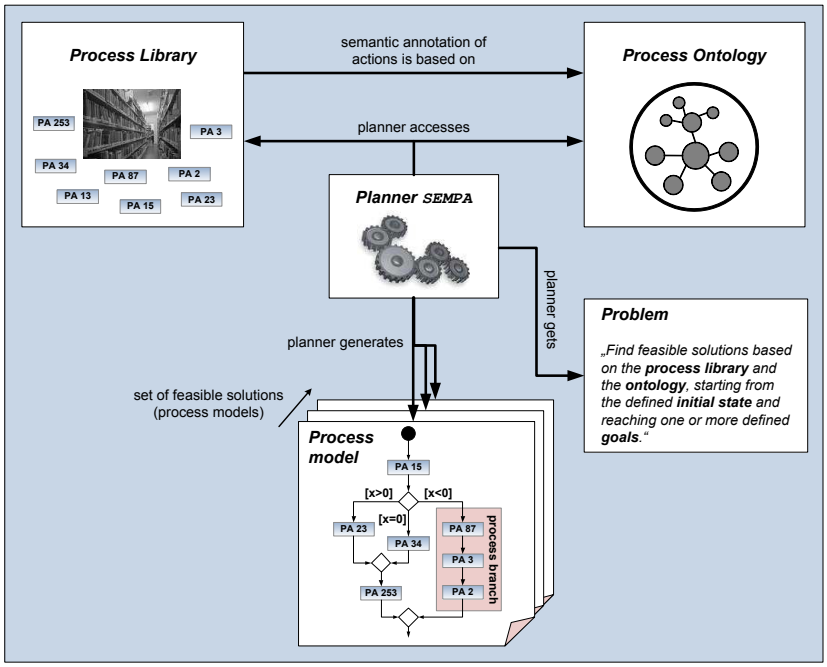

Figure 3.2: Basic ideas of the semi-automated planner SEMPA (Lautenbacher, 2010)

1. In order to find dependencies between process actions that occur due to input and outputs, an Action Dependency Graph (ADG) is built. SEMPA uses reasoning to find relationships of those inputs and outputs within an ontology. This ADG contains predecessor-successor relations between the process actions at the end.

2. To determine sequences within the ADG, a forward search algorithm is determined in this step to find all sequences of process actions from the initial state to the goals. The result is a so-called Action State Graph (ASG) that contains all feasible solutions to the corresponding planning problem.

3. In the last step, control structures are identified to build the final process model.

We will describe each phase in more detail in the following.

Action Dependency Graph

The ADG contains a set of process actions that depend on another one, i.e., process actions that provide one or more output parameters that are required as input by another process action. Such a dependency means that one process action *can* use the parameter of the other process action — but does not have to. So no control-flow is build using this input/output relationships between process actions as there might be several alternative process actions that provide (semantically) identical outputs.

In order to determine such dependencies between process actions, parameters of all actions are compared. Therefore, all outputs are compared to all inputs of process actions whether they are identical, equivalent or a specialization. These relationships are detected by querying ontology space where input and output parameters are stored.

Additionally, restriction of process actions have to be be considered. If the inputs and outputs of process actions match according to the ontology, the range of the parameters is checked.

Action State Graph

In this step, an algorithm is used that searches the results stored within the ADG in order to arrange process actions in a logical order. Therefore, a so called *state-space planning approach with forward search* is applied. A depth-first search on the results of the first step is applied: Starting with a given initial state that marks the starting point, the algorithm identifies possible successors. As input and output change while the algorithm walks through the graph, those parameters are collected during the algorithm. Whenever there are multiple possibilities, the graph is branched so these multiple possibilities result in multiple possible ASGs. The algorithm terminates when one of the specified goals are met.

Generation of process models

In the last step, control-flow structures are identified and the process model is built finally.

In order to identify the control structures, an algorithm analyzes the states and action nodes in the ASG as well as the partial and complete dependencies

between process actions in the ADG.

Sequences, for example, can be detected quite easily as process actions are in the correct order within ADG already. For the description of other control-flow structures (exclusive choice, simple merge) please refer to (Lautenbacher, 2010). The automatic generation of parallel split and synchronization patterns is currently developed by Bernd Heinrich and Felix Krause. An article containing results is not published yet.

3.5 Resource consideration and classification

There has been significant research activity on the general topic of combination of resources and processes in the past. We will describe different research areas and conclude with the research gap we will try to close.

Resource patterns Our work greatly differs from the well-known research about resource patterns by the group of van der Aalst (Russell et al., 2005b). Their work describes the different types of resource patterns in great detail (Russell et al., 2005b). Therefore, the work introduces 43 possibilities of how resources could be combined with process actions within process models. Those include the question of allocation of a resource to a process action, such as:

- **Direct Allocation** pattern which means that the model designer specifies the *identity of a resource* that will execute the task, e.g., 'PA must be undertaken by Max'.
- **Role-Based Allocation** pattern that defines which role is responsible and necessary for the completion of a process action, e.g., 'PA must be undertaken by a programmer'.
- **Capability-based Allocation** which describes which capabilities are necessary for the execution, e.g., 'PA must be undertaken by an auditor that is allowed to sign external contracts'.

Those patterns and — as mentioned — a large number of others are detailed and discussed in-depth in (Russell et al., 2005b). The work by Russel describes the enormous amount of different, occasionally highly complex

patterns. Though, the study by (zur Muehlen and Recker, 2008) showed that only a small fraction of the large amount of workflow patterns that exist for process modeling are used in reality. It can be assumed that the very same is true for resource patterns, as those are far less supported by process modeling languages today. Some patterns such as the above mentioned direct allocation pattern is an obvious pattern, yet it is comparatively inflexible because every time the resource allocated to a process element leaves the company or gets relocated into another department, all models containing the resource have to be adapted. Additionally, process actions with direct allocations would be quite inflexible, as a process instance would get stuck in case the assigned employee is on vacation, for example. Therefore, we argue that the holistic set of resource patterns described by Russell is not necessary for successful applications and integrations of resources and workflow patterns.

In the work presented in this thesis, we neglect the type of assignment and broaden the definition of resources to not only consider people or roles as resources. In our approach, any object that might be involved within a process should be able to be modeled as requirement and matched within a resource knowledge base. Additionally, we will present a way for resource requirement definition that allows description of complex requirement definitions, such as the requirement for a specific group or role with additional skills. Additionally, we focus on the matching of resource requirement and available resources to decide about the feasibility and possible allocation of a process instance. Thus, we consider both the requirement definition as well as matching a knowledge base describing all resources in detail.

Object Constraint Language (OCL) in BPMN (Awad et al., 2009) discusses an approach to define resource allocation of BPMN models. They state that BPMN "has little support for human resource modeling" as lanes can only be used to "loosely express roles or responsibilities". Based on the resource patterns (Russell et al., 2005b) they present an extension of the BPMN meta model to consider human resources by using OCL constraints to define resource requirements. The use of OCL might have advantages because of great expressiveness but this clearly comes with the downside of readability of the constraints. Furthermore, the usefulness suffers because it will be certainly challenging to explain OCL language details to BPMN modelers. Listing 3.1 shows an example taken from (Awad et al., 2009) which defines that a specific task should be executed by a member of the "Middle Management" or "Top Management". As one can see, those allocation constraints are extremely powerful. Though, those are highly sophisticated to read and complicated to

maintain.

```
1 context Task
2 inv roleLevelAuthorization:
3 self.Name = 'Process Leave Request' implies
4 self.workItem.performer->forAll(x | x.role->
5 select(name ='Middle Management' or name='Top Management')->size()
    >0)
```

Listing 3.1: OCL resource constraint for BPMN

Awad uses a similar approach in his PhD thesis to check compliance on process models (Awad, 2010). He states that compliance requirements are affected by resources in process models, e.g., regarding security policies.

In contrast to our approach, the presented work only considers human resources and neglects machines or other enterprise resources. Additionally, the resource knowledge base or resource database that holds information about resources is neglected as far as possible.

Resource Alignment Language (RAL) Another interesting approach to include resource requirements into process models was developed by Cabanillas et al. of the Applied Software Engineering research group at the University of Sevilla roughly at the same time our approach evolved (Cabanillas et al., 2011, 2012b).

The research group states, that currently, there is a lack of extensive resource definition within process modeling which is BPMN in the RAL approach (Cabanillas et al., 2012b). The approach consists of a Domain-specific language (DSL) named RAL that can be attached to process elements within BPMN 2.0. Resource Alignment Language (RAL) is a language for assigning resources to the process actions in business process models. It is build upon BPMN and enables the definition of resource requirements for the process models. Similar to the approach by Awad, RAL only considers human resources.

In (Cabanillas et al., 2011) an outlook is given where a mapping of RAL into OWL is shortly described. There is a technical report available since April 2012 that describes this mapping (Cabanillas et al., 2012a). The report describes the mapping of RAL into OWL which is based on Manchester Syntax to query the resource ontology (Horridge et al., 2006). Although the report mentions problems with control-flow-structures, e.g., if two tasks should be executed by the same person. Though, the handling of parallel

splits, for example, remains unclear in the technical report paper. Cabanillas et al. make use of the approach by Awad which, as discussed before, makes use of BPMN-Q together with Past Linear Temporal Logic (PLTL) formulas.

Although the approach by Cabanillas et al. is similar to ours at a first glance, there are some significant differences, though.

Our mapping is more generous regarding the possible modeling of resource constraints. While RAL is focused on human resources only, we tried to find a way to model arbitrary resource constraints that includes IT services, software tools or machinery to name a few. Additionally, we extended the approach by a value-based weighting and quantification of a set of process models. The latter is suitable to rank such a set of models by virtual any attribute that can be assigned to resources. Additionally, we combined the approach of resource consideration with the approach of ontology-based modeling presented in Chapter 4. Finally, RAL (at least in the version available at the time of this writing) does not support the definition of multiple required resource instances which is necessary to define a four-eyes principle, for example.

All approaches have in common, that they do not examine the use of additional parameters to decide which process model of several existing alternatives is better, e.g., regarding costs. Additionally, the design and integration of a resource modeling knowledge base is missing in the presented work.

3.6 Conclusion and research gap

Many approaches set the focus on semantic annotations and support for automatic service discovery and composition using, e.g., Semantic Web Service (SWS). Although those approaches are promising, it is widely unused in industry today and research on this specific topic significantly decreased since the research agenda from 2005. The missing usage in practice could be due to doubt and incertitude of humans when it comes to autonomic decisions made by IT systems. It is still arguable if people really want to pass over control about which services from whichever providers should be used during execution of process models.

While discussing with industry, the current approaches on using semantic technologies for enterprise description shape up to being unsatisfactory as they are heavyweight and therefore too complex while in addition, e.g., no

migration path from existing models is shown up.

On resource modeling in general, many research papers have been published. Though, (Hepp and Roman, 2007) states that "workflow-centric process representations and work on enterprise ontologies are largely unconnected, which contributes to two current shortcomings: Firstly, workflow-centric process representations are not very suitable for accessing the business process space at knowledge level (e.g., for the discovery of processes or process fragments that can serve a particular purpose). Secondly, models created by the enterprise ontology community cannot be used with current, workflow-centric BPM tools and infrastructure". Although these shortcomings were discovered, they are still unsolved today.

For combining resource requirements and process modeling notations, the only advanced approach that exists, is RAL, to the best of our knowledge. As discussed before, there are differences, though, as we expanded the existing planning and adaptation approach with the resource consideration and included an operationalization which allows selection of models considering their value.

The combination of a mapping of process models into ontologies as well as an extension to add resource requirements that check consistency of models given a set of available resources is a novel approach to the best of our knowledge.

This thesis provides several contributions that has not yet or insufficiently been discussed in literature:

1. Although ontologies exist that allow combination of process modeling with semantic technologies, most approaches are too complex for industrial usage. Furthermore, a formal specification of transformation patterns to assemble ontological representations of existing models within ontology space is missing which prohibits migration from traditional process modeling to a semantic-based, improved alternative.

2. Based upon the transformation patterns, we provide a framework and implementation details to automatically transform existing models from graphical process modeling languages into an OWL 2 ontology to ensure applicability of the approach.

3. We abstract the usage of semantic technologies within process modeling from the often positioned **business** process modeling. Although business process management is the most prominent application area

of process modeling, there are others, which also benefit from the mentioned techniques. We will show applicability of the techniques in other areas in Chapter 7.

4. We integrate a resource knowledge base that allows verification of executability of process models considering resource requirements at design time.

5. A novel approach to classify commodities is introduced that allows the semantic representation of process models to be classified as well.

6. We extended a web-based editor with the resource checking as well as value operationalization possibilities to demonstrate applicability of the latter.

Items one to three correspond to objective ①. The destination ontology is *PMon* as shown in Figure 3.3 while the formal specification and prototypical implementation is shown above the ontology in the image. Item four refers to objective ② and is handled by the resource ontology *RESon* and the matching of resource requirements with this ontology. Item five corresponds to objective ③ which consists of the classification approach itself where properties known from material resource research are formally described within ontology space to classify resources regarding criticality. Furthermore, we integrated the resource classification with the process modeling ontology *PMon*. Item six refers to the general objective which is to ensure that all parts of this thesis are realizable.

Table 3.1 gives an overview of features that are covered by this thesis in comparison with existing work. We took features discussed in (Filipowska et al., 2009a) and defined an additional set of features that represent the objectives as defined in Section 1.2 which are not or not sufficiently dealt with in past research. The comparison in Table 3.1 considers the best known approaches for enterprise and process modeling ontologies and features derived from this thesis' objectives. A positive symbol (✓) indicates, that the approach supports the respective feature while the second symbol (-) indicates, that the feature is not supported by the approach.

The feature about easy extensibility as well as the ontology language used are taken from (Filipowska et al., 2009a). We argue that our approach *is* easily extensible as we will show how further enterprise information is attachable to the presented approach in the next chapter. Our approach uses a W3C standardized, popular ontology language that has an active community

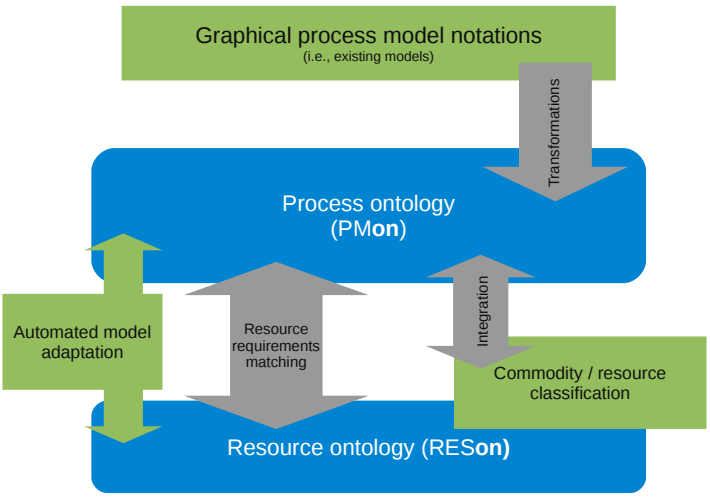

Figure 3.3: Overview of thesis' approach

and several tools ready to be used. Therefore, both the feature about available reasoners as well as about integrated querying is affirmed. Reasoners include, besides others, Pellet (Sirin et al., 2007), FaCT++ (Tsarkov and Horrocks, 2006) and HermiT (Shearer et al., 2008). The querying language which is supported by, e.g., OpenRDF Workbench (Broekstra et al., 2002) is also standardized and known as SPARQL. Additionally, we define transformation rules from existing process models into ontology space. We will demonstrate querying on our approach in the next chapter, too. As our approach includes resource consideration (which will be discussed in Chapter 5) we also present a resource knowledge base which is utilized to check process models for feasibility regarding resource requirements in our approach. We also discuss a technique to use the resource integration into process models to achieve an operationalization of the process models to compare a set of similar models regarding costs, for example. Finally, we demonstrate how to classify resources within the resource knowledge base regarding criticality.

Feature	TOVE	REA	The Enterprise Ontology	SUPER	Fellmann	Our approach
Easy extensible[a]	-	-	+/-	✓	✓	✓
Lightweight approach for modeling	-	-	-	-	-	✓
Ontology language[b] -	FOL	-	Hyprid[c]	WSML	OWL	OWL 2
Reasoner available	✓	-	-	✓	✓	✓
- Integrated querying	-	-	-	✓	✓[d]	✓
Integrated resource knowledge base	✓	✓	✓	-	-	✓
Formal specification of transformation patterns	-	-	-	-	-	✓
Model checking regarding resource requirements	-	-	-	-	-	✓
Operationalization of value of models	-	-	-	-	-	✓
Integration of commodity classification	-	-	-	-	-	✓

Table 3.1: Feature comparison of our approach and related work

[a](Filipowska et al., 2009a)
[b](Filipowska et al., 2009a)
[c]"Informal (text) and semiformal (Ontolingua)"(Filipowska et al., 2009a)
[d]Technically possible and shown using SPARQL in verification articles of the author (Fellmann and Thomas, 2011; Fellmann et al., 2011)

Chapter 4

Semantic process models

As depicted in Hepps' vision paper (Hepp et al., 2005) business processes and all corresponding artifacts should be computer-processable. Although the research community was quite active, there is still a gap between Hepps' vision and current possibilities in the area as shown in the previous chapter.

We will focus on the *modeling* of (business and software engineering) process models using semantic technologies in this section. The overall research question is **how formal logic in the form of ontologies can be used to improve handling of large model repositories and complex interdependencies with enterprise objects such as IT landscape**. This includes the following sub research questions: **(1)** How can semantic technologies be used to answer questions about process models such as

- Which IT-services are used within a certain process?
- Is a specific IT-service used by any process?
- Which stakeholders are involved within a specific process model?

Additionally, we want to show **(2)** how ontologies can be used to classify models and model elements automatically, e.g., regarding their validity: Are all process actions connected by an incoming and outgoing edge? Are all process actions supported by an IT service?

To summarize the before mentioned, using ontologies for defining process models we aim to:

- Utilizing *a single data repository* to define enterprise resources such as IT services, employees or machines and their skills and capabilities.

- To find a way to define process models using Semantic Web technologies on top of an *up-to-date, widely supported ontology language*.
- Depict an automated method to transform *graphical* process models into ontology space, e.g., into the *PMon* ontology.
- Integrate ontology-based enterprise modeling with process modeling.
- Demonstrate how to integrate business process modeling into existing enterprise ontologies.

In the following, we will present our approach named *PMon* in theory as well as implemented tooling support for mapping traditional graphical models into ontology-based definitions.

Our approach to combine process models and semantic technologies is to fully transform graphical process models into ontologies. Besides the beneficial usage of reasoning capabilities on the models this has one great advantage over other approaches. As the models are part of the ontology the latter can be used as a pivotal data and knowledge storage. This in turn minimizes the effort necessary to synchronize models in graphical notation languages and semantic information managed in ontologies which is often denoted as "annotations". Additionally, we will present an approach to define enterprise resources in the same ontological space therefore enabling the combination of both process models and enterprise resources. Furthermore, we will show an use case that enhances an automated planning approach for process models in Section 5.1.

Especially the mentioned combination and management of process modeling and IT services is handled in the area of enterprise modeling. We will present the underlying meta model in the next section and derive the ontology design afterwards.

4.1 Meta model

In this section, we will introduce the meta model and its concepts utilized to develop the formal model with an ontology language in Section 4.2. In the following, we will discuss well-known control-flow workflow patterns and our corresponding transformation patterns for the five basic workflow patterns:

- Sequence.
- Parallel Split.
- Synchronization.
- Exclusive Choice.
- Simple Merge.

Afterwards, we will demonstrate the transformations on the running examples that show the transformation methodology applied in an automated transformation tool. We will complete the use case showing how existing enterprise information can be added to the ontology core and discuss how the presented approach colludes with SBPM visions (Hepp et al., 2005).

Figure 4.1 shows the underlying meta-model that formalizes our approach. The central element within the meta model is `ModelElement` which is parent element for two further elements:

- `ProcessAction`.
- `ControlFlowElement`.

Those in turn are specialized into

- `Split`.
- `Join`.
- `Event`.

for the control flow elements and

- `AutomaticProcessAction`.
- `SemiAutomaticProcessAction`.
- `ManualProcessAction`.

for the different process actions.

The abstract `ModelElement` can be connected by edges where usually there is an incoming and an outgoing edge for most model elements. Though, there are exceptions where there is only an outgoing edge (e.g., start event nodes) and only one incoming edge respectively (e.g., end event nodes). Additionally, edges can have labels and conditions to restrict activation of edges after, e.g., `XORSplit` nodes. Furthermore, model elements can be part of a single swim-lane which in turn has a role that is responsible for the swim-lane.

Finally, the `ProcessModel` class is a container for models itself which have an arbitrary number of associated model elements and have at least one start and one end node.

In order to specify the transformations of process models and the adaptation approach more formally, we will describe process models by using the following notation.

A process model is a directed graph $\mathcal{G}$ which is denoted by $(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{E}$ the set of edges.

$\mathcal{N}$ consists of the disjoint subsets: $\mathcal{N}_{Start}$, $\mathcal{N}_{Stop}$, $\mathcal{N}_{Action}$, $\mathcal{N}_{ANDSplit}$, $\mathcal{N}_{ANDJoin}$, $\mathcal{N}_{XORSplit}$ and $\mathcal{N}_{XORJoin}$. The terms *action*, *decision*, describing control-flow structures are used in analogy to UML activity diagrams (OMG, 2009a) and BPMN process models (OMG, 2011). $\mathcal{N}_{Split}$ (also referred to as `ANDSplit`) and $\mathcal{N}_{XORSplit}$ (also referred to as `XORSplit`) are subsumed to `Split` nodes while $\mathcal{N}_{ANDJoin}$ (also referred to as `Join`) and $\mathcal{N}_{XORJoin}$ (also referred to as `Merge`) are subsumed to `Join` nodes within the meta model.

Each node $n \in \mathcal{N}$ has a set of incoming and outgoing edges denoted as $\mathcal{E}_{in}(n)$ and $\mathcal{E}_{out}(n)$ respectively. Furthermore, the graph $\mathcal{G}$ has to satisfy the following conditions:

- A process model graph $\mathcal{G}$ has an arbitrary number of nodes such that $n \in \mathcal{N} : |n \in \mathcal{G}| >= 0$
- Start nodes have no incoming and a single outgoing edge. End nodes have a single incoming and zero outgoing edges respectively. $|\mathcal{E}_{in}(n_{Start})| = 0 \wedge |\mathcal{E}_{out}(n_{Start})| = 1$ (called *entry edge* e_{entry}) and $|\mathcal{E}_{out}(n_{Stop})| = 0 \wedge |\mathcal{E}_{in}(n_{Stop})| = 1$ (called *exit edge* e_{exit}).
- Process actions have exactly one incoming and outgoing edge, such that $\forall n \in \mathcal{N}_{Action}$: $|\mathcal{E}_{in}(n)| = 1 \wedge |\mathcal{E}_{out}(n)| = 1$.

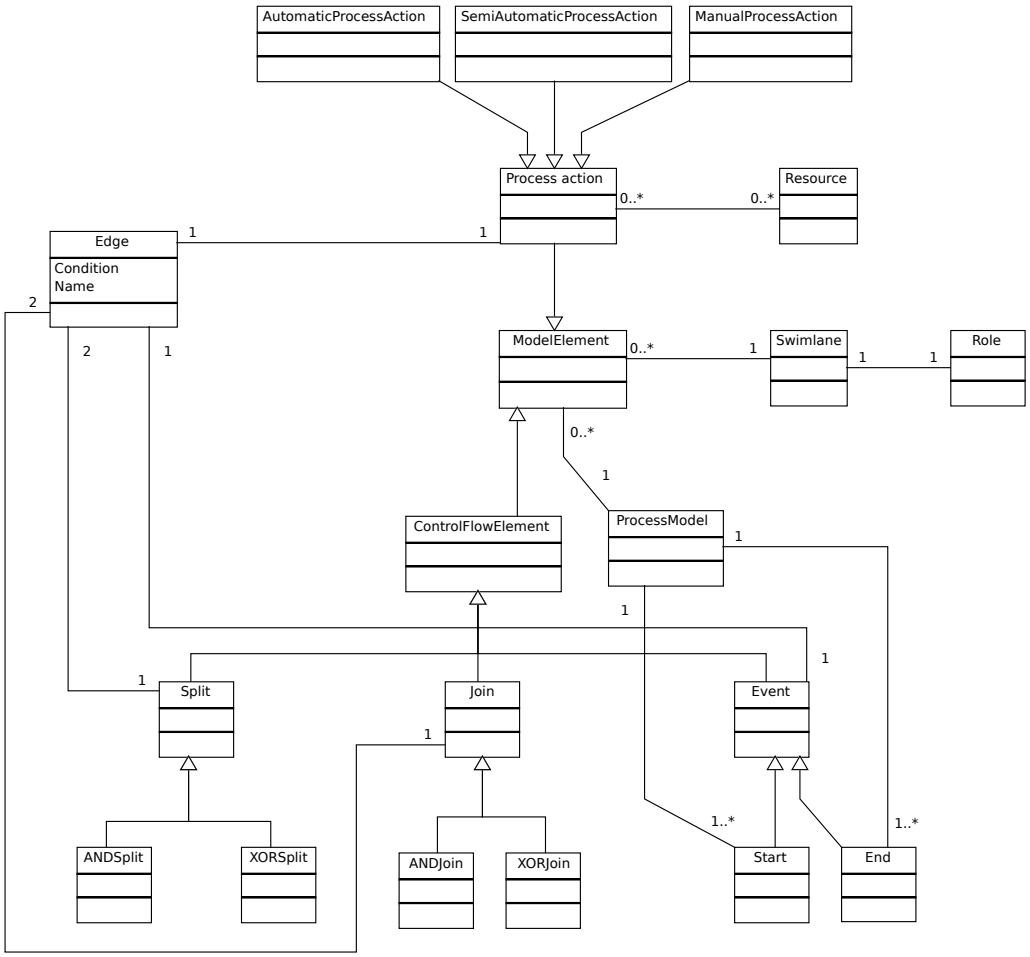

Figure 4.1: Meta model

- Split nodes have at least two outgoing and one incoming, join nodes have at least two incoming and one outgoing edges. $\forall n \in (\mathcal{N}_{Split} \cup \mathcal{N}_{XORSplit})$: $|\mathcal{E}_{in}(n)| = 1 \wedge |\mathcal{E}_{out}(n)| >= 2, \forall n \in (\mathcal{N}_{ANDJoin} \cup \mathcal{N}_{XORJoin})$: $|\mathcal{E}_{in}(n)| >= 2 \wedge |\mathcal{E}_{out}(n)| = 1$.

- All nodes must be reachable from the start node. $\forall n \in \mathcal{N} : \exists$ path $p = (n_{Start}, \ldots, n_{Stop}) \subseteq \mathcal{G}$: $n \in p$.

We will describe the components as well as the corresponding mapping into ontology space in more detail in the upcoming section.

4.2 Ontology design

In this section, the ontology design for our semantic process modeling approach is discussed. This **p**rocess **m**odeling **on**tology is also referred to as *PMon*. In order to represent process models in ontologies we need to transform the fundamental concepts of the meta model into ontology space. Basically every class from the meta model corresponds to an OWL class with one — important — exception that is necessary to describe the control flow of process models. We will discuss this later in this section.

The classes are containers for individuals that build a process model instance. We say an individual X is an individual and member of class by using this notation: $X : ProcessElement$. While the OWL classes define the taxonomy described in Section 4.1, individuals within the ontology describe a meta model instance. This corresponds to the well-known semantics of MDA and Model Driven Software Development (MDSD) (Kleppe et al., 2003; OMG, 2003).

Ontologies in contrast to taxonomies naturally consist of additional components as described in Section 2.3. Therefore, we defined object properties to build the control-flow of process models as well as further relations to model information, e.g., to define a process action to be part of a swim-lane or to express the requirement that it should be executed by a specific role. The integration of processes and arbitrary further enterprise information is accomplished by object properties, too. As the control-flow is fundamental to process models, we will describe the mapping of control-flow and respective structures in the following.

Control flow

Control flow is defined as follows (van der Aalst et al., 2003a).

Definition 4.2.1 (Control flow) *"The control-flow perspective (or process) perspective describes activities and their execution ordering through different constructors, which permit flow of execution control, e.g., sequence, choice, parallelism and join synchronization."*

In order to enable process models within *PMon* to support the most basic control-flow structures such as *sequence*, we make use of OWL object property assertions to represent some of them.

Object property assertions connect process elements such as process actions with other control flow elements or process actions in a sequence. Given a process action individual $n \in \mathcal{N}$ that is connected to another process action $n_2 \in \mathcal{N}$ by an edge $e \in \mathcal{E}$ we utilize an object property as defined in Listing 4.1.

```
1 <owl:ObjectProperty rdf:about="#hasSubsequentProcessAction">
2   <rdfs:range rdf:resource="#ProcessAction"/>
3   <rdfs:domain rdf:resource="#ProcessAction"/>
4 </owl:ObjectProperty>
```

Listing 4.1: Sequence control-flow object property

Utilizing this object property assertion we can build a simple sequence by defining $Enter_Payroll_Data$ *hasSubsequentProcessAction* $Approve_Payroll$ as defined in Listing 4.2.

```
1 <owl:NamedIndividual rdf:about="#Enter_Payroll_Data">
2   <rdf:type rdf:resource="#ProcessAction"/>
3   <hasSubsequentProcessAction rdf:resource="#Approve_Payroll"/>
4 </owl:NamedIndividual>
```

Listing 4.2: Sequence control-flow assertion

While utilizing object property assertions for defining the control-flow pattern *sequence*, we decided to define more complex patterns in a different way. The parallel split gateway pattern as can be seen in Figure 4.2 is called *implicit* modeling. This is due to the two outgoing edges from process action *PA* 1. There is no dedicated control-flow structure element following *PA* 1 that defines the semantics and behavior of the control-flow. This indicates, that the two edges should be activated in parallel after the completion of *PA* 1 (Fortis and Fortis, 2009; Wohed et al., 2006a).

Although it would be possible to use object property assertions to describe further control-flow patterns, we decided to use object properties for sequences only for simplicity and straightforwardness. Though, the representation as shown in Figure 4.2 would be easily possible within *PMon*. As process actions such as *PA* 1 are individuals, we can add any number of object property assertions to a process element. Accordingly, process action PA_1 in the example seen in Figure 4.2 could have two object property assertions such that

- PA_1 *hasSubsequentProcessAction* PA_2 and
- PA_1 *hasSubsequentProcessAction* PA_3.

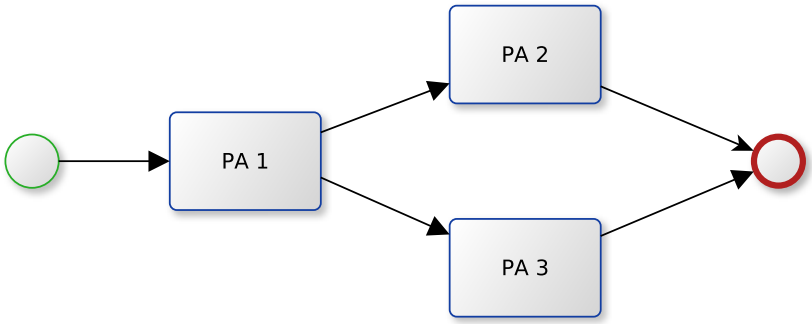

Figure 4.2: Implicitly modeled parallel split

Although this modeling is possible, we use dedicated control-flow elements which is known as explicit modeling as shown in Figure 4.3.

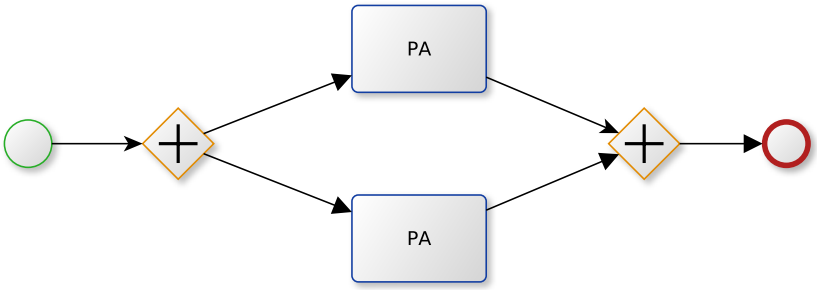

Figure 4.3: Explicitly modeled parallel split

Parallel splits for example are represented by an own class as modeled in Figure 4.1. Corresponding to the mapping of process actions, control-flow structures such as parallel splits are modeled as individuals within the ontology.

Furthermore, in the same way process actions are connected to other process actions in sequence, the link to other control flow elements such as parallel splits, events and so on is handled. We will describe the transformation patterns in greater detail and formally in Section 4.3.

Object properties

As discussed before, ontologies usually consist of a taxonomy, axioms and properties to describe their relations. *PMon* has a set of properties to describe elementary parts of process models such as control-flows as discussed in the chapter before Additionally information such as about

- stakeholders executing single process actions,
- swim-lanes grouping a set of process actions,
- which business/functional service is supported by an IT service,

and so on, require object properties and according object property assertions to represent this information.

The full set of object properties of *PMon* is shown in Table 4.1.

4.3 Transformation patterns

As mentioned before, lots of different workflow patterns exist (van der Aalst et al., 2000, 2003a). Process modeling notations such as BPMN and UML Activity Diagrams have been analyzed with regard to these patterns and whether the patterns are supported by the respective notation language (White, 2004a,b). Thom et al. showed (Thom et al., 2007) that a smaller subset of patterns may be sufficient to design real-world process models. This is backed by the empirical study of zur Muehlen et al. which states that a quite small number of workflow patterns is used in industry today (zur Muehlen and Recker, 2008). More precisely, the study showed that only nine elements (including workflow elements such as process actions as well as control flow structures) where used within BPMN models in reality.

Therefore, we will present detailed transformation patterns of the five basic workflow patterns and demonstrate that it is possible to represent more complex patterns as well. The five basic patterns are

- sequence,
- parallel split,

Table 4.1: Object properties within *PMon*

Domain	Property	Range	Comment
`ProcessModel`	*hasStartElement*	`StartEvent`	
`swim-lane`	*hasRole*	`Role`	
`ProcessElement`	*hasProceedingProcess Element*	`ProcessElement`	inverse to *hasSubsequentProcessElement*
`ProcessElement`	*hasSubsequentProcess Action*	`ProcessElement`	inverse to *hasProceedingProcessElement*
`ProcessElement`	*belongsToSwimlane*	`swim-lane`	
`ProcessElement`	*executes*	`Role`	inverse to *isExecutedBy*
`ProcessElement`	*isExecutedBy*	`Role`	inverse to *executes*

- synchronization,
- exclusive choice and
- simple merge.

We will formally describe the transformation from graphical notation languages (such as ARIS EPC, BPMN business process models or UML activity diagrams). Furthermore, we demonstrate the flexibility and power-fullness of the ontology by presenting transformations for swim-lanes and roles.

Sequence pattern

The sequence pattern is probably the most basic control-flow structure and defines that "an activity in a workflow process is enabled after the completion of a preceding activity in the same process." (Russell et al., 2006).

See Figure 4.4 for an example graphical notation. The transformation pattern is defined in Table 4.2.

Source	$n_1, n_2 \in \mathcal{N}, E_{out}(n_1) = n_2 \wedge E_{in}(n_2) = n_1$
Description	Process actions n_1, n_2 connected by edge e in a sequence control flow.
Destination	Create individuals n_1, n_2 as `ProcessAction`. Add object property assertion such that (e_1) n_1 *hasSubsequentProcessAction* n_2.

Table 4.2: Sequence pattern transformation rule

Figure 4.4: Sequence pattern: Graphical notation example

Parallel split pattern

The *Parallel Split pattern* is defined as the "divergence of a branch into two or more parallel branches each of which execute concurrently" (Russell et al., 2006).

See Figure 4.5 for an example graphical notation. The transformation pattern is defined in Table 4.3.

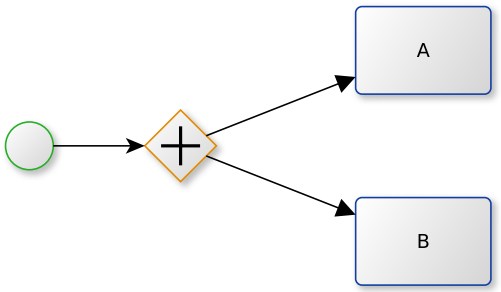

Figure 4.5: Parallel split: Graphical notation example

Source	$n_1, n_2, n_3 \in \mathcal{N}, s \in \mathcal{N}_{Split}, E_{out}(n_1) = s \wedge E_{out}(s) = [n_2, n_3], E_{in}(n_2, n_3) = s$
Description	Process action n_1 succeeded by n_2, n_3 which are to be executed in parallel, connected by edges e_1, e_2, e_3. n_2 and n_3 are activated in parallel, indicated by an explicitly modeled parallel split control flow pattern.
Destination	Create individuals n_1, n_2, n_3 as `ProcessAction` and s as `ANDSplit`. Add object property assertions such that • e_1: (n_1 *hasSubsequentControlFlowElement* s) $\wedge$ • e_2: (s *hasSubsequentProcessAction* n_2) $\wedge$ • e_3: (s *hasSubsequentProcessAction* n_3).

Table 4.3: Parallel split pattern transformation rule

Synchronization pattern

The *Synchronization pattern* is defined as "the convergence of two or more branches into a single subsequent branch such that the thread of control is passed to the subsequent branch when all input branches have been enabled" (Russell et al., 2006).

See Figure 4.6 for an example graphical notation. The transformation pattern is defined in Table 4.4.

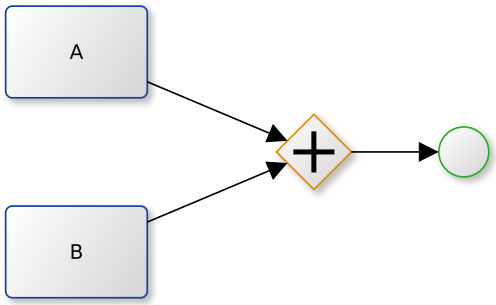

Figure 4.6: Synchronization: Graphical notation example

Exclusive choice pattern

The *Exclusive Choice pattern* is defined as "the divergence of a branch into two or more branches. When the incoming branch is enabled, the thread of control is immediately passed to precisely one of the outgoing branches based on the outcome of a logical expression associated with the branch" (Russell et al., 2006).

See Figure 4.7 for an example graphical notation. The transformation pattern is defined in Table 4.5.

Simple merge pattern

The *Simple Merge pattern* is defined as "the convergence of two or more branches into a single subsequent branch. Each enablement of an incoming branch results in the thread of control being passed to the subsequent branch." (Russell et al., 2006).

Source	$n_1, n_2, n_3 \in \mathcal{N}, j \in \mathcal{N}_{ANDJoin}, E_{out}(n_1, n_2) = j \wedge E_{out}(j) = n_3$
Description	Process actions n_1, n_2 succeeded by an AND join node j connected by edges e_1, e_2. j in turn is succeeded by node n_3, connected by edge e_1.
Destination	Create individuals n_1, n_2, n_3 as `ProcessAction` and j as `ANDJoin`. Add object property assertion such that • e_1: n_1 *hasSubsequentControlFlowElement* $j \wedge$ • e_2: n_2 *hasSubsequentControlFlowElement* $j \wedge$ • e_3: j *hasSubsequentProcessAction* n_3.

Table 4.4: Synchronization pattern transformation rule

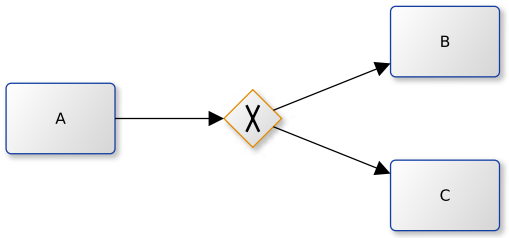

Figure 4.7: Exclusive choice: Graphical notation example

Source	$n_1, n_2, n_3 \in \mathcal{N}, x \in \mathcal{N}_{XORSplit}, E_{out}(n_1) = x \wedge E_{out}(x) = [n_2, n_3], e_1, e_2 \in E_{out}(x) \wedge$ Conditions c_1 on e_1, c_2 on e_2
Description	Process action n_1 succeeded by an exclusive choice (XOR) decision node x connected by edge e_1. x in turn is succeeded by nodes n_2, n_3, connected by edges e_2, e_3. Conditions c_1, c_2 defining which path to take should be added to edges e_2, e_3.
Destination	Create individuals n_1, n_2, n_3 as `ProcessAction` and x as `XORSplit`. Add object property assertion such that • n_1 *hasSubsequentControlFlowElement* $x \wedge$ • $edge_1$ as x *hasSubsequentProcessAction* $n_2 \wedge$ • $edge_2$ as x *hasSubsequentProcessAction* n_3. • Annotation *hasCondition* c_1 on object property assertion $edge_1$ • Annotation *hasCondition* c_2 on object property assertion $edge_2$

Table 4.5: Exclusive choice pattern transformation rule

See Figure 4.8 for an example graphical notation. The transformation pattern is defined in Table 4.6.

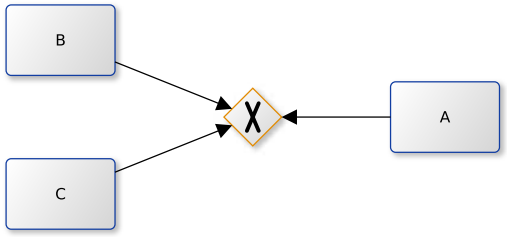

Figure 4.8: Simple merge: Graphical notation example

Source	$n_1, n_2, n_3 \in \mathcal{N}, m \in \mathcal{N}_{XORJoin}, E_{out}(n_1, n_2) = j \wedge E_{out}(m) = n_3$
Description	Process action n_1, n_2 succeeded by a merge node m connected by edges e_1, e_2. m in turn is succeeded by node n_3, connected by edges e_1, e_2, e_3.
Destination	Create individuals n_1, n_2, n_3 as `ProcessAction` and m as `XORJoin`. Add object property assertion such that • e_1: n_1 *hasSubsequentControlFlowElement* $m \wedge$ • e_2: n_2 *hasSubsequentControlFlowElement* $m \wedge$ • e_3: m *hasSubsequentProcessAction* n_3.

Table 4.6: Simple merge pattern transformation rule

Swim-lane and role patterns

The *Swim-lane and role pattern* are split into swim-lane and role pattern as their own patterns in the strict sens. We discuss them together because both are frequently used together in process modeling. *Swim-lanes* classify process

elements (usually process actions) into groups, often to demonstrate that such a single group or swim-lane is executed by a specific department, role or other group of people. While tasks within a swim-lane are associated to this group, a direct assignment of a role to a task denotes the same, but only for that single task. As depicted in Figure 4.9 "Role X" somehow executes task A while "Role Y" is responsible for task B. Swim-lane and role patterns represent the same behavior, while both have different suit of purpose in notation.

Roles describe which process actions are executed by which kind of role when the process is executed. Such roles typically describe qualifications and permissions such as power of procuration, approval for vacation requests and so on.

Russel et al. describe roles as follows: "A resource may have one or more associated roles. Roles serve as another grouping mechanism for human resources with similar job roles or responsibility levels e.g., managers, union delegates etc. Individual resources may also possess capabilities or attributes that further clarify their suitability for various kinds of work items. These may include qualifications and skills as well as other job-related or personal attributes such as specific responsibilities held or previous work experience" (Russell et al., 2005b).

Roles are of vital importance when instantiating and executing processes containing manual tasks such as approval for vacation requests. As human interaction is necessary, either the workflow engine or an operator has to assign the task to a genuine person. It would be quite inflexible to assign a concrete employee to a task in the process model itself as this required to change the process model every time an assigned person leaves the company, changes the department or field of activity. This is why today's workflow engines allow to assign groups to human tasks in the process model. Although this is better than assigning individuals, greater flexibility would be even more helpful.

We will discuss this issue and a semantic-based solution in Section 5.1.

See Figure 4.9 for an example graphical notation. The transformation pattern is defined in Table 4.7.

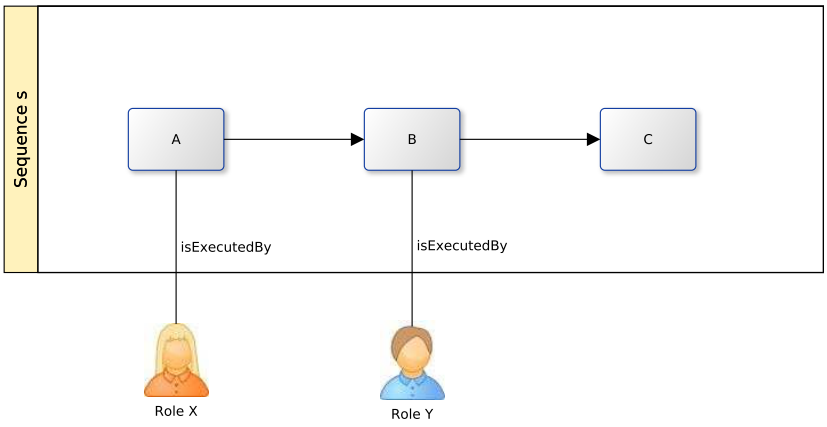

Figure 4.9: Swim-lane and roles: Graphical notation example

Conditions on control flow edges

Conditional flows are usually necessary to describe which paths should be activated at a `XOR` split. *PMon* supports this feature, too. In order to define conditions on edges (which are object property assertions technically) we use annotations on the particular object property assertion that in turn holds the respective condition.

Given two process actions $PA1, PA2 \in \mathcal{N}$ that have the condition $x == 1$ set on the edge an edge $e \in \mathcal{E}$ connecting both. Remember that the ontological representation defined by the patterns before maps process actions to individuals of the class `ProcessAction`. The individuals have an object property assertion *hasSubsequentProcessAction* set between them. The resulting code with the conditions set on the respective object property assertions can be seen in Listing 4.3.

That way any condition might be set on object property assertions.

Conclusion In this chapter we presented mappings of process model patterns that work for basically any kind of process notation language into ontology space. These mappings build upon the widely known workflow patterns (Russell et al., 2006; van der Aalst et al., 2003a). We presented three parts *source*, *description* and *destination* for each pattern.

As mentioned in the preliminary of this section most process models in

Source	$n \in \mathcal{N}, n \rightsquigarrow l \wedge l \in \mathcal{L}, n \longmapsto s \wedge r \in \mathcal{R}$
Description	Process action n is located within swim-lanes l and executed by role r.
Destination	Create individual n as `ProcessAction`. Find existing or create new swim-lane individual l and add object property assertion such that *n belongsToSwim-lane l*. Find or create role r and add object property assertion such that *n isExecutedBy r*.

Table 4.7: Swim-lane and role pattern transformation rules

industry can be represented with depicted workflow patterns and therefore be transformed into ontology space for further use. Nevertheless many other patterns can be transformed as well. Additionally, object property assertions can be used for, e.g., message flow, too. Furthermore, more complex workflow patterns influence runtime behavior only (e.g., Multi-Choice, Multi-Merge, Structured Discriminator, Blocking Discriminator, Thread Merge, Thread Split). Such behavior could be integrated into ontological representation if runtime accessed the respective ontological space, too.

4.4 Supported graphical notation languages

Although manual transformation of existing process models into ontology space already would be valuable in order to connect to other ontology-managed data and finally to make use of reasoning capabilities, the necessary effort to do so is enormous. Because manual transformation of existing models is expensive and error-prone, we developed a model-driven transformation method, enabling automatic transformation from several popular graphical notation languages into ontology space. We designed this method in a modular way so that additional notation languages can be added easily.

Currently the following notation languages are supported.

- Eclipse Java Workflow Tooling (JWT) (Eclipse, 2010),

```
1
2  <owl:Thing rdf:about="#PA1">
3      <rdf:type rdf:resource="#ProcessAction"/>
4      <hasSubsequentProcessAction rdf:resource="#PA2"/>
5  </owl:Thing>
6
7  <owl:Thing rdf:about="#PA2">
8    <rdf:type rdf:resource="#ProcessAction"/>
9  </owl:Thing>
10
11 <rdf:Description>
12   <rdf:type rdf:resource="&owl;Axiom"/>
13   <hasCondition>x == 1</hasCondition>
14   <owl:subject rdf:resource="#PA1"/>
15   <owl:object rdf:resource="#PA2"/>
16   <owl:predicate rdf:resource="#hasSubsequentProcessAction"/>
17 </rdf:Description>
```

Listing 4.3: Conditions on edges within PMon

- Business Modeling Notation (BPMN) or rather BPMN 2 using Oryx (Decker et al., 2008a,b),

- Architecture of Integrated Information Systems (ARIS) EPC (Davis, 2008)

We implemented tool support capable of transforming such models into the meta model presented in Section 4.1. From this intermediate model, a transformation into the destination ontology is realized. We will present the details of this implementation in the following.

4.5 Implementation

In order to demonstrate applicability of automatic transformations of existing workflow models into ontology space, we implemented the transformations using a model-driven approach. We will discuss the overall architecture of the implementation in the following and show details of the transformations in Section 4.5.2.

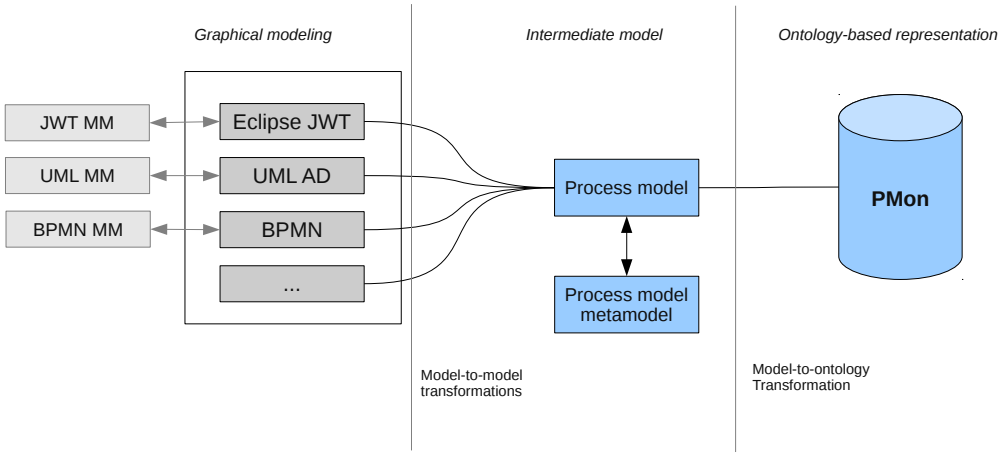

Figure 4.10: Big Picture: Transformations from graphical models into ontology space

4.5.1 Architecture

We will shortly describe architectural details of our implementation which is based on the Eclipse project and Eclipse Modeling Framework (EMF). As shown in Figure 4.10 the transformation process consists of two transformation steps. The first transformation into the intermediate model is necessary because of the following. As mentioned before, the implementation of the transformations currently supports multiple graphical notation languages as sources. Those in turn have different powerfulness in terms of graphical notation possibilities and therefore support different features. As the destination ontology has a defined (restricted) set of supported features, we used an initial transformation to smooth the source models to be consistent to our metamodel as shown in Figure 4.1. Additionally, the different graphical notation languages differentiate slightly in the concrete modeling syntax, e.g., explicit and implicit parallel split which is standardized in this step. Furthermore, this step allows adding more languages easily.

This initial transformation is implemented by using Extensible Stylesheet Language (XSL) Transformation (XSLT) and brings the various source model formats into EMF meta model (Ecore) format and our metamodel. BPMN, e.g., supports complex sub-process, pool, swimlane-pool and swimlane constructs

which we did not fully map to the ontology implementation. Technically the constructs can be represented within ontology space if necessary.

In the next step, OWLAPI (Horridge and Bechhofer, 2011) is used to transform the concepts into ontology space considering the presented transformation patterns. This is implemented by using Tefkat Query / View / Transformation (QVT) which is available as plugin for EMF. We will show some details of both transformations in the following.

4.5.2 Model Transformations

In this section we will show the two parts of transformation as well as a short example of both transformation steps. Please see Appendix 8.3 for a full overview of the concrete transformations.

Listing 4.4 shows the transformation of simple BPMN tasks from source models into the intermediate model.

```
1  <!-- Start events -->
2  <xsl:template match="process" mode="startEvent">
3    <xsl:for-each select = "./startEvent">
4      <xsl:element name="startEvent">
5        <xsl:attribute name="xsi:type">metaBPMN2:event</
             xsl:attribute>
6
7          <xsl:attribute name="processid">
8            <xsl:value-of select="./../@id"/>
9          </xsl:attribute>
10
11         <xsl:attribute name="id">
12           <xsl:value-of select="./@id"/>
13         </xsl:attribute>
14
15         <xsl:attribute name="name">
16           <xsl:value-of select="./@name"/>
17         </xsl:attribute>
18     </xsl:element>
19   </xsl:for-each>
20 </xsl:template>
21
22
23 <!-- Simple tasks -->
24 <xsl:template match="process" mode="task">
25   <xsl:for-each select = "./task">
26     <xsl:element name="task">
```

```
27        <xsl:attribute name="xsi:type">metaBPMN2:task</xsl:attribute
              >
28        <xsl:attribute name="processid">
29          <xsl:value-of select="./../@id"/>
30        </xsl:attribute>
31
32        <xsl:attribute name="id">
33          <xsl:value-of select="./@id"/>
34        </xsl:attribute>
35
36        <xsl:attribute name="name">
37          <xsl:value-of select="./@name"/>
38        </xsl:attribute>
39      </xsl:element>
40    </xsl:for-each>
41  </xsl:template>
```

Listing 4.4: Transformation into intermediate model: Start events

As mentioned in Section 4.5.1, the second step is implemented employing QVT transformations. We decided to use QVT in this step because of better readability of the rules and the possibility to easily integrate OWLAPI in order to access OWL ontologies. This transformation maps the intermediate model into ontology space finally. Listing 4.5 shows the transformation using QVT. The listing shows how two process actions PA_1 and PA_2 get converted while the control-flow is set by using the *hasSubsequentProcessAction* object property assertion as described in Section 4.3.

```
1  RULE SequenceTT(T, T)
2    FORALL  task T1, task T2
3    WHERE   linkedElements(T1,T2,KA)
4
5    MAKE    ProcessAction PA1,
6            ProcessAction PA2,
7            Edge K, Edge K1, Edge K2
8
9    SET     PA1.name=T1.name, PA1.id=T1.id, PA1.swimlaneid=T1.
         processid,
10           PA2.name=T2.name, PA2.id=T2.id, PA2.swimlaneid=T2.
                 processid,
11           K.name="hasSubsequentProcessAction", K.startid=PA1.id,
12           K.zielid=PA2.id, K.id=KA.id,
13
14           K1.name="isPartOfSwimlane", K1.startid=PA1.id,
15           K1.zielid=PA1.swimlaneid, K1.id=append(PA1.id, PA1.
                 swimlaneid),
16
```

```
17          K2.name="isPartOfSwimlane", K2.startid=PA2.id,
18          K2.zielid=PA2.swimlaneid, K2.id=append(PA2.id, PA2.
                swimlaneid)
19  ;
```

Listing 4.5: Transformation into ontology space: Two process actions connected by edge

A detailed list of the transformation patterns is shown in the appendix. We used the tooling described in this section to transform the running example into an OWL ontology to further show querying on the model. We will show details of this in the next section.

4.6 Running example

Using the transformations described in the section before, we transformed the running example as introduced in Section 2.4 into *PMon*. As mentioned in the introduction, one of the goals of transforming models into ontological space is to exploit results of reasoning functionality on the formal logic of process models. In order to demonstrate more realistic queries on the process model introduced, we slightly modified the running example by adding data inputs, outputs and IT-services to process actions.

The resulting example for this demonstration is shown in Figure 4.11.

After transforming the process model into ontology space, querying on the ontological representation can be achieved. As introduced during the motivation for our approach, enterprise architectures often get complicated and non-transparent. Thereby, questions that arise could include for example:

- Which processes make use of a certain IT service?
- How is input/output data used throughout a (large) process?

We implemented a set of demonstrating classification possibilities and querying examples. We used the Protégé-OWL editor (Knublauch et al., 2004) with various reasoners for Manchester Syntax queries. We tested and frequently used these reasoners: Pellet (Sirin et al., 2007), FaCT++ (Tsarkov and Horrocks, 2006) and HermiT (Shearer et al., 2008). We submitted SPARQL queries to OpenRDF Workbench (Broekstra et al., 2002) (formally

known as Sesame) which is primarily designed as RDF triple-store but can answer queries on OWL ontologies using the OWLIM plugin, too (Kiryakov et al., 2005).

Use of IT systems

In order to find out if an IT service is used, we query the ontology *which process actions use the Oracle RDBS service*?

In Manchester Syntax this is expressed as follows:

ProcessAction and *utilizes* some {**Oracle_RDBS_Service**}

In SPARQL the respective query looks as follows.

```
1 SELECT ?process_actions
2 WHERE {
3     ?process_actions rdf:type PMon:ProcessAction .
4     ?process_actions PMon:utilizes PMon:Oracle_RDBS_Service
5 }
```

Listing 4.6: SPARQL query

In our example, the reasoner returns two individuals *Update_personell_records* as well as *Enter_Payroll_Data* which matches our expectations as both were modeled to use the service in question. [1]

One could easily query for the inverse, i.e., which IT services are used by process actions. In order to do so, we define a property *itUtilizedBy* which is marked as an *inverse* property of *utilizes*.

Service and *isUtilizedBy* some {**ProcessAction**}

Listing 4.7 shows the SPARQL equivalent again.

```
1 SELECT ?services
2 WHERE
3 {
4   ?services rdf:type BusinessProcess:Service ,
5   ?x BusinessProcess:isUtilizedBy BusinessProcess:ProcessAction
```

[1] Please note that, we replace spaces in labels of process actions with underscore characters during the transformation process.

```
6 }
```

Listing 4.7: SPARQL query

The query returns the individual *Oracle_RDBS_Service* as expected.

Use of data

In order to query the ontology about which data is used as input throughout the process model, this query looked like the following:

DataElement and *isInputFor* some **{ProcessAction}**

Invalid process actions

Process actions can be invalid due to a multitude of possibilities. E.g., a process action having an edge to itself is obviously invalid. This can be expressed within the ontology be defining the following axiom.

InvalidProcessAction EquivalentTo
dlClassInvalidProcessAction and *hasSubsequentProcessAction* self

After running the reasoner's classification process on the ontology, the anonymous classes that define, e.g., invalid process actions as defined above or actions having IT support, contain those individuals that match the class description. These classes in turn can be used by consecutive steps, e.g., simply displaying the classification results or by including in other, more complex queries by referring to those classes.

We demonstrated some basic querying examples in this section that show how reasoning capabilities can be used for process models that are transformed into ontology space.

Admittedly, those examples are simple and the results are expected. Nevertheless, especially for larger process models that contain hundreds or thousands of process actions, data, IT services and stakeholders, such querying capabilities could assist in navigating through and understanding in-transparent correlations, detect unused parts and help to keep the models syntactically and semantically correct.

4.7 Conclusion

In this chapter we presented an approach to enable

- definition of process models including control flow in ontology space,
- transformation of graphical model notations to ontology-based models automatically and
- querying ontology space for model properties.

We demonstrated applicability of the transformations by two examples and showed how the questions compiled in the introduction of this section can be answered within ontology space.

On the one hand this approach allows bridging the world of formal enterprise modeling and graphical models:

1. We have shown how a graphical representation of a business process model can be converted into a purely semantic based representation that enables exploiting many of the envisioned benefits regarding the automatic process-ability of process models. The latter enables querying process models, e.g., regarding the use of resources, related services or responsibilities.
2. The presented intermediate meta model allows other representations of graphical modeling languages to adopt the methodology to achieve an automatic transformation.

After the definition of process models within ontological space and the transformation patterns for existing models presented in this chapter, we will continue with the integration and combination of resources with process modeling in the next chapter.

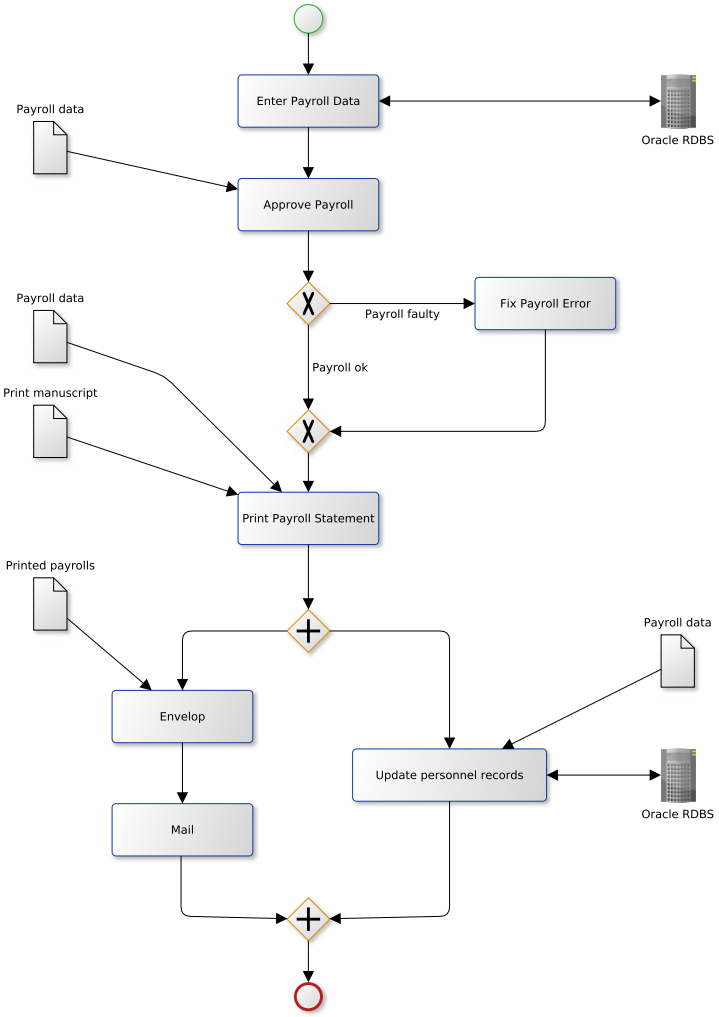

Figure 4.11: Running example extended by data inputs, outputs and IT services

Chapter 5

Resource constraints on process models

In this chapter we will introduce an approach to check executability of process models considering resource requirements *at design-time*. Therefore, we will introduce a lightweight approach to describe resources within ontologies, attach resource requirements within process modeling space and finally accomplish the checking. We will show how this approach also extends a semi-automatic adaptation approach. Finally, we describe prototypical implementations that shows the applicability of the theory.

5.1 Modeling resource constraints

As described in Chapter 2 process models are an abstraction while *executing* (or instantiating) process models means to put them into reality. At design-time, basically anything can be modeled because resources that execute and support the tasks are usually not considered. Though, those resources have to be available at execution time finally in order to successfully execute a process. Therefore, "it is not sufficient to simply focus on control-flow and data issues when capturing business processes, the resources that enable them need to be considered as well" (Russell et al., 2005b).

Typically, resources at companies include staff, machines, and many other services or objects necessary for daily business. We will discuss our working definition for resources hereafter.

Usually restrictions that appear and apply during runtime do not necessarily

influence design-time. That means that even if there is a process action in a model that demands a certain action to be completed, there is no safety or evidence that this really happens during execution of the process. Consequently, *resources* necessary for a process action can (but do not have to be) available at runtime. Figure 5.1 shows a very simple example with only one process action *PA*. Given a special person or machine is necessary to complete the process action *PA* (and this machine person is not available at the company) the model is perfectly valid at design-time but will not be executable at runtime due to the missing resource. Because of this information gap between both phases the value of process models suffers.

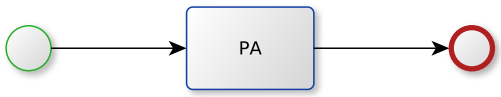

Figure 5.1: Simple process model, yet possibly non-executable.

It usually cannot be assumed that the person designing process models simply takes care of such problems at design-time: On the one hand, resources are hardly to be known by single individuals due to their enormous quantity especially in large enterprises. On the other hand, there might be several possibilities to satisfy a resource requirement: Let's assume a process action needs a software architect and a researcher to be completed and there are three people (human resources) that are worth considering to comply with the requirements.

- Person *A* is a researcher.
- Person *B* is a software architect.
- Person *C* is both a researcher and a software architect.

If a process model designer should attach one of the above concrete resources to the task requiring a software architect and a researcher, there would be already multiple possibilities in this simple case (*A* and *B* or *A* and *C* or *B* and *C* or probably only *C*). Obviously there are dozens of such roles, abilities, skills and authorities in reality so the possible combinations are tremendous. This is why we state that automated, computer-based support is necessary for the

matching of resource requirements with an appropriate resource knowledge base.

A manual assignment of resources to process actions implies a magnitude of possible solutions in process models which in turn results in even greater manual effort. Additionally, traditional graphical modeling languages such as BPMN do not require nor do most of them support integration of complex requirements into models. See Table 5.1 for an overview of process modeling languages capable of the features necessary to model resource requirements as well as to check models for such requirements. The table lists features and a number of popular and widespread notation languages. A positive symbol (✓) indicates, that the notation language supports the respective feature while the second symbol (-) indicates, that the feature is unavailable in the language.

The features listed include the most basic possibilities for resource requirement definition (swimlanes and roles) that are supported by basically any graphical notation languages. The feature 'Definition of complex resource constraints' refers to the possibilities to describe requirements that both include (multiple) roles together with an arbitrary number of skill requirements. The integration with a resource knowledge base describes the link of the resource requirements with a knowledge base in which the actual resources of an enterprise can be modeled. The 'Check model for resource consistency' describes the matching of the requirements within process models with this knowledge base. The requirement for an ontology based definition of resources is obvious while the last feature regarding quantification of models refers to the possibility to evaluate models with different resource assignments regarding costs, for example.

We developed an approach to decide about the feasibility of process models based on existing resources within enterprises represented in ontologies. Therefore, we combine two models which are expressed in two separate Technical Spaces (TSs): process models are normally represented in the Metamodeling Technical Space (MMTS), whereas a model of the available resources is expressed in the Ontological Technical Space (OTS). Both models are linked via a mapping which translates between both TSs. In contrast to complex resource planning approaches, our resource analysis approach is supposed to offer a fast and lightweight evaluation whether a company has, in principle, the necessary resources to execute a given process. That is,

[1]YAWL intentionally leaves this to the runtime environment that is shipped together with the modelling tool.

Feature	BPMN 1	BPMN 2.0	YAWL	UML AD	ARIS EPC	Our approach
Swimlanes	✓	✓	✓	✓	✓	✓
Roles	✓	✓	✓	✓	✓	✓
Definition of complex resource constraints	-	-	✓	✓	-	✓
Integration with resource knowledge base	-	-	✓	-	✓	✓
Check model for resource consistency	-	-	-[1]	-	-	✓
Ontology-based resource ontology	-	-	-	-	-	✓
Quantification of models based on resource properties	-	-	-	-	-	✓

Table 5.1: Process modeling languages supporting modeling of resources

instead of planning the execution of process actions, our approach validates that there is at least one resource available for each resource requirement of a process. On the one hand, this means that we neglect time constraints induced by, e.g., concurrently executed process instances. On the other hand, this focusing allows the presented approach to be used with less detailed processes. The result of the resource analysis is a set of non-executable process actions and the according missing resources.

This approach contributes to two major research areas. Firstly, the presented approach allows companies to face the following challenges:

1. *Assurance of feasibility of dedicated projects according to required internal or external resources*. Even prior to project realization, project managers must know, whether or not a project can be realized with the available resources. By applying our resource-based analysis, an automated and detailed feasibility evaluation becomes possible. Therefore, project teams with fitting skills can be arranged more purposefully.

2. *Assurance of standard compliance according to available resources*. Process

improvement frameworks, such as SPICE or CMMI, define the availability of particular artifacts and the execution of dedicated activities. This often neglects that all activities are enabled and executed by resources, which have to be available, too. By applying the automated resources analysis, the feasibility of all activities and the so produced artifacts can be ensured.

3. *Provisioning of a company- or customer-specific resource delta analysis*. By applying a company-wide resource analysis, it is possible to determine what kind of resources a company or a customer is missing to reach a particular compliance level or some other goal, which depends on individual resources. Additionally, such analysis allow detection of dependencies from single resources, i.e., resources that are required in highly critical processes, but are available only once.

Secondly, our approach also enables the automatic planner SEMPA to extend the planning of new process models with regard to available resources. Our extension selects those process models from the set of feasible solutions that are executable evaluating the available resources. From this set, we select the model that is valuated the "best" model, e.g., regarding costs or energy consumption. An overview of this extension is shown in Figure 5.2. The details of this enhancement will be discussed in this chapter.

In the next section we will describe details of our approach. We will introduce an ontology-based resource knowledge base, describe the modeling of resource requirements and the attachment of resource requirements to processes. Furthermore, we will introduce an approach to achieve the matching of requirements and available resources and show how this extends SEMPA. Finally, we will describe the algorithms as well as implementation details.

5.2 Ontology-based specification of resources

As discussed in Chapter 3, resources in general have been topic in numerous scientific publications and research areas.

Unfortunately none of the presented existing approaches build upon both standardized and actively supported Semantic Web technologies such as OWL which does satisfy those characteristics as exposed in Chapter 2. That is why we required the semantic definition of resources to be implemented by using OWL or rather OWL 2 (Hitzler et al., 2009a).

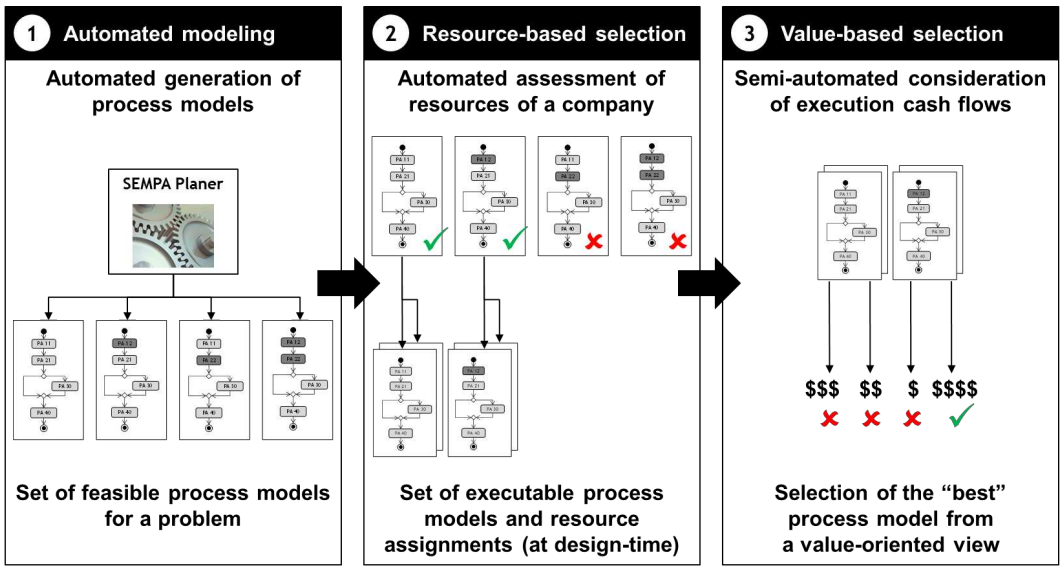

Figure 5.2: Approach to select executable process models with input from SEMPA

As mentioned before, the existing enterprise ontologies such as the ontologies from TOVE or REA are quite complex, very detailed and full-blown. Additionally, as discussed in the related work before, many approaches aim to build upper ontologies which result in a far more detailed description of, e.g., enterprises and markets than we need. We state that this complexity prevents widespread usage of such ontologies in industry today, because the barrier is too high — especially for small- and medium-sized enterprises.

We were striving for an agile, lightweight approach for both modeling resources as well as defining resource requirements to enable ontology-based, formal resource definition and usage to ensure the barrier for introducing modeling of resources is as low as possible. Another requirement was support for an open, standardized, formal and logics-based description language.

Basically there are two components necessary in order to define suchlike resources. *Abstract resources* cover the taxonomy, whereas *concrete resources* refer to actual existing real-world resources. We will use the following definitions of both terms:

Definition 5.2.1 (Abstract Resource) *An abstract resource $\mathcal{R}_a$ is a taxonomy concept that defines a class of resources.*

Definition 5.2.2 (Concrete Resource) *A concrete resource $\mathcal{R}_c$ is an existing physical or virtual asset that can be demanded during a business process. Each concrete Resource $\mathcal{R}_c$ must be assigned to one or more abstract resources $\mathcal{R}_a$.*

In order to determine the amount of resources available the quantity of an abstract resource $|r|$ is defined as the quantity of concrete resources of an abstract resource.

Although we aimed at enterprise resources as detailed above in the first step, we include all production facilities, utilities, and services that can be expressed with the model we described before.

Figure 5.3 shows the process modeling space on the left, the resource modeling space on the right and the mapping component in the middle of the picture. The right part depicts the *RESon*, i.e., a model of the resources within a company or department represented in the ontology. On the one hand, this model conceptualizes company-internal resources, and builds up a *resource taxonomy* of skills and infrastructural capabilities of interest which is what maps the concept of abstract resources ($\mathcal{R}_a$) as defined in Definition 5.2.1 into the ontology. On the other hand, it also comprises the *resource pool* of the company, i.e., concrete employees and infrastructure ($\mathcal{R}_c$). By putting both into one ontology, it is possible to assign each employee to the skills she has and, respectively, assign each tool to the capabilities it provides.

Abstract resources The **resource taxonomy** additionally contains a hierarchical definition of the skills, and the capabilities which are available or interesting within a company. Thereby, skills refer to concrete knowledge, capabilities, or access rights which an employee can have, e.g., tax law knowledge, project management skills, commercial procuration or access rights to personnel files to name a few. In order to come up with a reasonable taxonomy of the skills, those should be categorized and aggregated to umbrella terms as this is recommended from ontology modeling practice. For instance, programming skills in *Java* and *C* are classified as individuals of the more general skill *programming* which in turn is a subclass of the generic skill classification class *computer science*. Please note that we merge labels consisting of multiple parts by using the upper case delimiter strategy also known as CamelCase. Additionally, we usually attach the label of parent classes as it improves readability within continuous text. Therefore, we will refer to, e.g., the Java skill as JavaProgrammingSkill in the following. Refer to (Fliedl et al., 2007) for a discussion on naming strategies in ontologies.

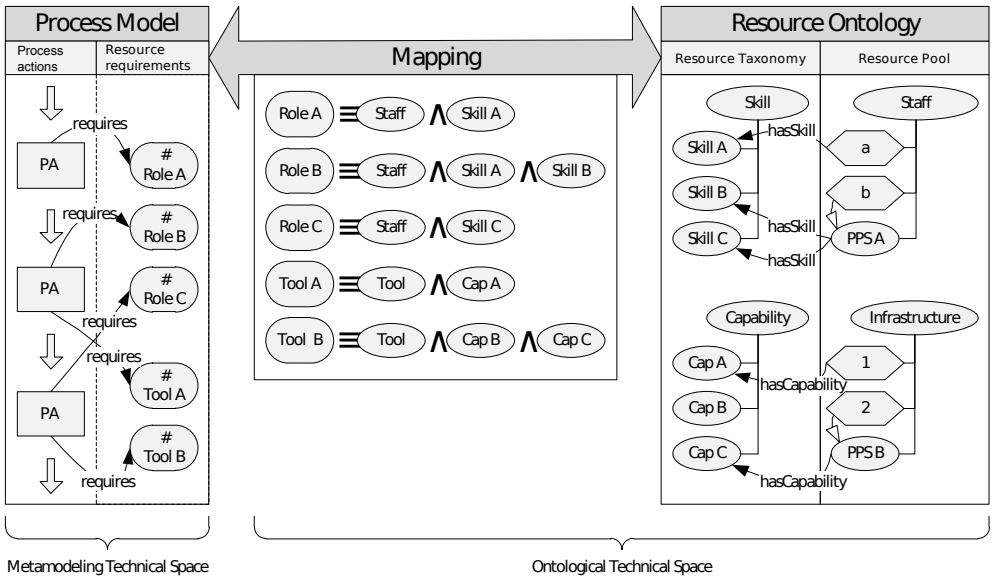

Figure 5.3: Process modeling, resources and mapping of both: Big Picture

Similar to the skill definition, the capabilities definitions are build. A capability refers to a functionality of infrastructure components such as a printing machine for example. The resource taxonomy defining abstract resources $\mathcal{R}_a$ is implemented by using OWL *subClassOf* relation definitions which describe the relationships as shown in Listing 5.1. The taxonomy components are depicted as ellipses on the right side of Figure 5.3.

The information for building the resource taxonomy can be gathered from experience or knowledge of experts. Another source of information (especially if experience in resource modeling is not available within an enterprise) are bodies of knowledge such as Software Engineering Body of Knowledge (SWEBOK) (Abran et al., 2004).

Concrete resources In our approach, the **resource pool** is a collection of concrete resources ($\mathcal{R}_c$). This consists of members of staff and instances of tools which are depicted as hexagons on the right side in Figure 5.3. For instance, there can be a representation of the real employee Bob and the Laboratory X.

Concrete resources $\mathcal{R}_c$ are modeled as OWL *individuals*. As defined in the

```
<owl:Class rdf:about="#ComputerScience" />
  <rdfs:subClassOf rdf:resource="#Thing"/>
</owl:Class>

<owl:Class rdf:about="#SoftwareDevelopmentSkill">
  <rdfs:subClassOf rdf:resource="#ComputerScience"/>
</owl:Class>

<owl:NamedIndividual rdf:about="#SoftwarePatternsSkill">
  <rdf:type rdf:resource="#SoftwareDevelopmentSkill"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="#JavaProgrammingSkill">
  <rdf:type rdf:resource="#SoftwareDevelopmentSkill"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="#CProgrammingSkill">
  <rdf:type rdf:resource="#SoftwareDevelopmentSkill"/>
</owl:NamedIndividual>
```

Listing 5.1: *RESon* fragment describing skills

OWL standard, an individual can be member in one or more classes which in turn has a positive effect on our modeling with regard to flexibility in our case (Hitzler et al., 2009a). For example, a concrete resource may be both a developer as well as project manager.

Each resource is linked to the skills and capabilities it provides via a hasSkill or hasCapability relationship, respectively. Both properties are object properties within OWL. The concrete definition of a single skill to a concrete resource is implemented by an object property assertion between the concrete resource (OWL individual) as source and the respective skill (individual as well) as destination.

Figure 5.4 shows an example model. Two concrete resources of the same kind Developer exist. Both Alice and Bob are classified as such Developers while Alice has a skill JavaProgrammingSkill associated and Bob has two skills JavaProgrammingSkill and CProgrammingSkill.

This allows a very detailed and concrete specification of the abilities of all concrete resources within the resource knowledge base.

Predefined Property Sets (PPSs) Usually resources are grouped with regard to their skills or capabilities. For example, an IT company might have

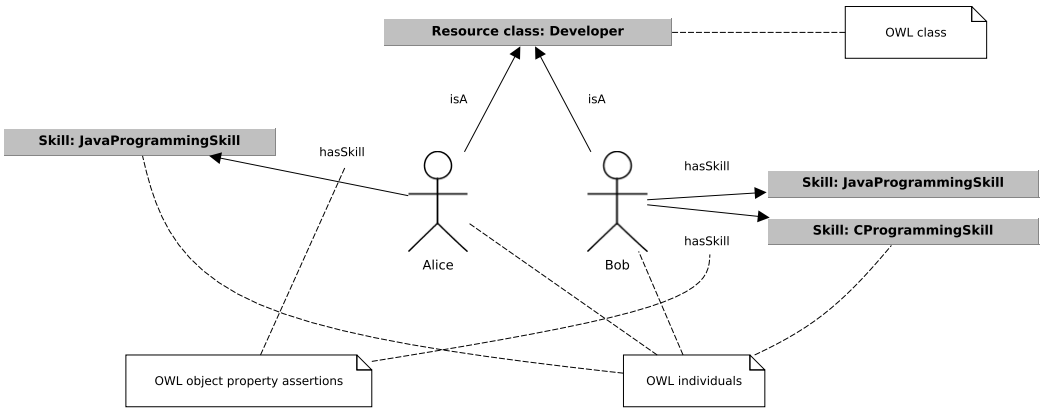

Figure 5.4: Ontology based resource modeling: Example

a software development and system operations department. Within those, the members of the software development department have knowledge in programming and software engineering methodologies while members of the system operations department are experts for distributed systems, database systems or Unix operations. All members of the departments have certain *skills* which makes them programmers or administrators respectively.

Those groups define a specific *set* of skills or capabilities. In our approach, we call them *Predefined Property Set*. PPSs are depicted as ellipses in Figure 5.3 below the resource taxonomy elements. Actually, PPSs can be seen as shortcuts for skill assignment by assigning a concrete resource to a PPS instead of repeatedly adding multiple skills.

We implemented this by using OWL 2 equivalent class definitions with value restrictions on the skill properties as shown in Listing 5.2.

Figure 5.5 shows an example PPS definition that corresponds to the PPS definition in Listing 5.2. In the example, the concrete resource Alice is member of the PPS class JavaProgrammer. This in turn causes the concrete resource to automatically inherit the skills JavaProgrammingSkill and SoftwarePatternsSkill as defined in the PPS.

Querying the ontology to return all resources that have the SoftwarePatternsSkill, Alice would be contained within the result-set because of the PPS assignment even though there is no explicit modeling of that skill within *RESon*.

```
<owl:Class rdf:about="&ResourceOntology;JavaProgrammer">
    <owl:equivalentClass>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&ResourceOntology;
                hasSkill"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:oneOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="&
                            ResourceOntology;JavaProgrammingSkill
                            "/>
                    </owl:oneOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </owl:equivalentClass>
    <owl:equivalentClass>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&ResourceOntology;
                hasSkill"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:oneOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="&
                            ResourceOntology;SoftwarePatternsSkill
                            "/>
                    </owl:oneOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="&ResourceOntology;Resource"/>
</owl:Class>
```

Listing 5.2: PPS example “Java Programmer”

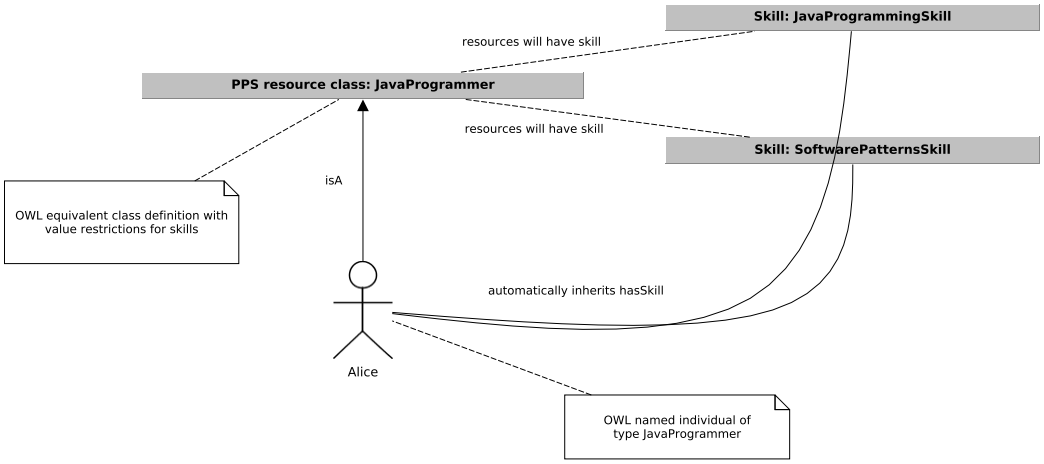

Figure 5.5: PPS example "Java Programmer"

An advantage of using ontologies for describing the resource ontologies are the possibilities to exploit advanced properties such as, e.g., to define equivalence of two different skills in case two departments get merged. The reasoner automatically handles both interchangeably then.

All in all, the *RESon* acts as a resource and skill database for human resources and, additionally, contains equipment and tools including their respective capabilities. This knowledge base is developed for a company once, and has to be kept up to date. Subsequently, it can be exploited in our approach to validate the feasibility of process models from a resource-oriented point of view.

5.3 Annotating models for resource consideration

In order to exploit the *RESon* for a feasibility analysis of a process, a translation between the resource requirements (also referred to as Resource References (RRs)) defined within the process models and the skills and capabilities in the resource ontology is necessary. We will refer to the requirements of a single process action PA as $RR(PA)$ in the following.

In order to provide a sound foundation, we took a well-known meta model by (Henderson-Sellers and Gonzalez-Perez, 2008) and added the components necessary for resource considerations. We chose this model because it is

generic and thus can be applied to any process modeling technique in various domains.

Therefore, we extend the metamodel with a *Resource Fragment (RF)* as shown in Figure 5.6 (b). The additional fragment indicates, that the selection of a Method Chunk (MC) is affected by resource-related factors, as well. A MC basically relates to a process model in our case. Similar to product and process fragments, the RF is a container that holds detailed resource-related information, which depend on the applied process definition metamodel. Put simply, a RF contains various resources, which are necessary to support the realization of the assigned MC. The information about resources is twofold: on the one hand, the RF holds RRs which represent specific roles or tools, and, on the other hand, an RR provides a cardinality attribute to indicate the required quantity of the respective resource. For example, a method like double-blind review needs two roles of the same type to realize the four-eyes principle.

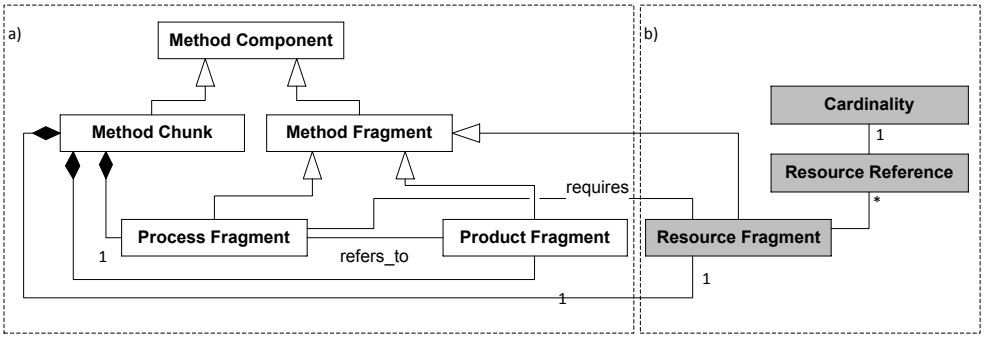

Figure 5.6: Meta model for method components according to (Henderson-Sellers and Gonzalez-Perez, 2008) extended by RFs

This is provided by the mapping depicted in the middle of Figure 5.3. It can be modeled as a separate ontology. Another way that is somewhat easier to implement is to store the requirements directly within process models using a description language (Manchester syntax) to define requirements. Whatsoever, it technically belongs to the OTS. The basic idea of the mapping is to define a RR as an aggregation of concepts from the resource taxonomy, i.e., classes for abstract resources or individuals for concrete resources, skills and capabilities.

A mapping between a concrete RRs and concrete skills or capabilities is represented as an ontological equivalence. Each RR is represented as a distinct concept in the mapping. Thereby, the matching between RRs in the OTS and in the MMTS can be based on, e.g., name equality. Now, the ontological RR is defined to be equivalent to either an employee with a set of skills or a tool with several capabilities. An example of such a mapping is:

```
software architect is equivalent to Staff
                                    and hasSkill software_architecture
                                    and hasSkill project_management .
```

To enable the usage of the resource taxonomy in the mapping, it imports the whole *RESon*. Hence, no specific matching between skills and tools in the mapping and in the resource taxonomy is necessary because both reside in the OTS.

In general, the set of skills and capabilities for a RR is interpreted as a conjunction, i.e., the employee or tool has to have *all* stated skills or capabilities. However, depending on the expressiveness of the utilized ontology language, this simple semantics can be extended by more complex structures like disjunctions as well. The cardinality property as shown in Figure 5.6 (b) that is necessary to define, e.g., a four-eyes principle, has to be a feature of the querying language or must be implemented using multiple queries.

This would allow to express a logic like a requirements engineer has a skill in use cases or user stories. Note, that the mapping defines a translation for RRs, hence, the cardinalities for the RFs are neglected.

This allows us to define arbitrary complex requirement definitions.

The mapping can be seen as the definition of a matching between the RRs, i.e., RFs, an arbitrary model definition language and the resource taxonomy of a company. Hence, the approach of decoupling process model and resource ontology allows reusing the company's body of knowledge for analyzing the feasibility for processes independently from the used model definition language. It is solely necessary to define the mapping once for each combination of model definition language and resource taxonomy. Note that, however, it has to be ensured that the names of the RRs in the language are unique, i.e., no two different RRs have the same name in case the RRs are kept within the separate ontology. If resource requirements are stored within models as discussed earlier, this is irrelevant.

In this section, we defined an elementary, yet powerful way to describe resource requirements for process models. This definition describes which resources are necessary to successfully execute a process action. We will continue with algorithms that allow to decide about feasibility of process models regarding resource constraints.

5.4 Consistency check for process elements

In this section, we show how the resource requirements definitions within process models can be used to check against the resource knowledge base *RESon* to decide about consistency of the model. Therefore, we firstly describe how a single process action having a RR attached gets checked. Secondly, we show how this mechanism is used for a full process model. Additionally, we will show how the extension can be used in conjunction with SEMPA.

Checking single process actions The analysis of a single process action is accomplished as follows: At first, the number of required resources is determined from the RR considering the cardinality attribute. All resources which match the required RRs are queried from *RESon*, i.e., staff or tools. The reasoner automatically returns resources considering the required skills and capabilities, i.e., only staff and tools are returned, that meet all skill and capability requirements. The amount of required resources is compared to the existing resources from *RESon* and marked as sufficient and therefore executable if the available number equals or exceeds the required resources. We will describe how to analyze process actions executed in parallel later on.

In order to gather the concrete number of resources from the ontology, the algorithm queries the *RESon*. Behind the scenes, this query matches the RR in the MMTS to the RR concept in the mapping, i.e., the OTS. As outlined, this can be accomplished by using name equality or storing queries directly within models. Subsequently, this concept can be translated via the mapping into either an employee with a given set of skills or a tool with a specific set of capabilities. Using logical reasoning, the resource pool can now be queried for all resources, i.e., staff or tools, which fulfill the requirements. The resulting resources are counted and their number returned to the algorithm. This simple algorithm is shown in Algorithm 1 and will be used in further constitutive algorithms in this chapter.

A resource requirement for a single process action *PA* is *satisfiable*, if all re-

Algorithm 1 query_resource_ontology

Input: required_resource
Input: required_amount
Output: true if enough resources within RESon exist, false otherwise

```
1: if OWLAPI::Query(RESon, required_resource) >= required_amount then
2:     return true
3: else
4:     return false
5: end if
```

quired resources are available within *RESon* with sufficient quantity. ($RR_S(PA)$: $\forall\ rr\ \in RR(PA) : \exists r \in RESon$ fulfilling requirement rr such that $|r| >=$ required quantity). As a process action possibly contains multiple resource requirements, checking a process action requires to iterate through each of the requirements and use the above algorithm to ensure executability of all individual requirements. The appropriate algorithm is shown in Algorithm 2.

Algorithm 2 Check executability of single process action

Input: Process action $n \in \mathcal{N}$
Output: true if process action is executable, false otherwise

```
1: executable = true
2: required_resoures = Hash.new
   // Collect required resources and amount into hash
3: for all r in n.required_resources do
4:     amount = n.required_resources.amount
5:     required_resoures[r] = amount
6: end for
7: for all resource, amount in required_resources do
8:     if check_resources_amount(resource, amount) == false then
9:         executable = false
10:    end if
11: end for
12: return executable
```

Checking entire process models Once, the *RESon* and the mapping are defined as outlined in the section before, this knowledge can be exploited to analyze whether a process model will be executable at runtime considering the required resources. This analysis is using available resources of a company

or department modeled within the ontology. Thereby, the definition of the *RESon* and the mapping in the OTS turns out to be an enormous advantage, since this allows exploitation of reasoning capabilities which enables an automation of this task.

Algorithm 3 shows the algorithm for this analysis in pseudo code. Given a set of process models (which is what SEMPA calls "set of feasible process models"), it returns the set of executable process models considering resource requirements and available resources.

Therefore, the algorithm checks whether each and every process action within the process model can be executed, i.e., whether there are enough resources in the resource pool for the RF assigned to the process actions. If there are insufficient resources available, the process action is marked as "red", denoting that the process action is not executable while "green" means that sufficient resources are available and the task will be executable.

The analysis is based on the analysis of single Single-entry Single-exit (SESE) fragments. This is necessary, because parallel threads (denoted by parallel split control-flow structures) require the resource requirements of the whole, parallelized fragment to be checked together and at once. Similarly, the SESE algorithm handles loops as it detects the loop as one fragment that can be checked that way.

Given the set of process actions B, C, D, E (arranged as shown in Figure 5.7) the definitions of common process modeling languages make no statement about the precise order those four process actions are executed (besides the sequence of $B \rightarrow C$ and $D \rightarrow E$ of course). Indeed, this is difficult in reality, as the single process actions might have different execution times in multiple instantiated processes. This might be, e.g., due to network latency for IT services or delayed execution for manual tasks due to high workload. Therefore, we have to assume, that process actions of both threads could be executed in parallel. For the example mentioned above, this means that process action B can be started in parallel to process actions D or E. Coincidentally, process action D could be executed in parallel to B or C.

The parallelization definition of UML Activity Diagram (AD) states, that parallel splits mean that the process actions **can** be executed in parallel - but they do not necessarily have to (OMG, 2009a). Additionally, we do not know about the execution time of the single process actions. This can lead to the following situation: Given the SESE fragment *[B, C, D, E]*. If *RESon*provides sufficient resources to execute B and C but there are not enough resources

Algorithm 3 Select executable process models regarding resource constraints

Input: Set of process models *models*
Output: true if process model is executable, false otherwise
Output: Missing resources $\mathcal{R}$
Output: Missing resources for full parallelization $\mathcal{RP}$

```
1: for all process_model in models do
2:     // A token-based algorithm is used to get all process' fragments
3:     fragments = get_sese_fragments(process_model)
4:     for all fragment in fragments do
5:         for all node in fragment.nodes do
6:             possible_resources = get_available_resources(node.required_resources)
7:             if possible_resources.empty? then
8:                 // process model is not executable ("red")
9:                 R.add(node.required_resources)
10:                return false
11:            else
12:                # store resources that could be used for execution
13:                node.possible_resources « possible_resources
14:            end if
15:        end for
16:        if fragment.is_parallel_fragment? then
17:            // Use simple depth-first search to gather and sum up
               amount of all resources in the parallel split
18:            required_resources = get_resources_in_all_parallel_paths(fragment)
19:            for all required_resource in required_resources do
20:                if query_resource_ontology(required_resource, re-
                   quired_resource.count) == false then
21:                    # fragment might not be executable ("orange")
22:                    mark_fragment(fragment, 'orange')
23:                    RP.add(required_resources)
24:                end if
25:            end for
26:        end if
27:    end for
28: end for
29: return true, R, RP
```

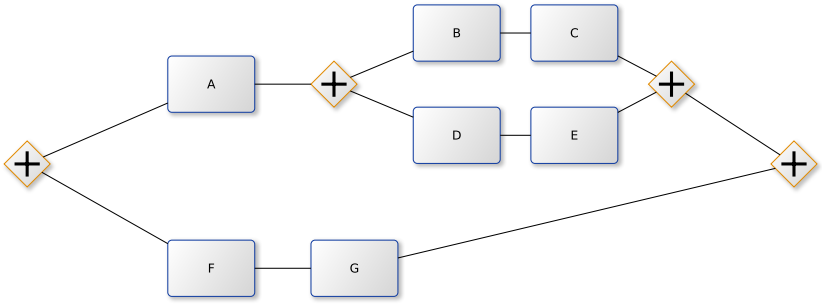

Figure 5.7: Process model fragment containing multiple parallel split patterns

for parallel execution of *B* and *D* we mark the fragment in a special state. As we could execute the process model (i.e., no resource is entirely missing) but cannot guarantee for the parallel execution of all possibilities of the fragment, we mark the fragment as "executable, but with restrictions" which is what we did by marking the fragment with an exclamation mark. See Chapter 7 for screenshots that show this detail.

See Figure 5.7 for an example model containing multiple parallel split patterns. Resources for process actions *B* and *D*, for example, have to be checked together as if both were merged into a single process action. This is due to the parallelization which makes resources of both process actions necessary — potentially — at the very same time. In order to accomplish this task, we use a token-based algorithm (Götz et al., 2009) that returns a segmentation of the process model as shown in Figure 5.8. After this segmentation, we traverse the SESE fragments from the innermost (i.e., the one having least process actions) to the outermost fragment. If an inner fragment is marked as not executable (red) the outer fragments will not be executable implicitly.

The result of the algorithm is that all infeasible process actions with the respective RRs that cannot be fulfilled by the resources in the resource pool are marked as non-executable in the given process model. This information can be utilized by the user to either re-design the process, e.g., replacing the non-executable process actions, planning a training program for developing the missing skills, or hire or respectively purchase new resources to fulfill the requirements. Additionally, the algorithm provides the specific resources from the resource pool which can perform a given process action.

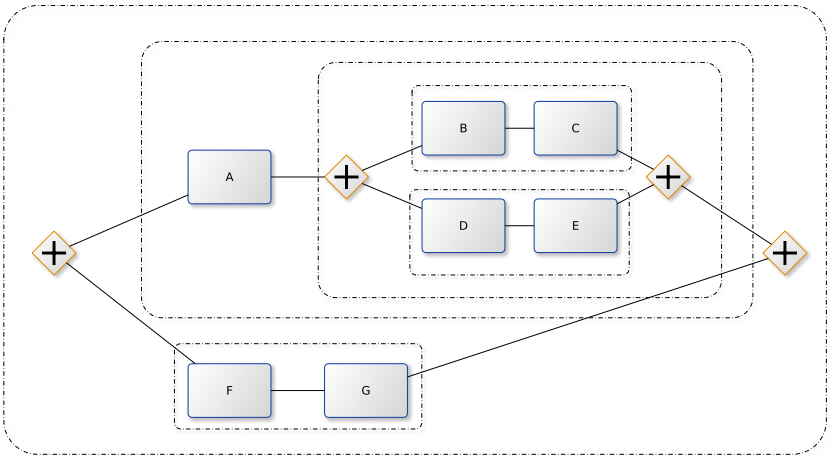

Figure 5.8: Process model segmented into SESE fragments

Valuating and selecting process models using resource properties

In the section before, we introduced an approach to check whether a process model is executable based on the definition of required resources and the matching with an ontology named *RESon*. In this section, we introduce a quantification for *a set of process models* in order to sort the models and the select the "best" one, e.g., regarding costs.

The topic of this so-called value-based perspective on process models has been extensively discussed in research: On the one hand, the costs within a process model have to be considered. This has been discussed in Activity Based Costing or Process Cost Accounting (Cooper and Kaplan, 1988). Though, when the value contribution of a model should to be recognized, the process revenues must be considered, too (vom Brocke et al., 2010; Kanevsky and Housel, 1994).

As we already have the process model including resource requirements and potential resources to fulfill the requirements on the one hand, and the ontology-based resource knowledge base on the other hand, we combined both to exploit that knowledge to quantify the model by using resource properties.

In order to determine the monetary value in terms of cash inflow (earnings) and cash outflow (costs) there are some assumptions necessary that are not required for other quantification properties. For further properties such as CO_2 or energy consumption there usually is no value *contribution* which makes the adaptation of the presented approach even easier. Nevertheless, we will shortly describe the assumptions necessary to determine cash flows: For the valuation of the cash-flow, we assume a risk neutral decision maker, so that the expected value of cash flows can be used as the basis for decisions and recommendations. This simplifies the demonstration of our extension without limiting its applicability in principle. (Bolsinger et al., 2011) describes how this assumption can be overcome and how the risk preferences of decision makers for process valuation can be included. To allow cash flows in different periods, we assume their net present values being used so that we can apply the valuation approach presented in (Bolsinger et al., 2011). Further requirements of the approach by Bolsinger for an application of the value-based process improvement approach are assumed to be given. This refers especially to the normal distribution of cash flows, which requires that the processes are executed in a sufficiently high quantity. From a value-based perspective, each executable process model can be characterized by the expected value of the cash flow for one process execution. This cash flow consists of three parts (Braunwarth et al., 2010).

- A direct process outcome D (e.g., a cash inflow that results from a customer paying for a service),
- an indirect process outcome I (e.g., a change in customer lifetime value due to a higher customer satisfaction and reduced churn probability),
- and an execution cash flow B resulting from out-payments for materials and resources.

Thinking of above mentioned properties without "contribution" such as energy or CO_2 consumption, there is no need to consider both D and I. The overall value $\mathcal{V}$ of a process and executable alternative j can be summarized as

$$\mathcal{V}_j = B_j + D_j + I_j$$

Each of the considered process model alternatives solve the same problem and thus fulfill the same process goal. Therefore, the direct process outcome D_j is identical for each model which is why D_j is not relevant for selecting the best alternative in our case.

Although the indirect process outcome I_j could differ because of, e.g., customer satisfaction might be different for a human voice answering a query compared to a computer voice), it is assumed to be constant for reasons of simplicity in the following. The value differences between executable process models thus stem solely from the difference of the execution cash flows B_j.

Each executable process model alternative j is composed of i process actions. Each process action can be fulfilled by k resource combinations denoted as $r_{i,k}$ which in turn consist of one or more roles.

Depending on possible execution paths resulting from different properties of process instances at decision nodes, each process action is executed with a certain probability $p_{i,j}$ that can differ between process model alternatives. The total (expected) execution cash flow B_j can be calculated this way:

$$B_j = \sum_i (min(M_{r_{i,1}j} + R_{r_{i,1}j} + ... + M_{r_{i,k}j} + R_{r_{i,k}j})) * p_{i,j}$$

The material-induced part ($M_{r_{i,k}j}$) of the execution cash flow results from out-payments for raw materials and supplies and must be defined in advance in the process library. The resource-induced part ($R_{r_{i,k}j}$) results from out-payments for the resources required during the process (resource-induced cash flows, e.g., employee salaries or service and maintenance costs of machines) and is derived semi-automatically using the resource information contained in the ontology and the process actions.

Process of quantification Given the set of alternative process models generated by SEMPA from which the best one should be selected. At first, the algorithm searches those process models for all existing combinations of resources and process actions. In the second step, the algorithm uses the ontology to determine, e.g., the cost unit rate for the resources gathered in the first step. Step three uses this information to calculate the cost unit rate for a specific combination of resource and process action. If more than one resource is assigned to a process action, the cost unit rates are summed up resulting in one or more resource-induced cash flows $R_{r_{i,k}j}$.

Of these k possibilities, the alternative with the lowest out-payments is selected. If an executable process model j contains decision nodes, an estimation of the probability for the possible execution paths must be provided. Based on this information, overall path probabilities can be calculated and linked to the process actions. The last step is the calculation and comparison of B_j

for all executable models. The model with the lowest B_j is the best one and should be selected.

As mentioned before, this approach can be easily abstracted and used for any numerical property of resources. This includes for example electrical energy consumption or CO_2 as stated before.

Algorithms

In order to show the practicability of our approach, we will show implementations of the necessary algorithms in pseudo code notation in the following.

Algorithm 3 shows the decision algorithm which calculates whether a process model is executable or not. As mentioned before, this step can be executed after SEMPA planned a set of process models automatically to further select the executable models.

Algorithm 4 shows the algorithm to calculate the cash flow of a single model. It takes a model with probabilities on all edges as input and returns the execution cash flow of the model as output. It is important to mention, that the calculation might be more difficult in detail as certain control-flow structures influence the algorithm. Though, this is out of scope of this thesis. See (Bolsinger et al., 2011) for an in-depth analysis and description of this part.

Algorithm 5 shows the obvious combination of both algorithms: The algorithm takes a set of process models and uses Algorithm 3 to decide about the practicability and selects the best model from a value-oriented point of view by using Algorithm 4 afterwards.

Additionally, we implemented the algorithms prototypically which will be described in the following section.

5.5 Tool support

We prototypically implemented parts of the presented approach.

The implementation consists of two larger components. The first one is used to modify the resource knowledge base *RESon*. It is necessary to model (read: create, read, update, delete) both the abstract resource taxonomy ($\mathcal{R}_a$) as well as concrete resources ($\mathcal{R}_c$). Additionally to the resources available within

Algorithm 4 Calculate cash flow of single process model

Input: Single process model *N* incl. probabilities on process actions
Output: Execution cash flow of process model *N*

```
model_cash_flow=0.0
for all node in N.nodes do
    node.cash_flow = ∞
    # calculate cash flow of all possible resource combinations
    for all resource in node.possible_resources do
        cash_flow = resource.type.cost_unit_rate * node.probability
        if cash_flow < node.cash_flow then
            node.cash_flow = cash_flow
        end if
    end for
    model_cash_flow+=node.cash_flow
end for
return model_cash_flow
```

Algorithm 5 Select process model

Input: Set of process models *N* incl. cash flows
Output: Best model regarding quantification (here: cash flow)

```
best_model = N[0]
for all model in N do
    if model.is_executable? then
        if model.cash_flow < best_model.cash_flow then
            best_model = model
        end if
    end if
end for
return best_model
```

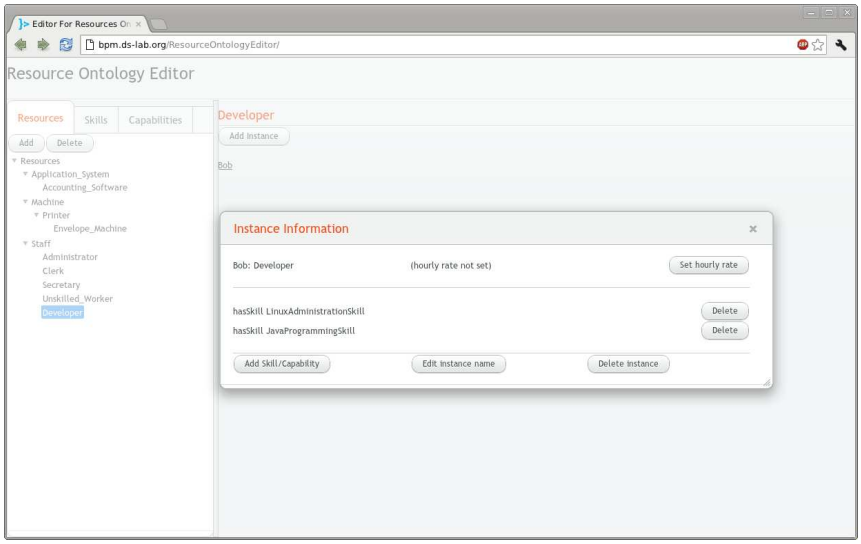

Figure 5.9: Detail information for resource individual

enterprises skills or capabilities have to be linked to concrete resources. This is possible with the "resource ontology editor" tool we will describe in more detail in the following.

The second component is an extension to an existing process modeling toolkit named Oryx (Decker et al., 2008a,b). The extension enables attachment of resource requirements to process models and checking of the models to decide about executability of models considering *RESon* which is build and handled with the before-mentioned resource ontology editor.

We will describe both components in the following.

Resource ontology editor

We implemented a domain-specific editor for the resource ontology *RESon* and skills or capabilities that can be added to resources.

It is web-based employing the Vaadin framework[2] and Java Application programming interface (API) for modifying OWL Ontologies (OWLAPI) to read and modify the underlying OWL 2 ontology.

Figure 5.9 shows the dialog of a concrete resource individual **Bob** that is a **Developer** and has two skills associated within the ontology.

[2]https://vaadin.com (access as of 2012-08-28)

Both resources as well as skills can be edited with the editor. Therefore, the two tabs "Skills" and "Capabilities" enable definition of the respective parts of *RESon*.

The full underlying ontology that was active at the time the screenshots were taken can be found in Listing 5 in the appendix.

A demo installation of the resource ontology editor is — at the time of this writing — available at `http://bpm.ds-lab.org/ResourceOntologyEditor/`.

Process modeling editor

Based upon the web-based process model editor Oryx we implemented functionality to define resource requirements and check against the resource knowledge base for feasibility of the model regarding required resources.

The architecture of Oryx is split into two parts.

- The frontend visible to users is implemented using Hypertext Markup Language (HTML) and JavaScript which is quite obviously for web-based applications.
- The backend is a Java application, run in typical application servers such as Tomcat. Models can be serialized into a PostgreSQL database.

In order to enable adding resource constraints, we extended the frontend as shown in Figure 5.10. It is possible to add an arbitrary number of resource requirements necessary to complete a single task.

In order to check the process model requirements with the ontology built with the resource editor, we extended the backend to handle this check. Usually, communication between the frontend and backend is done by serializing the data into JavaScript Object Notation (JSON) syntax to transfer it to the server. This is used, e.g., by the saving feature of Oryx.

In order to check whether the required resources are modeled within the resource ontology, we also transfer the whole model as well as resource requirements to the backend where the resource ontology is accessed and queried. After the JSON data is parsed and an internal representation is build, the model is split up into SESE fragments as described before.

Figure 5.11 shows a zoomed, detailed screenshot of the process modeling tool. The process model is a simple model consisting of a single process action

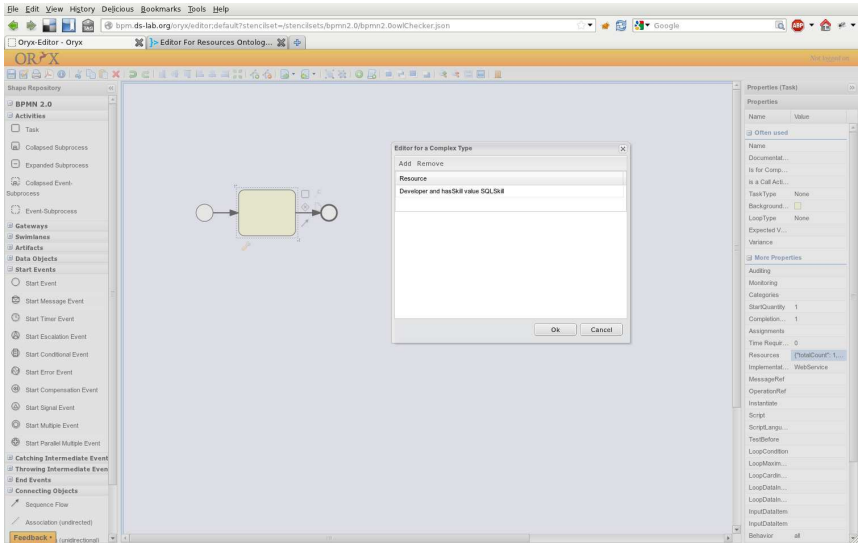

Figure 5.10: Modifying resource constraints within Oryx

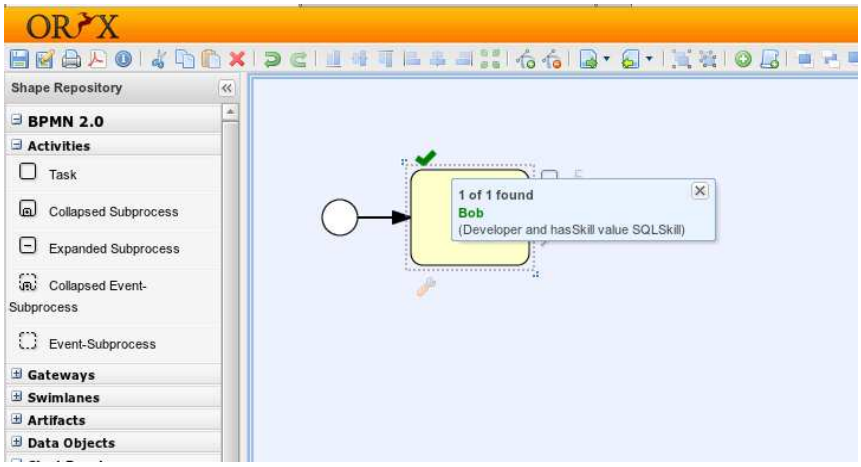

Figure 5.11: Single process element after checking for feasibility regarding resources

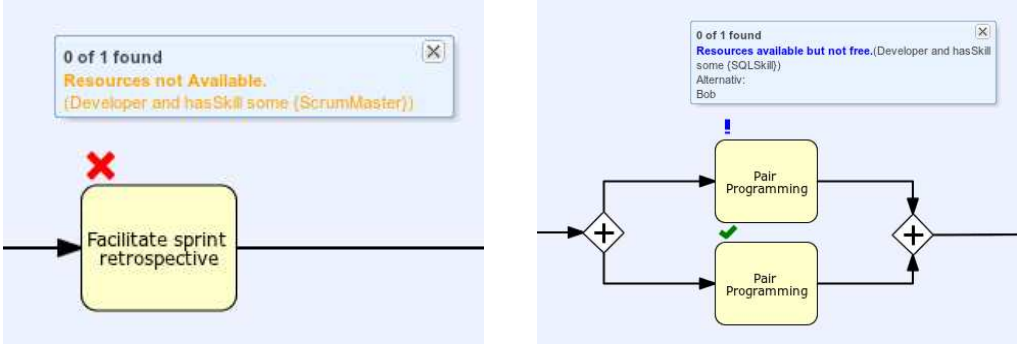

Figure 5.12: Insufficient resources (left) and insufficient for parallel execution (right)

that has one resource requirement (Developer and hasSkill value SQLSkill). The green tick on the upper left corner indicates that the process action is executable.

An example for a non-satisfiable resource requirements (i.e., no resources within *RESon* match so the requirements can be fulfilled) can be seen in the left part of Figure 5.12, which shows that a certain resource is not available at all.

In case the resource is available within the resource ontology, but this specific individual resource is already scheduled to execute another process action that runs in parallel, the process action is marked with an exclamation mark. An example for this can be seen in the right part of Figure 5.12.

Figure 5.13 shows a more complex model including checks as well as assigned resources to the specific requirements as well as the challenge of parallel threads that where discussed before and solved by segmenting the model. In the top right corner, resource requirements for a process action were modeled, that actually are present within the resource knowledge base, yet might be used in a parallel thread of the model. Therefore, this segment is marked with the exclamation mark.

The full flow of data between frontend, backend and some internals of the backend implementation as well as involved components can be seen in Figure 5.14.

In this section, we described details about theory and application of the resource definition and model checking approach and showed how the approach can be applied. In the next section, we will introduce a novel

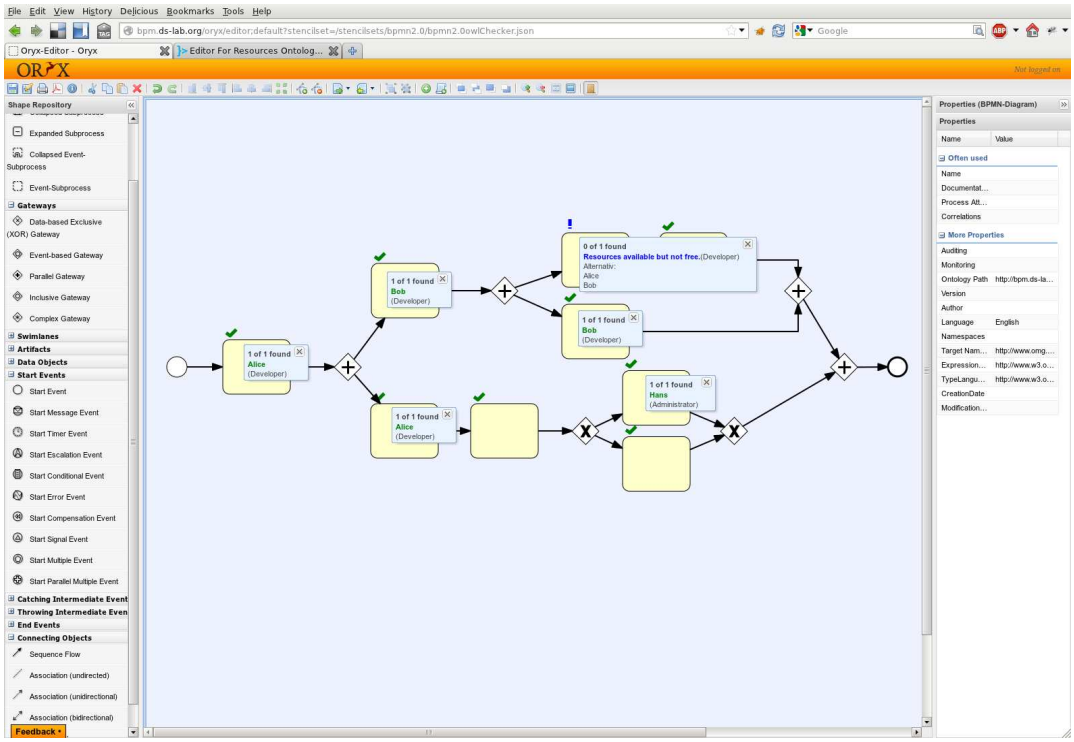

Figure 5.13: More complex model including parallel and XOR patterns

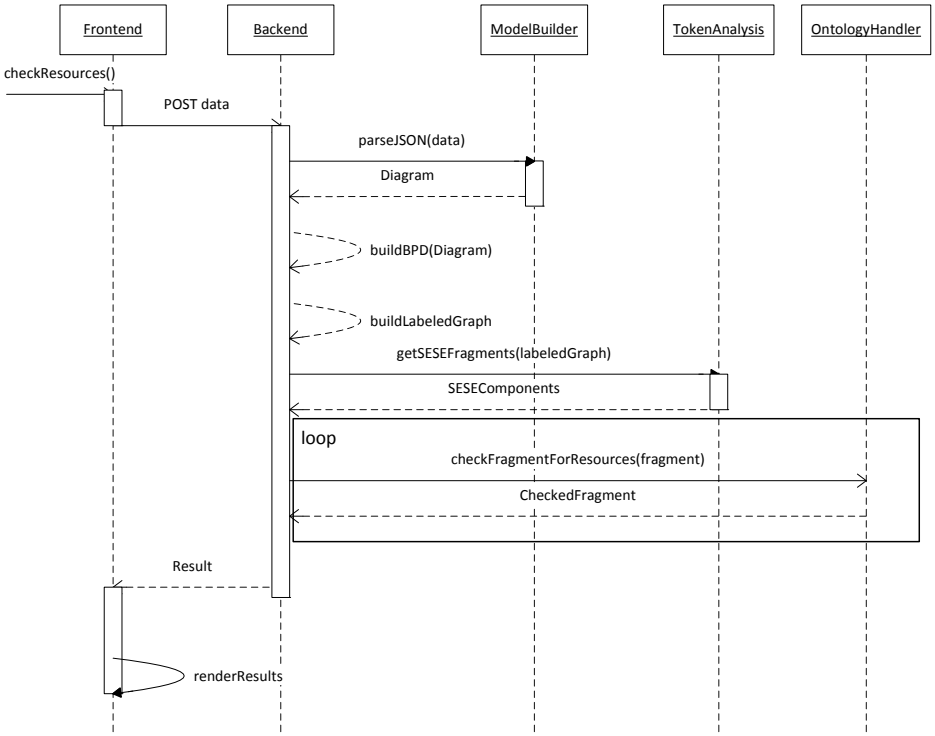

Figure 5.14: Sequence diagram of resource checking prototype

approach the automatically adapt process models considering resources as described in this section.

5.6 Adapting process models considering resources

The techniques presented in Section 3.4 describe an approach for semi-automated planning of process models using semantically annotated input and outputs on process actions. We used this technique to solve another issue that is time-consuming and error-prone: Adaptation of existing process models while considering resource constraints for process actions as described in Section 5.1.

When process actions have been changed (either in their implementation as, e.g., discovered by process mining techniques, due to management decisions or regulatory necessities), the respective actions need to be identified in all process models and automatically adapted where "adapting" means removing parts of the process or replacing a process action or a larger fragment with a new process component.

Basically, there are multiple alternatives for adapting process models. A simple, yet extensive adaptation method would be to substitution a complete process model by a completely re-planned version. Though, there are more fine-grained methods, such as adaptation of single process actions. As we are interested in adapting only parts of a model, our approach assumes that a set of process actions are given which that require adaptation because of any of the before-mentioned reasons.

5.6.1 Automatically adapting process models

An easy, fast and reproducible way to adapt process models can be a benefit and advantage for companies over competitors. This kind of flexibility within process modeling has been widely discussed in information systems whereas two areas of flexibility are commonly distinguished (cf. Hanseth et al., 1996; Gebauer and Schober, 2006).

Flexibility in the pattern of use and flexibility for future changes. Flexibility-to-use is the range of process requirements that is supported by Information Systems (IS) without requiring a major change of the respective IS. On the other side, flexibility- to-change requires a major change of the IS considering

the flexibility of the IT personnel, the integration of data and functionality and the modularity of system components (Gebauer and Schober, 2006).

For process models we can transfer these definitions. A process model has an internal flexibility-to-use, if different kinds of processes that only vary slightly are covered. It has an inherent flexibility-to-change, if most parts of the process model stay the same for major changes of the business process and only some parts need to be adapted. Finally, a process model is not flexible at all, if the complete model needs to be redesigned, when changes of the underlying process appear.

After transferring these definitions we would name a process model inflexible, if all process actions of the model have been changed ($\forall n \in \mathcal{N} : n \in lib_{CA}$) where elements stored in the set called lib_{ca} are those, that have been changed and need adaptation. If all process actions of a process model are within this set, this would result in a replacement of the complete process model. In contrast, we would name a process model flexible-to- change if single process action or fragments changed and possibly need adaptation. This adaptation can be of different kind:

- Deleting a single process action without further adaptation.
- Replacing a process action with another action out of lib_A.
- Marking a process action to require re-planning and adaptation that possibly leads to deletion or replacement of the process action or process fragments according to the re-planning

In the following, we will focus on flexibility-to-change, i.e., only some parts of the process model need to be adapted to fit to the business changes again.

Deleting and replacing single process actions does not necessarily induce a complex re-planning if input and output of the process action are identical or a produced output is not necessary any longer. Given a process action n_{del} to be deleted, an incoming edge to this node ($\mathcal{E}_{in}(n_{del})$) and the successor process action (or another arbitrary process model node) n_{succ} of n_{del} such that $\mathcal{E}_{out}(n_{del}) = \mathcal{E}_{in}(n_{succ})$.

All process actions in our graph have at most one incoming and one outgoing edge. To perform the deletion, the incoming edge to $\mathcal{E}_{in}(n_{del})$ is connected to n_{succ} and n_{del} is removed from the model.

The replacement of single process actions is even easier. The incoming edges to the node n_{old} to be replaced are connected to the new node n_{new}. The

outgoing edges of n_{old} are connected to n_{in} and finally n_{old} is deleted. After integrating the new node no re-planning is necessary. Please note that this obvious deletion and replacement of process actions is possible only if the following assumptions apply:

- Output produced by the deleted or replaced process action is not required at any other node in the process model.
- Input and Output fit flawlessly from predecessor to successor nodes when deleting nodes.
- Input and Output of a replacement node is semantically identical to the node to be replaced.

As those assumptions induce significant human effort to identify suitable alternative process actions, we look into the last and most interesting possibility when it comes to process model adaptation. This is when the adaptation of certain process actions is necessary but it is unknown how this adaptation such as replacing a process fragment with a new one could look like. This is the usual case as we suppose the set of process actions in enterprises to be extraordinary large and complex. Furthermore, there might be two or more process actions that need to be adapted due to legal and/or business driven changes. As a result, the most appropriate way would be to mark all process actions that need adaptation. More technically we collect all those process actions in a library $lib_{CA} \subseteq lib_A$ as introduced before.

In order to automatically adapt a process model to changing requirements, we assume that the process model has already been planned and therefore all process actions are described at least with their semantic inputs and outputs using ontology concepts ($SemAn$). Additionally, we assume that resource requirements for the process actions are attached to the actions as described in Section 5.3. Furthermore, we assume that we know the process actions that have been changed and how they have changed. Despite this assumption we will describe how this identification could take place and how the used semantics could help to improve this manual process. Since new regulations or customer requests can mostly be reduced to changes on a few actions, this assumption is not too restrictive anyway. After the changed actions have been identified in the existing process models (which is a simple depth-first search with matching process actions' labels), the adaptation process can start.

Following up the described requirements, the tasks for process model adaptation are the following:

1. Qualifying set of process actions in lib_{CA} regarding resources.
2. Identification and collection of process actions that have been changed.
3. Computation of the fragments that need to be adapted.[3]
4. Identification of the initial state of each process fragment.
5. Calculation of the goal state of each process fragment.
6. Re-planning the process fragments considering the changed process actions.
7. Integration of the planning result into the process model.
8. Validation of the adapted process model.

We will now elaborate each task in further detail.

5.6.2 Tasks for automated adaptation

1. Identification of changed process actions

As explained in the section before, our approach expects changed process actions to be in a set which we will refer to as lib_{CA} in the following. The first step in our approach is about the collection of such process actions, which are added to the library lib_{CA} at the end.

The most obvious way to fill lib_{CA} is simply to put the process actions out of lib_A manually, i.e., the human modeler needs to look through all process actions that have been stored in the process library lib_A, select those that have been changed and copy them to the library lib_{CA}. This is a potentially time consuming task depending on the power of lib_A. Therefore, we will show more efficient approach for this.

As the approach is based on semantic annotations, those annotations can be used for the identification. We assume that we do not know each task that has been changed, but at least what the changes are about. Concretely this means that we are able to identify the concept c in our ontology that is used for annotation and is subject of the change. Using this concept and the

[3]Where a fragment can be more than the known process action that changed. Basically, it could be made up of several process actions around the changed one.

underlying ontology it becomes possible to automatically identify process actions that take the concept as input or output using the filter functions for semantic annotations: $\forall n \in lib_A : SemIn(n) \cap C \neq \emptyset \vee SemOut(n) \cap C \neq \emptyset \Rightarrow n \in lib_{CA}$ with C being the set of changed concepts. Using this method it is possible to identify process actions that might have changed and need adaptation. Those are added to lib_{CA} and are to be considered in the following phases.

2. Qualifying set of process actions in lib_{CA} regarding resources

As discussed before, the actually existing resources within enterprises limit the possibilities of what can be executed. We introduced the resource knowledge base *RESon* to describe resources within enterprises that could also be used when adapting process models. Thus, if resource requirements for a single process action cannot be fulfilled, i.e., not enough resources with sufficient skills are available within *RESon*, this action should not be embedded into process models during the adaptation approach.

Therefore, we remove those process actions from lib_{CA} such that resource requirements for all process actions within lib_{CA} are satisfiable: $\forall\ n\ \in lib_{CA} : RR_S(n)$. Thereby, we ensure that we only exclude process actions, that cannot be executed under any circumstances within process models. Though, the resulting process models could still be problematical as parallel modeling could limit the executability. Step 7 takes care of this special case.

3. Computation of fragments to be adapted

All process actions having been identified in the step before have been changed and need further processing. In the next step we need to search for the entailing fragments of these process actions in order to re-plan the fragments. The identification of the surrounding fragment allows us to start planning with one initial state and a single goal state and makes the integration of the planning result easier afterwards.

A process fragment (SESE fragment) can be either a single process action or a part of the model that has only one single incoming edge (the entry edge e) and one single outgoing edge (the exit edge e'). It can be defined as a non-empty subgraph of $\mathcal{G}$ with $\mathcal{N}' \subseteq \mathcal{N}$ and $\mathcal{E}' = \mathcal{E} \cap (\mathcal{N}' \times \mathcal{N}')$ such that there exist edges $e, e' \in \mathcal{E}$ with $\mathcal{E} \cap ((\mathcal{N} \backslash \mathcal{N}') \times \mathcal{N}') = \{e\}$ and

$\mathcal{E} \cap (\mathcal{N}' \times (\mathcal{N} \backslash \mathcal{N}')) = \{e'\}$ (cp. (Vanhatalo et al., 2007)). In our further work we are only interested in canonical process fragments, i.e., fragments that do not overlap and are either nested or disjoint.

We calculate the (canonical) process fragments as well as the Strongly Connected Components (SCCs) of the process actions that have been changed. SCCs of a directed graph $\mathcal{G}$ are the maximal strongly connected subgraphs, i.e., there is a path p from each node in the subgraph to every other node in the same subgraph. The computation of fragments and SCCs is done using a token-flow algorithm. Please refer to (Götz et al., 2009), (Eisenbarth et al., 2011) and (Lautenbacher, 2010) for details of the token flow algorithm.

By using the described algorithm we identify the SESE fragments that contain changed process actions and therefore need to be adapted. We will calculate the initial and goal state of each fragment in the next steps.

4. Initial states of process fragments

In this step we need to identify the initial state of the process fragment that should be adapted. A state s is a subset of the set of parameters P, $s \subseteq P$. For this identification we have two possibilities:

The first one would be to compute the state that has been reached where the process fragment starts. This state can then be used as initial state for the re-planning. Therefore, we build a planning graph starting with the first action of the process model until the beginning of the fragment has been reached. The disadvantage of this solution is, that, if we have a rather big process model, probably hundreds of process actions and states need to be computed again. Additionally, we face the problem how to compute the initial state if the process action that has been changed is the first one of the whole process that has been executed. It would be impossible to determine this state.

The second possibility only analyzes the process fragment that needs to be re-planned. This process fragment very likely contains less process actions than the whole process model which in turn results in a faster computation than for the whole process model. Here, we compute all input parameters of the actions of this fragment. We remove the generated output parameters again, because those had been created as part of the fragment before. Thus, these are not available for the initial state during re-planning anymore.

This leads to a set of parameters that were necessary for the execution of the

process actions of the fragment before adaptation and should be sufficient for the re-planning, too.

Formally, we use the following as initial state:
$init := SemIn(n_1) \cup \bigcup_{\forall i=2..|\mathcal{N}|}(SemIn(n_i) \setminus SemOut(n_{i-1}))$.

5. Goal states of process fragments

Goal states $= G_1, G_2, \ldots, G_k$ specify the set of $k \in \mathbb{N}$ different (sets of) parameters $G_x \subseteq P$ (with $x = 1, \ldots, k$) which should be reachable in an adapted fragment.

The calculation of the goal state works analogous to the computation of the initial state. As described in the step before, we have two possibilities. The first possibility computes the initial state of the rest of all process actions in the process model that follow the fragment. The better way considers only the process fragment that should be adapted and selects all outputs of all actions in this fragment ($\bigcup_{i=1..|\mathcal{N}|} SemOut(n_i)$).

Again, we prefer the second method.

6. Re-planning process model fragments

Now that we have the initial state and the goal state of the process fragment we can re-plan the process fragment using SEMPA as introduced in Section 3.4.

Note that all existing process actions $n \in \mathcal{N}_{action}$ that have been stored in the process library lib_A are considered. The library lib_{CA} was only used for identification of the process fragments. In order to complete re-planning of the fragment, all process actions in the library are required.

In order to identify dependencies between process actions regarding their input and output parameters, `SEMPA` computes an ADG through backwards traversing beginning from the goal state. This ADG includes process actions and corresponding parameters as nodes. We use semantic reasoning for this part of the algorithm. This means input and output parameters of the process actions stored in the process library and their relationships in the ontology are analyzed ($SemIn(n)$ and $SemOut(n)$ respectively).

As the ADG does not describe direct sequences of process actions, a forward search algorithm is used to determine all sequences of process actions lead-

ing from the initial to the goal state. As result, we obtain an ASG which has two partitions: the process actions $\mathcal{N}_{action}$ and states which capture the state of the world after the execution of the action considering $SemIn$ and $SemOut$. Thereby, the algorithm performs a non-deterministic planning (Ghallab et al., 2004) with initial state uncertainty (Bonet and Geffner, 2001). This graph comprises all feasible solutions to the corresponding planning problem.

The action-state-graph is the basis to build the (syntactically correct) process model in the last step and to identify control structures such as, e.g., $\mathcal{N}_{XORSplit}$ or $\mathcal{N}_{ANDJoin}$.

7. Integrating the planning result considering resource availability

The result of the planning is first put into an own embedded sub-process to enable an isolated consideration and possible further (manual) changes of the new planned part. The two nodes $\mathcal{N}_{start}$ and $\mathcal{N}_{stop}$ of the sub-process are not used in further steps and can therefore be removed. The remaining sub-process is integrated into the original model as follows: As defined in Section 5.6.2 the parts of the original process we identified for re-planning have only one entry edge e and one exit edge e' (as the fragments are SESE fragments). Those are connected to the entry edge e_{entry} of the new planned fragment. The exit edge e_{exit} is connected respectively.

It is possible that the original SEMPA planning algorithm returns multiple results. As described before, we're applying the approach presented in Section 5.1 to deal with this and select only those models from the set that are feasible considering the resources modeled within ontology space.

A straightforward approach could be to count the number of process actions in the new fragments as parameter for the decision. The strategy and this parameter could be to chose and integrate the fragment that has the fewest process actions. A far better strategy would be to include economic parameters in the decision. Processing time or the cost of the planned fragments could affect the strategy which fragment to implement. We introduced a valuation mechanism by using the attached resources and their properties such as costs based on (Bolsinger et al., 2011) that helps to decide which model alternative should be selected from the set of possible solutions.

Instead of choosing one fragment automatically, a more pragmatic approach could be to integrate all fragments in copies of the original process model. The different alternatives are shown to the user who decides which fragment

conforms best to the business requirements.

If SEMPA does not return a result at all, e.g., because not all actions to achieve the goal states exist in the process library, the planning is stopped and the user is notified which parameter could not be fulfilled. This part of the algorithm that locates the position where to integrate the new fragment can be achieved by depth-first search basically. Another possibility would be to store the identified SESE nodes in an efficient data structure like a hash map.

8. Validating the adapted process model

The resulting process model needs to be validated in order to ensure that all dependencies are fulfilled after adaptation. First, it is validated automatically by evaluating whether each process action has all necessary input parameters. Additionally, the resulting model is checked for deadlocks or lack of synchronization. Therefore, we apply the algorithm presented in (Götz et al., 2009) again. For computing deadlocks and lack of synchronization we applied a method similar to (Vanhatalo et al., 2007) that has been adapted to conform to the token analysis algorithm introduced before.

If no errors have been detected, the resulting process is shown to the business analyst who can then decide how to further proceed with the adapted process model and further refine the selection of the set of feasible solutions to the one that adds most value, costs less or has fewest energy or CO_2 consumption based on the resources' properties.

In this section, we presented an approach for adaptation of process models. We described the eight necessary tasks to achieve automated adaptation of process models. In the next section, we will sum up the findings of the whole chapter.

5.7 Conclusion

In this chapter, we proposed an approach to analyze process models regarding their feasibility based on a semantically-defined resource ontology. We successfully showed, that process models from the MMTS can be combined with semantical knowledge from the OTS to assure that process resource demands are met. Therefore we introduced *RESon*, an ontology for modeling a resource taxonomy as well as concrete resources together with their

skills or capabilities. Additionally, we described a way to define resource requirements, attach them to process modeling space and finally match them with the resource knowledge base to decide about executability of a complete process model. We demonstrated applicability with algorithms as well as a prototypical implementation of both a *RESon*management tool as well as a process modeling extension.

Based upon this, we proposed a way to valuate models using the assigned resources and properties assigned to resources. This can be of great benefit for selecting a model out of a set of automatically generated models or model fragments.

Based on the presented mapping of resources and models, many further improvements can be implemented. E.g., an analysis of the weak points of a companies' resource landscape can be easily generated to reveal which skills or resources are available but could be critical because many processes depend on those while the number of available resources is low. Additionally, scheduling of resources for process execution could be added to the approach.

Furthermore, we described an approach for adapting semantic process models using the existing SEMPA algorithm as a central component. If the demands of customers change, new jurisdiction, and regulations appear or a supplier adapted its processes, the presented approach allows to adapt an existing fragment of a process model to the changed environment. As part of this adaptation approach, we combined the approach that checks for executability of models regarding resource constraints with the adaptation.

We identified the phases of this adaptation mechanism and described the details of each phase.

Chapter 6

Resource classification

As discussed in the last chapters, we use ontologies as a central knowledge database for both process model components as well as resources. Therefore, we searched for possibilities to further exploit the advantages, ontologies and underlying formal logic provides.

An interesting topic heavily discussed today, is the problem of critical commodities, i.e., mineral raw materials being used within industrial production processes. Those are subject to price and availability fluctuations indicating how critical such a commodity is at a given point in time. This in turn makes certain commodities business critical, and — at the end — the processes that require the commodity.

Although several manual approaches exist to identify the criticality of commodities, they rarely benefit from IT-based decision support. The fact that the criticality is the sum of a magnitude of properties that change frequently, shows the need for an automated classification.

Additionally, we state that such a classification should be integrated into the process modeling life-cycle to recognize which parts of a process are subject to problems when criticality of commodities changes.

We will present an approach that shows how to use semantic technologies to accomplish such resource classification in the following. This perfectly integrates into the approach to define resource requirements into process models as introduced in Section 5.1.

6.1 Motivation

Today, industry is heavily based upon mineral raw materials which are the basis for economic wealth in turn. Price and availability of such commodities is subject to increasing fluctuations, not only due to speculations at capital markets. For example, the prices of the industrial metals Copper and Tin have more than doubled (Exchange, 2012) within the last three years. At the same time, the price of the rare metal Neodymium (which is used for wind power plants for instance) has multiplied by a factor of six, while the price of Indium, an essential component of liquid crystal displays, has fluctuated between $ 300 and $ 800 per kilogram (Metal-Pages, 2012).

Both government and enterprises realized the risks of such criticality changes of raw materials lately. For example, the renewable energies sector abandoned highly efficient technologies due to raw materials shortage (Polinder et al., 2006).

In general, public opinion tends to overestimate bad news, especially, if empirical evidence is missing (Simon, 1980). Therefore, a number of empirically based indicators have been developed to rate commodities' risks and impact for a company or an entire economy (Rosenau-Tornow et al., 2009; Committee on Critical Mineral Impacts of the U.S. Economy, 2008; European Commission, 2010). Those indicators (usually represented by simple figures) are usually referred to as "criticality". Criticality characterizes a commodities' risk for upcoming shortages or price fluctuation as mentioned above. These criticality indicators often aggregate two or more dimensions of relevant key figures, e.g., supply risk and elasticity of demand. Today, companies and governments already use some of the many published criticality indicators. In doing so, they hope to "better understand the weaknesses of these markets which may lead to future supply shortages thus influencing (the) price" (Rosenau-Tornow et al., 2009). Especially enterprises with a large number of complex, resource-handling processes are highly vulnerable to supply shortages and fluctuations of prices. This in turn requires enterprises to discover which of their processes utilize potentially critical commodities, and thus require special attention.

However, criticality measurement of raw materials is a complex task (Achzet et al., 2011; Working Group of the Raw Materials Supply Group, 2010; European Commission, 2010). Analysts without specific knowledge about raw materials are confronted with a number of challenges. This includes research for data availability, data collection, and the definition of suitable

aggregate functions. Even raw materials experts struggle with the need for large amounts of manual data processing, and the need for up-to-date data to build contemporary, ideally realtime analysis of criticality indicators. As potentially critical commodities could be required in various parts of enterprises, *many* business processes that handle the resources might be affected. Thus, the decision about criticality has to be taken into account at different places within the enterprise (Bhattacharjee et al., 2010). The effort of characterizing criticality of raw materials used within an enterprise and additionally the check of processes that might use those materials is a substantial, labor-intensive, and error-prone task, as it easily leads to incorrect or inconsistent results.

As we both have processes as well as resources combined within an ontology as discussed in the chapters before, we were striving to exploit the formal logic and reasoning for an approach, that enables automatic categorization of resources' criticality. After the classification of commodities took place, we will show that we can easily use this information to detect which *processes* require critical resources.

After a short background on criticality of non-renewable resources in Section 6.2, we will present the approach that automatically calculates aggregated values to define criticality of commodities in Section 6.3. We will show details of the implementation in Section 6.4 and present the before-mentioned integration into our semantic process definition approach from Chapter 4 in Section 6.5. Finally, we will conclude our findings in Section 6.7

6.2 Theoretical background

Large parts of the value creation in economies especially in producing companies depends on the affordable availability of non-renewable resources. We will provide some background knowledge on non-renewable resources and existing criticality classification possibilities in the following to better understand our approach on the automated classification approach presented in the following sections. This background information is necessary to understand the classification rules we will introduce afterwards.

Metals are an integral part of many products. While industrial metals like iron, aluminum, copper, zinc or tin are very well known components of many products, there is a great number of minor metals like indium, neodymium or tantalum that are less known but essential for, e.g., any computer system

or cell phone that is manufactured today (Reller et al., 2009). At present, the mining and refinement of these metals is distributed globally and to a high degree. This makes the detection of the metals' origin, reserves, stocks, demand and supplies a rather complex and often in-transparent task.

However, many producing companies do not have extraordinary experience or know-how about supply chains of non-renewable resources. Though, often, they are very susceptible to supply shortages and price fluctuations. Thus, companies as well as economies, need some kind of *aggregated indicators*, which enable them to manage the inherent risks of any non-renewable resource they are utilizing today or planning to do so in the future.

Searching for such indicators is certainly not a new topic. In 1972 Meadows et al. published their widely noticed study entitled "Limits to Growth". The authors warn of a number of metals that will extinguish (Meadows, 1972). The authors heavily relied on the so-called "reserves-to-production" ratios in this study, which indicate how long the reserves of a certain resource are going to last while assuming, that the current rate of consumption stays at this level constantly. However, many of their forecasts turned out to be false, which is not to a small extent due to the inherent shortcomings of the reserves-to-production ratio (Feygin and Satkin, 2004) as well as fundamental criticism about forecasting resource consumption (Neumayer, 2000). Thus, while the price is certainly a sound approximation of a commodity's criticality (Tilton, 2002), a number of more in-depth and more sophisticated indicators for the criticality of non-renewable resources have been developed. For instance, the European Union (European Commission, 2010) as well as the Committee on Critical Mineral Impacts on the U.S. Economy use a simple metric, that incorporates both supply risk and economic importance (Committee on Critical Mineral Impacts of the U.S. Economy, 2008). The U.S. Department of Energy in turn presents a more detailed criticality analysis, that assigns different percentage weights to a number of indicators like availability, concentration of producers or political influence, for instance by export quotas (Bauer et al., 2010). In a similar approach (Rosenau-Tornow et al., 2009) present a methodology that aggregates five main criticality indicators (supply and demand, production costs, geostrategic risks, market power, supply and demand trends) in a spider chart.

Obviously, as non-renewable resources were matter of public concern over many years, a lot of research has been done in this area so nowadays, there is no lack of methodologies for criticality assessment any more. However, these criticality assessments usually require detailed knowledge in the domain of

non-renewable resources and access to a number of different data sources. In most cases, the input data is processed manually and criticality assessments are performed only sporadically. This often leads to outdated and erroneous results, as criticality assessment is a complex and time-consuming task requiring many intermediate steps in data processing. The capacity of manual criticality assessments simply constitutes an upper limit on the bearable complexity and in many cases, e.g., in global enterprises with thousands of processes, products and components, this limit is significantly below what actually would be needed for sensible long-term planning. Therefore, we state that many companies search for fast and reliable measures for criticality assessment. Volkswagen AG for instance supported a published study (Rosenau-Tornow et al., 2009) on a criticality analysis.

We will present some existing resource classification indicators such as the mentioned reserves-to-production ratio as well as other, more sophisticated ones in the following.

As mentioned before, developing a sound definition for the criticality of a non-renewable resource is an intricate task that has been addressed by a number of distinguished researchers in the past. Additionally, the approach presented in this thesis is independent from the concrete details of the criteria to a great extend. Therefore, we do not intend to present a complete and novel definition of criticality itself, but rather make use of existing indicators and use a working definition that incorporates the most important, existing criticality criteria.

We will combine the results of the approaches not just as prerequisite for our criticality assessment algorithm, but also as template that can be further extended and customized.

Reserves-to-production ratio

The reserves-to-production ratio specifies the remaining years that a resource's reserves will last at the current rate of consumption while assuming a fixed amount of reserves. The main question is how many remaining years are to be regarded as critical.

As mentioned in the motivation, the reserves-to-production ratio has failed as a solitary indicator. Though, it still is a powerful indicator amongst others that is used for criticality assessments in science as well as in practice (Achzet et al., 2011).

Tilton and Lagos state, that mining companies do not consider investing in reserve additions, i.e., exploration of new deposits, if the reserves-to-production ratio exceeds 20-30 years (Tilton and Lagos, 2007). Thus, a reserves-to-production ratio of over about 25 years could be regarded as non-critical. On the other hand, lower reserves-to-production ratios indicate, that companies have been unsuccessful in discovering new deposits. Taking into account common product life cycles, e.g., within the automotive sector, a reserves-to-production ratio of less than 10 years seems highly critical, as the supply of current product lines may be endangered before new product lines with possible substitutes can be deploy into production. Of course, this number has to be specifically altered for any industrial sector with longer or shorter product line cycles. This demonstrates that the presented criticality definition depends on industry sector and products and should be adjusted accordingly.

Market power and country concentration

Among many other scarce metals, the case of the Rare Earth Elements (REEs) has shown the impacts of market power and country concentration.

China mines about 97% of the world's rare earth production (European Commission, 2010). Thus, China has a virtual monopoly, that makes the rest of the world highly dependent on Chinese market regulations. For instance, as China has repeatedly lowered its export quotas, companies, and economies around the world are searching for substitutes for REE, and investments in new deposits (Metal-Pages, 2012). Obviously, both market power, as well as country concentration at these levels are to be considered as highly critical. To measure the general concentration of more than one country or company, the Herfindahl–Hirschman Index (HHI) is used (Hirschman, 1964). This HHI adds the squared shares as percentages of any country and company, respectively. Thus, a HHI of 10.000 denotes, that a country (or company), respectively controls the whole world market.

Rosenau-Tornow et al. consider a HHI of above 2.000 as highly critical, while values of over 1.000 are regarded as moderately critical (Rosenau-Tornow et al., 2009). We use these numbers as a reasonable approximation.

Stock of inventory

Obviously stockpiling influences criticality when stock of inventory is used as a criticality indicator. It is rather self-evident that stockpiling can reduce the criticality of a non-renewable resource, as stocks can be used to absorb short-term supply shortages. While stockpiling is also subject to speculation in some cases and thus could also intensify supply shortages, a responsible stockpiling generally prevents shortages, and even in case of speculation, stocks are only profitable if they are sold at some time. Naturally, no stocks at all can be regarded as critical, while on the other hand this would render a large part of the chemical elements critical, as in many cases there is no data on stocks. Thus, only stocks where data is available can be used in order to prevent a large number of false positives. (Rosenau-Tornow et al., 2009) measure stocks in relation to the overall production. In their work, they consider stocks of less than 10 percent of the annual production as highly critical, while less than 15 percent are moderately critical. Of course, these numbers can vary substantially depending on the respective sector or company. However, they can be used as a reasonable approximation for our case.

Price

From an economic point of view, the price is an aggregation of all information influencing current and future demand and supply on any subject on a market. Though, prices is often disregarded in criticality assessments. Nevertheless, the price itself can be used as criticality indicator (Tilton, 2002). While there are company or sector specific thresholds that might render a non-renewable resource critical, from a general point of view, rising prices can be considered to be critical, while falling prices can be regarded as non-critical. This approximation provides a rather robust assessment while abstaining from unnecessary assumptions.

Further criteria and criteria selection

Of course, there are many other potentially useful indicators. For instance, governance or corruption indices, as well as data on mining investments or demand and supply trends. We limited the selection of indicators to the most important ones to demonstrate the approach. As the main focus of

this approach is to show how criticality criteria can be used to automatically classify criticality and show integration into semantic process modeling, a selection of important and automatically processable indicators seems adequate.

Aggregation

As the indicators described above do not include all aspects of criticality if used individually, they are normally aggregated into an overall criticality measure. There are several methods to aggregate these indicators, from geometrical aggregations like spider charts (Rosenau-Tornow et al., 2009) to weighted averages (Erdmann and Graedel, 2011). We used a simple point system for the approach presented in the following: An indicator that is highly critical equals two points, a medium criticality indicator equals one point. A resource that has at least four points is considered *highly critical*, while a commodity that has two or three points is considered *medium critical*. All other resources are considered as *non-critical*. Of course, this aggregation can be tailored to the specific needs of an enterprise, government or whatever environment it is used for. In our case even such an elementary aggregation provides decent results as will be shown in an evaluation in Chapter 7. Finally, we use the aggregated criteria to declare a process model to be critical, if it makes use of one or more highly critical resources.

6.3 Approach

We will show the details of the approach in the following. We will start with describing the semantic specification of above presented indicators as well as the implementation.

Describing criticality indicators with OWL and SWRL

For the implementation of this definition, we obviously use semantic technologies as we did in the chapters before. Though, even considered stand-alone, this makes sense for a number of reasons:

1. Semantic Web technologies are designed and built for maximum inter-

operability and enable automated gathering and processing of heterogeneous and distributed information.

2. As we use OWL-DL, our ontology is based on a formally defined and decidable logical language, which enables automated consistency-checks and thus guarantees consistent knowledge and consistent rules.

3. OWL and SWRL not only enable inference of new knowledge from existing knowledge, but also the answering of queries. In addition, there is an explain function that describes the logical steps that lead to a certain decision.

Another important argument is that the use of OWL and SWRL allow the integration of criticality into semantic process management and thus provide the interface between criticality research and business process modeling. So the criticality classification perfectly colludes with the presented work in the chapters before.

As described in introductory sections, OWL offers a wide range of semantic features, that enable automated reasoning for the detection of inherent correlations of classes, individuals (i.e., instances) and properties (i.e., relations).

Therefore, we defined a class Commodity which holds all resource instances that should be classified. Instances include all metals that we imported into ontology space such as lithium, aluminum, chromium, and so on. Altogether we imported **42** metal individuals to test our approach. Additionally, we imported information about the metals such as HHI, reserves, prices, world mine production, stocks, and so on. We added **10** data property assertions to each of the commodity individuals to represent this information.

We tried to implement the above criticality criteria directly in OWL first. And, as a matter of fact, at least one of our criteria can easily be implemented in OWL: The market power indicator produces a simple inequality that provides thresholds for medium and high criticality. Using Manchester Syntax, this criterion can be described as follows:

```
1  Commodity and (hasHhiOfCountries some float(> 0.2f))
```

Listing 6.1: Example criteria definition

However, the other three criteria tend to be more complex. For instance, the stock of inventory criterion basically consists of a simple threshold. But the reserves-to-production ratio requires a division of the reserves quantity

Indicator	**Operationalization**		
	high criticality	*medium* criticality	*no* criticality
Reserves-to-production ratio	=< 10 years	> 10 years, < 25 years	>= 25 years
Market power and country concentration	HHI >= 2000	1000 <= HHI < 2000	HHI < 1000
Stock of inventory, in percent of production	=< 10%	10% < stocks <= 15%	> 15%
Price development	Strongly rising prices	Rising and falling prices	Falling prices

Table 6.1: Criticality indicators and their operationalization.

by the annual consumption by definition. Additionally, the stocks have to be divided by the annual consumption to give a meaningful figure. While this certainly does not sound too complex, it is a problem as OWL cannot perform suchlike arithmetic operations. Finally, the price criterion requires the comparison of two or more prices, a feature that is not provided by OWL, too.

Therefore, we decided to use SWRL for this task. With the addition of so-called built-ins (Horrocks et al., 2004) it is also possible to use mathematical functions. It is important to note that these SWRL rules seamlessly integrate into an OWL-based ontology and allow for automated reasoning, too. Based on these technologies, we defined a number of rules for each of the mentioned criteria. Table 6.1 shows the indicators as well as the respective operationalization. The SWRL rules and a short description of the rules is given later in this section.

Based upon the indicators and operationalization as shown in Table 6.1 we defined an ontology that enables description of commodities, the respective properties of the commodities, criticality indicators, and SWRL rules. We will show the ontology design and rules including a short description of the respective rule in the following.

Ontology design Within the commodity classification ontology we defined an individual for each criticality indicator such as ReservesHighCriticalityIndication or ConcentrationHighCriticalityIndication. In order to describe that a commodity has a criticality indicator such as ConcentrationHighCriticalityIndication we simply use an object property hasCriticalityIndication. E.g., if the commodity Indium has a high HHI an object property assertion is added so that

Indium
hasCriticalityIndication ConcentrationHighCriticalityIndication

As the name of the criticality class already says, ConcentrationHighCriticalityIndication is an indicator of *high* criticality. The individual ConcentrationHighCriticalityIndication is put into an appropriate OWL class **HighCriticalityIndication**. We defined appropriate further indicators for both other criticality properties as well as levels.

In order to put commodities into groups of medium and high critical commodities, we defined that *medium* criticality means that a commodity has one or more highly critical indicator *or* two or more medium critical indicators as shown in Table 6.1). This is defined as an equivalent class definition:

mediumCriticalCommodity ≡ **Commodity**
and ((*hasCriticalityIndication* min 1 **HighCriticalityIndication**)
or (hasCriticalityIndication min 2 **MediumCriticalityIndication**))

High criticality in turn means that a commodity has two high critical or four medium critical indicators:

highlyCriticalCommodity ≡ **Commodity**
and ((*hasCriticalityIndication* min 2 **HighCriticalityIndication**)
or ((*hasCriticalityIndication* min 4 **MediumCriticalityIndication**)

Please note that the defined limits for medium or highly critical commodities are arbitrary and can be changed to whatever values seem reasonable. Additionally, further classes can easily be added to enable a more fine-granular classification.

We will describe details of the rules shown in Table 6.1 in the following.

Rules for Reserves-to-production ratio indicators Listing 6.2 describes the rule for a commodity that has fewer reserves than the accumulated usage of ten years. It is marked with an indicator for high criticality.

```
1 Commodity(?c),
2 hasReserves(?c, ?reserves), hasWorldMineProduction(?c, ?
    annualUsage),
3 divide(?range, ?reserves, ?annualUsage), lessThan(?range, 10)
4 → hasCriticalityIndication(?c, ReservesHighCriticalityIndication)
```

Listing 6.2: SWRL rule for <= 10 years RTP ratio

Listing 6.3 is the second rule that describes that reserves for a commodity last between ten to 25 years.

```
1 Commodity(?c),
2 hasReserves(?c, ?reserves), hasWorldMineProduction(?c, ?
    annualUsage),
3 divide(?range, ?reserves, ?annualUsage), greaterThan(?range, 10),
    lessThan(?range, 25)
4 → hasCriticalityIndication(?c, ReservesMediumCriticalityIndication
    )
```

Listing 6.3: SWRL rule for >= 10 <= 25 years RTP ratio

Rules for HHI indicators As mentioned before, an HHI of 1000 to 2000 is considered critical while an HHI above 2000 is considered highly critical (Rosenau-Tornow et al., 2009).

Thus, the rule specified in Listing 6.4 marks commodities with an HHI above 2000 (here: 0.20) as high criticality.

```
1 Commodity(?c),
2 hasHhiOfCountries(?c, ?hhi), greaterThan(?hhi, 0.20)
3 → hasCriticalityIndication(?c,
    ConcentrationHighCriticalityIndication)
```

Listing 6.4: SWRL rule for HHI > 2000

Rules price deviation indicators A commodity whose prices have been rising for the last two years is marked with an indicator for high criticality. This is defined in Listing 6.5.

```
1 Commodity(?c),
2 hasPrice(?c, ?p), hasPrice1(?c, ?p1), hasPrice2(?c, ?p2),
3 greaterThan(?p, ?p1), greaterThan(?p1, ?p2)
4 → hasCriticalityIndication(?c, PricesHighCriticalityIndication)
```

Listing 6.5: SWRL rule for rising prices

Rules for stock/annual usage indicators Listing 6.6 shows the rule defining a commodity whose stocks are less than 10% of its annual usage to be marked as highly criticality.

```
1 Commodity(?c),
2 hasStocks(?c, ?s), hasWorldMineProduction(?c, ?p),
3 divide(?stockrange, ?s, ?p),
4 lessThan(? stockrange, 0.10)
5 → hasCriticalityIndication(?c, StocksHighCriticalityIndication)
```

Listing 6.6: SWRL rule for low stocks considering production and annual usage

For reasons of brevity, we only presented one rule for each criterion, as the other rules differ in the used thresholds, only. The other rules that translate the indicators into ontology space can be found in Appendix 8.3.

After execution of the ruleset on commodities, criticality indicators are attached to commodity individuals by adding object property assertions.

We will describe the setup of our implementation in more detail in the following.

6.4 Implementation

For the automated classification to take place, we need to import relevant data such as HHI, prices and so on into ontology space. Therefore, we wrote a simple parser to import this data being represented as Comma-Separated Values (CSV) files containing time series data. We utilized Java programming language and OWLAPI for this task (Horridge and Bechhofer, 2011). This parser inserts the commodities into the ontology as individuals and adds

information such as the as data property assertions. After this step, all information necessary to decide about criticality with the reasoner is represented within the ontology. Based on this data the reasoning and the mentioned extensions compute the classification of the commodities into classes such as highlyCriticalCommodity. We use the Pellet OWL 2 reasoner software for criticality classification (Sirin et al., 2007). Figure 6.1 shows an overview of all involved components and the flow of information within the implemented tool.

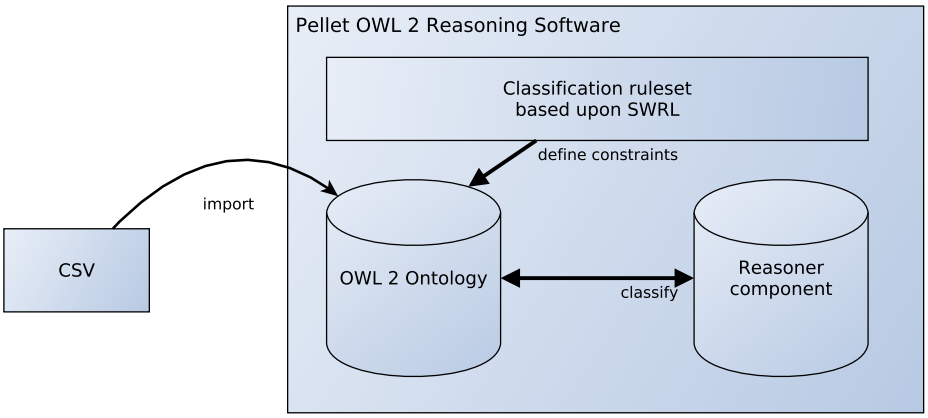

Figure 6.1: Classification tool components: Big picture

A by-product provided by the reasoning software is a formal consistency check of the ontology and rules, i.e., the ontology cannot contain contradictory facts or axioms. This is especially helpful when it comes to complex criticality definitions. Given, for example, it is necessary to add further criticality indications to the equivalent class definition, and one would add the following rules.

Commodity and (*hasCriticalityIndication*
max 3 **MediumCriticalityIndication**)
Commodity and (hasCriticalityIndication
min 4 **MediumCriticalityIndication**)

Given such rules within the ontology, reasoning software such as the utilized Pellet Reasoner would complain about an inconsistency of the ontology and points to the minCardinality restriction, that is larger than a maxCardinality restriction. Additionally, a reasoner typically checks whether it is possible for

```
1  Commodity(?c),
2    hasPrice(?c, ?p), hasPrice1(?c, ?p1), hasPrice2(?c, ?p2),
3    greaterThan(?p, ?p1), greaterThan(?p1, ?p2)
4    → hasCriticalityIndication(?c, PricesHighCriticalityIndication)
5
6  Commodity(?c),
7    hasStocks(?c, ?s), hasWorldMineProduction(?c, ?p),
8    divide(?stockrange, ?s, ?p), lessThan(?stockrange, 0.1)
9    → hasCriticalityIndication(?c, StocksHighCriticalityIndication)
10
11 Lithium hasPrice2 8.169053
12 Lithium hasPrice1 8.39841
13 Lithium hasPrice 8.418477
14 Lithium hasStocks 0.0
15 Lithium hasWorldMineProduction 301000.0
16 PricesHighCriticalityIndication DifferentFrom
       StocksHighCriticalityIndication
17 PricesHighCriticalityIndication Type HighCriticalityIndication
18 StocksHighCriticalityIndication Type HighCriticalityIndication
19 highlyCriticalCommodity EquivalentTo Commodity
20   and (hasCriticalityIndication min 2 HighCriticalityIndication)
```

Listing 6.7: Explanation path for Lithium being highly critical

a class to contain any individuals (concept satisfiability). In our approach this ensures that no rules or class definitions are contradictory or inconsistent.

Another interesting feature of a reasoner is the possibility to explain certain inferred facts. This is especially useful when classification of a resource is completed and one wants to know why the resource is classified into a specific criticality class. This is a common feature of reasoning software and implemented, e.g., in the Protégé Ontology Editor. Ten out of the complete set of modeled metals we put into our ontology for classification are classified as *highly critical*. Lithium is one of them, the so-called "explanation path" of the individual for the classification as highly critical can be seen in Listing 6.7.

Using the explain feature, the decision process of the reasoning component can be easily reproduced.

6.5 Integration with semantic process models

As mentioned before, the ontology-based classification approach of resources colludes great with the ontology-based process modeling as introduced in Chapter 4.

If resources are attached to elements within process models already, object property assertions can be used to attach the resource information to semantic modeled process actions.

As the running example as introduced in Section 2.4 does not include resources and the process is more a service than a producing process, we will show the integration of criticality classification of this section with semantic process modeling with another example.

Resources that are needed within a process model are connected to a specific process action usually. As demonstrated in Section 5.1 this combination can be achieved by using object property assertions betweens process actions, and the necessary resources. Within ontology space, this would be modeled as follows.

n Type **ProcessAction**
c Type **Commodity**
n *hasResourceRequirement* **c**

That way, the necessity and demand for the resource is put into ontology space. Given the results of the classification process mark c as critical such that inferred

c Type **highlyCriticalCommodity**

Then the definition of a critical process action within the ontology is fairly easy. The equivalent class definition of **CriticalProcessAction** should include all process actions of process models, that have critical commodities attached, i.e., having properties indicating that the respective process actions require the commodities. Therefore, a critical process action is a subclass of process action and is defined as follows.

CriticalProcessAction Type **ProcessAction**
and *hasResourceRequirement* some **highlyCriticalCommodity**

After reasoning is applied to the ontology, **CriticalProcessAction** includes process actions that require critical commodities.

The same principle can be applied to the full process model. E.g., to define that process models should be classified as critical if there are any *process actions* within the process model that are critical, the *whole model* should be marked as critical as well.

CriticalProcessModel Type **ProcessModel**
and *contains* some **CriticalProcessAction**

Please note this last feature requires an additional property added to *PMon* so that a process model individual holds object property assertions (*contains*) to all process action individuals of the model. This is easily achievable, though.

Figure 6.2 shows the integration of process modeling with the resource classification approach graphically.

6.6 Demonstration of applicability

We will demonstrate how to combine semantic process modeling as discussed earlier and the classification approach of commodities in this section. As the running example is a service process that hardly requires any of the presented raw materials we will show the approach using a straightforward and domain-specific process model as shown in Figure 6.3. The process itself consists of fictitious tasks but show how the combination of the resource classification and semantic process modeling is achieved.

The ontology-based modeling of the process itself — according to the concepts presented within Chapter 4 — would look like

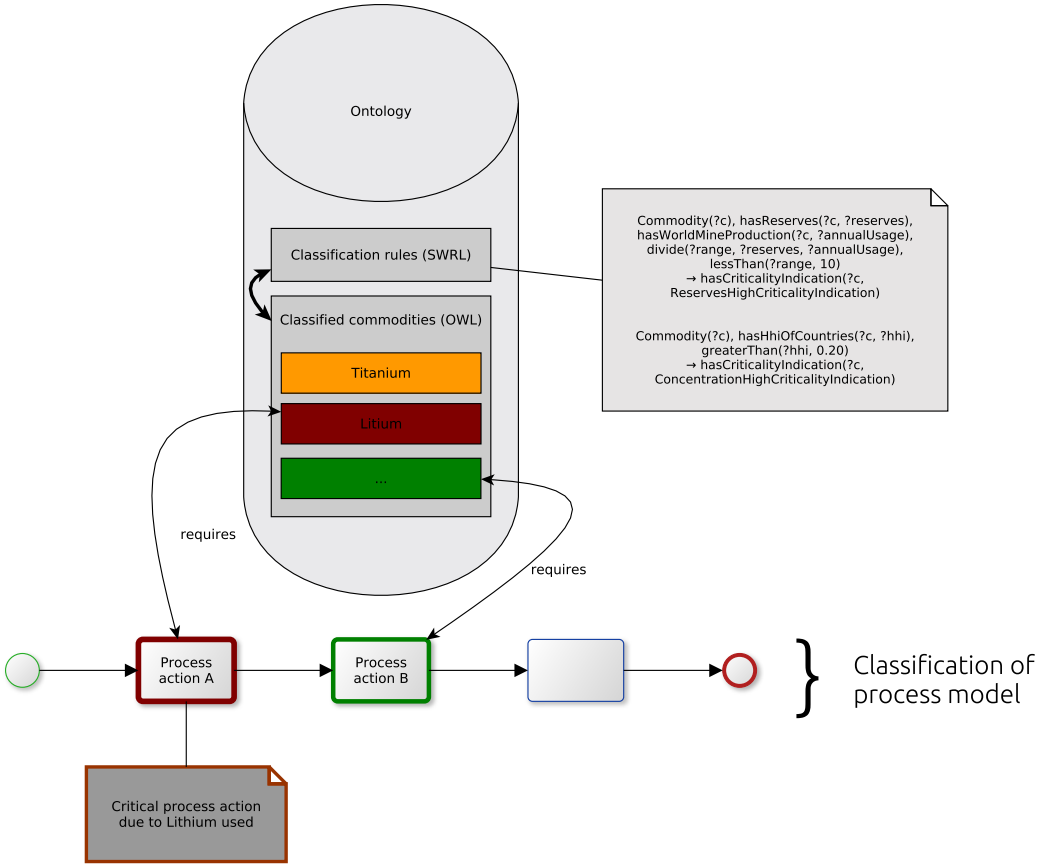

Figure 6.2: Integration of resource classification with semantic process modeling

StartEvent Type **StartEvent**
Prepare_board Type **ProcessAction**
Prepare_battery Type **ProcessAction**
Assemble_parts Type **ProcessAction**

StartEvent *hasSubsequentProcessAction* **Prepare_board**
Prepare_board *hasSubsequentProcessAction* **Prepare_battery**
Prepare_battery *hasSubsequentProcessAction* **Assemble_parts**

Prepare_battery *hasResourceRequirement* **Lithium**
Assemble_parts *hasResourceRequirement* **Copper**

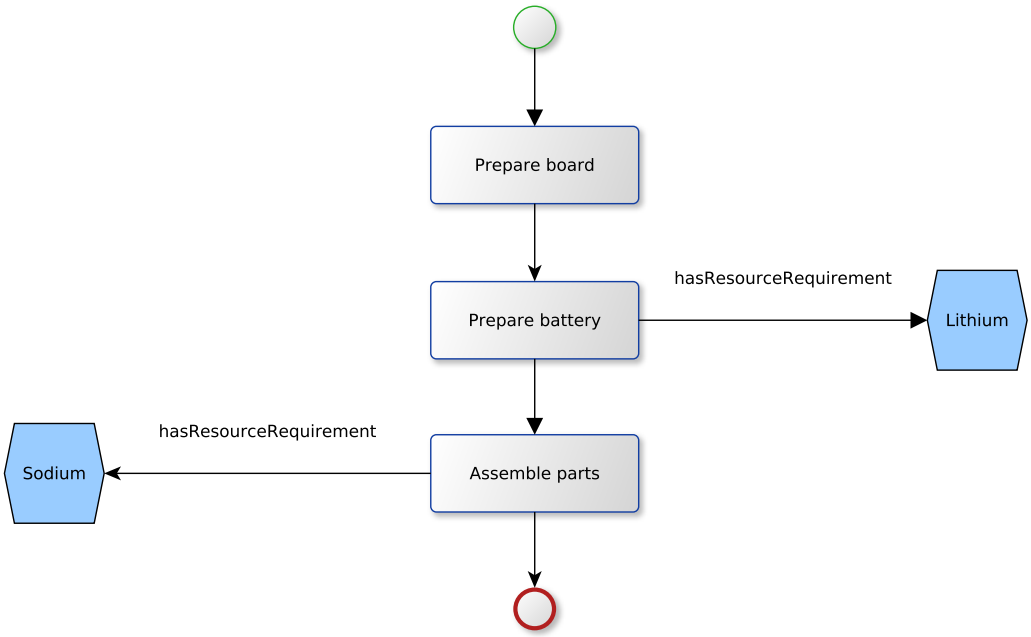

Figure 6.3: Demonstration example

As shown in Listing 6.7 Lithium is considered to be highly critical. The same applies for Sodium, the explanation path can be found in Listing 4 in Appendix 8.3.

Due to the fact both process actions use a commodity that is classified as highly critical, we expect both process actions **Manufactoring_step_2** and **Manufactoring_step_3** to be critical alike. Querying the ontology for individuals of **CriticalProcessAction** lists both individuals as expected. Adding the *contains* property as discussed earlier, the classification of the whole model is performed as expected, as well.

6.7 Conclusion

In this chapter, we presented an approach to classify commodities regarding their criticality. Our approach builds upon criticality indicators, well-known from raw-materials research. We have shown a way to classify resources utilizing semantic technologies, namely OWL and SWRL which are well

suited for this task as they can handle great amounts of data while providing enough calculation and logics to complete the task. Additionally, the integration into process modeling as introduced in Chapter 4 as well as features such as the explanation path show the benefits of the presented method. Our approach enables the combination of automated data processing as known from classical database systems with the large expressive power of formal logics. Additionally, we showed how to integrate this functionality with semantic process modeling where process models are represented within ontology space solely. In the presented approach, the classification is done on a coarsely granular level. Though, extending the approach by more fine-grained levels that indicate detailed levels of criticality can be easily achieved by adding appropriate classes that have different levels of criticality equivalence, e.g., commodities having only one medium criticality indicator.

While the presented implementation is intended to offer a first approximation for non-experts, it might also help experts to fine-tune indicators instead of losing time with manual data processing.

Especially when handling hundreds or thousands of different processes and components, the latter can turn into a competitive advantage.

Chapter 7

Evaluation

This chapter demonstrates the applicability of the work presented in this thesis in two case studies from different domains. We will describe an implementation of the resource matching approach to decide about feasibility of processes in the following. We show the applicability of the approach by implementing a case study in the domain of automotive embedded systems as well as the eHealth domain.

Additionally, we will provide some application scenarios that show the possibilities enabled by the solutions developed in this thesis. Those scenarios are not intended to be full evaluations but to stimulate further thoughts for areas of application in which the approaches may be applied in.

We will describe the results of a quantitative evaluation in Section 7.1 that was carried out for the commodity classification approach. We compared the results of the automated classification approach on the resource ontology *RESon* using SWRL rules with an expert group, which conducted the criticality classification of resources manually.

Additionally, we will present a scenario based evaluation to show how the presented approaches support varying situations in Section 7.2.

7.1 Case studies and evaluation

In the following, we will show the application of the presented resource consistency approach in the *software engineering* (Section 7.1.1) and in the *health care* (Section 7.1.2) domain.

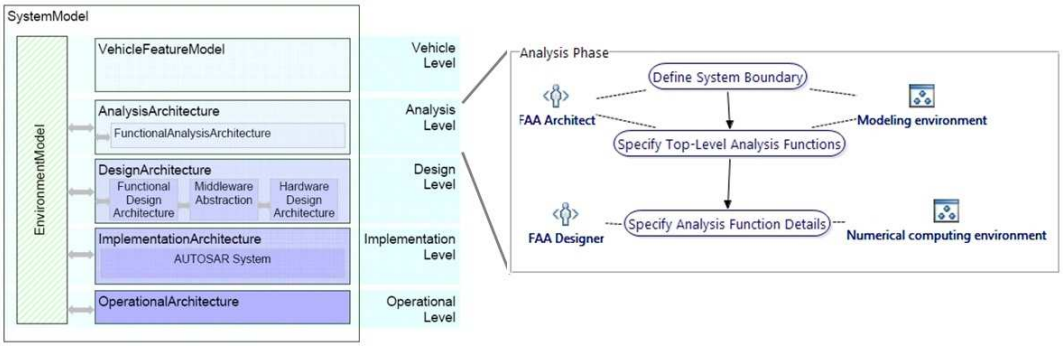

Figure 7.1: Example process: EAST-ADL abstraction levels (The ATESST2 Consortium, 2010)

Additionally, we conducted a quantitative expert evaluation on the resource classification approach which we will discuss in Section 7.1.4. Therefore, we compared the results of the presented tool in Chapter 6 and the results of participating experts in a setting as realistic and uninfluenced as possible.

7.1.1 Software engineering processes

As many of today's methodologies in the software engineering domain are realized using Eclipse Process Framework (EPF) (Eclipse, 2008), we used this framework for the following software engineering use case. This enables processing already available methodologies, that were modeled using EPF and its underlying meta model SPEM (Object Management Group, 2008), where various *MethodComponentElements*, e.g., *RoleDefinition* and *ToolDefinition*, realize the RFs in the MMTS as introduced in Chapter 5. The taxonomy of resources within the OWL 2 ontology is implemented using classes as well as subclass relationships, i.e., inheritance dependencies between those as discussed in Chapter 5. As stated in Chapter 5, the resource description within *RESon* varies in different domains. Therefore, we slightly modified this ontology for this use case: *RESon* contains a class `Resource` that in turn has two subclasses `Staff` and `Infrastructure`. Both contain the concrete resources $\mathcal{R}_c$ (such as employees, machines, tools) modeled as OWL individuals, respectively.

Process model

The process example in this use case is shown on the right side of Figure 7.1. Although, our approach would be applicable to the entire lifecycle, we picked out the analysis phase to demonstrate the matching of required resources because of the large dimensions of the entire process.

The analysis phase is basically composed of a sequence of the three process actions

- *Define System Boundary*,
- *Specify Top-Level Analysis Functions*,
- and *Specify Analysis Function Details*.

Those are detailed by various resource fragments in turn. Hereby, *FAA Architect* and *FAA Designer* represent human resources, i.e., roles to paraphrase responsibilities, and required skills of the primary performer of a task. Unfortunately, the methodology only provides little information about applicable tools and infrastructure resources. Hence, we introduced *Modeling environment* and *Numerical computing environment*, which support the respective process fragment, for demonstration purposes.

Resource ontology

Figure 7.2 depicts the fictive, company-specific resource pool ontology *RESon*, which has been developed along the guidelines presented before. The screenshot is taken from Protégé-OWL editor (Knublauch et al., 2004) which has been used to model the ontology parts of the use case.

The ontology comprises several skills and features to describe available resources. This resource taxonomy was derived on the one hand from personal experiences, and on the other hand from EAST-ADL-related deliverables, such as (Feiertag et al., 2009) and (The ATESST2 Consortium, 2010). As stated before, both the resource taxonomy as well as the details about skills and capabilities are very domain specific and have to be extended and adapted by enterprises. Using these skills and features, the concrete company-specific resources, e.g., Bob, can be defined in detail. As presented in Chapter 5, skills are assigned to individual resources by defining type axioms. For instance, Bob is defined as follows.

Bob Type Staff
and hasSkill some ASICSkill
and hasSkill some TimingSkill
and hasSkill some ProjectManagement .

Another possibility to define skills for Bob is by putting the individual into a prepared, specialized class, such as *HardwareArchitect*, which we called *PPSs* in the theoretical description of the approach in Chapter 5. In OWL 2 the implementation of this concept is a class having the following SubClassOf axioms attached:

HardwareArchitect SubClassOf hasSkill some AsicSkill
and hasSkill some TimingSkill .

Each individual, being member of class HardwareArchitect will have both object property assertions with both skills, automatically. For instance, given we have an individual Alice that is member of class HardwareArchitect, the OWL individual Alice has both AsicSkill and TimingSkill attached as object property assertions attached.

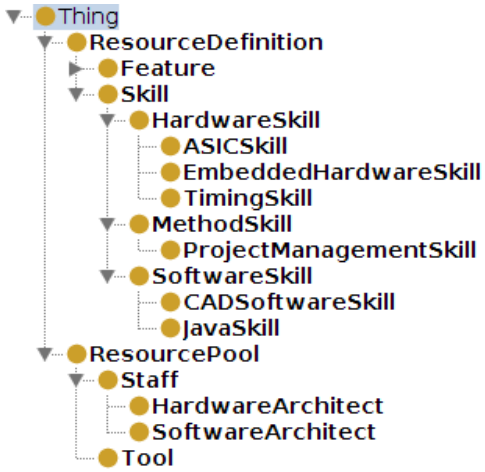

Figure 7.2: *RESon*of software engineering case study

Mapping and Analysis

The mapping between the process models and the *RESon* is defined in a separate ontology in this use case, which includes the *RESon*. Such a mapping

axiom is expressed as an equivalent class axiom in OWL 2, which consists of an atomic class and a complex class expression. The atomic class represents the resource reference of the same name in the process model. The complex class expression, however, is a logical combination of classes in the resource taxonomy. Usually, this is a conjunction of skills or capabilities which defines a subclass in the resource model and, hence, subsumes several resource taxonomy classes.

For instance, the following definition determines that the resource fragment FAA_Architect as depicted in Figure 7.1 is mapped to, i.e., is equivalent to, a member of Staff who has knowledge about ASICSkill and TimingSkill.

```
FAA_Architect EquivalentTo Staff
                           and hasSkill some ASICSkill
                           and hasSkill some TimingSkill .
```

At runtime, the algorithm which was introduced in Chapter 5 queries the *RESon* for all resource fragments of the process, and determines feasibility of the fragments and the complete process. Hence, it evaluates that a resource in the pool, concretely Bob, satisfies the requirements of the FAA_Architect. Since this is also the case for the other RFs in our example process, the algorithm concludes that the process is, in general, feasible. If this would not be the case, then the algorithm would inform the user about the respective unsatisfiable fragments. In this case, the user can utilize this information, in order to mitigate the problems, e.g., by adapting the process or developing the resource pool.

In this use case from the software engineering domain, we showed a slightly modified implementation of the resource matching approach. We utilized Eclipse and used an intermediate ontology for matching requirements with *RESon*.

In the next section, we will discuss the second use case from the eHealth domain.

7.1.2 eHealth process

In this section, we will demonstrate applicability of the resource matching approach in the second use case taken from the eHealth domain.

According to (Oh et al., 2005) a heavily used definition for "eHealth" is one given by Eysenbach (Eysenbach, 2001):

Definition 7.1.1 (eHealth) *"E-health is an emerging field in the intersection of medical informatics, public health and business, referring to health services and information delivered or enhanced through the Internet and related technologies. In a broader sense, the term characterizes not only a technical development, but also a state-of-mind, a way of thinking, an attitude, and a commitment for networked, global thinking, to improve health care locally, regionally, and worldwide by using information and communication technology."*

As the definition says, eHealth is a discipline attempting to combine healthcare and computer sciences and finally aims to improve healthcare by using IT. A countless number of scientific publications target on patient safety and improving quality of care. Additionally, there are numerous research projects planned or started as of today (European Commission, Information Society, 2012).

A report by the European Union (EU) published in 2012 entitled "Business Models for eHealth" shows that the usage of process modeling within the eHealth domain is an interesting topic. The report states that "the identification of potential links between the eHealth literature and literature associated with business modeling is complex". Though, there is initial work published lately, that describes the usage of well-known process modeling languages for the eHealth domain (Müller and Rogge-Solti, 2011).

We will show how the resource consideration approach, as pointed out in Chapter 5, can be applied to guidelines or reference processes in the eHealth domain.

Although research for reference process models or reference guidelines for eHealth is at its beginnings, there is some research available on that topic (Hübner-Bloder et al., 2005). The usage of process modeling and the impact of enterprise modeling enabling a holistic view on dependencies is also discussed (Haux et al., 2004). We will continue with an example process within a so-called Hospital information system (HIS). A HIS is defined as "a sociotechnical subsystem of a hospital, which comprises all information processing as well as the associated human or technical actors in their respective information processing roles" (Haux et al., 2004).

Process model

We will demonstrate our approach on an example process model showing a part of the admission process in the Department of Child and Juvenile Psychiatry at Plötzberg Medical Center and Medical School (PMC) (Haux et al., 2004, p. 71). A very similar model was used in the subsequent edition of the book where the modeling is considerably better in terms of clarity and semantics (Winter et al., 2010, p. 50). A representation of the model described in the books can be seen in Figure 7.3. We utilized the extended Oryx process modeling environment as introduced in in Chapter 5 for this use case. The process describes how admission of patients is organized. The process starts with a patient calling the secretary where the need for admission is checked first. In case the need for admission is negative, the call is finished and the process ends. In case the secretary determines a first need for admission, the call is forwarded to a physician that again checks the need for admission. In the negative case the process again ends with the call being finished. In case the admission diagnosis of the physician is positive, the patient is forwarded to the secretary again, where an appointment is arranged. In the next step, the administration must be informed, so the patient is forwarded to the administration where it is checked whether the patient has been at the hospital before. If so, the old patient record must be found in order to attach the information about the new appointment. Otherwise a new patient record is prepared.

When looking at the graphical notation of the example model, several details about requirements remain hidden. The obvious resources which are required are the people that are modeled by the swim-lanes. In order to successfully execute the model, three people must be available.

- Administrative staff
- Secretary
- Physician

Additionally, some IT support is obviously necessary, although this is not represented anywhere within the model. The task to *arrange appointment* assumes that there is some kind of groupware or calendar system multiple people have access to, as secretary staff arranges those appointments, but physicians are those who medicate patients. Additionally, there must be an

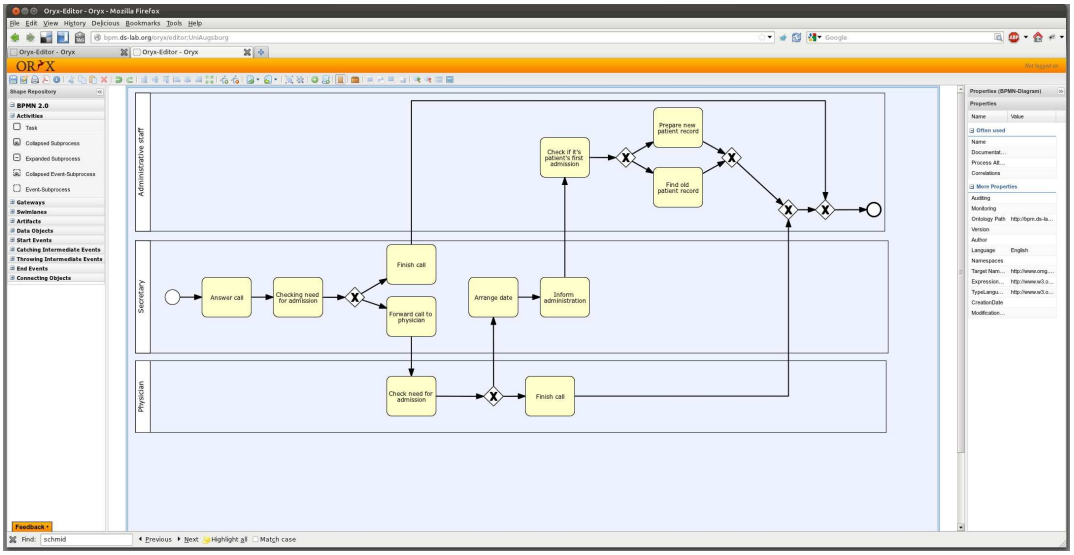

Figure 7.3: Admission process model example from eHealth domain

information system (HIS) to handle patient records which is handled within the lane of the administrative staff.

We modeled the obvious requirements such as calendar and patient record systems within the resource ontology *RESon* to show the resource feasibility analysis in the following. In addition, a magnitude of resource and skill requirements could be thought of in the eHealth process model of this use case. As mentioned before, there are obvious requirements such as several people, a HIS, calendar and a telephone switchboard. Though, it could be better if the "check for admission" task that is handled by a physician should be carried out by someone who specialized in this kind of work and maybe achieves faster and better results because of specialized trainings in patient admissions. In order to demonstrate how our approach could help HIS, we assumed the following resource requirements for the process model.

- All tasks within the particular swim-lanes can be accomplished by someone who is in the respective class, with the exception that the physician must be one that has a *specialized training in admittance*.
- We do *not* model the telephone switchboard.
- The tasks to arrange an appointment requires an *IT system* that has a capability named "GroupwareCalendar".

- The tasks for the administrative staff are accomplished within *one system* for patient records.

Therefore, we modeled the resource requirements shown in Table 7.1 into the Oryx process editor as a second alternative implementation to the environment used in the SE use case. The left column lists the process actions within the admission process while the respective requirements are shown in the right column.

Process action	**Requirement** (Manchester Syntax)
• Finish call	Staff
• Answer call • Check need for admission	Secretary
• Check need for admission (Physician)	Physician and hasSkill some {AdmissionTrainingSkill}
• Arrange date	hasCapability some {GroupwareCalendar}
• Check if it is patient's first admission • Prepare new patient record • Find old patient record	AdministrativeStaff

Table 7.1: Resource requirements definition for eHealth use case process

Resource ontology

The resource ontology for this use case again consists of a slightly modified modeling within *RESon*. In order to fit the eHealth domain and admission process, we added a category Groupware as subclass of ApplicationSystem within the taxonomy. We modeled one concrete resource MicrosoftExchange of type Groupware that additionally has a groupware calendar capability. Furthermore, we added two new classes Secretary and Physician as subclasses of Staff to the taxonomy. The concrete resources Lisa and Heidi are both secretaries while Mike is an physician who also has the specialized admission training. Figure 7.4 shows an excerpt from the resource knowledge base for this use

case. On the left, the taxonomy can be seen while on the right side, details for the concrete resource Mike are displayed.

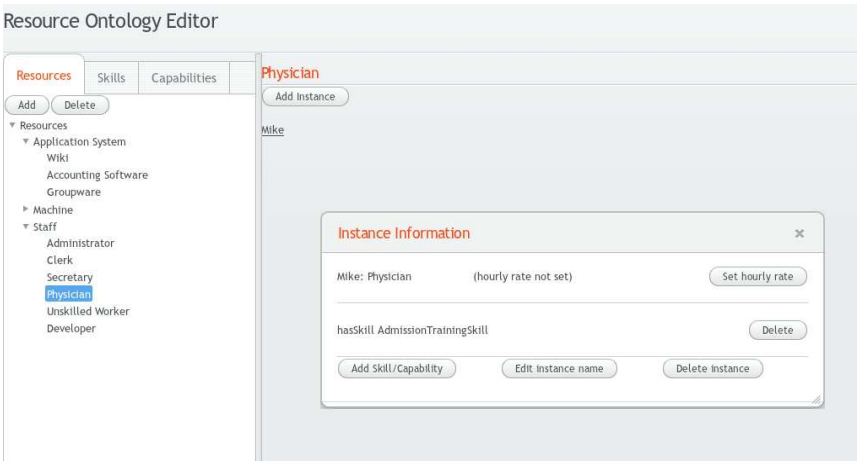

Figure 7.4: *RESon* modeled for eHealth use case

Mapping and Analysis

After the check for fulfillment of the resource requirements is accomplished, the Oryx editor shows at each process action whether it is executable or not. In the example (see Figure 7.5 for the example result after the resource check) the whole process is fully executable, as can be seen by the green ticks at the upper left corner of each process action. If it was not executable at all, a red cross would be displayed.

Additionally, the concrete resources that can handle the appropriate tasks are displayed within the editor. For example, “Heidi” as secretary for the first task to answer the call, or “Microsoft Exchange” which has been modeled to be able to handle groupware calendar capabilities.

To sum it up, the presented process model is only an excerpt from a process that is — compared to other processes within hospitals, where patient data might be delivered by multiple external systems and more people getting involved — elementary for demonstration purposes. Though, the checking for feasibility using resource requirements, can handle large resource knowledge bases or process models and therefore add great benefit to process modeling in the eHealth domain in our opinion.

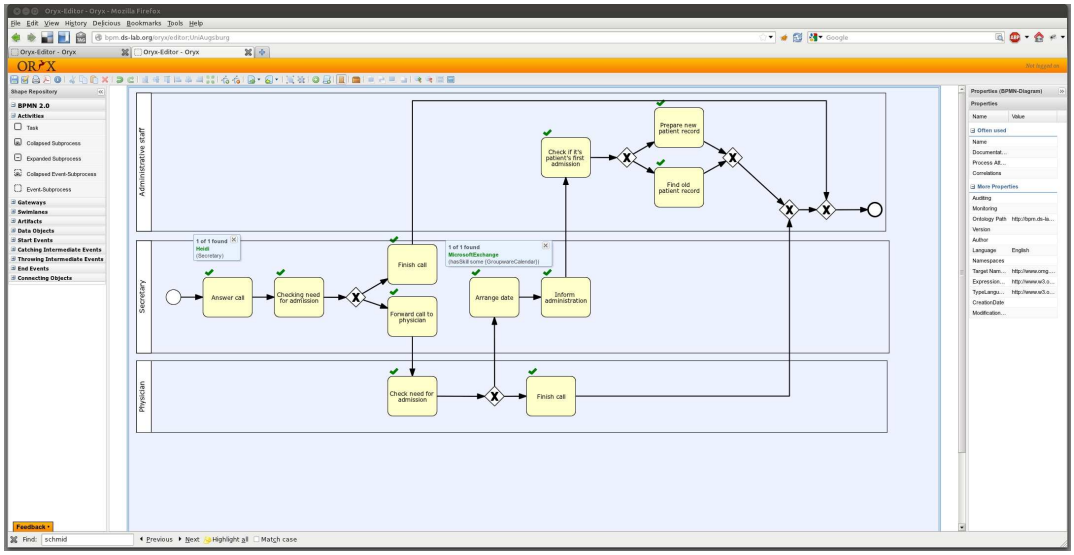

Figure 7.5: eHealth example process model after resource feasibility check

7.1.3 Application scenarios

In this section, we will show some diverging application scenarios that offer different perspectives on certain challenges within enterprises today. For each scenario, we will demonstrate how the approaches presented throughout this thesis, can support to solve problems and improve handling of the respective scenario.

Vacations

This scenario describes a frequent issue in companies: Whenever employees apply for leave, the question arises whether the processes, the respective employee is involved in, are still fully executable during the requested vacation.

Although sometimes employees name a colleague who should be responsible during leave, this is problematic as it is difficult to know whether the representation really has all skills and competencies that are necessary to support all processes someone is involved in. Additionally, in larger teams, the absences are not necessarily known to the whole team. In case someone else who is involved in a four-eyes principle is already on vacation or sick, this also influences the consistency of processes when another employee leaves.

In the worst case, processes are not executable because of missing competencies or skills and have to be paused until a person returns from vacation. In case of service processes this leads to delays at least. In processes where certain time constraints have to be met, such pauses can have much more significant consequences.

Solution statement Given the processes are modeled together with resource requirement definitions as introduced in Chapter 5, the problem can be handled by simulating the absence by removing the person from the resource pool and checking the process models repository for consistency. If the removal of the respective person does not have any influence on the consistency of the processes, no problems should come up during a requested vacation.

Restructuring departments

A quite similar application scenario as the one discussed before is the following. Given that departments are restructured such that, e.g., two departments get merged into one or one department gets shut down completely, the handling of the processes in which the respective departments are involved in must be checked for consistency and probably adapted.

Solution statement Again, given the processes have resource requirement definitions attached, the first objective can be accomplished by the resource checking approach described in the application scenario before. The replanning of processes can be achieved by the process model adaptation approach presented in Section 5.6.

Consolidating IT

SOA has been one of the primarily influencing IT initiatives in the past decade that offers great benefit on the one hand, but renders IT operation into an increasingly complex task on the other hand.

SOA attempted to break up large, monolithic software systems into a *larger number of smaller, distributed* services that cooperate. Additionally, the concept of SOA aims to allow arbitrary users within (or even outside of) the enterprise to use the software services. This encounters the frequent problem in large enterprises, where the very same service is developed a number of times

independently, because the existence is unknown, or the service is not usable due to technically implementation.

SOA adds great benefit to the consumers of the offered resources. Though, it adds a large number of unmanageable dependencies to the providers of the resources: SOA pursues an intensive usage of services, which obviously leads to dependencies of such services. In a third party funded project with a large German car manufacturer, the challenge regarding which processes call which services when and by whom was described as one of the major problems in highly heterogeneous environments.

Given an IT department is in need of consolidating equipment, e.g., IT hardware or software services, it becomes a serious challenge to decide if equipment might get removed or it is still in use due to the extended coherence of IT landscape.

Solution statements When managing process models within ontology space as discussed in Chapter 4, either by transforming existing process models into a semantically equivalent mapping, or by describing new processes directly with semantic technologies, this problem can be mitigated. Although the mentioned interdependencies will not disappear by only handling information about the context and correlation within ontology space, the possibilities of using querying technologies on a technological foundation that allows easy integration of other information adds benefit: When consolidation requires downtimes of single machines or services, the consumers can be easily identified within the ontology to get notified. Additionally, failures can be communicated better by automatically targeting all users. Additionally, single point of failures for critical process can be detected easily.

Resource market fluctuations

As already mentioned in Chapter 6, the commodities market is subject to consistent fluctuations. Given the presented approach for classifying commodities leads to a change of a certain commodity that is (according to the rules defined by the enterprise) classified to be highly critical. Let's assume that it is decided to replace the commodity by a substitute, it is likely that the underlying process has to be adapted because of the new commodity.

Solution statement The identification of criticality of commodities by rules is already used in the assumption of this application scenario. Additionally, the process model adaptation approach presented in Section 5.6 can be used to re-plan the respective fragments.

7.1.4 Quantitative evaluation

As discussed in Chapter 6, the core result of the approach is a classification of a number of non-renewable raw materials as presented in Table 7.2. As this classification is performed automatically, it enables up-to-date results as well as nearly effortless distribution within large companies, or even complex supply-chains. The explanation feature of ontology reasoners allows unexperienced users to understand why, e.g., a product or (even more interesting) a process that makes use of certain commodities is classified to be critical, for example.

Highly critical raw materials	Cobalt, Chromium, Germanium, Lithium, Tungsten
Medium critical raw materials	Copper, Indium, Platinum, Vanadium
Non-critical raw materials	Silver

Table 7.2: Criticality classifications as returned by tooling support

Thus, the presented architecture allows real-time checking of processes handling commodities and explanation of results based upon consistent logical rules that are inspected and implemented by an automated reasoner. Though, these possibilities depend on the reliability of the system results, which are evaluated in the following sections.

Evaluation method

In order to validate the suitability of the approach presented in Chapter 6 and the tool, we conducted an expert evaluation based upon eleven experts from industry and science.

We will discuss three findings of this evaluation in the following.

- The criticality assessments of the mentioned expert group.

- The criticality assessment of the tool implementation.
- The criticality assessment of two published studies (Pfleger et al., 2009; Achzet et al., 2011) serving as a benchmark.

We will compare the results from the first and second observations to the benchmark studies to determine if the automated classification, or human experts are closer to scientific criticality assessments. All participating experts have some degree of experience working with non-renewable resources, be it in their everyday work or as a topic of their research, respectively.

Quantitative evaluation

We provided our experts with a description of the task and a questionnaire, a set of data regarding the ten relevant non-renewable resources, and four published studies or guidelines regarding the definition and analysis of criticality. We explicitly refrained from giving any definition or description of criticality that we made up ourselves to prevent any bias. Instead, the studies we provided represent a selection of recognized approaches that an expert would probably refer to based on his expert knowledge or a short search for specific literature.

Those studies are listed in the following in detail.

- "Assessing the long-term supply risks for mineral raw materials—a combined evaluation of past and future trends" (Rosenau-Tornow et al., 2009)
- "Critical raw materials for the EU" (Working Group of the Raw Materials Supply Group, 2010)
- An excerpt from (Pfleger et al., 2009) showing the weighting of risk indicators that, e.g., suggest country risk and reserves-to-production ratio to be both 12,5% of the overall risk indication.
- An excerpt from (Committee on Critical Mineral Impacts of the U.S. Economy, 2008) that describes a possible criticality matrix as an example.

Thereby, the experts had the identical information available as the tool as well as some additional data that our tool did not utilize.

We designed the evaluation to encourage our experts to rely on quantitative indicators and their personal knowledge, or their knowledge from the provided studies on the rating of these indicators exclusively. Therefore, in order to prevent the influence of previous knowledge about criticality of commodities, we made the names of the metals anonymous and used letters (*A* to *J*) instead of the commodity labels.

We asked the experts to assess the criticality of the ten presented metals named *A* to *J* by assigning them *high*, *medium* or *low criticality* consulting the supporting documents attached to the questionnaire.

The results are shown in Table 7.3. It can be seen that the manual criticality assessment generates rather heterogeneous results. This can be interpreted as a consequence of equally heterogeneous definitions of criticality that are used in industry and science.

At the same time, we used our tool and our semantic rules to calculate the criticality of the ten metals at hand. The results of the automated classification are the cells marked with gray background cells in Table 7.3. E.g., silver was classified as *not critical* by the automated tooling.

It can be seen that the most common answers by the human experts are not identical with the automated criticality assessment in every case. Thus, the question arises, if the manual or automatic classification achieves better results.

As mentioned before, we used two published papers about criticality analyses (Pfleger et al., 2009) referenced to as "*VBW*" and (Achzet et al., 2011) referenced to as "*BP*". These studies classified all the used metals as highly, medium or non-critical as a baseline for comparison with the classification with our experts.

Afterwards, we performed an analysis of the precision and recall of the tool and the human experts, respectively, in comparison with this baseline. In contrast to classical analyses determining precision and recall, we have three possible states (*highly critical*, *medium critical*, *non-critical*) instead of two (true/false or critical/non-critical). Thus, we consider both incorrect states as false and the one correct state as true.

The results of this analysis are shown in Table 7.4.

Metal	**Code**	**High criticality**		**Medium criticality**		**low / no criticality**	
		#Experts	*Studies*	*#Experts*	*Studies*	*#Experts*	*Studies*
Silver	A	2	-	1	VBW	8	BP
Cobalt	B	1	VBW + BP	4	–	6	–
Chromium	C	1	VBW + BP	4	–	6	–
Copper	D	2		5	VBW	4	BP
Germanium	E	4	VBW + BP	5	–	2	–
Indium	F	4	VBW	6	BP	1	–
Lithium	G	2	VBW	4	BP	5	–
Platinum	H	6	VBW + BP	3	–	2	–
Vanadium	I	0	–	6	BP	5	–
Tungsten	J	3	VBW	7	BP	1	–

Table 7.3: Evaluation results: Human and automated criticality assessments in comparison.

Criticality assessment	Human experts (average values)				Tool			
	VBW study		BP study		VBW study		BP study	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
High criticality	84%	27%	48%	27%	100%	71%	60%	75%
Medium criticality	13%	27%	51%	52%	50%	25%	50%	50%
No criticality	0%	0%	31%	54%	0%	0%	100%	50%

Table 7.4: Evaluation results

Discussion

As can be seen from the evaluation results, the tool performs quite well compared with our human experts. Generally, there is only one case where the results returned by the tool do not fit with any study at all (platinum). In all other cases, the results fit with the findings of a single or even both studies.

It is interesting to note that the human experts did especially well at resources with medium criticality - perhaps due to the error of central tendency. However, in the case of resources with high criticality, which account for 40% or 70% of all resources, depending on the respective study, our tool performed considerably better. In six out of ten classifications, the tool shows a higher precision or recall than the human experts.

In addition, while in these cases the tool performs considerably better than the human experts, the inverse case is contrary: In all four cases where the human experts performed better than the tool, those results are only less than 10% better than the tool. Two further cases are not relevant for our analysis as all values are zero, due to the fact that in the VBW study, none of our resources where considered as non-critical.

Additionally, it is worth mentioning that definitions of criticality and of criticality criteria still seem to be at an early stage. While we tried to adapt the definitions given in literature to provide more consistency with published studies on the criticality of non-renewable resources, this issue shows that further research is necessary on the functional definition of criticality, which is out of scope of this thesis. Here, the study we selected as reference for our comparisons might be disputable, while this problem would probably apply to all other studies as well. In particular, providing a transparent, detailed and explicit specification of possible criticality criteria seems to be an indispensable first step towards further automated criticality assessments, while purely qualitative criteria are likely to be hardly digestible for a fully automated classification approach as presented.

On the other hand, if some criteria cannot be specified quantitatively at all, it is very likely that those would not be reproducible as well. In addition, as expected, some experts performed better than our tool while others did not. Thus, one central result is that only *experienced* experts can provide a reliable criticality assessment, while employees that mainly have other areas of responsibility tend to produce a considerable amount of false classifications. It is certainly not a new finding of this evaluation that experts are better at

performing complex tasks (as classifying commodities as discussed here) than non-experts. Though, we showed that the classification itself is a highly complex tasks that should not be performed by non-experts if the results are of any concern.

A good combination for criticality definition could be experts that feed tools such as the one presented, and enable integration into areas where only classification results and explanations are used.

In addition, data availability is an important issue. While it is subject to further research if an expanded data set would in turn lead to more precise criticality assessments, an improved availability of data, especially in machine readable form, would certainly ease and accelerate IT-based criticality assessments while enabling further criticality criteria. Currently a number of freely available documents had to be processed manually into formats that can be automatically processed.

An interesting evolution that could enable such automated processing enormously is the research done at Linked Data where Semantic Web technologies such as RDF are used to describe knowledge in a computer-understandable way on the web. Although this had a quite slow start, there is currently quite some activity as governments open data in form of Linked Data. Given data such as prices, stock, reserves would be published based on Linked Data, the automated subsequent processing would greatly benefit. We will discuss this in the outlook and future work in greater detail.

We presented an quantitative evaluation of the resource classification approach discussed in Chapter 6. Therefore, an expert group conducted the classification of the raw materials manually. We compared these results with the automated tooling from our approach.

We will discuss findings of a scenario-based evaluation in the next section.

7.2 Scenario-based evaluation

Using so-called scenarios for evaluation is mainly known from evaluations of software architectures and comparison of design alternatives (Kazman et al., 1996). Scenarios serve as brief descriptions of anticipated or desired use cases of systems or software architectures.

This approach is usually used in early phases of software development cycles

to expose problems within software architecture, and see impacts on the architecture in case of changes. It has been shown that such evaluations achieve positive results on various system properties such as maintainability or performance (Babar and Gorton, 2004).

There are several different evaluation methods.

- Architecture Level Modifiability Analysis (ALMA)
- Architecture Tradeoff Analysis Method (ATAM)
- Active Reviews for Intermediate Designs (ARID)
- Software Architecture Analysis Method (SAAM)
- Performance Assessment of Software Architecture (PASA)

While the methods obviously differ in detail, a comparison of different evaluation methods states that certain activities such as *scenario development*, and *scenario evaluation* are common between different approaches while "techniques of performing these activities are quite different" (Babar and Gorton, 2004). As our goal is to demonstrate benefits of the presented approaches and not to estimate performance or maintenance costs, we chose *modifiability*, *functionality* and *variability* from the possible set of quality attributes for evaluation (Clements et al., 2002)

(Kazman et al., 1996) describes several examples for evaluation of software architectures that were conducted. The repeating elements where the following:

- Develop scenarios.
- Perform Scenario Evaluations.
- Discussion "What did we learn".

Very similar to the template used in the evaluations in (Kazman et al., 1996), we will describe a scenario that would heavily influence the presented architecture. We will describe the second and third step of the scenarios in the following.

There are special technology changes that have great influence on the architecture of the presented approaches. We identified two architectural details

that are prone to changes. The formal language used to describe ontology space in our approach is the first one. The other one is about the graphical notation languages that change quite frequently. We will describe both in detail in the following.

Ontology language A significant part our the approaches presented throughout this thesis are build upon OWL or rather OWL 2 as the ontology description language on top of DL. Changing this component can be necessary because of a magnitude of reasons.

- Upcoming new technologies with an extended feature set.
- Better tool or commercial support.
- Existing know-how, or other commitment within enterprises.

These are only few possibilities for the demand to switch to a new or another technical implementation language.

In order to face the process of replacing the implementation language, we identified the parts that would require re-work.

- *PMon* **description language**. Given the current implementation of *PMon* should be replaced by another language, basically there are few features the replacement language has to support to successfully replace OWL. The possibility to define objects that correspond to individuals within OWL, and a taxonomy concept similar to classes as well as a feature corresponding to object properties and object property assertions between the individuals to define the control-flow and connection to other enterprise architecture objects. Furthermore, discrete data should be able to be integrated into ontology space to, e.g., attach information about usage, prices and the commodity information into ontology space. This is done using data properties in OWL.
- **Querying on top of** *PMon*. The presented possibilities to query for usage of connected objects with processes or the integration of the resource classification approach requires a querying language on top of the ontology language. Used utilized Manchester Syntax as well as SPARQL for these queries.

- **Querying language for resource requirements definition**. We used Manchester Syntax (Horridge et al., 2006) for requirements definition because it enables directly executable queries within ontology space. A replacement querying language should be as easily readable as Manchester Syntax.

- **Resource ontology** *RESon*. The modeling of resources has very similar requirements as those necessary for potential replacement languages of *PMon*. The PPS feature that enables quick assignment of properties to resources requires that these properties can be defined upon abstract classes. Individuals within the respective classes inherit the properties automatically.

- **Input and Output parameter description** to enable process model adaptation considering resource requirements has similar requirements as those necessary for *PMon* or the resource ontology.

- **Rule language**. In order to integrate the commodity classification approach as discussed in Chapter 6, a rule language is necessary that is capable of arithmetic operations on properties (data properties in OWL) for classification.

To sum it up, the ontology description language is an integral part of the approaches. Though, given the above mentioned requirements are fulfilled by a replacement technology, the ontology language can be substituted.

For the general architecture, this implies significant changes although none are impossible as far as we could see in the scenario analysis.

Graphical notation languages The second fundamental architectural change concerns graphical notation languages that were suspect to changes in the past years. For example, BPMN as one of the well-known and commonly used notation languages was continuously emerging over the past decade: BPMN 1.0 was published in 2004 (BPMI, 2004), version 1.1 was published in 2008 (OMG, 2008) while 2.0 was published only three years later (OMG, 2011). UML, which allows process modeling by using activity diagrams, was developed further and enhanced at a similar rate.

Therefore, one might expect further changes to existing notation languages or even new notations coming up. This again means that the presented transformations into ontology space might have to deal with these changed

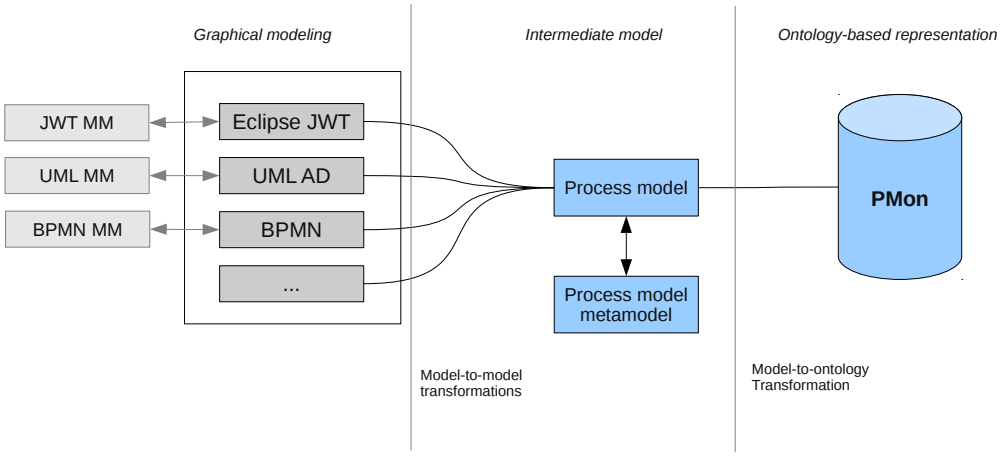

Figure 7.6: Transformations from graphical models into ontology space

or new languages as well. We will show how this influences the presented approach and how one can deal with the changing environment.

The approach presented in Chapter 4 uses a modular way for the transformation. Therefore, the left part of the transformation as shown in Figure 7.6 has to be replaced by an initial transformation into the intermediate model.

Given that the new notation language is standardized and computer-readable, e.g., in XML format, the existing transformations into the intermediate level should be easily adapted to enable transformation of changed or new graphical notation languages into ontology space. Therefore, changing graphical notation languages could also be handled.

In this section we discussed two major influencing scenarios on architectural components of our approach. We can state that both scenarios would significantly influence the approach. Though, if substituting techniques or tools comply with the given requirements, the changes could be handled and the approach still be applied.

Chapter 8

Conclusion

In this chapter, we will show solutions for the problems disclosed in Section 1.1 and give a short summary of the presented approaches. Thereafter, we will discuss the findings presented in this thesis and show future work.

8.1 Summary

In Section 1.2, the following objectives were defined for this thesis.

- Integrating process models with a resource knowledge base to check consistency of a model at design-time.
- A dense reference ontology for modeling process models including resource constraints.
- A formal transformation specification for graphical process models into the reference ontology.
- An advanced specification and implementation of an algorithm capable of planning process models considering resource constraints.
- A classification system to categorize resources, e.g., commodities used within process models.
- A way to query process models, e.g., regarding utilized IT services.

In order to encounter the problems discussed, we used semantic technologies to build the following solutions.

Process model reference ontology "*PMon*" We defined an ontology for process models based on OWL 2. It allows definition of process actions, complex control-flow structures such as parallelization and XOR gateways as well as control-flow of the processes. It is integrated with the resource classification approach as models using critical resources are marked as being critical as well.

Automated transformation of process models We defined transformation patterns to map existing process models from various modeling languages into ontology space. Therefore, we formally described necessary transformation steps for process elements supported by *PMon*, and described a framework to automatically accomplish the transformations on top of QVT and XSLT. After transforming models into ontology space, querying for relations of models with, e.g., IT landscape is easily possible. We demonstrated this functionality using the running example.

Resource knowledge base "*RESon*" We defined a resource knowledge base within an ontology based on OWL 2. The ontology allows description of enterprise resources such as human resources and machinery. The resource ontology supports detailed description of skills and capabilities of resources. This can be used to describe such properties in arbitrary detail. In order to accelerate the definition process we introduced a new concept called PPS to allow definition of properties on an abstract class level where individuals of the respective class inherit the properties.

Integration of process modeling and resource knowledge base We integrated process modeling and the resource knowledge base so that consistency of a process model is determinable at design-time. This means that on the one hand, process models can be extended by resource requirements. This is accomplished using a standardized, well-established language. On the other hand, these requirements can be matched against the resource knowledge base *RESon* to decide about realizability of a process model considering the resource requirements. Finally, we added an operationalization of resource properties to select the "best" model from a set of alternatives considering, e.g., costs.

Extension of automatic planning approaches We extended an existing planning approach (SEMPA) to enable adaptation of process models on the one hand. Therefore, we use the existing planning algorithm and applied it to the domain of process model adaptation. Additionally, we extended this approach by the before-mentioned resource checking component.

Evaluation

We evaluated the solutions in Chapter 7. Therefore, we presented two case studies as well as further application areas. The use cases demonstrate applicability of the presented theory within the software engineering domain (Section 7.1.1) and within the eHealth domain (Section 7.1.2). Furthermore, we presented a quantitative evaluation of the classification approach (Section 7.1.4) and concluded with a scenario-based evaluation approach in Section 7.2.

8.2 Discussion and future work

We presented several approaches to improve process modeling using semantic technologies in this thesis. Though, there are limitations that require future work. We will discuss these in the following.

Considering the process model life-cycle as introduced in Section 2.2, this thesis clearly focused on the modeling phase. Given process models are serialized into ontology space (for example by using the approach to transform models shown in Chapter 4) a semantically based process runtime environment (process engine) could access those models within the ontology directly. Such an engine could add runtime logging information into ontology space, and enable powerful process mining on this data exploiting reasoning capabilities. The automated mapping of models into ontology space is a starting point for companies that tend to use ontologies as a central information repository attempting to also include process information within ontologies. For models containing complex control-flow patterns for which no appropriate mapping has been defined in Chapter 4, additional mappings of the information within ontology space as well as transformations for those patterns must be build. Though, as we defined the most important patterns, we state, that this is no limitation for industrial use. Furthermore, definition of additional patterns can be accomplished by taking those presented in this

thesis as foundation.

Regarding the resource consideration and process consistency checking approach pointed out in Chapter 5, there are conceptual limitations we will discuss in the following.

Consideration of runtime parameters is often neglected during the design-phase of processes. This includes a large number of parameters, that have great influence on the processes, however. For example, usually no time characteristics of processes are considered. This includes execution time of single process actions as well as whole process models. In addition, usually no assumptions are made regarding the time *when* processes get started, nor about the amount of processes that run in parallel. Though, process models are often simulated in order to get information about runtime characteristics.

This is a problem as a matter of principle and also affects the consideration of resources as presented in our approach. Given two or more processes get instantiated in parallel, there could be more resources necessary than expected when analyzing a single process model. In order to encounter this problem, assumptions that are used for simulating processes could be considered at design-time, too. For example, an obvious solution could be to add assumptions for the typical number of parallel running processes to the checking component. A more advanced approach could consider starting times of processes and use execution times and typical control-flows gathered from process logs, e.g., using process mining techniques. Further consideration of runtime properties would add great benefit to the approach. In general, an integration of information from different phases of the process modeling life-cycle might have great benefit and requires further work on possible integration possibilities.

An interesting application scenario, e.g., applicable within the eHealth domain that we did not examine until now, is to enhance existing reference process models or guidelines with resource requirements. Given that a hospital or other medical facility has a fully populated resource knowledge base *RESon* available. If such facilities would like to adapt a reference processes or guideline that comes with the resource requirements defined, it would be easy to compare the required to the currently available resources at the facility. This way, it would be clear to see which resources within the facility are missing to fully enable a reference process to be deployed.

The approach to classify commodities regarding criticality as discussed in Chapter 6 shows the possibilities of the ontology as central knowledge base.

The approach describes usage of SWRL to map classification rules into ontology space to enable reasoners to subdivide the set of commodities modeled within ontologies into subsets with regard to risk classification. We used real data from, e.g., stock markets gathered from various sources, to allow a close to reality analysis. As the evaluation with an expert group showed, the results returned from the reasoner on top of the rules are promising. We demonstrated that the approach can be used as a decision support system in commodity-utilizing processes within enterprises.

In order to further improve the approach, additional rules would help to refine the results. In general, more classification properties would back the classification decisions. The decision to use three levels of criticality could also be investigated, as more levels might also improve the overall performance of the approach.

A next step for this research is to fully automate the gathering of data necessary to classify criticality. This could be accomplished by using semantic technologies on the Internet, often referred to as Linked Data (Berners-Lee, 2006; Bizer et al., 2008, 2009; Heath and Bizer, 2011). As a kind of successor of the Semantic Web initiative, Linked Data becomes more and more popular today: There are already several governmental as well as industrial institutions that publish data in a computer-understandable way on the web today. E.g., The German National Library announced in January 2012 that its bibliographic data will be published using Semantic Web (or Linked Data) technologies (Hauser, 2012). This includes millions of entries of bibliographic data, as well as information about titles that are collected at the national library. In summary, there is an enormous growth of datasets as well as RDF triples within the Web of Linked Data. In 2007, 500.000.000 triples where counted. In the subsequent years, three-digit rates of growth led to 31.634.213.770 triples as of September 2011 (Bizer et al., 2011). The information gets more valuable if it is interconnected with with other large data sources. E.g., The British Library as well as the Dutch National Archive besides many others also publish their data using semantic technologies.

For our approach, using Linked Data to retrieve information about markets, prices, mining and usage details is a promising next step. This information available as Linked Data on the Internet could be retrieved and inserted into the ontology automatically. This would enable a fully automatic update of the information available, as well as an up-to-date analysis. This finally could lead to realtime analysis of commodities on top of that automatically gathered data.

8.3 Outlook

Currently, we are working on extending the web-based editor that was used for a prototypical implementation of the resource checking approach in Chapter 5 to serialize models directly into ontology space. We are using the rules for the transformation of process models introduced in Chapter 4 the before-mentioned chapter to save models using OWLAPI. This enhancement is especially useful if no process models exist which in turn means there is no need for automatic transformation from existing models in notation languages such as BPMN. Using this direct serialization into ontology space allows the full exploitation of the reasoning and querying capabilities as presented exemplary.

We implemented parts of the approaches presented in this thesis in several third-party projects with an industry partner in the automotive domain. The semantic-based representation of models is utilized at the car manufacturer within a large enterprise ontology to enable querying on top of process models.

It is very likely that research in the area of Semantic Web technologies in conjunction with enterprise applications will continue. Additionally, applications using such technology are likely to show up in the next years, since both libraries to build appropriate applications as well as toolkits such as graphical editors for ontologies are technically mature and ready to be used in production.

A research project that is — to some extend —- build upon results from research presented throughout this thesis is Semantic Enterprise Architecture Management (SEAM). SEAM is a publicly supported research project aimed to apply Semantic Web technologies in the area of Enterprise Architecture Management (EAM). The intended goal is to formally describe enterprise architectures using ontology languages in order to support planning of changes, especially in the area of IT landscapes. By describing architectures formally, planning and preparation of such changes can be considered thoroughly, embracing all technical dependencies and impacts. Altogether it is expected, that complex projects such as legacy system decommissioning can be mastered more easily. The presented work in this thesis is used as foundation to further investigate advantages of semantic technologies in enterprise architectures.

Hence, we state that research on enterprise applications utilizing Semantic Web technologies in general should continue in certain areas. Concrete further

development of the approaches presented in this thesis is supposable, too, as an application within industry was successful and showed off considerable advantages. Additionally, a subsequent research project to further investigate the topic is launched.

This thesis contributed with different approaches that improve process management using semantic technologies based upon formal logic and serves as a foundation for further application, for example within semantic enterprise architecture management.

Bibliography

W. M. P. v. Aalst, K. M. v. Hee, J. M. v. Werf, and M. Verdonk. Auditing 2.0: Using process mining to support tomorrow's auditor. *Computer*, 43(3): 90–93, Mar. 2010. ISSN 0018-9162.

W. Abramowicz, A. Filipowska, M. Kaczmarek, and T. Kaczmarek. Semantically enhanced Business Process Modeling Notation. In M. Hepp, K. Hinkelmann, D. Karagiannis, R. Klein, and N. Stojanovic, editors, *Semantic Business Process and Product Lifecycle Management (SBPM 2007)*, volume 251 of *CEUR-WS*, 2007.

A. Abran, P. Bourque, R. Dupuis, J. W. Moore, and L. L. Tripp. *Guide to the Software Engineering Body of Knowledge - SWEBOK*. IEEE Press, Piscataway, NJ, USA, 2004 version edition, 2004. ISBN 0769510000.

B. Achzet, A. Reller, V. Zepf, C. Rennie, and M. Simmons. Materials critical to the energy industry. An introduction. ON Communication, 2011.

S. Alexakis, M. Bauer, A. Pace, A. Schumacher, A. Friesen, A. Bouras, and D. Kourtesis. Application Of The Fusion Approach For Assisted Composition Of Web Services. In L. Camarinha-Matos, H. Afsarmanesh, P. Novais, and C. Analide, editors, *Establishing The Foundation Of Collaborative Networks*, IFIP International Federation for Information Processing, pages 531 – 538. Springer Boston, 2007.

D. Allemang and J. Hendler. *Semantic Web for the Working Ontologist: Modeling in RDF, RDFS and OWL*. Safari Books Online. Morgan Kaufmann Publishers/Elsevier, 2008. ISBN 9780123735560.

A. Awad, A. Polyvyanyy, and M. Weske. Semantic Querying of Business Process Models. In *Proceedings of the 2008 12th International IEEE Enterprise Distributed Object Computing Conference*, pages 85 – 94, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3373-5.

A. Awad, A. Grosskopf, A. Meyer, and M. Weske. Enabling Resource Assignment Constraints in BPMN. Technical report, Hasso Plattner Institute, Potsdam, Germany, 2009.

A. M. H. A. Awad. *A compliance management framework for business process models*. PhD thesis, Hasso Plattner Institute, University of Potsdam, 2010.

F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation And Applications*. Cambridge University Press, New York, NY, USA, 2003. ISBN 0-521-78176-0.

F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, New York, NY, USA, 2nd edition, 2010. ISBN 0521150116, 9780521150118.

M. A. Babar and I. Gorton. Comparison of Scenario-Based Software Architecture Evaluation Methods. In *Proceedings of the 11th Asia-Pacific Software Engineering Conference*, APSEC '04, pages 600–607, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2245-9.

W. Bandara, S. Chong, M. Indulska, M. Rosemann, and S. Sadiq. Major Issues In Business Process Management: An Australian Perspective. In S. Spencer and A. Jenkins, editors, *Proceedings of the 17th Australasian Conference on Information Systems*, pages 1 – 10, Australia, South Australia, Adelaide, 2006. Australasian Association for Information Systems.

S. C. Bandinelli, A. Fuggetta, and C. Ghezzi. Software Process Model Evolution in the SPADE Environment. *IEEE Transactions on Software Engineering*, 19(12):1128 – 1144, Dec. 1993. ISSN 0098-5589.

B. Bauer, T. Eisenbarth, C. Frenzel, and B. Honke. Resource-oriented consistency analysis of engineering processes. In *Proceedings of the 14th International Conference on Enterprise Information Systems (ICEIS)*, 2012.

D. Bauer, D. Diamond, J. Li, D. Sandalow, P. Telleen, and B. Wanner. Critical materials strategy. *U.S. Department of Energy*, December 2010. URL `http://energy.gov/sites/prod/files/edg/news/documents/criticalmaterialsstrategy.pdf`. (accessed April 16, 2012).

J. Becker, M. Kugeler, and M. Rosemann. *Process Management: A Guide for the Design of Business Processes*. Springer, 2003.

D. Beckett and B. McBride. RDF/XML syntax specification (revised). W3C Recommendation, February 2004. URL `http://www.w3.org/TR/rdf-syntax-grammar/`. (accessed November 27, 2011).

T. Berners-Lee. Linked data - design issues, 2006. URL `http://www.w3.org/DesignIssues/LinkedData.html`. (accessed March 18, 2011).

T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34 – 43, May 2001. ISSN 0036-8733.

S. Bhattacharjee, J. Cruz, and H. Singh. Closed loop supply chains: A systems dynamics model for analyzing sustainable business policies for shared partners in a chain. In *International Workshop on Supply Chain Models For Shared Resource Managemen*, 2010.

C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee. Linked data on the web. In *Proceeding of the 17th international conference on World Wide Web*, pages 1265 – 1266. ACM, 2008.

C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):2 – 2, 2009.

C. Bizer, A. Jentzsch, and R. Cyganiak. State of the LOD Cloud, 09 2011. URL `http://www4.wiwiss.fu-berlin.de/lodcloud/state`. (accessed September 25, 2012).

M. Bolsinger, M.-A. Bewernik, and H. U. Buhl. Value-based process improvement. In V. K. Tuunainen, M. Rossi, and J. Nandhakumar, editors, *ECIS*, 2011.

B. Bonet and H. Geffner. GPT: A Tool for Planning with Uncertainty and Partial Information. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 82 – 87, 2001.

A. Bouras, P. Gouvas, and G. Mentzas. ENIO: An Enterprise Application Integration Ontology. In *Database and Expert Systems Applications, 2007. DEXA '07. 18th International Workshop on*, pages 419 – 423, 09 2007.

B. P. M. I. BPMI. Business Process Modeling Notation (BPMN) Version 1.0, May 2004. URL `http://www.omg.org/bpmn/Documents/BPMN_V1-0_May_3_2004.pdf`. (accessed September 29, 2012).

K. S. Braunwarth, M. Kaiser, and A.-L. Müller. Economic evaluation and optimization of the degree of automation in insurance processes. *Business & Information Systems Engineering*, 2(1):29–39, 2010.

D. Brickley and R. Guha. RDF vocabulary description language 1.0: RDF Schema. W3C Recommendation, February 2004. URL `http://www.w3.org/TR/rdf-schema/`. (accessed October 10, 2011).

S. Brockmans, M. Ehrig, A. Koschmider, A. Oberweis, and R. Studer. Semantic alignment of business processes. In *Proceedings of the 8th International Conference on Enterprise Information Systems (ICEIS 2006)*, pages 191 – 196, Paphos, Cyprus, 2006. INSTICC Press.

J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In I. Horrocks and J. Hendler, editors, *The Semantic Web - ISWC 2002*, volume 2342 of *Lecture Notes in Computer Science*, pages 54 – 68. Springer Berlin / Heidelberg, 2002. ISBN 978-3-540-43760-4.

A. Bögl, M. Schrefl, G. Pomberger, and N. Weber. Semantic Annotation of EPC Models in Engineering Domains to Facilitate an Automated Identification of Common Modelling Practices. In J. Filipe and J. Cordeiro, editors, *Enterprise Information Systems*, volume 19 of *Lecture Notes in Business Information Processing*, pages 155–171. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-00669-2.

C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Towards the definition and analysis of resource assignments in BPMN 2.0. *9th International Conference on Business Process Management BPM11*, 2011.

C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. Towards A Semantic Approach for the Definition and Automated Design-Time Analysis of Resource Assignments in Business Processes. Technical report, Department Computer Languages and Systems, University of Sevilla, Apr 2012a.

C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. RAL: A high-level user-oriented resource assignment language for business processes. In F. Daniel, K. Barkaoui, and S. Dustdar, editors, *Business Process Management Workshops*, volume 99 of *Lecture Notes in Business Information Processing*, pages 50 – 61. Springer Berlin Heidelberg, 2012b. ISBN 978-3-642-28107-5.

L. Churliov, D. Neiger, M. Rosemann, and M. Z. Muehlen. Integrating Risks in Business Process Models with Value focused Process Engineering. In

J. Ljungberg and M. Andersson, editors, *Proceedings of the 14th European Conference on Information Systems*, pages 1 – 10, Sweden, Goteborg, 2006. IT University of Goteborg.

P. Clements, R. Kazman, and M. Klein. *Evaluating software architectures: methods and case studies*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002. ISBN 0-201-70482-X.

N. R. C. Committee on Critical Mineral Impacts of the U.S. Economy, Committee on Earth Resources. *Minerals, Critical Minerals, and the U.S. Economy*. The National Academies Press, 2008. ISBN 9780309112826.

R. Cooper and R. S. Kaplan. Measure Costs Right: Make the Right Decisions. *Harvard Business Review*, 66(5):96 – 103, 1988.

T. H. Davenport and J. E. Short. The New Industrial Engineering: Information Technology and Business Process Redesign. *Sloan Management Review*, 31 (4):11 – 27, 1990.

R. Davis and E. Brabnder. *ARIS Design Platform: Getting Started with BPM*. Springer Publishing Company, Incorporated, 1st edition, 2007. ISBN 1846286123.

R. I. Davis. *ARIS Design Platform: Advanced Process Modelling and Administration*. Springer, 2008. ISBN 978-1-8480-0110-7.

T. Debevoise, R. Geneva, and R. Welke. *The Microguide to Process Modeling in BPMN 2.0*. CreateSpace, 2011. ISBN 9781463511357.

G. Decker, H. Overdick, and M. Weske. Oryx – Sharing Conceptual Models on the Web. In Q. Li, S. Spaccapietra, E. Yu, and A. Olivé, editors, *Conceptual Modeling - ER 2008*, volume 5231 of *Lecture Notes in Computer Science*, pages 536 – 537. Springer Berlin / Heidelberg, 2008a. ISBN 978-3-540-87876-6.

G. Decker, H. Overdick, and M. Weske. Oryx – An Open Modeling Platform for the BPM Community. In M. Dumas, M. Reichert, and M.-C. Shan, editors, *Business Process Management*, volume 5240 of *Lecture Notes in Computer Science*, pages 382 – 385. Springer Berlin / Heidelberg, 2008b. ISBN 978-3-540-85757-0.

R. Dijkman, M. Dumas, B. van Dongen, R. Käärik, and J. Mendling. Similarity of Business Process Models: Metrics and evaluation. *Information Systems*, 36(2):498 – 516, 2011. ISSN 0306-4379. Special Issue: Semantic Integration of Data, Multimedia, and Services.

M. Dumas, M. Rosa, J. Mendling, and H. Reijers. *Fundamentals of Business Process Management*. Springer London, Limited, 2013. ISBN 9783642331428.

Eclipse. Eclipse Process Framework Project (EPF). Available at http://www.eclipse.org/epf., Feb. 2008. URL `http://www.eclipse.org/epf`. (accessed September, 13 2012).

Eclipse. Eclipse SOA Java Workflow Tooling project (JWT). Available at http://www.eclipse.org/jwt., Feb. 2010. URL `http://www.eclipse.org/jwt`. (accessed September, 11 2010).

M. Ehrig, A. Koschmider, and A. Oberweis. Measuring similarity between semantic business process models. In *Fourth Asia-Pacific Conference on Conceptual Modeling (APCCM 2007), Ballarat, Victoria, Australia*, January 2007.

T. Eisenbarth, F. Lautenbacher, and B. Bauer. Adaptation of Process Models - a Semantic-based Approach. *Journal of Research and Practice in Information Technology*, 43:5 – 23, 2011.

M. El Kharbili and S. Stein. Policy-Based Semantic Compliance Checking for Business Process Management. In P. Loos, M. Nüttgens, K. Turowski, and D. Werth, editors, *MobIS Workshops*, volume 420 of *CEUR Workshop Proceedings*, pages 178 – 192. CEUR-WS.org, 2008.

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 17 2008a.

M. El Kharbili, S. Stein, I. Markovic, and E. Pulvermüller. Towards Policy-Powered Semantic Enterprise Compliance Management. In *In 3rd International Workshop on Semantic Business Process Management*, pages 16 – 21, 2008b.

M. El Kharbili, S. Stein, and E. Pulvermüller. Service Contract Compliance Management. In *Proceedings of the 3rd Workshop on Emerging Web Services Technology (WEWST 2008)*, Dublin, Ireland, November 2008c.

L. Erdmann and T. E. Graedel. Criticality of Non-Fuel Minerals: A Review of Major Approaches and Analyses. *Environmental science technology*, 45(18): 7620 – 7630, 2011.

European Commission. Report of the ad-hoc working group on defining critical raw materials, June 2010. URL `http://ec.europa.eu/enterprise/policies/raw-materials/files/docs/report_en.pdf`. (accessed June 14 2012).

European Commission, Information Society. EU projects on eHealth interoperability, 2012. URL `http://ec.europa.eu/information_society/activities/health/policy/interoperability_and_standardisation/iop_projects/index_en.htm`. (access September 14, 2012).

G. Eysenbach. What is e-health? *Journal of Medical Internet Research*, 3(2):2 – 20, Jun 2001.

F. G. Fadel, M. S. Fox, and M. Grüninger. A Resource Ontology for Enterprise Modelling. *Proceedings of the Third Industrial Engineering Research Conference*, 1994a.

F. G. Fadel, M. S. Fox, and M. Grüninger. A Generic Enterprise Resource Ontology. *Proceedings of the Third Workshop on Enabling Technologies - Infrastructures for Collaborative Enterprises*, 1(April):117 – 128, 1994b.

N. Feiertag, A. Hamann, D. Karlsson, J. Kemmerich, S. Kuntz, H. Lönn, E. Löschner, J. Neumüller, N. Salem, and M. Schlager. Methodology version 2. Timmo (ITEA 2 project 06005), Deliverable D7, August 2009. URL `http://timmo-2-use.org/timmo/pdf/D7_TIMMO_Methodology_Version_2_v10.pdf`. (accessed September 13, 2012).

M. Fellmann and O. Thomas. Process Model Verification with SemQuu. In M. Nüttgens, O. Thomas, and B. Weber, editors, *EMISA*, volume 190 of *LNI*, pages 231–236. GI, 2011. ISBN 978-3-88579-284-0.

M. Fellmann, O. Thomas, and B. Busch. A Query-Driven Approach for Checking the Semantic Correctness of Ontology-Based Process Representations. In W. Abramowicz, editor, *Business Information Systems*, volume 87 of *Lecture Notes in Business Information Processing*, pages 62–73. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-21829-3.

M. Feygin and R. Satkin. The Oil Reserves-to-Production Ratio and Its Proper Interpretation. *Natural Resources Research*, 13:57 – 60, 2004. ISSN 1520-7439. 10.1023/B:NARR.0000023308.84994.7f.

A. Filipowska, M. Hepp, M. Kaczmarek, and I. Markovic. Organisational Ontology Framework for Semantic Business Process Management. In *BIS*, pages 1 – 12, 2009a.

A. Filipowska, M. Kaczmarek, and S. Stein. Semantically Annotated EPC within Semantic Business Process Management. In D. Ardagna, M. Mecella, J. Yang, W. Aalst, J. Mylopoulos, M. Rosemann, M. J. Shaw, and C. Szyperski, editors, *Business Process Management Workshops*, volume 17 of *Lecture Notes in Business Information Processing*, pages 486–497. Springer Berlin Heidelberg, 2009b. ISBN 978-3-642-00328-8.

G. Fliedl, C. Kop, and J. Vöhringer. From OWL Class and Property Labels to Human Understandable Natural Language. In Z. Kedad, N. Lammari, E. Métais, F. Meziane, and Y. Rezgui, editors, *Natural Language Processing and Information Systems*, volume 4592 of *Lecture Notes in Computer Science*, pages 156–167. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-73350-8.

A. Fortis and F. Fortis. Workflow patterns in process modeling. *Ann. Univ. Tibiscus Comp. Sci. Series*, 6:81 – 94, 2009.

M. S. Fox. The tove project towards a common-sense model of the enterprise. In *IEA/AIE '92: Proceedings of the 5th international conference on Industrial and engineering applications of artificial intelligence and expert systems*, pages 25 – 34, London, UK, 1992. Springer-Verlag. ISBN 3-540-55601-X.

F. Gailly, W. Laurier, and G. Poels. Positioning and formalizing the REA enterprise ontology. *Journal of Information Systems*, 22(2):219 – 248, 2008.

D. Gasevic. Petri Nets on the Semantic Web - Guidelines and Infrastructure. *International Journal on Computer Science and Information Systems*, 1(2):127–151, Nov. 2004. InternalNote: Submitted by: gasevic@yahoo.com.

J. Gebauer and F. Schober. Information System Flexibility and the Cost Efficiency of Business Processes. *Journal of the Association for Information Systems*, 3:122 – 147, 2006.

G. L. Geerts and W. E. McCarthy. The Ontological Foundation of REA Enterprise Information Systems. Technical report, Michigan State University, 2000.

D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. In *DISTRIBUTED AND PARALLEL DATABASES*, pages 119 – 153, 1995.

M. Ghallab, D. Nau, and P. Traverso. *Automated Planning*. Elsevier, San Francisco, 2004.

F. Gottschalk and M. L. Rosa. Process configuration in YAWL. In A. H. ter Hofstede, W. M. van der Aalst, M. Adams, and N. Russell, editors, *Modern Business Process Automation : YAWL and its Support Environment*, pages 313–382. Springer, London, 2010.

M. Götz, S. Roser, F. Lautenbacher, and B. Bauer. Token Analysis of Graph-Oriented Process Models. In *Proceedings of DDBP 2009, Auckland, New Zealand*, September 2009.

B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. OWL2: The next step for OWL. *Journal on Web Semantics*, 6(4):309 – 322, November 2008.

T. R. Gruber. Ontolingua: A Mechanism to Support Portable Ontologies. Technical report, University of Massachusetts, 1992.

A. Hallerbach, T. Bauer, and M. Reichert. Context-based Configuration of Process Variants. In *3rd International Workshop on Technologies for Context-Aware Business Process Management (TCoB 2008)*, pages 31–40, June 2008.

O. Hanseth, E. Monteiro, and M. Hatling. Developing Information Infrastructure: The Tension between Standardization and Flexibility. *Science, Technology and Human Values*, 11(4):407 – 426, 1996.

P. Harmon and C. Wolf. *The State of Business Process Management - 2008*. Business Process Trends, 2008.

P. Harmon and C. Wolf. *The State of Business Process Management - 2010*. Business Process Trends, 2010.

J. Hauser. The german national bibliography as linked open data, 01 2012. URL `http://lists.w3.org/Archives/Public/public-lld/2012Jan/0004.html`. (accessed September 25, 2012).

R. Haux, A. Winter, E. Ammenwerth, and B. Brigl. *Strategic Information Management in Hospitals: An Introduction to Hospital Information Systems*. Health Informatics. Springer, 2004. ISBN 9780387403564.

T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web. Morgan & Claypool Publishers, 2011.

J. Heflin. OWL Web Ontology Language Use Cases and Requirements, 02 2004. URL `http://www.w3.org/TR/webont-req/`. (accessed August 4, 2012).

B. Heinrich, M. Henneberger, A. Krammer, M. Bewernik, and F. Lautenbacher. SEMPA – Ein Ansatz des Semantischen Prozessmanagements zur Planung von Prozessmodellen. *Wirtschaftsinformatik*, November/December, 2008.

B. Henderson-Sellers and C. Gonzalez-Perez. *Comparison of Method Chunks and Method Fragments for Situational Method Engineering*, pages 479 – 488. IEEE Computer Society, 2008.

M. Hepp and D. Roman. An Ontology Framework for Semantic Business Process Management. *Proceedings of Wirtschaftsinformatik*, 2007, 2007.

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 *Proceedings of the IEEE International Conference on e-Business Engineering*, pages 535 – 540, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2430-3.

A. O. Hirschman. The paternity of an index. *American Economic Review*, 54(5): 761 – 762, 1964.

P. Hitzler, M. Krötzsch, S. Rudolph, and Y. Sure. *Semantic Web: Grundlagen (eXamen.press)*. Springer, 1 edition, Oct. 2007. ISBN 3540339930.

P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 2009a. URL `http://www.w3.org/TR/owl2-primer/`. (accessed May 28, 2012).

P. Hitzler, M. Krötzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC Textbooks in Computing. CRC Press, 2009b. ISBN 9781420090505.

H. Hoang, P.-C. Tran, and T. Le. State of the Art of Semantic Business Process Management: An Investigation on Approaches for Business-to-Business Integration. In N. Nguyen, M. Le, and J. Swiatek, editors, *Intelligent Information and Database Systems*, volume 5991 of *Lecture Notes in Computer Science*, pages 154 – 165. Springer Berlin / Heidelberg, 2010.

M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL ontologies. *Semantic Web*, 2(1):11 – 21, 2011.

M. Horridge, N. Drummond, J. Goodwin, A. L. Rector, R. Stevens, and H. Wang. The Manchester OWL Syntax. In *OWLED*, 2006.

I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3c member submission, World Wide Web Consortium, 2004. URL `http://www.w3.org/Submission/SWRL`. (accessed May 28, 2012).

G. Hübner-Bloder, E. Ammenwerth, B. Brigl, and A. Winter. Specification of a Reference Model for the Domain Layer of a Hospital Information System. *Stud Health Technol Inform*, 116:497 – 502, 2005.

N. Josuttis. *SOA in practice*. O'Reilly, first edition, 2007. ISBN 9780596529550.

V. Kanevsky and T. K. Housel. Value-Based Business Process Reengineering: An Objective Approach to Value Added. *International Business and Management Forum*, 6:2 – 8, 1994. ISSN 09158235.

R. Kazman, G. Abowd, L. Bass, and P. Clements. Scenario-Based Analysis of Software Architecture. *IEEE Softw.*, 13(6):47–55, 1996. ISSN 0740-7459.

S. Kent. Model driven engineering. In M. Butler, L. Petre, and K. Sere, editors, *Integrated Formal Methods*, volume 2335 of *Lecture Notes in Computer Science*, pages 286 – 298. Springer Berlin / Heidelberg, 2002. ISBN 978-3-540-43703-1.

A. Kiryakov, D. Ognyanov, and D. Manov. OWLIM – a pragmatic semantic repository for OWL. In M. Dean, Y. Guo, W. Jun, R. Kaschek, S. Krishnaswamy, Z. Pan, and Q. Sheng, editors, *Web Information Systems Engineering – WISE 2005 Workshops*, volume 3807 of *Lecture Notes in Computer Science*, pages 182 – 192. Springer Berlin / Heidelberg, 2005. ISBN 978-3-540-30018-2.

A. G. Kleppe, J. Warmer, and W. Bast. *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003. ISBN 032119442X.

H. Knublauch, M. A. Musen, and A. L. Rector. Editing description logic ontologies with the Protégé OWL plugin. In *Proc. of International Conf. on Description Logics*. CEUR-WS.org, 2004.

D. Kourtesis and I. Paraskakis. Web Service Discovery in the FUSION Semantic Registry. In W. Aalst, J. Mylopoulos, N. M. Sadeh, M. J. Shaw, C. Szyperski, W. Abramowicz, and D. Fensel, editors, *Business Information Systems*, volume 7 of *Lecture Notes in Business Information Processing*, pages 285 – 296. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-79396-0.

D. Krafzig, K. Banke, and D. Slama. *Enterprise SOA: Service-Oriented Architecture Best Practices*. The Coad series. Prentice Hall Professional Technical Reference, 2005. ISBN 9780131465756.

S. Kunz, F. Brecht, B. Fabian, M. Aleksy, and M. Wauer. Aletheia–Improving Industrial Service Lifecycle Management by Semantic Data Federations. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 1308 – 1314, april 2010.

D. Kuropka. *Modelle zur Repräsentation natürlichsprachlicher Dokumente: Ontologie-basiertes Information-Filtering-und-Retrieval mit relationalen Datenbanken*. Advances in information systems and management science. Logos-Verl., 2004. ISBN 9783832505141.

M. La Rosa, M. Dumas, A. ter Hofstede, J. Mendling, and F. Gottschalk. Beyond Control-Flow: Extending Business Process Configuration to Roles and Objects. In Q. Li, S. Spaccapietra, E. Yu, and A. Olivé, editors, *Conceptual Modeling - ER 2008*, volume 5231 of *Lecture Notes in Computer Science*, pages 199–215. Springer Berlin / Heidelberg, 2008a. ISBN 978-3-540-87876-6.

M. La Rosa, F. Gottschalk, M. Dumas, and W. M. P. Van Der Aalst. Linking Domain Models and Process Models for Reference Model Configuration. In *Proceedings of the 2007 international conference on Business process management*, BPM'07, pages 417–430, Berlin, Heidelberg, 2008b. Springer-Verlag. ISBN 3-540-78237-0, 978-3-540-78237-7.

F. Lautenbacher. *Semantic Business Process Modeling - Principles, Design Support and Realization*. PhD thesis, University of Augsburg, 2010.

F. Lautenbacher, T. Eisenbarth, and B. Bauer. Process Model Adaptation using Semantic Technologies. In *Vocabularies, Ontologies and Rules for the Enterprise (VORTE), Auckland, New Zealand*. IEEE, September 2009.

Y. Lin, D. Strasunskas, S. Hakkarainen, J. Krogstie, and A. Solvberg. Semantic Annotation Framework to Manage Semantic Heterogeneity of Process Models. In *Proceedings of the 18th international conference on Advanced Information Systems Engineering*, CAiSE'06, pages 433–446, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-34652-X, 978-3-540-34652-4.

I. Markovic. Advanced Querying and Reasoning on Business Process Models. In W. Abramowicz, D. Fensel, W. Aalst, J. Mylopoulos, M. Rosemann, M. J. Shaw, and C. Szyperski, editors, *Business Information Systems*, volume 7 of

Lecture Notes in Business Information Processing, pages 189 – 200. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-79396-0.

I. Markovic. *Semantic Business Process Modeling*. PhD thesis, Karlsruhe Institut für Technologie (KIT), 2009.

E. A. Marks and M. Bell. *Service-Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology*. John Wiley & Sons, June 2006. ISBN 0471768944.

D. H. Meadows. *The Limits to growth: a report for the Club of Rome's project on the predicament of mankind*. Universe Books, New York, 1972. ISBN 0876631650 0856440086.

Metal-Pages. http://www.metal-pages.com/, 04 2012. URL `http://www.metal-pages.com/`. (accessed April 18, 2012).

R. Müller and A. Rogge-Solti. BPMN for Healthcare Processes. In D. Eichhorn, A. Koschmider, and H. Zhang, editors, *Proceedings of the 3rd Central-European Workshop on Services and their Composition, ZEUS 2011, Karlsruhe, Germany, February 21–22, 2011*, volume 705 of *CEUR Workshop Proceedings*, pages 65–72. CEUR-WS.org, 2011.

M. Mühlen and R. Shapiro. Business Process Analytics. In J. vom Brocke and M. Rosemann, editors, *Handbook on Business Process Management 2*, International Handbooks on Information Systems, pages 137–157. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-01982-1.

E. Neumayer. Scarce or Abundant? The Economics of Natural Resource Availability. *Journal of Economic Surveys*, 14(3):307 – 335, 2000. ISSN 1467-6419.

Object Management Group. Software & systems process engineering metamodel specification. *OMG document*, 2008. URL `http://www.omg.org/spec/SPEM/2.0`. (accessed September 13, 2012).

H. Oh, C. Rizo, M. Enkin, and A. Jadad. What is ehealth (3): A systematic review of published definitions. *Journal of Medical Internet Research*, 7(1):e1, Feb 2005.

OMG. MDA Guide Version 1.0.1, June 2003. URL `http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf`. (accessed September 21, 2012).

OMG. Business process modeling notation, v1.1. OMG Specification, January 2008. URL http://www.omg.org/spec/BPMN/1.1/PDF/. (accessed September 29, 2012).

OMG. Unified modeling language (UML) superstructure, version 2.2. OMG Specification, February 2009a. URL http://www.omg.org/spec/UML/2.2/Superstructure/PDF/. (accessed February 27, 2010).

OMG. Business Process Modeling Notation, V1.2. OMG Specification, January 2009b. URL http://www.omg.org/spec/BPMN/1.2/. (accessed May 26, 2010).

OMG. Business process model and notation (BPMN) version 2.0. OMG Specification, January 2011. URL http://www.omg.org/spec/BPMN/2.0/. (accessed March 2nd, 2012).

OMG. Business Process Modeling Notation (BPMN) Information Frequently Asked Questions, 03 2012. URL http://www.omg.org/bpmn/Documents/FAQ.htm. (accessed March 23, 2012).

Oracle Corp. State of the business process management market 2008, 08 2008.

C. Ouyang, M. Dumas, and A. H. M. ter Hofstede. From BPMN Process Models to BPEL Web Services. In *In Proceedings of the 4th International Conference on Web Services*, pages 285 – 292. IEEE Computer Society, 2006.

C. Ouyang, M. Dumas, A. H. M. ter Hofstede, and W. M. van der Aalst. Pattern-based translation of BPMN process models to BPEL web services. *International Journal of Web Services Research (JWSR)*, 5(1):42 – 62, 2007.

M. P. Papazoglou. Service-oriented computing: Concepts, characteristics and directions. In *WISE*, pages 3 – 12, 2003.

M. P. Papazoglou. *Web Services - Principles and Technology*. Prentice Hall, 2008. ISBN 978-0-321-15555-9.

P. Pfleger, K. Lichtblau, H. Bardt, and A. Reller. *Rohstoffsituation Bayern - Keine Zukunft ohne Rohstoffe*. VBW (Vereinigung der Bayerischen Wirtschaft), 2009.

H. Polinder, F. F. A. van der Pijl, G.-J. de Vilder, and P. J. Tavner. Comparison of direct-drive and geared generator concepts for wind turbines. *IEEE Transactions on Energy Conversion*, 21(3):725 – 733, August 2006.

A. Reller, T. Bublies, T. Staudinger, I. Oswald, S. Meisharpner, and M. Allen. The mobile phone: Powerful communicator and potential metal dissipator. *GAIA - Ecological Perspectives for Science and Society*, 18(2):127 – 135, 2009. ISSN 0940-5550.

M. L. Rosa. *Managing variability in process-aware information systems*. PhD thesis, Queensland University of Technology, 2009.

M. Rosemann and W. van der Aalst. A configurable reference modelling language. *Information Systems*, 32(1):1 – 23, 2007. ISSN 0306-4379.

M. Rosemann and M. zur Muehlen. Integrating Risks in Business Process Models. In B. Campbell, H. Underwood, and D. Bunker, editors, *Proceedings of the 16th Australasian Conference on Information Systems (ACIS 2005)*, pages 1 – 10, Australia, New South Wales, Sydney, 2005. Australasian Chapter of the Association for Information Systems.

D. Rosenau-Tornow, P. Buchholz, A. Riemann, and M. Wagner. Assessing the long-term supply risks for mineral raw materials—a combined evaluation of past and future trends. *Resources Policy*, 34(4):161 – 175, 2009.

N. Russell, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst. Workflow data patterns: Identification, representation and tool support. In *IN 'PROCEEDINGS OF THE 25TH INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING (ER'2005*. Springer, 2005a.

N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and D. Edmond. *Workflow Resource Patterns: Identification, Representation and Tool Support*, volume 3520 of *Lecture Notes in Computer Science*, chapter 16, pages 216 – 232. Springer Berlin, Heidelberg, 2005b. ISBN 978-3-540-26095-0.

N. Russell, Arthur, W. M. P. van der Aalst, and N. Mulyar. Workflow control-flow patterns: A revised view. Technical report, BPMcenter.org, 2006.

S. Schulz, B. Suntisrivaraporn, and F. Baader. SNOMED CT's problem list: ontologists' and logicians' therapy suggestions. *Studies in health technology and informatics*, 129(1):80 – 2, 2007.

R. Shearer, B. Motik, and I. Horrocks. HermiT: A Highly-Efficient OWL Reasoner. In A. Ruttenberg, U. Sattler, and C. Dolbear, editors, *Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 EU)*, Karlsruhe, Germany, October 26–27 2008.

B. Silver. *BPMN Method and Style, Second Edition, with BPMN Implementer's Guide*. Cody-Cassidy Press, second edition, Jan. 2012.

J. L. Simon. Resources, Population, Environment: An Oversupply of False Bad News. *Science*, 208(4451):1431 – 1437, 1980.

E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51 – 53, 2007. ISSN 1570-8268.

K. A. Spackman, K. E. Campbell, , and R. A. Côté. SNOMED RT: A reference terminology for health care. In *Journal of the American Medical Informatics Association*, pages 640 – 644, 1997.

S. Stein, J. Lauer, and K. Ivanov. ARIS Method Extension for Business-Driven SOA. *Wirtschaftsinformatik*, 50:436 – 444, 2008. ISSN 0937-6429. 10.1365/s11576-008-0090-5.

S. Stein, C. Stamber, and M. Kharbili. ARIS for Semantic Business Process Management. In W. Aalst, J. Mylopoulos, N. M. Sadeh, M. J. Shaw, C. Szyperski, D. Ardagna, M. Mecella, and J. Yang, editors, *Business Process Management Workshops*, volume 17 of *Lecture Notes in Business Information Processing*, pages 498 – 509. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-00328-8.

SUPER project. List of deliverables, April 2011. URL `http://www.ip-super.org/content/view/32/66/`. (accessed June 8, 2012).

The ATESST2 Consortium. EAST-ADL Domain Model Specification. ATESST2, Deliverable D4.1.1, June 2010. URL `http://www.atesst.org/home/liblocal/docs/ATESST2_D4.1.1_EAST-ADL2-Specification_2010-06-02.pdf`. (accessed March 13, 2012).

L. H. Thom, C. Iochpe, and M. U. Reichert. *Workflow patterns for business process modeling*, volume 1, pages 349 – 358. Tapir Academic Press, 2007.

O. Thomas and M. Fellmann. Semantic EPC: Enhancing Process Modeling Using Ontology Languages. In M. Hepp, K. Hinkelmann, D. Karagiannis, R. Klein, and N. Stojanovic, editors, *SBPM*, volume 251 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

O. Thomas and M. Fellmann. Semantic Process Modeling – Design and Implementation of an Ontology-based Representation of Business Processes. *Business & Information Systems Engineering*, 1:438–451, 2009. ISSN 1867-0202.

J. Tilton. *On Borrowed Time?: Assessing the Threat of Mineral Depletion*. Resources for the Future Press Series. Resources for the Future, 2002. ISBN 9781891853579.

J. E. Tilton and G. Lagos. Assessing the long-run availability of copper. *Resources Policy*, 32(1-2):19 – 23, 2007.

D. Tsarkov and I. Horrocks. Description logic reasoner: System description. In U. Furbach and N. Shankar, editors, *Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*, pages 292 – 297. Springer Berlin / Heidelberg, 2006. ISBN 978-3-540-37187-8.

R. Uba, M. Dumas, L. Garcia-Banuelos, and M. La Rosa. Clone Detection in Repositories of Business Process Models. In S. Rinderle-Ma, F. Toumani, and K. Wolf, editors, *Business Process Management*, volume 6896 of *Lecture Notes in Computer Science*, pages 248 – 264. Springer Berlin / Heidelberg, 2011. ISBN 978-3-642-23058-5.

M. Uschold, M. Uschold, M. King, M. King, S. B. R. House, S. Moralee, S. Moralee, Y. Zorgios, and Y. Zorgios. The enterprise ontology. *The Knowledge Engineering Review*, 13:31 – 89, 1995.

M. Uschold, M. King, S. Moralee, and Y. Zorgios. The enterprise ontology. *The Knowledge Engineering Review*, 13:31 – 89, March 1998. ISSN 0269-8889.

W. M. P. van der Aalst. Challenges in Business Process Analysis. In J. Filipe, J. Cordeiro, J. Cardoso, W. Aalst, J. Mylopoulos, M. Rosemann, M. J. Shaw, and C. Szyperski, editors, *Enterprise Information Systems*, volume 12 of *Lecture Notes in Business Information Processing*, pages 27–42. Springer Berlin Heidelberg, 2009. ISBN 978-3-540-88710-2.

W. M. P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, Berlin, 2011. ISBN 978-3-642-19344-6.

W. M. P. van der Aalst and K. Bisgaard Lassen. Translating Unstructured Workflow Processes to Readable BPEL: Theory and Implementation. *Inf. Softw. Technol.*, 50(3):131 – 159, 2008. ISSN 0950-5849.

W. M. P. van der Aalst and A. H. M. ter Hofstede. YAWL: yet another workflow language. *Information Systems*, 30(4):245 – 275, 2005. ISSN 0306-4379.

W. M. P. van der Aalst, A. Barros, A. H. M. ter Hofstede, and B. Kiepuszewski. Advanced workflow patterns. In P. Scheuermann and O. Etzion, editors, *Cooperative Information Systems*, volume 1901 of *Lecture Notes in Computer Science*, pages 18 – 29. Springer Berlin / Heidelberg, 2000. ISBN 978-3-540-41021-8.

W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distrib. Parallel Databases*, 14(1):5 – 51, 2003a. ISSN 0926-8782.

W. M. P. van der Aalst, A. H. M. ter Hofstede, and M. Weske. Business process management: A survey. In *Business Process Management*, volume 2678 of *LNCS*, pages 1 – 12. Springer Verlag Berlin Heidelberg, 2003b.

B. van Dongen, R. Dijkman, and J. Mendling. Measuring Similarity between Business Process Models. In Z. Bellahsène and M. Léonard, editors, *Advanced Information Systems Engineering*, volume 5074 of *Lecture Notes in Computer Science*, pages 450 – 464. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-69533-2.

F. van Harmelen, F. van Harmelen, V. Lifschitz, and B. Porter. *Handbook of Knowledge Representation (Foundations of Artificial Intelligence)*. Elsevier Science, San Diego, USA, 1 edition, 2007. ISBN 0444522115, 9780444522115.

J. Vanhatalo, H. Völzer, and F. Leymann. Faster and More Focused Control-Flow Analysis for Business Process Models Through SESE Decomposition. In *ICSOC*, 2007.

K. Vergidis, A. Tiwari, and B. Majeed. Business process analysis and optimization: Beyond reengineering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 38(1):69–82, 2008.

J. vom Brocke and M. Rosemann. *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*. Springer Publishing Company, Incorporated, 1st edition, 2010. ISBN 3642004156, 9783642004155.

J. vom Brocke, J. Recker, and J. Mendling. Value-oriented Process Modeling: Integrating Financial Perspectives into Business Process Re-design. *Business Process Management Journal*, 16(2):333 – 356, 2010.

I. Weber, G. Governatori, and J. Hoffmann. Approximate compliance checking for annotated process models. In *Proceedings of the 1st International Workshop on Governance Risk and Compliance—Applications in Information Systems GRCIS'08*, volume 339, pages 46 – 60, 2008.

M. Weidlich, G. Decker, A. Großkopf, and M. Weske. BPEL to BPMN: The myth of a straight-forward mapping. In R. Meersman and Z. Tari, editors, *On the Move to Meaningful Internet Systems: OTM 2008*, volume 5331 of *Lecture Notes in Computer Science*, pages 265 – 282. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-88870-3.

M. Weske. *Business Process Management - Concepts, Languages, Architectures*. Springer Verlag, Berlin Heidelberg, 2007.

B. Wetzstein, Z. Ma, A. Filipowska, M. Kaczmarek, S. Bhiri, S. Losada, J. manuel Lopez-cob, and L. Cicurel. Semantic Business Process Management: A Lifecycle Based Requirements Analysis. In *European Semantic Web Symposium / Conference*, 2007.

S. A. White. Process Modeling Notations and Workflow Patterns. *Business*, March(1999):1 – 25, 2004a.

S. A. White. Workflow Patterns with BPMN and UML. *IBM January*, 2004b.

A. Winter, R. Haux, E. Ammenwerth, B. Brigl, N. Hellrung, and F. Jahn. *Health Information Systems: Architectures and Strategies*. Health Informatics. Springer, 2010. ISBN 9781849964401.

P. Wohed, W. M. P. van Der Aalst, M. Dumas, A. H. M. ter Hofstede, and N. Russell. Pattern-based Analysis of BPMN – an extensive evaluation of the Control-flow, the Data and the Resource Perspectives. *BPM Center Report*, 06-17, 2006a.

P. Wohed, W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, and N. Russell. On the suitability of BPMN for business process modelling. In S. Dustdar, J. Fiadeiro, and A. Sheth, editors, *Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, pages 161 – 176. Springer Berlin / Heidelberg, 2006b. ISBN 978-3-540-38901-9.

E. U. Working Group of the Raw Materials Supply Group. Critical raw materials for the EU. Report of the Ad-hoc Working Group on defining critical raw materials, 2010.

E. Zolin. Complexity of reasoning in description logics, 11 2011. URL http://www.cs.man.ac.uk/~ezolin/dl/. (accessed August, 4 2012).

M. zur Muehlen and J. C. Recker. How much language is enough? Theoretical and practical use of the Business Process Modeling Notation. In *20th International Conference on Advanced Information Systems Engineering*, pages 465 – 479, Montpellier, France, 2008. Springerlink.

Acronyms

ALMA Architecture Level Modifiability Analysis

ATAM Architecture Tradeoff Analysis Method

ARID Active Reviews for Intermediate Designs

ABox Assertional box

ARIS Architecture of Integrated Information Systems

API Application programming interface

AD Activity Diagram

ADG Action Dependency Graph

ASG Action State Graph

ANSI American National Standards Institute

BPEL Business Process Execution Language

BPM Business Process Management

BPMN Business Modeling Notation

BPMN-Q BPMN Query

BORO Business Object Reference Ontology

BPMO Business Process Management Ontology

CEO Core Enterprise Ontology

CWA Closed World Assumption

CSV Comma-Separated Values

DARPA Defense Advanced Research Projects Agency

DAML DARPA Agent Markup Language

DFG Deutsche Forschungsgemeinschaft (Germany's funding organisation)

DL Description Logic

DSL Domain-specific language

eTVSM enhanced Topic-based Vector Space Model

ERP Enterprise Resource Planning

EMF Eclipse Modeling Framework

Ecore EMF meta model

EPC Event-driven Process Chain

EPF Eclipse Process Framework

EU European Union

EAM Enterprise Architecture Management

EA Enterprise Architecture

FEA Federal Enterprise Architecture

FOL First-order logic

HR Human Resources

HE Hardware Engineering

HHI Herfindahl–Hirschman Index

HTML Hypertext Markup Language

HIS Hospital information system

IT Information Technology

IS Information Systems

IEEE Institute of Electrical and Electronics Engineers

JWT Java Workflow Tooling

JSON JavaScript Object Notation

KIF Knowledge Interchange Format

KB Knowledge Base

LUBM Lehigh University Benchmark

MC Method Chunk

MDE Model Driven Engineering

MDA Model Driven Architecture

MDSD Model Driven Software Development

MRP Manufacturing Resource Planning

MMTS Metamodeling Technical Space

ME Method Engineering

OCL Object Constraint Language

OTS Ontological Technical Space

OMG Object Management Group

OWL Web Ontology Language

OWLAPI Java API for modifying OWL Ontologies

OWA Open World Assumption

OIL Ontology Interchange Language

PPS Predefined Property Set

PMod Process Model

PLTL Past Linear Temporal Logic

PASA Performance Assessment of Software Architecture

QVT Query / View / Transformation

RAL Resource Alignment Language

RDF Resource Description Framework

RDFs RDF Schema

RF Resource Fragment

RDF Resource Description Framework

RR Resource Reference

REE Rare Earth Element

REA Resources, events, agents

SAAM Software Architecture Analysis Method

SBPM Semantic Business Process Management

SCC Strongly Connected Component

SE Software Engineering

SEBIS Semantics in Business Information Systems

SEMPA Semantic Based Planning Approach

SEAM Semantic Enterprise Architecture Management

SESE Single-entry Single-exit

SEMPRO Semantic based Modeling, Selfcomposition, and Selfconfiguration of Reference Processes

SUPER Semantics Utilised for Process Management within and between Enterprises

SME Small and Medium-sized Enterprises

SWS Semantic Web Service

SOA Service Oriented Architecture

SNOMED CT Systematized Nomenclature of Medicine – Clinical Terms

STI Semantic Technology Institute

SPARQL SPARQL Protocol And RDF Query Language

SWRL Semantic Web Rule Language

SWEBOK Software Engineering Body of Knowledge

TS Technical Space

TBox Terminological box

TOVE Toronto Virtual Enterprise

TOGAF The Open Group Architecture Framework

UML Unified Modeling Language

URI Uniform Resource Identifier

WSMO Web Service Modeling Language

WSML Web Service Modeling Language

WSMO Web Service Modeling Ontology

W3C World Wide Web Consortium

XML Extensible Markup Language

XSL Extensible Stylesheet Language

XSLT XSL Transformation

XPDL XML Process Definition Language

YAWL Yet Another Workflow Language

List of figures

List of tables

Listings

XSL transformations

```
1  <?xml version="1.0"?>
2  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:metaBPMN2="http://bpmn2"
3                  xmlns:xsi="http://www.w3.org/2001/
                       XMLSchema-instance" version="1.0">
4    <xsl:template match="/">
5      <root>
6        <xsl:apply-templates select="//process" mode="lane"/>
7        <xsl:apply-templates select="//process" mode="task"/>
8        <xsl:apply-templates select="//process" mode="startEvent"/>
9        <xsl:apply-templates select="//process" mode="endEvent"/>
10       <xsl:apply-templates select="//process" mode="dataObject"/>
11       <xsl:apply-templates select="//process" mode="sequenceFlow"/
             >
12       <xsl:apply-templates select="//process" mode="
             dataInputAssociation"/>
13       <xsl:apply-templates select="//process" mode="
             dataOutputAssociation"/>
14       <xsl:apply-templates select="//process" mode="
             exclusiveGateway"/>
15       <xsl:apply-templates select="//process" mode="
             parallelGateway"/>
16     </root>
17   </xsl:template>
18   <xsl:template match="process" mode="startEvent">
19     <xsl:for-each select="./startEvent">
20       <xsl:element name="startEvent">
21         <xsl:attribute name="xsi:type">metaBPMN2:event</
               xsl:attribute>
22         <xsl:attribute name="processid">
23           <xsl:value-of select="./../@id"/>
24         </xsl:attribute>
25         <xsl:attribute name="id">
26           <xsl:value-of select="./@id"/>
27         </xsl:attribute>
28         <xsl:attribute name="name">
```

```
29            <xsl:value-of select="./@name"/>
30          </xsl:attribute>
31        </xsl:element>
32      </xsl:for-each>
33    </xsl:template>
34
35    <xsl:template match="process" mode="endEvent">
36      <xsl:for-each select="./endEvent">
37        <xsl:element name="endEvent">
38          <xsl:attribute name="xsi:type">metaBPMN2:event</
                xsl:attribute>
39          <xsl:attribute name="processid">
40            <xsl:value-of select="./../@id"/>
41          </xsl:attribute>
42          <xsl:attribute name="id">
43            <xsl:value-of select="./@id"/>
44          </xsl:attribute>
45          <xsl:attribute name="name">
46            <xsl:value-of select="./@name"/>
47          </xsl:attribute>
48        </xsl:element>
49      </xsl:for-each>
50    </xsl:template>
51
52    <xsl:template match="process" mode="dataObject">
53      <xsl:for-each select="./dataObject">
54        <xsl:element name="dataObject">
55          <xsl:attribute name="xsi:type">metaBPMN2:dataObject</
                xsl:attribute>
56          <xsl:attribute name="processid">
57            <xsl:value-of select="./../@id"/>
58          </xsl:attribute>
59          <xsl:attribute name="id">
60            <xsl:value-of select="./@id"/>
61          </xsl:attribute>
62          <xsl:attribute name="name">
63            <xsl:value-of select="./@name"/>
64          </xsl:attribute>
65        </xsl:element>
66      </xsl:for-each>
67    </xsl:template>
68
69    <xsl:template match="process" mode="task">
70      <xsl:for-each select="./task">
71        <xsl:element name="task">
72          <xsl:attribute name="xsi:type">metaBPMN2:task</
                xsl:attribute>
73          <xsl:attribute name="processid">
```

```
74          <xsl:value-of select="./../@id"/>
75        </xsl:attribute>
76        <xsl:attribute name="id">
77          <xsl:value-of select="./@id"/>
78        </xsl:attribute>
79        <xsl:attribute name="name">
80          <xsl:value-of select="./@name"/>
81        </xsl:attribute>
82      </xsl:element>
83    </xsl:for-each>
84  </xsl:template>
85
86  <xsl:template match="process" mode="parallelGateway">
87    <xsl:for-each select="./parallelGateway">
88      <xsl:element name="parallelesGateway">
89        <xsl:attribute name="xsi:type">metaBPMN2:parallelesGateway
            </xsl:attribute>
90        <xsl:attribute name="processid">
91          <xsl:value-of select="./../@id"/>
92        </xsl:attribute>
93        <xsl:attribute name="id">
94          <xsl:value-of select="./@id"/>
95        </xsl:attribute>
96        <xsl:attribute name="gatewayDirection">
97          <xsl:value-of select="./@gatewayDirection"/>
98        </xsl:attribute>
99        <xsl:attribute name="name">
100         <xsl:value-of select="./@name"/>
101       </xsl:attribute>
102     </xsl:element>
103   </xsl:for-each>
104 </xsl:template>
105
106 <xsl:template match="process" mode="exclusiveGateway">
107   <xsl:for-each select="./exclusiveGateway">
108     <xsl:element name="exclusiveGateway">
109       <xsl:attribute name="xsi:type">
            metaBPMN2:datenbasiertesExklusivesGateway</
            xsl:attribute>
110       <xsl:attribute name="processid">
111         <xsl:value-of select="./../@id"/>
112       </xsl:attribute>
113       <xsl:attribute name="id">
114         <xsl:value-of select="./@id"/>
115       </xsl:attribute>
116       <xsl:attribute name="gatewayDirection">
117         <xsl:value-of select="./@gatewayDirection"/>
118       </xsl:attribute>
```

```
119
120         <xsl:attribute name="name">
121           <xsl:value-of select="./@name"/>
122         </xsl:attribute>
123       </xsl:element>
124     </xsl:for-each>
125   </xsl:template>
126
127   <xsl:template match="process" mode="dataInputAssociation">
128     <xsl:for-each select="./task/dataInputAssociation">
129       <xsl:element name="dataInputAssociation">
130         <xsl:attribute name="xsi:type">
              metaBPMN2:dataInputAssociation</xsl:attribute>
131         <xsl:attribute name="id">
132           <xsl:value-of select="./@id"/>
133         </xsl:attribute>
134         <xsl:attribute name="sourceRef">
135           <xsl:value-of select="./sourceRef"/>
136         </xsl:attribute>
137         <xsl:attribute name="targetRef">
138           <xsl:value-of select="./targetRef"/>
139         </xsl:attribute>
140       </xsl:element>
141     </xsl:for-each>
142   </xsl:template>
143
144   <xsl:template match="process" mode="dataOutputAssociation">
145     <xsl:for-each select="./task/dataOutputAssociation">
146       <xsl:element name="dataOutputAssociation">
147         <xsl:attribute name="xsi:type">
              metaBPMN2:dataOutputAssociation</xsl:attribute>
148         <xsl:attribute name="id">
149           <xsl:value-of select="./@id"/>
150         </xsl:attribute>
151         <xsl:attribute name="sourceRef">
152           <xsl:value-of select="./sourceRef"/>
153         </xsl:attribute>
154         <xsl:attribute name="targetRef">
155           <xsl:value-of select="./targetRef"/>
156         </xsl:attribute>
157       </xsl:element>
158     </xsl:for-each>
159   </xsl:template>
160
161   <xsl:template match="process" mode="sequenceFlow">
162     <xsl:for-each select="./sequenceFlow">
163       <xsl:element name="sequenceFlow">
```

```
164         <xsl:attribute name="xsi:type">metaBPMN2:sequenceFlow</
                xsl:attribute>
165         <xsl:attribute name="id">
166           <xsl:value-of select="./@id"/>
167         </xsl:attribute>
168         <xsl:attribute name="name">
169           <xsl:value-of select="./@name"/>
170         </xsl:attribute>
171         <xsl:attribute name="sourceRef">
172           <xsl:value-of select="./@sourceRef"/>
173         </xsl:attribute>
174         <xsl:attribute name="targetRef">
175           <xsl:value-of select="./@targetRef"/>
176         </xsl:attribute>
177       </xsl:element>
178     </xsl:for-each>
179   </xsl:template>
180
181   <xsl:template match="process" mode="lane">
182     <xsl:element name="swimlane">
183       <xsl:attribute name="xsi:type">metaBPMN2:lane</xsl:attribute
              >
184       <xsl:attribute name="id">
185         <xsl:value-of select="@id"/>
186       </xsl:attribute>
187       <xsl:attribute name="name">
188         <xsl:value-of select="@id"/>
189       </xsl:attribute>
190     </xsl:element>
191   </xsl:template>
192 </xsl:stylesheet>
```

Listing 1: XSL transformations

QVT transformations

```
1  TRANSFORMATION bp2bpo : bp -> bpon
2
3  NAMESPACE http://bp
4  NAMESPACE http://extendedBPOuri
5
6
7  RULE SequenceET(E, T)
8      FORALL   event E , task T
9      WHERE    linkedElements(E,T,KA)
10
11     MAKE     Event BE ,
12                  Edge K , Edge K1, Edge K2,
13                  ProcessAction PA
14     SET      BE.name = E.name , BE.id = E.id , BE.swimlane_id=E.
           processid ,
15                  PA.name=T.name , PA.id=T.id , PA.swimlane_id=T.
                        processid ,
16                  K.name="hasSubsequentProcessAction", K.source_id=
                        BE.id , K.dest_id=PA.id , K.id=KA.id ,
17                  K1.name="belongsToSwimlane", K1.source_id=BE.id ,
                        K1.dest_id=BE.swimlane_id , K1.id=append(BE.id ,
                         BE.swimlane_id),
18                  K2.name="belongsToSwimlane", K2.source_id=PA.id ,
                        K2.dest_id=PA.swimlane_id , K2.id=append(PA.id ,
                         PA.swimlane_id)
19     ;
20
21
22 RULE SequenceTE(T, E)
23     FORALL   event E , task T
24     WHERE    linkedElements(T,E,KA)
25
26     MAKE     Event BE ,
27                  Edge K , Edge K1, Edge K2,
28                  ProcessAction PA
```

```
29      SET       BE.name = E.name , BE.id = E.id , BE.swimlane_id=E.
            processid,
30                  PA.name=T.name , PA.id=T.id , PA.swimlane_id=T.
                        processid,
31                  K.name="hasSubsequentProcessAction", K.source_id=
                        BE.id , K.dest_id=PA.id , K.id=KA.id,
32                  K1.name="belongsToSwimlane", K1.source_id=BE.id ,
                        K1.dest_id=BE.swimlane_id , K1.id=append(BE.id,
                         BE.swimlane_id),
33                  K2.name="belongsToSwimlane", K2.source_id=PA.id ,
                        K2.dest_id=PA.swimlane_id , K2.id=append(PA.id,
                         PA.swimlane_id)
34    ;
35
36
37  RULE SequenceTT(T, T)
38      FORALL    task T1 , task T2
39      WHERE     linkedElements(T1,T2,KA)
40
41      MAKE      ProcessAction PA1 ,
42                  ProcessAction PA2 ,
43                  Edge K,  Edge K1, Edge K2
44
45      SET       PA1.name=T1.name , PA1.id=T1.id , PA1.swimlane_id=T1.
            processid,
46                  PA2.name=T2.name , PA2.id=T2.id , PA2.swimlane_id=
                        T2.processid,
47                  K.name="hasSubsequentProcessAction", K.source_id=
                        PA1.id , K.dest_id=PA2.id , K.id=KA.id,
48                  K1.name="belongsToSwimlane", K1.source_id=PA1.id ,
                         K1.dest_id=PA1.swimlane_id , K1.id=append(PA1.
                        id, PA1.swimlane_id),
49                  K2.name="belongsToSwimlane", K2.source_id=PA2.id ,
                         K2.dest_id=PA2.swimlane_id , K2.id=append(PA2.
                        id, PA2.swimlane_id)
50    ;
51
52
53
54
55  RULE ParallelGateIn1()          //Task linked mit PG (Diverging)
56      FORALL    task T , ANDSplit PG
57      WHERE     linkedElements(T,PG,KA) AND PG.gatewayDirection="
            Diverging"
58      MAKE ANDSplit AS , Edge K, ProcessAction PA ,  Edge K1, Edge
            K2
59
```

```
60      SET       PA.name=T.name , PA.id=T.id , PA.swimlane_id=T.
            processid ,
61                  AS.name=PG.name , AS.id=PG.id , AS.swimlane_id=PG.
                        processid ,
62                  K.name="hasSubsequentProcessAction", K.source_id=
                        PA.id , K.dest_id=AS.id , K.id=KA.id,
63                  K1.name="belongsToSwimlane", K1.source_id=PA.id ,
                        K1.dest_id=PA.swimlane_id , K1.id=append(PA.id,
                         PA.swimlane_id),
64                  K2.name="belongsToSwimlane", K2.source_id=AS.id ,
                        K2.dest_id=AS.swimlane_id , K2.id=append(AS.id,
                         AS.swimlane_id)
65
66  ;
67
68
69  RULE ParallelGateIn2()          //Task linked mit PG (Converging)
70      FORALL    task T , ANDSplit PG
71      WHERE    linkedElements(T,PG,KA) AND PG.gatewayDirection="
            Converging"
72      MAKE ANDJoin AJ , Edge K, ProcessAction PA ,  Edge K1, Edge K2
73
74      SET       PA.name=T.name , PA.id=T.id , PA.swimlane_id=T.
            processid ,
75                  AJ.name=PG.name , AJ.id=PG.id , AJ.swimlane_id=PG.
                        processid ,
76                  K.name="hasSubsequentProcessAction", K.source_id=
                        PA.id , K.dest_id=AJ.id , K.id=KA.id,
77                  K1.name="belongsToSwimlane", K1.source_id=PA.id ,
                        K1.dest_id=PA.swimlane_id , K1.id=append(PA.id,
                         PA.swimlane_id),
78                  K2.name="belongsToSwimlane", K2.source_id=AJ.id ,
                        K2.dest_id=AJ.swimlane_id , K2.id=append(AJ.id,
                         AJ.swimlane_id)
79
80  ;
81
82
83
84
85  RULE ExclusiveGateIn1()          //Task linked mit EG (Converging)
86      FORALL    task T , databasedXORGateway EG
87      WHERE    linkedElements(T,EG,KA) AND EG.gatewayDirection="
            Converging"
88      MAKE XORJoin XJ , Edge K, ProcessAction PA ,  Edge K1, Edge K2
89
90      SET       PA.name=T.name , PA.id=T.id , PA.swimlane_id=T.
            processid ,
```

```
91                  XJ.name=EG.name , XJ.id=EG.id , XJ.swimlane_id=EG.
                        processid ,
92                  K.name="hasSubsequentProcessAction", K.source_id=
                        PA.id , K.dest_id=XJ.id , K.id=KA.id,
93                  K1.name="belongsToSwimlane", K1.source_id=PA.id ,
                        K1.dest_id=PA.swimlane_id , K1.id=append(PA.id,
                         PA.swimlane_id),
94                  K2.name="belongsToSwimlane", K2.source_id=XJ.id ,
                        K2.dest_id=XJ.swimlane_id , K2.id=append(XJ.id,
                         XJ.swimlane_id)
95 ;
96
97
98
99  RULE ParallelGateOut1()     // PG (Diverging) linked mit T
100     FORALL    ANDSplit PG, task T
101     WHERE     linkedElements(PG, T, KA) AND PG.gatewayDirection="
            Diverging"
102     MAKE ANDSplit AS , Edge K, ProcessAction P,  Edge K1, Edge K2
103
104     SET       P.name=T.name , P.id=T.id , P.swimlane_id=T.processid
            ,
105                 AS.name=PG.name , AS.id=PG.id , AS.swimlane_id=PG.
                        processid ,
106                 K.name="hasSubsequentProcessAction", K.source_id=
                        AS.id , K.dest_id=T.id , K.id=KA.id,
107                 K1.name="belongsToSwimlane", K1.source_id=P.id ,
                        K1.dest_id=P.swimlane_id , K1.id=append(P.id, P
                        .swimlane_id),
108                 K2.name="belongsToSwimlane", K2.source_id=AS.id ,
                        K2.dest_id=AS.swimlane_id , K2.id=append(AS.id,
                         AS.swimlane_id)
109 ;
110
111
112 RULE ParallelGateOut2()     // PG (Converging) linked mit T
113     FORALL    ANDSplit PG, task T
114     WHERE     linkedElements(PG, T, KA) AND PG.gatewayDirection="
            Converging"
115     MAKE ANDJoin AJ , Edge K, ProcessAction P,  Edge K1, Edge K2
116
117     SET       P.name=T.name , P.id=T.id , P.swimlane_id=T.processid
            ,
118                 AJ.name=PG.name , AJ.id=PG.id , AJ.swimlane_id=PG.
                        processid ,
119                 K.name="hasSubsequentProcessAction", K.source_id=
                        AJ.id , K.dest_id=T.id , K.id=KA.id,
```

```
120                 K1.name="belongsToSwimlane", K1.source_id=P.id ,
                        K1.dest_id=P.swimlane_id , K1.id=append(P.id, P
                        .swimlane_id),
121                 K2.name="belongsToSwimlane", K2.source_id=AJ.id ,
                        K2.dest_id=AJ.swimlane_id , K2.id=append(AJ.id,
                         AJ.swimlane_id)
122 ;
123
124
125 RULE ParallelGateOut3()     // PG (Diverging) linked mit EG (
        Converging)
126     FORALL    ANDSplit PG, databasedXORGateway EG
127     WHERE    linkedElements(PG, EG, KA) AND PG.gatewayDirection="
            Diverging" AND EG.gatewayDirection="Converging"
128     MAKE ANDJoin AJ , Edge K, XORJoin XJ,  Edge K1, Edge K2
129
130     SET      XJ.name=EG.name , XJ.id=EG.id , XJ.swimlane_id=EG.
            processid ,
131                 AJ.name=PG.name , AJ.id=PG.id , AJ.swimlane_id=PG.
                        processid ,
132                 K.name="hasSubsequentProcessAction", K.source_id=
                        AJ.id , K.dest_id=XJ.id , K.id=KA.id,
133                 K1.name="belongsToSwimlane", K1.source_id=XJ.id ,
                        K1.dest_id=XJ.swimlane_id , K1.id=append(XJ.id,
                         XJ.swimlane_id),
134                 K2.name="belongsToSwimlane", K2.source_id=AJ.id ,
                        K2.dest_id=AJ.swimlane_id , K2.id=append(AJ.id,
                         AJ.swimlane_id)
135 ;
136
137
138
139 RULE ExclusiveGateOut1()     // EG (Diverging) linked mit T
140     FORALL    databasedXORGateway EG, task T
141     WHERE    linkedElements(EG, T, KA) AND EG.gatewayDirection="
            Diverging"
142     MAKE XORSplit XS , Edge K, ProcessAction P,  Edge K1, Edge K2
143
144     SET      P.name=T.name , P.id=T.id , P.swimlane_id=T.processid
            ,
145                 XS.name=EG.name , XS.id=EG.id , XS.swimlane_id=EG.
                        processid ,
146                 K.name="hasSubsequentProcessAction", K.source_id=
                        XS.id , K.dest_id=T.id , K.id=KA.id,
147                 K1.name="belongsToSwimlane", K1.source_id=XS.id ,
                        K1.dest_id=XS.swimlane_id , K1.id=append(XS.id,
                         XS.swimlane_id),
```

```
148                 K2.name="belongsToSwimlane", K2.source_id=P.id ,
                        K2.dest_id=P.swimlane_id , K2.id=append(P.id, P
                        .swimlane_id)
149 ;
150
151 RULE ExclusiveGateOut2()     // EG (Converging) linked mit T
152     FORALL    databasedXORGateway EG, task T
153     WHERE     linkedElements(EG, T, KA) AND EG.gatewayDirection="
            Converging"
154     MAKE XORJoin XJ , Edge K, Edge K1, Edge K2, ProcessAction P
155
156     SET       P.name=T.name , P.id=T.id , P.swimlane_id=T.processid
            ,
157                 XJ.name=EG.name , XJ.id=EG.id , XJ.swimlane_id=EG.
                        processid ,
158                 K.name="hasSubsequentProcessAction", K.source_id=
                        XJ.id , K.dest_id=T.id , K.id=KA.id,
159                 K1.name="belongsToSwimlane", K1.source_id=XJ.id ,
                        K1.dest_id=XJ.swimlane_id , K1.id=append(XJ.id,
                         XJ.swimlane_id),
160                 K2.name="belongsToSwimlane", K2.source_id=P.id ,
                        K2.dest_id=P.swimlane_id , K2.id=append(P.id, P
                        .swimlane_id)
161 ;
162
163
164 RULE SwimLane(S)
165     FORALL    lane S
166     MAKE      Swimlane SL
167     SET       SL.name=S.name, SL.id=S.id
168   ;
169
170
171
172 RULE BPMNdataObject(D)
173     FORALL    dataObject D
174     MAKE      Datenelement DE1
175     SET       DE1.id=D.id , DE1.name=D.name, DE1.swimlane_id=D.
            processid
176   ;
177
178
179
180 RULE BPMNdataInputAssociation(D)
181     FORALL    dataInputAssociation D
182     MAKE      Edge K
183     SET       K.name="isInputFor", K.id=D.id , K.source_id = D.
            sourceRef, K.dest_id = D.targetRef
```

```
184     ;
185
186
187 RULE BPMNdataOutputAssociation(D)
188       FORALL    dataOutputAssociation D
189       MAKE      Edge K
190       SET       K.name="hasOutput", K.id=D.id  , K.source_id = D.
                sourceRef, K.dest_id = D.targetRef
191     ;
192
193
194 PATTERN linkedElements(E1,E2, F)
195       FORALL  sequenceFlow F
196       WHERE
197               (
198                 F.sourceRef=E1.id AND F.targetRef=E2.id
199               )
200 ;
201
202
203 PATTERN moreThanOneIn(GW)
204       FORALL  sequenceFlow SF1, sequenceFlow SF2
205       WHERE
206               (
207                  SF1.targetRef=GW.id AND SF2.targetRef=GW.id  AND NOT
                        SF1.id = SF2.id
208               )
209 ;
```

Listing 2: QVT transformations

SWRL rules for criticality classification of resources

```
<!-- Price development rules -->
Commodity(?c),
hasPrice(?c, ?p), hasPrice1(?c, ?p1), hasPrice2(?c, ?p2),
greaterThan(?p, ?p1), lessThan(?p1, ?p2)
→ hasCriticalityIndication(?c, PricesMediumCriticalityIndication)

Commodity(?c),
hasPrice(?c, ?p), hasPrice1(?c, ?p1), hasPrice2(?c, ?p2),
greaterThan(?p1, ?p2), lessThan(?p, ?p1)
→ hasCriticalityIndication(?c, PricesMediumCriticalityIndication)

Commodity(?c),
hasPrice(?c, ?p), hasPrice1(?c, ?p1), hasPrice2(?c, ?p2),
greaterThan(?p, ?p1), greaterThan(?p1, ?p2)
→ hasCriticalityIndication(?c, PricesHighCriticalityIndication)


<!-- Stock of inventory rules -->
Commodity(?c),
hasStocks(?c, ?s), hasWorldMineProduction(?c, ?p),
divide(?stockrange, ?s, ?p), greaterThan(?stockrange, 0.1),
    lessThan(?stockrange, 0.15)
→ hasCriticalityIndication(?c, StocksMediumCriticalityIndication)

Commodity(?c),
hasStocks(?c, ?s), hasWorldMineProduction(?c, ?p),
divide(?stockrange, ?s, ?p), lessThan(?stockrange, 0.1)
→ hasCriticalityIndication(?c, StocksHighCriticalityIndication)


<!-- Reserves-to-production ratio rules -->
Commodity(?c),
hasReserves(?c, ?reserves),
hasWorldMineProduction(?c, ?annualUsage),
```

```
34 divide(?range, ?reserves, ?annualUsage),
35 greaterThan(?range, 10), lessThan(?range, 25)
36 → hasCriticalityIndication(?c, ReservesMediumCriticalityIndication
       )
37
38 Commodity(?c),
39 hasReserves(?c, ?reserves), hasWorldMineProduction(?c, ?
       annualUsage),
40 divide(?range, ?reserves, ?annualUsage), lessThan(?range, 10)
41 → hasCriticalityIndication(?c, ReservesHighCriticalityIndication)
42
43
44 <!-- Market power and country concentration rules -->
45 Commodity(?c),
46 hasHhiOfCountries(?c, ?hhi),
47 greaterThan(?hhi, 0.1), lessThan(?hhi, 0.2)
48 → hasCriticalityIndication(?c,
       ConcentrationMediumCriticalityIndication)
49
50 Commodity(?c),
51 hasHhiOfCountries(?c, ?hhi), greaterThan(?hhi, 0.2)
52 → hasCriticalityIndication(?c,
       ConcentrationHighCriticalityIndication)
```

Listing 3: SWRL rules for resource classification

Commodity classification explanation path

```
1  Commodity(?c), hasPrice(?c, ?p), hasPrice1(?c, ?p1), hasPrice2(?c,
       ?p2), greaterThan(?p, ?p1), greaterThan(?p1, ?p2) ->
     hasCriticalityIndication(?c, PricesHighCriticalityIndication)
2
3  Commodity(?c), hasStocks(?c, ?s), hasWorldMineProduction(?c, ?p),
     divide(?stockrange, ?s, ?p), lessThan(?stockrange, 0.1) ->
     hasCriticalityIndication(?c, StocksHighCriticalityIndication)
4
5  Sodium Type Commodity
6
7  Sodium hasApparentConsumptionOfUs 4950000.0f
8  Sodium hasPrice 4.962845f
9  Sodium hasStocks 0.0f
10 Sodium hasPrice2 4.7361984f
11 Sodium hasSecondaryProduction 0.0f
12 Sodium hasPrice1 4.905275f
13 Sodium hasReserves 3.3E9f
14 Sodium hasWorldMineProduction 4.41E7f
15
16 PricesHighCriticalityIndication Type HighCriticalityIndication
17
18 StocksHighCriticalityIndication Type HighCriticalityIndication
19
20 highlyCriticalCommodity EquivalentTo Commodity and (
     hasCriticalityIndication min 2 HighCriticalityIndication)
```

Listing 4: Explanation path for Sodium being highly critical

Resource ontology serialization

Please note that the ontology was generated by the resource ontology editor as described in Section 5.5 using OWLAPI. In order to save space, some comments have been removed from the original serialization and the fontsize is decreased intentionally.

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#"
     xml:base="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl"
     xmlns:Ontology1346170549107="http://www.semanticweb.org/ontologies/2012/7/Ontology1346170549107.owl#"
     xmlns:ontologies="http://www.semanticweb.org/ontologies/2012/4/25/"
     xmlns:Payroll="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Enter_Payroll"
     xmlns:Envelope="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Envelope"
     xmlns:Accounting="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Use_Accounting"
     xmlns:Ontology133796774647="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:Accounting2="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Accounting"
     xmlns:Enveloper="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Enveloper"
     xmlns:Application="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Application"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:Printer="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Printer"
     xmlns:Unskilled="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Unskilled">
    <owl:Ontology rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl"/>



    <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#hasCapability -->
    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
        hasCapability"/>

    <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#hasSkill -->
    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
        hasSkill">
        <rdfs:range rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
            Skills"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#hasName -->
    <owl:DatatypeProperty rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
        hasName">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
            Resources"/>
        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:DatatypeProperty>

    <!-- http://www.semanticweb.org/ontologies/2012/7/Ontology1346170549107.owl#hasHourlyRate -->
    <owl:DatatypeProperty rdf:about="http://www.semanticweb.org/ontologies/2012/7/Ontology1346170549107.owl#
        hasHourlyRate"/>
```

```
40
41
42      <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Accounting_Software -->
43      <owl:Class rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
            Accounting_Software">
44          <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl
                #Application_System"/>
45      </owl:Class>
46
47      <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Administrator -->
48      <owl:Class rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
            Administrator">
49          <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.
                owl#Staff"/>
50      </owl:Class>
51
52      <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Application_System -->
53      <owl:Class rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
            Application_System">
54          <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.
                owl#Resources"/>
55      </owl:Class>
56
57      <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Clerk -->
58      <owl:Class rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Clerk">
59          <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.
                owl#Staff"/>
60      </owl:Class>
61
62      <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Developer -->
63      <owl:Class rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Developer">
64          <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.
                owl#Staff"/>
65      </owl:Class>
66
67      <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Envelope_Machine -->
68      <owl:Class rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
            Envelope_Machine">
69          <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl
                #Printer"/>
70      </owl:Class>
71
72      <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Machine -->
73      <owl:Class rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Machine">
74          <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.
                owl#Resources"/>
75      </owl:Class>
76
77      <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Printer -->
78      <owl:Class rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Printer">
79          <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl
                #Machine"/>
80      </owl:Class>
81
82      <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Printing -->
83      <owl:Class rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Printing">
84          <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.
                owl#Capabilities"/>
85      </owl:Class>
86
87      <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Secretary -->
88      <owl:Class rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Secretary">
89          <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.
                owl#Staff"/>
90      </owl:Class>
91
92      <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Unskilled_Worker -->
93      <owl:Class rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
            Unskilled_Worker">
94          <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.
```

```
                owl#Staff"/>
95  </owl:Class>
96
97  <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Capabilities -->
98  <owl:Class rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
        Capabilities">
99      <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
100 </owl:Class>
101
102 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Resources -->
103 <owl:Class rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Resources
        "/>
104
105 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Skills -->
106 <owl:Class rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Skills"/>
107
108 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Staff -->
109 <owl:Class rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Staff">
110     <rdfs:subClassOf rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.
            owl#Resources"/>
111 </owl:Class>
112
113
114 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#ARISModelingSkill -->
115 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
        ARISModelingSkill"/>
116
117 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Ada -->
118 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
        Ada"/>
119
120 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Alice -->
121 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
        Alice">
122     <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            Developer"/>
123 </owl:NamedIndividual>
124
125 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#BPMNModelingSkill -->
126 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
        BPMNModelingSkill"/>
127
128 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Bill -->
129 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
        Bill">
130     <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Clerk
            "/>
131 </owl:NamedIndividual>
132
133 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Bob -->
134 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
        Bob">
135     <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            Developer"/>
136     <Ontology1346170549107:hasHourlyRate rdf:datatype="http://www.w3.org/2001/XMLSchema#double">90.0</
            Ontology1346170549107:hasHourlyRate>
137     <hasSkill rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            JavaProgrammingSkill"/>
138     <hasSkill rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            LinuxAdministrationSkill"/>
139 </owl:NamedIndividual>
140
141 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#CGuru -->
142 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
        CGuru"/>
143
144 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Enter_Payroll_Data -->
145 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
        Enter_Payroll_Data">
146     <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
```

```
                Skills"/>
147     </owl:NamedIndividual>
148
149     <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Envelope -->
150     <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            Envelope">
151         <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
                Skills"/>
152     </owl:NamedIndividual>
153
154     <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Enveloper_A01 -->
155     <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            Enveloper_A01">
156         <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
                Envelope_Machine"/>
157     </owl:NamedIndividual>
158
159     <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Hans -->
160     <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            Hans">
161         <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
                Administrator"/>
162     </owl:NamedIndividual>
163
164     <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Heidi -->
165     <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            Heidi">
166         <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
                Secretary"/>
167     </owl:NamedIndividual>
168
169     <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#JavaGuru -->
170     <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            JavaGuru"/>
171
172     <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#JavaProgrammingSkill -->
173     <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            JavaProgrammingSkill">
174         <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
                Skills"/>
175     </owl:NamedIndividual>
176
177     <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Jeff -->
178     <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            Jeff">
179         <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Clerk
                "/>
180     </owl:NamedIndividual>
181
182     <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Jenny -->
183     <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            Jenny">
184         <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
                Unskilled_Worker"/>
185     </owl:NamedIndividual>
186
187     <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#LinuxAdministrationSkill -->
188     <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            LinuxAdministrationSkill">
189         <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
                Skills"/>
190     </owl:NamedIndividual>
191
192     <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Lisa -->
193     <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            Lisa">
194         <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
                Secretary"/>
195     </owl:NamedIndividual>
196
```

```
197 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Mark -->
198 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
        Mark"/>
199
200 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Micor_Money_4 -->
201 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
        Micor_Money_4">
202     <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
            Accounting_Software"/>
203 </owl:NamedIndividual>
204
205 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Micro_Mone_3 -->
206 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
        Micro_Mone_3">
207     <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
            Accounting_Software"/>
208 </owl:NamedIndividual>
209
210 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Micro_Money_1 -->
211 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
        Micro_Money_1">
212     <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
            Accounting_Software"/>
213 </owl:NamedIndividual>
214
215 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Micro_Money_2 -->
216 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
        Micro_Money_2">
217     <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
            Accounting_Software"/>
218 </owl:NamedIndividual>
219
220 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#OntologyModelingSkill -->
221 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
        OntologyModelingSkill"/>
222
223 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Print -->
224 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
        Print">
225     <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
            Skills"/>
226 </owl:NamedIndividual>
227
228 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Printer_MFC-201 -->
229 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
        Printer_MFC-201">
230     <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
            Printer"/>
231 </owl:NamedIndividual>
232
233 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Printer_MFC-202 -->
234 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
        Printer_MFC-202">
235     <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
            Printer"/>
236 </owl:NamedIndividual>
237
238 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#SQLGuru -->
239 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
        SQLGuru"/>
240
241 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#SQLSkill -->
242 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
        SQLSkill">
243     <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
            Skills"/>
244 </owl:NamedIndividual>
245
246 <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#Testimies -->
247 <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
        Testimies"/>
```

```
248
249     <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Thomas -->
250     <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            Thomas"/>
251
252     <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#UMLModelingSkill -->
253     <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            UMLModelingSkill"/>
254
255     <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#UberDeveloper -->
256     <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            UberDeveloper"/>
257
258     <!-- http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#Use_Accounting_Software -->
259     <owl:NamedIndividual rdf:about="http://www.semanticweb.org/ontologies/2012/4/25/Ontology133796774647.owl#
            Use_Accounting_Software">
260         <rdf:type rdf:resource="http://www.semanticweb.org/ontologies/2012/4/25/Ontology1337967746472.owl#
                Skills"/>
261     </owl:NamedIndividual>
262 </rdf:RDF>
```

Listing 5: Full resource ontology