Modularization of Process Models Using Natural Language Techniques

Master Thesis

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MASTER OF SCIENCE (M.Sc.)

IN INFORMATION SYSTEMS

AT THE SCHOOL OF BUSINESS AND ECONOMICS OF
HUMBOLDT-UNIVERSITÄT ZU BERLIN

submitted by

Henrik Leopold

Matriculation number 522866

Supervisor: Prof. Dr. Jan Mendling

Berlin, September 15th 2010
Acknowledgement

On this page I would like to take the opportunity to thank those people who were indirectly involved in this thesis. At the beginning, I would like to thank my supervisor Prof. Dr. Jan Mendling for his extraordinary commitment in lectures, seminars and our joint paper projects. Due to his inspiring attitude I discovered my passion for research in the area of BPM and really enjoyed to collaborate on my first publications. Second, I want to thank my associate supervisor Dr. Hajo Reijers for enabling my research stay in Eindhoven and for supporting my thesis in the context of various discussions, feedback and valuable ideas.

Further, I would like to thank Thomas, Christopher, Fabian and Matthias for their support, feedback and ideas. Without the study groups, various discussions about exams and collaboration on term papers, the study time would have been incomparably harder. However, also the time outside the university, including several trips and nice evenings, contributed to the motivation for accomplishing my studies and this thesis. Particularly, I want to thank Fabian for his proof read.


-i-
Abstract

In order to successfully compete in a globalized business environment and to continuously satisfy the needs of their customers, companies are often required to dynamically and rapidly adapt their business processes. Business process management and business process modeling are frequently considered to be key concepts for managing, understanding and changing business processes. The main problem attached with business process models is their quickly increasing complexity when they are representing real-world business processes. Therefore, the techniques of modularization and abstraction are often employed to reduce the complexity of these process models by introducing subprocesses or by aggregating certain process parts. From a structural perspective, already several solutions for implementing these techniques exist. However, the current available algorithms lack the computation of the labels of activities. This is quite frequently required as newly created activities are introduced to represent subprocesses or a set of collapsed activities. As the determination of an appropriate activity label is an expensive task for the modeler, we aim to develop an algorithm for automating this labeling task. In order to accomplish the development of such an algorithm, we perform an analysis on existing process model collections to identify existing labeling practices. Based on the results, we develop our labeling approach which automatically computes a set of potentially suitable labels which are provided to the modeler. As a consequence, the effort for the modeler is reduced to select the most proper label for the given context. In order to demonstrate the capability of our labeling approach we additionally present a validation experiment.
# Table of Contents

List of Acronyms vi

List of Figures vii

List of Tables viii

1 Introduction 1

1.1 Motivation ................................. 1

1.2 Research Contribution ......................... 3

1.3 Research Methodology .......................... 4

1.4 Structure of this Thesis ......................... 4

2 Background 6

2.1 Business Process Management ...................... 6

2.1.1 Role and Definition of Business Processes .......... 6

2.1.2 Business Process Management as a Life Cycle .......... 7

2.2 Business Process Modeling ...................... 9

2.2.1 The Concept of Business Process Models .......... 9

2.2.2 The Process of Modeling .................... 11

2.3 Modularization in Business Process Models .......... 12

2.4 Abstraction in Business Process Models .......... 14

2.5 Linguistics ................................ 19

2.5.1 Introduction to Linguistics .................... 19

2.5.2 Relevance of Linguistics for Business Process Modeling 20

2.5.3 Lexical Semantics .......................... 21

2.5.4 Natural Language Processing .................. 23

2.6 Related Work of NLP in Business Process Models .... 27
# TABLE OF CONTENTS

## 3 Derivation and Classification of Manual Naming Strategies

3.1 Analyzed Process Model Collections .................................................. 30
3.2 Methodology of the Analysis ................................................................. 32
3.3 Resulting Classification from Analysis ................................................... 34
3.4 Towards a Theoretical Foundation for the Identified Naming Strategies ... 40
3.5 Summary and Implications ................................................................. 44

## 4 Strategy Development for Automated Labeling

4.1 Preconditions ............................................................... 46
4.2 Overview about Proposed Approach .................................................... 48
4.3 Phase 1: Annotation ................................................................. 50
4.4 Phase 2: Information Extraction ..................................................... 52
  4.4.1 Dominating Element Extraction ................................................ 53
  4.4.2 Subordinate Element Extraction ................................................ 54
  4.4.3 Lexical Conjunction ............................................................... 55
  4.4.4 Logical Conjunction ............................................................... 56
  4.4.5 Label Repository ................................................................. 58
  4.4.6 Event Extraction ................................................................. 59
  4.4.7 Main Activity Extraction ........................................................ 61
  4.4.8 Combining All Approaches .................................................... 62
4.5 Phase 3: Proposal List Creation ....................................................... 64
4.6 Application on an Exemplary Process Model ....................................... 65
4.7 Summary ................................................................. 67

## 5 Validation ................................................................. 69

5.1 Methodology ................................................................. 69
5.2 Results ................................................................. 71
5.3 Assessment of Validity ............................................................... 74
# TABLE OF CONTENTS

5.3.1 Internal Validity ............................................. 75  
5.3.2 External Validity ............................................. 76  
5.3.3 Construct Validity ............................................. 77  
5.4 Summary ......................................................... 80  

6 Conclusions 82  
6.1 Summary of Results ............................................. 82  
6.2 Discussion ..................................................... 83  
6.3 Future Research ............................................... 85  

References 87  

A Java Classes of Prototype Implementation 101  
A.1 Package epcModel ............................................ 101  
A.2 Package phase1 ............................................... 103  
A.3 Package phase2 ............................................... 103  
A.4 Package phase3 ............................................... 104  
A.5 Package main .................................................. 104  

B Structure of the Enclosed CD 106
### Nomenclature

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>BPM</td>
<td>Business Process Management</td>
</tr>
<tr>
<td>BPMA</td>
<td>Business Process Model Abstraction</td>
</tr>
<tr>
<td>EPC</td>
<td>Event-Driven Process Chain</td>
</tr>
<tr>
<td>EPML</td>
<td>Event-Driven-Process-Chain-Markup-Language</td>
</tr>
<tr>
<td>JWNL</td>
<td>Java WordNet Library</td>
</tr>
<tr>
<td>KCPM</td>
<td>Klagenfurt Conceptual Predesign Model</td>
</tr>
<tr>
<td>MDA</td>
<td>Model Driven Architecture</td>
</tr>
<tr>
<td>MOF</td>
<td>Meta Object Facility</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>YAWL</td>
<td>Yet Another Workflow Language</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Business Process Management Life Cycle, adapted from [Men08, p. 5]</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>Process of Modeling, adapted from [FvdW06]</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>Abstraction Artifacts on the MOF Levels, adapted from [SRNW10]</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>Illustration of Elimination</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>Illustration of Aggregation</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>Illustration of Full Aggregation</td>
<td>18</td>
</tr>
<tr>
<td>7</td>
<td>SAP Model <em>Purchase Requisition</em></td>
<td>36</td>
</tr>
<tr>
<td>8</td>
<td>SAP Model <em>Change in Material Price</em> (adapted)</td>
<td>36</td>
</tr>
<tr>
<td>9</td>
<td>SAP Model <em>Date Planning</em></td>
<td>37</td>
</tr>
<tr>
<td>10</td>
<td>SAP Model <em>Backorder Processing</em></td>
<td>37</td>
</tr>
<tr>
<td>11</td>
<td>SAP Model <em>Customer Inquiry and Quotation Processing</em></td>
<td>38</td>
</tr>
<tr>
<td>12</td>
<td>SAP Model <em>Shipping</em> (adapted)</td>
<td>38</td>
</tr>
<tr>
<td>13</td>
<td>SAP Model <em>Invoice Verification</em> (adapted)</td>
<td>39</td>
</tr>
<tr>
<td>14</td>
<td>Overview about the Proposed Labeling Approach</td>
<td>49</td>
</tr>
<tr>
<td>15</td>
<td>SAP Model <em>Capacity Planning</em></td>
<td>66</td>
</tr>
<tr>
<td>16</td>
<td>SAP Model <em>Analysis</em></td>
<td>78</td>
</tr>
<tr>
<td>17</td>
<td>SAP Model <em>Information System</em></td>
<td>79</td>
</tr>
</tbody>
</table>
List of Tables

1  Mapping of Naming Strategies to Theories of Meaning       43
2  Frequency Distribution of Naming Strategies              44
3  Overview about Activity Label Structures                 50
4  Overview about Event Label Structures                    51
5  Mapping of Naming Strategies to Automated Sub Approaches 52
6  Computed Proposal List for SAP Model *Capacity Planning* 67
7  Results of the Validation Approach                       72
B8  Content of the Enclosed CD                              106
1. Introduction

This section provides an introduction to this master thesis. We start by giving a general motivation about the topic. Subsequently, we present the research contribution of this thesis and continue with an introduction to our research methodology. We close this section by outlining the structure of this thesis.

1.1. Motivation

Business process management represents an important concept to enable companies to cope with the increasing dynamics and challenges in the days of globalization. In order to keep up with competitors, business processes need to be adjusted rapidly and costs have to be reduced at the same time. For the purpose of gaining understanding and performing detailed analyses of business processes, they are mapped to business process models. Nevertheless, this often entails several problems.

One of them is the definition of an adequate level of granularity. Top level managers are usually interested in coarse grained process models in order to gain a brief overview. By contrast, other people in the organization, as for instance clerks, require a very exhaustive description of the process steps. Another problem is that real world process models quickly become tremendously complex and important aspects like understandability and maintainability significantly suffer from this fact. In order to cope with these challenges, the technique of process model modularization is employed [KB07]. By splitting up large processes into smaller subprocesses, the complexity can be reduced and certain factors such as understandability and maintainability are increased. In addition, by introducing a main process where the activities are linked to the created subprocesses, different levels of granularity are provided. As pointed out by [PSW09], the formal relation between these models is necessary to assure the consistency between them. Otherwise, the maintenance of these models would become a time-consuming and error-prone task. A technique which is quite related to modularization, and may also help to solve the discussed problems, is given by the concept of business process model abstraction (BPMA). In the context of
BPMA, process models are reduced to the information which is required for a particular purpose, while dispensable details are omitted.

In order to use these techniques, most of the popular modeling languages, such as the Event-driven Process Chains (EPCs) [DB07, pp. 258-266] and the Business Process Modeling Notation (BPMN) [Obj06a, pp. 56-64], allow to structure process models hierarchically. Hence, process modelers can decompose a complex process model and connect the different subprocesses to a main process. Moreover, various approaches for automatically applying abstraction methods on process models exist [EG08, PSW08a, BRB07]. However, these algorithms focus on the structural aspects of abstraction and miss to include the labeling of new activities, which are created in the context of the aggregation of several activities. The same problem also occurs in other scenarios. For instance, if a process model is decomposed into subprocesses in the context of modularization, each of the corresponding activities in the main process requires a label which adequately describes the content of the underlying subprocess. A further example for this labeling problem is given in more general setting. In some cases, a name for a whole process model is required, e.g. for the purpose of identifying a model in a process model repository. However, in all cases a particular set of activities, which may even represent a whole process model, are collapsed into one single activity and therefore require an appropriate label.

As emphasized by former research, this labeling task can be considered to be of crucial importance to assure the quality of process models since the labeling of activities significantly contributes to the overall understanding of a process model [MRR09]. However, so far this labeling task must be performed manually by a modeler. This is usually expensive as it may be time consuming to adequately determine a label for activities which need to describe whole process fragments or even a complete model. Motivated by these facts, we attempt to develop an automated solution for this labeling task.
1.2. Research Contribution

The research goal of this thesis is the development of an algorithm which automatically computes a set of label proposals for a given set of activities, which is collapsed into one single activity. As already mentioned above, this approach can then be applied in the three uses cases of abstraction, modularization and also for naming complete process models. Using our approach, the modeler is provided with a set of potentially suitable labels from which the modeler can select the most adequate label for the given context. As a result, the manual step of labeling is reduced to one single click.

In the context of achieving this research goal, this thesis provides the following contributions:

- **Classification of Labeling Practices**: As a result of the analysis of multiple process model collections, we detect patterns of existing labeling strategies conducted by humans. Based on the results, we provide a classification of these labeling strategies.

- **Extension of Label Structure Classification**: A side contribution is the extension of the labeling structure classification provided in [LSM09]. As it is an important requirement for our labeling approach, we extend the existing classification for activity labels by investigating and adding the structure of event labels.

- **Labeling Approach**: According to our research goal we provide an algorithm which is capable of automatically computing a set of label proposals for a given set of labels as for instance a process fragment or subprocess.

- **Prototype of Labeling Approach**: Based on the developed labeling approach, we provide a prototypical Java implementation.

- **Validation Experiment**: In order to demonstrate the capability of our labeling approach for computing suitable labels, we conduct a validation experiment.
1.3. Research Methodology

According to Hevner et al. [HMPR04] there are two paradigms characterizing most of the research which is conducted in the field of information systems: behavioral science and design science. Behavioral science is concerned with developing and verifying theories explaining the behavior of humans and organizations. By contrast, design science tries to increase the capabilities of humans and organizations by developing new and innovative artifacts. Based on this classification, we can assign our work to the latter category.

In order to achieve the research goal defined above, we perform the following three steps:

1. **Analysis and Classification**: In order to learn how activities are labeled in practice, when they are used to describe the contents of a subprocess, we conduct an analysis on three real world process model collections. The identified approaches will be structured, classified and furthermore used to develop our semi-automated approach.

2. **Development of Labeling Approach**: Utilizing the elicited knowledge from the first step, we develop and construct a semi-automated labeling approach. Thereby, we compose several sub approaches to an overall labeling approach.

3. **Validation Experiment**: In order to demonstrate the capability of the developed labeling approach, we conduct an evaluation. In the context of the evaluation, we apply the developed approach to one of the given process model collections and compare the computed results with the process names, which were originally given in the model collection.

1.4. Structure of this Thesis

This thesis is organized in six sections. We start with a background section giving a general overview about the discussed concepts in this thesis. In alignment with our research methodology we proceed with the analysis of labeling practices in real world
process model collections. Based on the insights from this analysis, we develop our labeling approach in the following section. Subsequently, we present and discuss our validation experiment before we finish with our conclusion.

- **Section 1: Introduction** In this section we sketch and discuss the motivation of this work and present the outline of the thesis.

- **Section 2: Background** The background section introduces the scientific domains which form the basis of this work. This includes disciplines from the field of business process management, modeling and linguistics.

- **Section 3: Derivation and Classification of Manual Naming Strategies**: This section presents an analysis on three real world process model collections. Goal of the analysis is to elicit existing labeling practices conducted by humans in order to employ them for the automatization.

- **Section 4: Strategy Development for Automated Labeling**: In this section we develop our semi-automated labeling approach based on the results of the previous section. As the main approach includes several sub approaches, we introduce each part in detail using pseudo code.

- **Section 5: Validation**: In this section we aim for demonstrating the capability of our labeling approach to provide useful labels. Therefore, we conduct a validation experiment. Moreover, we discuss certain validity aspects of our experiment in order to accurately assess the results.

- **Section 6: Conclusion**: Finally, we summarize the findings of this thesis and sketch the possibilities for future research.
2. Background

This section aims at providing the background which is required for understanding the concepts and solutions discussed in this thesis. As already indicated in the introduction, the development of the labeling approach is an interdisciplinary task. Therefore, this section introduces the domains of business process management, business process modeling, modularization and abstraction in business process models and also aspects from the field of linguistics.

2.1. Business Process Management

Globalization forces companies to adjust their business processes increasingly fast to keep up with their competitors. The field of business process management is concerned with enabling companies to effectively manage and implement business processes and their changes. In order to provide a theoretical foundation for the concepts presented in this thesis, this section introduces two aspects. First of all, we discuss the term *business process* and attempt to find a proper definition. Secondly, we introduce the activities related with business process management.

2.1.1. Role and Definition of Business Processes

Every product introduced to a market requires several activities to be performed. Such business process activities can be either performed manually by employees or supported by information systems. In order to achieve a company’s objectives, it is of crucial importance that people and resources, such as information systems, collaborate in an effective manner. Business processes are considered to be one of the key instruments to organize and understand the relationships between the performed activities and to successfully accomplish such effective collaboration [Wes07, p. 4].

In literature various definitions for the term *business process* can be found, each emphasizing the importance of different aspects. For instance, Hammer and Champy focus on the input and output characteristics by defining a business process as a ”collection
of activities” taking one or more types of input and creating a valuable output for the customer [HC93, p. 38]. Thereby, it is worth noticing that the term collection is not very specific in regard to how these activities are related. By contrast, Davenport delimits the relation between the tasks more specifically, by defining a business process as ”a set of logically-related tasks” which are performed in order to achieve a defined business goal [Dav90]. Similar to Hammer and Champy, Davenport additionally mentions the importance of the customers of a process and the characteristic that a process may cross organizational boundaries.

Based on process features identified by former definitions, Weske provides a condensed but also very comprehensive definition, which we will adapt for our work:

**Definition 1**: A business process consists of a set of activities which are performed in an organizational and technical environment in a coordinated fashion. In this manner, they jointly achieve a desired business goal. Although each business process is performed by a single organization, it may interact also with business processes from other organizations [Wes07, p. 5].

Having defined the term business process, we turn to the field of business process management in the next paragraph.

2.1.2. Business Process Management as a Life Cycle

As already indicated, business process management is associated with a set of specific activities. For instance, Mendling defines business process management in a very general manner as all management activities, which are concerned with business processes [Men08, p. 5]. Other definitions, as for example provided by van der Aalst et al. [vdAtHW03] or Weske [Wes07, p. 5], explicitly name these activities such as analysis, design, enactment etc. These typical business process management activities are generally arranged in a life cycle. Such business process management life cycles were proposed by several authors in different variants [vdAtHW03, zMR04, Wes07, Men08]. However, we will follow the life
cycle proposed by Mendling [Men08, p. 5] as it combines multiple life cycle concepts and focuses on both, business process management activities and outcomes of each activity. The life cycle is depicted in Figure 1. The solid arcs indicate the order of the activities while the dashed arcs represent feedback loops. The following activities are part of the life cycle:

- **Analysis**: The life cycle is entered with the analysis of the technical and organizational environment. As a result, a list of business process requirements is created.

- **Design**: Driven by these requirements, the business process is identified and subsequently formalized as a business process model.

- **Implementation**: Based on the created process model, the implementation is conducted. This includes several activities such as technical implementation and configuration of the systems, creation of a suitable infrastructure, but also training of
the employees participating in the process.

- **Enactment**: Once the implementation has been carried out, the enactment of the process can start. Hence, business process instances are enacted using the infrastructure. During this phase, several case data is produced which can subsequently be used as input for the activities monitoring and evaluation.

- **Monitoring**: This activity comprises the continuous monitoring and evaluation of process metrics with regard to the process instances. If problematic situations occur, counteractions are triggered.

- **Evaluation**: In the evaluation step the case data is considered on an aggregated level and compared with the initial requirements. As a result, new requirements are created which serve as input for the next iteration of the life cycle.

Considering the activities of the business process management life cycle, the research question discussed in this thesis can be assigned to the design step. In particular, we are concerned with automating steps of the creation of business process models, which are used for representing business processes. For this reason, we will present the concepts of business process modeling in the following section.

### 2.2. Business Process Modeling

This section aims at providing a deeper understanding of the concept of business process modeling. Therefore, two major aspects will be discussed. First of all, we introduce the concept of business process models and attempt to find a definition for this term by starting from a very general point of view. Secondly, we will briefly introduce the steps which are contained in the process of modeling

#### 2.2.1. The Concept of Business Process Models

In order to find a proper definition for the term business process model, we shortly investigate the meaning of the term model in a general setting. According to Stachowiak
a model has three basic characteristics [Sta73, p. 132]:

- **Mapping feature**: A model is derived from an original.
- **Abstraction feature**: A model only provides the relevant information from the original.
- **Pragmatic feature**: A model is created for a specific purpose.

Hence, according to Stachowiak a model can be defined as a simplification of an original, created for a specific purpose. This very general view on models is also supported by more specific model definitions from the field of Model Driven Architecture (MDA). For instance, Brown describes a model as an abstraction from a physical system that focusses on relevant information and ignores extraneous details and enables engineers to reason about that system [Bro04]. Another definition is given by Bézivin and Gerbé who define a model as a simplification of a system which was created with an intended goal [BG01]. However, as also discussed by Mendling [Men08, p. 7], this view is for instance criticized by Schuette and Rotthowe [SR98]. They define a model not as a mapping from the real world, but as a result created by a modeler for a particular purpose, with a specific language and at a given point of time [SR98]. Their argument is that a model cannot be defined as a mapping from reality as this requires the reality to be independent from their observers and the subjective perception of all individuals to be congruent. Following this line of argumentation, we define the term business process model as follows:

**Definition 2**: A *business process model* is a construct resulting from the mapping of a business process. Thereby, the business process may be a real-world business process according to the modelers perception, or a business process which was conceptualized by the modeler [Men08, p. 7].

After we have found a definition for the term *business process model*, the next paragraph focusses on the creation of such models and introduces the according steps.
2.2.2. The Process of Modeling

In literature modeling is considered to be a complex task due to the fact that several people with different backgrounds and perceptions are involved [Rit07, HPvdW05]. Nevertheless, the goal of this section is to present the steps which are included in the modeling process. A definition for the task of modeling, or in particular for the task of business process modeling, is given by van der Aalst et al.:

**Definition 3**: Business process modeling describes the manual task of identification and specification of business processes [vdAtHW03].

Although this definition gives already some idea what to expect from the modeling process, it does not provide any specific information about how the task of business process modeling can actually be accomplished. Frederiks and van der Weide [FvdW06] describe the process of information modeling which we will adapt for the business process modeling domain. Basically, they differentiate four different phases: Elicitation, modeling, verification and validation.

In the *elicitation* phase the important information is collected from the so called universe of discourse. Subsequently this information is verbalized, unified and stored in an informal specification. In the succeeding *modeling* phase the informal specification is transformed into a formal specification. This is accomplished by mapping the sentences on modeling concepts, i.e. a business process modeling language. In the *verification* phase the formal specification is checked for internal consistency. Afterwards, the formal specification is validated by comparing the informal specification with a paraphrased version of the formal specification. Figure 2 illustrates the process of modeling graphically.

Relating the introduced process of modeling to the task of naming process models, we can assign the naming task to the modeling phase as it is one particular activity of mapping the informal specification to the modeling language. However, in the first place
this phase only includes the labeling of activities. Only after a modeler decided to split a business process into different fragments, the use case of naming whole process model parts arises. For this reason, the next subsection will introduce the technique of business process model modularization as one important use case for the labeling task.

2.3. Modularization in Business Process Models

Modularity is a general design principle for managing complexity [Lan02]. The idea behind this concept is to split up a complex system into smaller subsystems which can be managed independently, but still function as a whole [RM08]. The application of the modularity principle can be found in various domains [LAYT07]. For instance, in the field of product development the principle of modularity has been suggested to cope with the design and production of increasingly complex technology, short product life cycles and rapidly rising development costs [BC97, DJT01]. As a result of modularization, products
can be recomposed to several product variants which allows to achieve customization, while still gaining economies of scale since components can be reused in several products [Pin99, p. 196],[MG03].

However, the concept of modularity is also applied to business processes. The resulting modules of a process model are called subprocesses and entail various advantages. For instance, it is argued that subprocesses support a modeling style of stepwise refinement, encourage to reuse process models, may reduce the development time of the whole process model and are also supposed to increase the understandability of process models [vdAvH04, p. 48],[RM08].

Based on the aspects proposed by Mendling and Reijers [RM08], we discuss three major issues in the context of process model modularization: When should we consider a process model for modularization, which parts of a process model are suitable for modularization and how can we finally accomplish the modularization in the sense of implementation.

- **Modularization Prerequisites:** One of the key challenges is to decide when a process model should be modularized. Focussing on the intention of a process model to convey understandable information, the resulting question is, when the understandability of a process actually suffers from its complexity. In their work on process models guidelines in [MRvdA10], the authors argue that a process model should be decomposed if it contains more than 50 elements. This recommendation is motivated by the empirical insight that process models with more than 50 elements tend to have an error probability of above 50% [MNvdA07].

- **Modularization Selection:** Having decided to decompose a process model into subprocesses entails the selection of suitable process parts. As pointed out in [RM10], currently there is no dedicated approach to guide modularization based on metrics. Nevertheless, there are still some guidelines on how to select good candidates for subprocesses. For instance, in [VVL10, VVK09] it is suggested to decompose the model into single-entry-single-exit (SESE) fragments. Another pro-
posal is made in [Dav03, p. 278] where the author emphasizes the benefits of long and thin models as opposed to square models since these tend to have many interconnections or failure loops.

- **Modularization Implementation**: The implementation of modularization can be discussed on different levels. First of all, we can state that several modeling languages such as EPCs [DB07, pp. 258-266], BPMN [Obj06a, pp. 56-64], and Yet Another Workflow Language (YAWL) [vdAtH05] provide the possibility of implementing subprocesses. Nevertheless, the possibilities of employing an operator to automatically implement modularization, are still limited. Recently, the extraction of subprocesses was introduced as a change pattern for process aware information systems [WRR07]. But still, the support of modularization by modeling tools does rely on the modeler. For instance, ARIS offers two possibilities for creating subprocesses. Either the modeler refines an existing function or segments a model into different parts [Dav03, pp. 243-250].

Summarizing this subsection, we can state that modularization of business process models is an important technique to reduce the complexity of business process models. However, the implementation of this technique is in most cases done manually. Even if an algorithm automatically selects suitable parts and sets up a process hierarchy, we would still need to determine proper labels for the newly created activities which link to the subprocesses. The next subsection introduces another method which is used manage the complexity of processes. Moreover, it also represents a second use case for the labeling task in process models.

2.4. Abstraction in Business Process Models

This subsection will introduce the technique of business process model abstraction and how it is related to the topic of this thesis.
Informally, BPMA can be described as an operation on process models preserving relevant process properties and excluding insignificant details. As a result, BPMA provides a particular level of information which is required for a given purpose [SDMW10]. Hence, the loss of information and the reduction of complexity in the abstracted model is a desired outcome of BPMA [PSW09]. The model elements, such as functions or events, which are abstracted in the context of the abstraction operation, are referred to as abstraction objects.

Approaching the field of abstraction from a formal perspective, BPMA can be described as follows. We consider a process model $m$ of type $M$ and a set containing the possible abstractions of $m$, denoted with $\text{abstr}(m)$. In order to obtain an abstraction $m_a \in \text{abstr}(m)$ of model $m$, we introduce a basic abstraction function $\alpha_0 : M \rightarrow M$. Thereby, the function $\alpha_0$ transforms the process model $m$ into $m_a$ by abstracting from an insignificant abstraction object $o$. In order to illustrate the concept of abstraction we allocate the abstraction artifacts on the different levels of the Meta Object Facility (MOF) [Obj06b] as proposed by [Küh06] in a general setting and by [SRNW10] in the context of BPMA (see Figure 3). The level M0 captures a set of process instances $\text{inst}(m)$ which refer to the actual occurrence of a business process. Hence, if we consider a car production pro-
cess, an instance of this process refers to the production of a specific car with concrete attributes, as for instance the color *black* and the serial number *123*. On layer M1 we allocate the original process model and one of its possible abstraction models. Although the abstraction is derived from the original model and provides less details, both models are still referring to the same set of instances. The original model $m$ and its abstraction $m_a$ are conforming to the modeling notation they are described with - the metamodel $n$. Important to notice is that both, $m$ and $m_a$ need to conform to the same metamodel. Hence, if the original model is modeled using an EPC, its abstraction model must adhere to an EPC as well.

In order to obtain an abstraction $m_a \in \text{abstr}(m)$ from an original model $m$, we introduce two basic abstraction operations: The elimination ($\pi$) and the aggregation ($\sigma$) [SRNW10].

A basic abstraction operation on model $m$ is an elimination $\pi : M \rightarrow M$, if the abstraction model $m_a$ does not provide any information about the abstraction object $o$, while all other model elements are preserved. Figure 4 depicts the effect of the elimination operation on a given process model fragment $m$. As a consequence of the operation, the abstraction object activity $B$ is omitted in the abstraction $m_a$ of the model fragment $m$.

As opposed to the elimination, aggregation preserves the information about all model elements including the abstraction object $o$. Hence, a basic abstraction operation is an aggregation $\sigma : M \rightarrow M$, if the abstraction object $o$, together with other abstraction

![Figure 4: Illustration of Elimination](image-url)
objects, is replaced with a newly created abstraction object $o'$. Thereby, object $o'$ inherits the properties of all objects it aggregates. Figure 5 depicts the aggregation operation on a given process model fragment $m$. The abstraction object given by the activities $B$ and $C$ are replaced with the new activity $BC$.

![Figure 5: Illustration of Aggregation](image)

Relating the introduced basic abstraction operations to the labeling task we aim to automate, we can add the following thoughts. Independent from the three use cases we sketched in the introduction, we can state that the labeling task does structurally always refer to a situation, where a set of activities is collapsed into one single activity. It is important to notice that this scenario equally applies for all of the mentioned use cases. In the abstraction case only a few activities are affected, but still all of them are aggregated. In the modularization case a whole subprocess is collapsed into one activity. Accordingly, the naming of a whole process model can be analogously considered as the aggregation of all activities into one. Hence, from a structural perspective we consider the labeling task as a so called full aggregation of the affected activities. Figure 6 illustrates this operation in line with the previous figures. The remaining task of finding an appropriate label for this single activity, will be discussed in the course of this thesis.

Shifting the focus from the abstraction operations to the application of BPMA in practice, a wide range of use cases can be sketched. In [SRNW10] different use cases of BPMA are developed and classified into four groups. These groups include use cases such as filtering process activities depending on the purpose, preserving relevant process
instances and the creation of a process quick view. Especially the latter one is a very prominent use case [BRB07, EG08].

In order to implement such use cases, a huge variety of abstraction approaches exist. In earlier research, reduction rules for model simplification were already proposed for Petri nets [Mur89]. Later on, such reduction rules were also defined for workflow graphs [SO00], EPCs [vDJVVvdA07] and YAWL [WVvdA+09]. In recent research, a further very basic approach for EPCs is presented by Polyvyanyy et al. [PSW08b]. They introduce abstraction solutions for elementary patterns such as sequence or blocks. In another work Polyvyanyy et al. [PSW08a] propose an abstraction approach where an activity is aggregated with another activity if a particular abstraction criterion, as for instance the relative effort of an activity, falls below a certain threshold. By varying this threshold using a slider, the degree of aggregation can be adjusted to the current needs. In a very recent paper Smirnov et al. [SWM10] present an abstraction approach based on behavioral profiles. In their approach, a modeler selects those activity groups which shall be aggregated. Subsequently, the control flow dependencies between the activities are automatically derived and employed for constructing the abstract process model.

Although all these papers address the problem how an abstracted model can be structurally derived from a given original, all these approaches miss the computation of according labels for aggregated activities. If two activities are merged, the simple connection of both labels might be a solution. However, if three or more labels are aggregated, this
BACKGROUND

approach does not yield a satisfying result. One approach which is able to circumvent this problem, is the abstraction approach presented by Smirnov et al. [SDMW10]. In their approach the meronymy relationship between activities is utilized to find suitable aggregation candidates. Nevertheless, the labeling problem remains for the majority of the presented approaches.

Before we start to develop our semi-automated labeling approach, the next section will introduce the required concepts from the field of linguistics. These concepts are of crucial importance as they provide insights into the structure of language and hence provide an important basis for the development of our labeling approach.

2.5. Linguistics

As the labeling approach developed in this thesis builds on textual information captured in process model labels, it is essential to be familiar with two domains. On the one hand, it is necessary to know about linguistic and semantic relations which exist between words, as these relations are significant for the development of the approach and also for the analysis in section 3. On the other hand, we require tools which are capable of detecting these relations in an automated fashion. For this reason, this section provides an overview about both areas in the following subsections.

2.5.1. Introduction to Linguistics

Linguistics is the scientific study of the human language with all its facets such as structure, usage and history [FHC01, p. 3]. In many cases, humans are not aware of the knowledge areas which are required for conducting a conversation. However, when it comes to enable computers to analyze or even recognize the human language, we need to be acquainted with all these knowledge domains. As an example, assume somebody says a simple sentence such as ”Order the goods, please.”. To be able to understand this spoken sentence we need to know about phonetics and phonology, as we have to assign a set of words to what we hear acoustically. Once we have mapped the acoustic signal to a
set of words, we need to interpret these words in a correct manner. Hence, to group and order the words, knowledge about *syntax* is required. To interpret the verb *order* in the sense of *creating a purchase order* and not in the sense of *arranging goods in the correct order*, requires knowledge about the meaning of words, also referred to as *semantics*. In addition, to recognize that *goods* is the plural of the noun *good* we need to know about morphology. The usage of the word *please* for the purpose of being polite derives from our knowledge about *pragmatics*. Summarizing the discussed aspects, Jurafski et al. classify all facets of knowledge which are required to engage in complex language behavior into six categories [JM08, pp. 3-4]:

- **Phonetics and Phonology**: Knowledge of linguistic sounds.
- **Morphology**: Knowledge about meaningful components of words.
- **Syntax**: Knowledge about the structural relationships between words.
- **Semantics**: Knowledge about meaning.
- **Pragmatics**: Knowledge about the relationship between the meaning and the goals and intention of the speaker.
- **Discourse**: Knowledge about linguistic units larger than a single utterance.

### 2.5.2. Relevance of Linguistics for Business Process Modeling

The role of natural language in process models is of crucial importance. As illustrated in Subsection 2.2.2, business process models represent a formal specification, describing how certain aspects of a domain are related. These aspects, as for instance activities, are themselves described by natural language in labels. Hence, natural language is one important mean of business process models for describing the domain. Moreover, the natural language in the labels significantly contributes to the overall understanding of the process model. In an experiment Mendling et al. [MRR09] showed that the use of different language structures amongst labels lead to different levels of understanding and that
clear and unambiguous labels are of significant importance for the model understanding. Because of the importance of natural language in business process models, we investigate linguistic concepts which are important for our purposes and employ them in our automated labeling approach. The following subsections will introduce the linguistic domains used in this thesis.

2.5.3. Lexical Semantics

Lexical semantics is a branch of linguistic semantics and deals with the study of the meaning and relation among words [JM08, p. 611]. The field of lexical semantics is of crucial importance for this thesis as the relations between words are one major source of the presented labeling approach.

When speaking about language, we often use the term word. The problem attached to this term is that it is used in various different contexts and it is, hence, hard to disambiguate its different meanings [JM08, p. 611]. Thus, in order to precisely describe the concepts of lexical semantics, we need to employ more specific terms. Therefore, we introduce the following concepts:

- Lexeme,
- word form,
- and lemma.

The most significant term is the notion of a lexeme. According to Lipka a lexeme can be defined as an abstract unit of the language system representing a group of different word forms [Lip02, p. 89]. Particularly for the purpose of dictionaries, a lexeme is represented by a citation form which is also referred to as lemma [JM08, pp. 611-612].

For illustrating the introduced terms consider the verbs: do, does, doing, did and done. Applying the given definitions, these verbs are different word forms of a lexeme which is represented by the lemma do. The word forms associated with the lexeme are
derived via inflection of the lemma. This means a word can, but not necessarily has to, carry an inflectional ending such as \textit{ing} in \textit{doing}. As already mentioned, the lemma can be particularly found in dictionaries used as key word for representing the lexeme. The word form used for the lemma is in many languages given by the infinitive of the verb or the singular form of the noun. The process of mapping a word form to the according lemma is called \textit{lemmatization} [JM08, p. 611]. The problem attached with this mapping is that the lemmatization is a non-deterministic process. As a consequence, depending on the context the lemma of the word form \textit{found} could be \textit{find}, in the sense of to locate something, but also \textit{found} if the context is about setting up a company. In addition, lemmatization is part of speech-specific. For instance, the word form \textit{tests} may either refer to the lemma of the verb \textit{test} since \textit{tests} refers to the third person form of the verb, or \textit{tests} may refer to the plural form of the noun \textit{test}.

Having introduced the basic terms of lexical semantics, we can turn to the meaning component of lexemes, which is also referred to as \textit{sense} [JM08, p. 612]. The complexity of this field is for example emphasized by the noun \textit{race}. Depending on the context it may be used in the sense of a taxonomic group that describes the division of a species in biology. Nevertheless, \textit{race} could also be used to describe a contest of speed using cars. This characteristic of a lexeme having multiple distinct unrelated senses is called \textit{homonymy} [PB96, p. 2]. Thereby, the unrelatedness of the senses is a decisive criterion. The feature of a lexeme having multiple related senses is referred to as \textit{Polysemy}. For instance, the lexeme \textit{wood} may be used in the sense of a piece of a tree or in the sense of a geographical area with many trees. However, both senses have the same historic origin and are thus, polysemes and not homonyms.

Besides the homonymy and polysemy relationship, there also exist several other lexical relations, which are significant for this thesis. Therefore, we will discuss the lexical relations synonymy, hyponymy and meronymy in the following paragraphs.

A very common definition for the \textit{synonymy} relation is that two different lexemes are
synonyms if both lexemes have the same meaning [JM08, p. 615]. However, the resulting question is, when do two words actually have the same meaning. One approach is to require the words to be mutually substitutable without changing the meaning of a given sentence. Nevertheless, of course it is always possible to find an example where this does not work. For instance, consider the words *pupil* and *student*. Although it is obvious that both words can be used interchangeably, it is also conspicuous that this does not work in every case. For example, a pupil who is going to school can also be called a student, but a student going to university will not be referred to as a pupil. Hence, the definition was extended [JM08, pp. 615-616]. Accordingly, two words are synonyms if they are substitutable in at least one context. Consequently, as *pupil* and *student* are substitutable in at least one case, they are considered to be proper synonyms.

If two lexemes are closely related in the sense that one lexeme is a sub class of the other, this relationship is called *hyponymy* [JM08, p. 616]. As an example consider the lexemes *car* and *vehicle*. As *car* is a more specific term for a *vehicle*, we observe a hyponymy relationship between both lexemes. As hyponymy is an asymmetric relationship we call the more specific lexeme a hyponym and the more general term a hypernym. Hence, *car* is a hyponym of *vehicle* and *vehicle* is a hypernym of *car*.

A more specific relation is given by the part-whole relationship which is called *meronymy* [CC04, pp. 159-160]. For instance, a *finger* is part of a *hand*. As this relationship is also asymmetric, a *finger* is a meronym of *hand* and *hand* is a holonym of *finger*.

Having explored which basic relations can be found among words and their senses, the next subsection illustrates how these relationships can be detected in an automated manner.

2.5.4. Natural Language Processing

Natural Language Processing (NLP) is a research field that studies how systems can be enabled to analyze, understand and produce the human language [All93]. NLP is a highly interdisciplinary research area as its foundation lies in various disciplines such as computer
science, linguistics, logic, artificial intelligence and psychology [Jos91]. The application fields of NLP are diverse and can for example be found in natural language text processing [JM02, FAA+94], machine translation [Hut86, BCDP+90] and speech recognition [RJ93, Jel98].

However, for the purpose of this work we focus on the field of natural language text processing as this domain provides the methods which we require for our analysis and our final approach. In order to automatically analyze text, several tools exist, each focussing on different aspects. For instance the Stanford Tagger [TM00, TKMS03] attempts to assign the according part of speech to each word in a given text. The Stanford Parser [KM03a, KM03b] additionally determines the underlying structure of sentences. Nevertheless, as shown in [LSM09] the analysis of business process model activities using the Stanford tools turned out to be problematic. Since activities in process models tend to be rather short and usually not form complete sentences (consider for instance the activity Printing Notification from the SAP Reference Model), the analysis using natural language processing tools was not very fruitful. Hence, in order to analyze the relations between activities in process models, we will employ the lexical database WordNet [Mil95]. The following paragraphs will briefly introduce the concept of WordNet and how WordNet can be accessed using a Java program.

_Concept of WordNet._ WordNet is a lexical database for the English language developed at the Princeton University [Fel98]. WordNet organizes nouns, verbs, adjectives and adverbs into logical groups called _synsets_. Each synset consists of a set of synonymous words or collocations (a combination of words with a specific meaning such as _fast food_ or _think tank_). As one single word may have various meanings in different contexts, each synset only comprises words or collocations having the same meaning in a particular context and are thus interchangeable. As a result, for each sense of a given word, one synset exists.

As an example for the synset concept, consider the noun _shot_. According to WordNet there exist 17 different senses and thus 17 synsets for this noun. Two of these 17 synsets
representing two particular senses of shot are \{shot, pellet\} and \{shot, injection\}. Listing 1 shows the WordNet entries for these synsets. Besides the elements of the synset and the according part of speech of the index word shot, WordNet provides a dictionary like definition called gloss. This gloss contains an explanation and an example for the use of the synset elements. From these entries we can learn that shot might be replaced with pellet if shot is used in the sense of a solid missile and that it can be replaced with injection if shot is used in the context of a vaccination.

The most significant and remarkable feature of WordNet is the existence of lexical and semantic relations as introduced in 2.5.3. As semantic relations such as hyponymy or meronymy are relations among the meaning of words, they are defined between synsets. By contrast, relations like synonymy and antonymy are defined between word forms.

Accessing WordNet. The WordNet database is freely available and accessible in various ways. One of the easiest possibilities is the use of the online interface provided by the University of Princeton\(^1\). However, this online interface is not sufficient as we need to directly access and evaluate the WordNet database contents in our own program code. Therefore, a more flexible option is given by downloading the WordNet database and accessing it using a programming language. For this purpose, a large set of application programming

\(^1\)http://wordnetweb.princeton.edu/perl/webwn
interfaces (APIs) for different programming languages are provided on the web. As the analytical tools and the prototype of this thesis will be implemented with Java, Listing 2 demonstrates how the WordNet database can be accessed using the Java Wordnet Library (JWNL). JWNL is a freely available WordNet API which provides detailed functionality to access all required contents of the WordNet database [DW10].

In the beginning, WordNet is initialized using the property file (lines 2-3). Line 6 illustrates how an arbitrary word can be looked up in WordNet. Using the function `getIndexWord`, we can look up any combination of word form and part of speech. If WordNet provides a lexeme for this combination, it is returned using an `IndexWord` object. If WordNet does not contain a lexeme referring to such a word form, the returned object will be `null`. As already pointed out, for each sense of a given lexeme, WordNet provides one synset. Using the function `getSenses`, these synsets can be acquired and are provided in the form of an array (line 7). After we obtained one particular synset (line 8), we can also derive lexical relations for this lexeme-sense combination. For example, with the help

```java
// Initialize JWNL and create Dictionary object
JWNL.initialize(new FileInputStream("src/file_properties.xml"));
Dictionary wordnet = Dictionary.getInstance();

// Define word which is looked up
IndexWord word = wordnet.getIndexWord(POS.NOUN,"invoice");
Synset[] senses = word.getSenses();
Synset firstSense = senses[0];

// Get list of direct hypernyms
PointerTargetNodeList directHypernyms = PointerUtils.getInstance().
    getDirectHypernyms(firstSense);

// Go through list of hypernyms and print each to standard out
Iterator<PointerTargetNode> iterator = directHypernyms.iterator();
while (iterator.hasNext()) {
    Synset directHypernym = iterator.next().getSynset();
}
```

Listing 2: Usage of JWNL
of the class `PointerUtils` we can ask for the direct hypernyms which are returned in a list of synsets, the `PointerTargetNodeList` (lines 11-12). By iterating over this list, we can access each hypernym synset accordingly.

In this section we investigated how the different relations among words can be detected in an automated fashion. In this vein, we demonstrated how the concepts from the lexical semantics can be derived using WordNet and Java. Before we turn to the analysis section, the following subsection closes the background part by providing a summary on related work in the area of NLP in business process models.

2.6. Related Work of NLP in Business Process Models

The application of natural language processing techniques on process models can be found in different streams of business process modeling.

One important area is the similarity measurement between business process models. For instance, Ehrig et al. propose an approach for measuring the similarity between business process models, modeled with the Web Ontology Language (OWL), in order to support model interconnectivity and interoperability [EKO07]. To accomplish this goal, they introduce different similarity measures. Besides metrics concerned with structural and syntactic quality, they also measure the linguistic similarity by employing WordNet to detect synonyms and homonyms of process labels. Another approach is proposed by Koschmider and Oberweis in [KO07], who pursue the goal of facilitating process reuse. They introduce an algorithm for semi-automatically detecting process variants by determining linguistic similarities between process model labels using WordNet. An algorithm which is also building on similarity measurement, is proposed by Awad et al. in [APW08]. They present an automated approach for querying a business process model repository. For this purpose a user formulates a BPMN-Q query and subsequently receives a list of relevant models based on structural and semantical aspects. Thereby, the measurement of the semantic similarity is conducted with the help of WordNet.

Another widely discussed stream of business process modeling is the semantic annota-
tion of business process models [BDW07, FRST09, LD05]. However, former approaches did either rely on existing ontologies or they did not employ natural language techniques to construct them. By contrast, in recent research Francescomarino and Tonella present an automated technique to support the domain ontology creation and the semantic annotation of BPMN process models. Therefore, they apply natural language processing to process labels in order construct a domain ontology. In addition, semantic annotations are suggested to the modeler in an automated fashion [FT09].

A quite recent endeavor is the derivation of process models from natural language descriptions. For instance, in [dAGSB09, dAGSB10] de A.R. Gonçalves et al. propose an approach for automatically generating process models from group stories. For this purpose, people write down the activities which describe their work. By using the techniques of text mining and natural language interpretation, BPMN models are created from these texts. However, the examples illustrated in [dAGSB09] indicate that only the basic notation elements are used. An alternative is proposed by the research team from the university of Klagenfurt, who developed a tool [FKM+05, KVH+05] that combines linguistic analysis techniques and the consultation of users, to transform textual requirement specifications into a meta-model called Klagenfurt Conceptual Predesign Model (KCPM) [MK98]. Using this model, different conceptual models as UML activity diagrams [SCK04] and UML class diagrams [FKM+07] can be created. Another approach is presented by Ghose et al. in [GKC07b, GKC07a]. They propose the Rapid Business Process Discovery (R-BPD) framework, which employs natural language analysis techniques for constructing BPMN models from heterogeneous information resources, such as corporate documentations or web-content. However, their focus is on providing models that can be incrementally adjusted and corrected by analysts. Hence, the results rather represent model snippets than complete process models.

A last application of NLP techniques can be found in different settings of process model quality considerations. For instance, Friedrich [Fri09] introduces different quality metrics
utilizing information from WordNet. With the goal of minimizing misunderstandings, the
metrics reflect and penalize the ambiguous usage of synonyms and words from different
levels of specificity. Similar considerations have been undertaken by our previous research
in [LSM09]. In these papers we tried to automatically determine the structural quality
of process labels with the use of different NLP techniques. Based on insights about
label structures, we also provided a refactoring approach for process model activity labels
[LSM10].
3. Derivation and Classification of Manual Naming Strategies

Due to the high complexity and huge size of real world business processes, many of them contain abstraction levels, and hence provide labels for the abstracted subprocesses. In this section we utilize this fact by conducting an analysis on existing process model collections. The goal is to find linguistic and semantic relations between the process name and the process elements in order to learn how existing labeling approaches can be included in the development of our approach. In this context, three different real world process model collections are investigated.

The section is organized as follows. In the first subsection the analyzed model collections are introduced. Subsequently, we explain how the analysis is conducted from a methodical point of view. Finally, the results are presented and the implications are discussed.

3.1. Analyzed Process Model Collections

In order to detect structures which do not only reflect the setup of one particular process model collection, we decided to employ three different model collections for our analysis:

- **SAP Reference Model**: The SAP Reference Model is a model collection capturing the business processes supported by the SAP R/3 system in its version from the year 2000 [KT98, pp. 145-164]. The collection contains in total 604 Event-driven Process Chains (EPCs) organized in 29 functional branches of an enterprise such as sales, accounting and other areas. The total number of activity labels amounts to 2433. Although not every model in the SAP Reference Model includes subprocesses, each EPC carries a name describing the content of the model. Hence, according to the process naming use case sketched in the introduction, we investigate the relation between the process name and the contained activities.
• **Model Collection from Utility Vehicle Manufacturer**: The model collection from the utility vehicle manufacturer contains process models from the procurement domain. It consists of one main procurement process containing nine subprocesses which include altogether 115 activity labels. The models are provided in the format of the Protos tool which is a modeling and analysis tool developed by Pallas Athena [VvHRdM05]. The models are similar to EPCs and thus contain events and activities, but do not adhere to the strict alternation of events and activities. However, the models are suitable for the purpose of our analysis of the relations between the name of the process and its activities.

• **Model Collection from IT Service Provider**: The third model collection contains the incident management process from an IT service provider. The process is depicted using an EPC and consists of three abstraction layers, contains 88 subprocesses and include in total 293 activity labels. As opposed to the other model collections, the model elements are labeled in German. As WordNet only includes entries for the English language, we would have to consult Germanet, an according lexical dictionary for the German language. However, as GermaNet is not freely available, we analyze the processes manually.

While the SAP Reference Model is a freely available model collection capturing processes from different branches, the other model collections reflect specific processes from practice partners which are restricted in terms of disclosure. For this reason, we will not present any models from the latter two collections. However, we include these models in our analysis, as we aim to cover different types of model collections in order to achieve substantial results. Furthermore, the comparison between the different model collections may also help to argue about the relevance of a detected labeling strategy.

The next section will continue by introducing the methodology of the analysis. Please note, that in the rest of the thesis the term *process model name* is used to refer to the label which was defined for describing a set of activities as for instance a subprocess.
This includes *real* activities which are linked to a subprocess, but also the names of the processes of the SAP Reference Model which are only interpreted as such. Consequently, the labeling of a set of activities will be also referred to as *naming* of a process model.

### 3.2. Methodology of the Analysis

In order to decide how the process model names were derived from the contained activities, different steps are required. In general, the following steps are performed in the context of the analysis:

2. Automated search for matches and linguistic/semantic relations.
3. Manual investigation how the model name was derived.

In the following paragraphs we will introduce each step in detail.

*Annotation of Process Model Names.* As process model names represent the label of an activity or can at least be interpreted accordingly, they should include an action and a business object to which the action refers. Hence, in order to process this information, the first step consists of the annotation of all process model names with action and business object. In order to avoid derivation errors caused by automated approaches, this step is performed manually. As a result, we are able to precisely determine whether a given word corresponds to the action or to the business object of the process model name and can thus draw accurate conclusions.

*Automated Search for Matches and Relations.* In order to identify the matches and relations between the process model name and its activities and events, a Java analysis tool was developed. After the process model has been imported from a file in the Extensible Markup Language (XML) format to our Java process model data structure, this tool consults WordNet to implement the comparison between words and the detection of linguistic and semantic relations.
Algorithm 1 Determination of Matches and Relations

1: findMatchesAndRelations(ProcessModelCollection modelCollection)
2: matches = new List();
3: for all models m in modelCollection do
4:   modelA = WordNet.getLemma(m.getName().getAction());
5:   modelBO = WordNet.getLemma(m.getName().getBusinessObject());
6: for all activities and events element in m do
7:   elementA = WordNet.getLemma(element.getAction());
8:   elementBO = WordNet.getLemma(element.getBusinessObject());
9: if WordNet.hasRelation(elementA, modelA, "Verb") then
10:   relType = WordNet.getRelationType(elementA, modelA);
11:   matches.add(new Match(element, elementA, relType));
12: if WordNet.hasRelation(elementBO, modelBO,"Noun") then
13:   relType = WordNet.getRelationType(elementBO, modelBO);
14:   matches.add(new Match(m, element, elementBO, relType));
15: return matches;

Algorithm 1 illustrates the basic steps using pseudo code. The algorithm expects a model collection and returns a list of matches between the model name and the element of the process. Thereby, each match object contains the affected activity or event, the according model and the type of relationship. Relationship types may include a complete match, synonymy, holonymy and hypernymy. As a model collection contains various models, all models are analyzed in succession (line 3). For each model the according action and business object are stored in the variables modelA and modelBO (lines 4-5). Important to notice is, that not the words themselves, but their lemmata are saved. This ensures that the comparison between two words, provided in different word forms, will still result in a complete match. For instance Analysis and analyze may refer to the same verb with the lemma analyze. After having derived action and business object from the model name, all activities and events in the model are considered. First of all, analogously to the model name, the lemmata of action and business object are stored in variables (lines 7-8).
3 DERIVATION AND CLASSIFICATION OF MANUAL NAMING STRATEGIES

Subsequently, WordNet is consulted for identifying the relations between the action and business object of the model name and its elements (lines 9-14). The distinction between action and business object is necessary, as WordNet requires the information about the part of speech of the parameters of the relation. If a potential relation was identified, all necessary information about the relation is stored in a match object which is then appended to the match list (lines 11,14). After all models and all their elements were inspected, the match list is returned (line 15).

Manual Investigation of Model Name Derivation. To obtain a fundamental outcome from the analysis, we decided to additionally conduct a manual inspection of each model using its graphical representation. Although, in some cases the matches of the automated analysis clearly indicated how the model name was determined, other models required further investigation. One example for an obvious naming strategy can be found in the SAP model *Purchase Requisition*. Each activity included in the model contains the business object *Purchase Requisition*. However, in order to not miss other useful but maybe less obvious patterns, we manually inspected each model.

By evaluating the matches and relations identified by Algorithm 1 with the graphical representation of the model, we determined for each model how its name could be explained using the contained activities. After we finished the analysis, we clustered these labeling strategies to a set of six basic approaches. The next subsection will present a detailed overview about each of the identified labeling strategies.

3.3. Resulting Classification from Analysis

As indicated in the last subsection, in the context of the manual analysis we observed different labeling concepts which we classified into six basic approaches. The following paragraphs introduce each concept in detail and provide an example from the analyzed model collections for illustration purposes.
3 DERIVATION AND CLASSIFICATION OF MANUAL NAMING STRATEGIES

_Dominating Action or Business Object._ In most activities the action is not given in isolation, but is performed on a business object. If one particular business object is used more often among all activities of the process than the other business objects, it is considered to be the _dominating business object_ of the process. In the analyzed process model collections the dominating business object was often used for naming the process model if such a dominating business object existed. As an example, consider the SAP process model _Purchase Requisition_ depicted in Figure 7. The business object _Purchase Requisition_ is used both activities and also in three of five events. Hence, the name _Purchase Requisition_ of the process model can be explained with the concept of the dominating business object.

Analogously to the concept of the dominating business object, there also exist process models where one particular action is performed more often than all other actions. Consequently, we refer to this action as _dominating action._ Similarly to the dominating business objects, the dominating action is used for naming the process model.

_Main Activity._ In some processes there is one particular activity which is of fundamental importance for the process. By contrast, the remaining activities have the character of side activities supporting, preparing or evaluating the result of the main activity. Consequently, this main activity is used for naming the process model. An illustrating example can be found in the SAP process model _Change in Material Price_ depicted in Figure 8. Besides the change of the material price, this process also contains an activity which is concerned with activating the future material price. However, from the labeling behavior of the modeler we can assume that this activity only plays a subordinate role while the focus is on the activity of changing the material price.

_Start or End Events._ Especially when the state at the beginning or at the end of process are of capital importance, the name of the process model may capture the content of the start or end event. For instance, the end event of the SAP process model _Date Planning_, depicted in Figure 9, states _Date planning is performed_. As the accomplishment of the date planning is obviously the outcome of the process, it is consequently used for naming
Analogously to end events, this labeling practice can also be applied using start events. For example the SAP process model *Budget Planning* contains the start event *Business must be planned*. Hence, the start event states what is going to happen during the process, what is consequently used for naming the process model.

**Only Existing Activity.** A significant share of the analyzed processes or subprocesses only comprised one single activity. In a vast majority of these cases the process was also named according to this single activity while the surrounding events were ignored. One example of such a process can be found in the SAP Processes *Backorder Processing* depicted in Figure 10. The process contains four events and only one activity. Although the events capture useful information, as for instance that there are two outcomes of the missing parts situation, this is not captured in the process name and the label of the only existing
activity is adapted.

Conjunction of Activities. If the same action is performed on different business objects or different actions are applied on the same business object, these activities can be easily described by connecting the different actions or business objects. Alternatively, whole activity labels may be connected if the resulting name is not too complex. An example for the conjunction concept can be found in the SAP model Customer Inquiry and Quotation Processing depicted in Figure 11. In this model the action Processing is performed on two business objects: Customer Inquiry and Customer Quotation which is also referred to as Quotation. Considering the process name, we can conclude that the process name was created using the conjunction of the business objects Customer Inquiry and Customer Quotation in connection with the action Processing.

Semantic Naming. The previously introduced concepts always explicitly refer to the textual description of at least one element in the process model. By contrast, the concept of semantic naming does not refer to one or more model elements, but uses the broader context of the activities for naming the process model. In particular, we can think of a part-of relationship between the activities and the given process name, as for instance
described in [RLvdA03]. Hence, the process name, which is itself representing an activity, consists of the given activities in the model and is therefore employed for the naming task. As an example, consider the SAP Process Shipping depicted in Figure 12. It consists of five events and the two activities Delivery for Returns and Goods Receipt Processing for Returns. Apparently, the action shipping is not mentioned in any of the process elements. Nevertheless, when assigning the process steps to a superordinate concept, we can argue that the term shipping contains of the stated activities and may hence be used to describe the contents of this process model on a semantic level.

Concept Combinations. In some cases two of the introduced concepts were combined in order to find a name for the process model. For example, in the process Invoice Verification, depicted in Figure 13, the business object Invoice is referenced in each activity and thus represents the dominating business object. By contrast, the action verify is not
mentioned at all. The actions performed on the business object Invoice include actions such as process and release. Inspecting the model, it becomes apparent that the action verify was derived from the semantics of the model. Hence, the concept of the dominating business object was combined with semantic naming. Another example of a combination approach is the usage of a dominating action and a dominating business object. For instance in the model Cost Planning each of the three activities contains the action plan and the business object cost.
3.4. Towards a Theoretical Foundation for the Identified Naming Strategies

In the last subsection we detected and classified different labeling approaches based on linguistic or pragmatic relations between the name and the elements of the process model. However, this classification lacks a theoretical foundation for the applied approaches. Hence, this subsection aims at supporting the identified approaches by connecting them with different Theories of Meanings. A Theory of Meaning can be considered as a concept which tries to determine the meaning of a term or sentence from the natural language [Dum96, pp. 1-3]. Such theories have a long history in philosophy, psychology and other related disciplines and a huge number of them have been developed during this period [Eve08].

The arising question is, how we can apply theories which are concerned with the meaning of words to process model labeling approaches. In order to answer this question we return to the labeling task itself. In the initial situation, a modeler is confronted with a process model that is to be replaced with a single activity. As process models are usually created with the intention of being understandable for many people, we can assume that the modeler aims at finding an activity which describes the underlying process model in an intuitive and understandable fashion. This is the case if the activity adequately describes the semantics or, if we use the terms of the Theory of Meaning, the meaning of the process model. If this meaning is subsequently verbalized into an activity label, the labeling task is accomplished. As a result, we obtain an activity label by implicitly or explicitly applying different Theories of Meaning on a process model.

The following paragraphs introduce different Theories of Meaning and their relation to the introduced labeling approaches from section 3.3. Although a large number of theories has been developed, we will only present those theories which are relevant for our purposes.

Early Pragmatist Theory of Meaning. Pragmatist theories focus on the ability of words to create effects in the world [Eve08]. The Early Pragmatist Theory [Wit09, pp. 5-
DERIVATION AND CLASSIFICATION OF MANUAL NAMING STRATEGIES

10][Dew58, pp. 184-186] focusses on the observable effects of words and sentences in the world. For instance, a stop sign is considered to be meaningful, because of the effect it creates.

If we apply the Early Pragmatist Theory on process models, this means that we focus on the observable effects of the execution of a process model. Hence, we are basically concerned with changes of states in the real world as a result of the process model execution. As the last events of process models often explicitly capture the effects of the process steps by providing information about the final state, we can draw a connection between the Last Event labeling approach and the Early Pragmatist Theory. In both approaches the meaning of the underlying concept is derived based on the observable effects.

Late Pragmatist Theory of Meaning. As opposed to the early pragmatist theory, the late pragmatist theory [Aus75, pp. 67-82], [Gri89, pp. 213-224] states that the meaning of a word or phrase is not its actual effect, but its intended effect in the real world [Eve08]. For instance, the meaning of the phrase take it can be determined by the intention this phrase represents.

The Late Pragmatist Theory can be applied to process models in a similar fashion as the Early Pragmatist Theory. As a result, we focus on the intended effect of the execution of a process model. As the intention of the steps of a process model is often captured in its start events, we can analogously draw a connection between the Late Pragmatist Theory and the Start Event labeling approach. In both cases the meaning of the underlying concept is derived by its intended effect.

Prototype Theory of Meaning. According to the Prototype Theory the Meaning of a term is given by that instance which possesses most of the typical characteristics [Ros75a, Ros75b]. It is based on the observation that some instances of a concept are better representatives than others. Rosh investigated this field by asking students to assess how well they consider 60 different items to fit into the category furniture [Ros75a]. As a result the word chair was voted to fit best, while the word bed was on position 13 and
Applied to the labeling approaches, we can create a connection between the Prototype Theory and the Main Activity approach. As in the Main Activity approach a single activity is chosen to represent the whole process, we may assume that this activity captures most of the typical features of the this process. As for instance in the Main Activity example, the process named *Change in Material Price* contains different types of activities. However, the activity *Change in Material Price* was chosen for describing the contents of the process model. Although the term *instance* from the Prototype Theory is not applicable, as we do not refer to a process instance, but to a specific process activity, the pattern of choosing a particular part to represent a whole concept remains. As a result, we can support the Main Activity Approach by the Prototype Theory.

**Feature Theory of Meaning.** The philosopher Gottlob Frege suggests that the meaning of a word or phrase is given by its *sense* [Fre92]. Thereby, the term *sense* was further defined by Rusell as a function of logical operators, predicates and referents a phrase is composed of [Rus05, Rus56]. In more general terms, we can define a concept using a set of necessary attributes which sufficiently describe the underlying concept. For example, a bird may be characterized by having wings, feathers, its ability to fly and to lay eggs etc. [Eve08].

To some extent this theory of meaning can be related to the *Dominating Element* approaches. If we apply this theory on the process model domain in a very stringent manner, a process model would acquire its meaning through those activities which sufficiently describe its overall semantics. Assuming that this applies for those activities containing the dominating element, we can draw a connection to the Feature Theory of Meaning. Of course, the assumption that the activities containing the dominating element are more significant than the others, may not be applicable in every case. However, if the dominating element was actually used for naming the process, its importance cannot be denied.
Another relation of the Feature Theory of Meaning can be identified to the *Conjunction* approach. As multiple activities are connected in order to describe the process, we can assume that these activities sufficiently reflect the contents of the process. As a result, the Feature Theory of Meaning could also be mapped to the *Conjunction* approach.

*Knowledge-based Theory of Meaning.* According to Knowledge-based Theories of Meaning a term acquires its meaning through relations to other words or statements which are captured in an underlying concept [Eve08]. Thus, for instance the meaning of *water* is provided by chemists who define, that something which is water must definitely consist of hydrogen and oxygen.

Mapping this concept to the introduced labeling approaches, we can draw a connection to the semantic naming approach. If the activities of a process model can be associated with one underlying concept, this concept may adequately represent the meaning of the process model. For example, if all activities relate to the maintenance of a particular business object, the term *maintenance* may serve as a suitable description, although the maintenance itself is never mentioned. This scenario is quite similar to a case where we had to assign a concept to the elements hydrogen and oxygen, whose combination could properly be described with water.

Table 1 summarizes the mapping considerations of this subsection. The only labeling
strategy which is missing, is given by the concept *Only Existing Activity*. This is caused by the fact that using the only activity for naming a process model is rather a pragmatic procedure than a sophisticated labeling approach deriving from the meaning of a process model.

### 3.5. Summary and Implications

The structure analysis on the three process model collections uncovered existing relations between the name of a process model and its elements. Although it is not possible to conclude that these concepts have been consciously applied while the processes were created, these relations provide a valuable basis for the automatization of the task of naming process models.

Table 2 summarizes the frequency of each concept among the analyzed process model collections. In addition to the basic approaches, the table also includes the concept combinations. In the table header the SAP Reference Model is denoted with *SAP*, the

<table>
<thead>
<tr>
<th>Naming Strategy</th>
<th>Total</th>
<th>SAP</th>
<th>UVM</th>
<th>ITSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dominating Business Object</td>
<td>66</td>
<td>64</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Dominating Action</td>
<td>28</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Main Activity</td>
<td>60</td>
<td>54</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Start Event</td>
<td>30</td>
<td>29</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>End Event</td>
<td>50</td>
<td>26</td>
<td>3</td>
<td>21</td>
</tr>
<tr>
<td>Only existing Activity</td>
<td>119</td>
<td>76</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>Conjunction</td>
<td>11</td>
<td>9</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Semantic Naming</td>
<td>224</td>
<td>211</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>Dominating Action + Dominating Business Object</td>
<td>32</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dominating Action + Semantic Naming</td>
<td>27</td>
<td>27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dominating Business Object + Semantic Naming</td>
<td>54</td>
<td>48</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td><strong>Sum</strong></td>
<td><strong>701</strong></td>
<td><strong>604</strong></td>
<td><strong>9</strong></td>
<td><strong>88</strong></td>
</tr>
</tbody>
</table>
models from the utility vehicle manufacturer with UVM and the models from the IT service provider with ITSP. Considering the results, we can state the following. Out of the 701 analyzed models, in 477 cases (68%) the full model name or at least parts of it can be explained with an approach which relies on information which is explicitly given in the model. In particular these labeling strategies can be utilized for developing an approach for automatically naming process models. As the remaining 224 model names (32%) were derived from the semantics of the models, it is hard to develop an algorithm for this procedure. This is especially impeded by the fact that in all these cases WordNet did not provide any relations between the name of the model and its elements. However, the next section builds on this elicited knowledge and develops an approach for the semi-automated naming of process models.
4. Strategy Development for Automated Labeling

In this section we present the development of the approach for semi-automatically naming process models. For this purpose, we build on the labeling strategies conducted by humans, which we identified in the last section. This section starts with a discussion of the requirements for such a labeling approach. Subsequently, we give a general overview about our approach and proceed by introducing each phase in detail. Thereby, particular emphasis is put on the different sub approaches which constitute the main part of the approach.

4.1. Preconditions

As we are aiming for creating an approach, which automatically computes a set of activities, a very crucial aspect is given by the question what we actually consider to be a desired outcome from the perspective of quality. A frequently discussed concept in the context of quality for conceptual models is given by the SEQUEL framework. It has been proposed in different versions by Krogstie et al. in [LSS94, KSJ06, KLS95]. The basic idea of the framework is to evaluate the quality of conceptual models along the three dimensions Syntax, Semantics and Pragmatics. Thereby, Krogstie et al. define relations between the model and the sets domain, language and audience interpretation which reflect the quality. The relations which are of importance for our considerations are given by the pragmatic quality and the semantic quality.

In the context of the SEQUEL framework the pragmatic quality is given by the degree of correspondence between the model and the audience interpretation, i.e. the pragmatic quality reflects to what extent the model has been understood by the reader. The semantic quality defines the degree of correspondence between the model and the domain. Put differently, the semantic quality reflects to what extent the model contains those statements which could be made about the domain. Thereby, the authors focus on the notion of feasibility as a model containing all possible statements would not be easily understandable. Hence, a model is feasible complete if adding a new statement entails more
drawbacks than benefits. Applied to our approach, we can conclude that the computed labels have to fulfill two requirements. First of all, we have to assure that the labels are understandable and that users are capable of interpreting them correctly. Secondly, we need to ensure that a computed label conveys the concepts of the underlying steps as comprehensive as possible. The challenge in this context is, that we only use one label to describe a domain and not a set of statements. Having defined the requirements on a general level, we need to refine these requirements for the purpose of implementing them in our approach. The following paragraphs illustrate how we implement the notion of pragmatic and semantic quality.

Pragmatic Quality. Aiming for the optimization of the pragmatic quality of a label, the resulting question is, which factors actually influence the pragmatic quality, i.e. the understandability of a label. In order to be meaningful, recent research [MRR09] suggests that a label has to contain two elements: an action and a business object the action refers to. As an example consider the label Prepare Business Case. The action is given by the verb prepare which is referring to the business object business case. Some activities also provide some additional information in order to accurately specify the purpose of the activity. For example we could extend the given label to Prepare Business Case for Project. Thus, the phrase for Project increases the specificity of the label. However, for the purpose of computing a label which describes a set of abstracted activities, we consider the elements action and business object to provide sufficient information. Besides the elements of the label, a second important aspect is given by the composition of the elements in the label. In a study on activity labels three main label structures have been identified[MRR09]:

- verb-object labels,
- action-noun labels
- and the rest category.
Verb-object labels such as the exemplary label *Prepare Business Case* used above, provide the action as verb in the beginning of the label. By contrast, in action noun labels like *Printing Notification* the action is captured as a noun. In labels belonging to the rest category no action is provided at all. In a study on the impact of these label styles on the understandability it was uncovered that verb-object labels support the model understanding best, while action-noun and rest category labels caused misunderstandings due to the ambiguity associated with their structure. The problem becomes obvious when considering the exemplary action-noun label *Printing Notification*. It could either advice to print a notification or to notify for a printing job. Without the context of the model, this is not decidable. Applying the findings of former research to our approach, we aim at constructing a verb-object label containing both, action and business object.

*Semantic Quality.* As already pointed out, the semantic quality refers to the expressiveness of the label in regard to the contents it is describing. Due to the fact, that the expressiveness in regard to the semantics is, at least to some extend, a subjective aspect, it is hard to define concrete requirements. Apparently, there is not only one appropriate label for a given set of activities, but several possibilities which describe the underlying contents equally well. However, the requirement for the labels to describe the underlying contents of the process model in a very substantial manner can still be made, although the assessment of this criterion is left to the subjective interpretation of humans.

4.2. Overview about Proposed Approach

The main idea of our labeling approach is to automatically compute a set of potentially useful labels for a given process model. Subsequently, the modeler can decide which of the proposed labels fits best and select the according label. Consequently, the whole procedure of naming a given process model is reduced to a semi-automatic procedure, where the modeler is only involved to select the best proposal.

In the context of the development of this approach we included the quality considerations discussed in subsection 4.1. In order to accomplish the generation of the proposal.
list, we propose a three step approach as depicted in Figure 14. The first phase serves as preparation for the main information extraction. In this phase all activities, start events and end events of the given process model are annotated with their respective action and business object. The second phase contains the main information extraction using different sub approaches to derive meaningful process labels. In this context we explicitly build on the labeling concepts identified in the last section and aim for automating them. As a result, we developed a set of sub approaches, each pursuing a different strategy to generate suitable names for the given process model. While some sub approaches rely on the input of other sub approaches other act completely independently. Finally, in the third phase, this information is employed to create a list of label proposals following the verb-object style.

The following sections introduce each phase of the labeling approach in detail. Due to its significance for the total approach, particular emphasis is put on phase 2 and its sub approaches. Each sub approach will be explained and discussed using an algorithm.
4.3. Phase 1: Annotation

All sub approaches presented in the next section have one commonality: They require information about the performed actions and referenced business objects of the given process model as input. Therefore, it is necessary to automatically derive this information from the model activities and those events which are used in the Event Extraction approach. An algorithm for eliciting action and business object from activity labels was presented in [LSM10]. It is based on the observation that there exist different structure types amongst activity labels. While the action in verb-object labels is per definition always given in the beginning, the structure of action-noun labels is more complex. Nevertheless, it was uncovered that action-noun labels can be classified into five categories, which can be quite easily recognized and directly indicate where action and business object are positioned within the label. Table 3 gives an overview about the identified activity label structures and states an example for each of the types. In the given structure definitions the action is denoted with \( A \), business object with \( BO \), prepositions with \( PP \) and additional information is abbreviated with \( ADD \). Optional elements are stated in round brackets.

While the algorithms building on this structures are capable of identifying action and business object in activity labels, they are not applicable to events as their structure significantly deviates from those of activities. Hence, we extended the derivation algorithm

<table>
<thead>
<tr>
<th>Name</th>
<th>Structure Pattern</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verb-Object</td>
<td>( A ) (imperative) + ( BO ) + (( PP ) + ( ADD ))</td>
<td>Evaluate Receipts</td>
</tr>
<tr>
<td>Noun Phrase</td>
<td>( BO ) + ( A ) + (( PP ) + ( ADD ))</td>
<td>Goods Receipt Processing with Reference</td>
</tr>
<tr>
<td>Preposition-’of’</td>
<td>( A ) + ’of’ + ( BO ) + (( PP ) + ( ADD ))</td>
<td>Monitoring of Budget Planning</td>
</tr>
<tr>
<td>Gerund</td>
<td>( A ) (gerund) + ( BO ) + (( PP ) + ( ADD ))</td>
<td>Generating Rent Adjustment Data</td>
</tr>
<tr>
<td>Irregular</td>
<td>-</td>
<td>FIFO: Valuation</td>
</tr>
</tbody>
</table>
for start and end events based on the event label structure in the investigated process model collections. In order to be able to accomplish this extension, we had to learn about the general structure of start and end events. An analysis on event structures with the focus on start events was already conducted by Decker and Mendling in [DM09]. However, in order to provide structures which are also applicable to end events, we analyzed and classified all start and end events according to the internal structure of their elements. In total, three main structures were detected in the investigated model collections.

In the first type, which we will refer to as event-participle style, the action is given in the past participle verb form at the end of the label while the business object is stated in the beginning. As an example consider the label Asset to be created. Apparently, the business object is given by asset at the beginning and the action by create at the end. However, the remarkable characteristic of this label is the construct to be in the middle. There is a large number of different constructs which can be found among labels following this type. In start events, this includes must be, is to be and needs to be and in end events we encountered constructs as was or has been.

As opposed to the first type, in the second event structure the action is given as a noun. Therefore, we will refer to it using the term event-noun style. As an example consider the label Post Capitalization is carried out. In this label the business object is given in the beginning succeeded by the nominalized action. Similarly to the first event type, this structure contains a signalizing construct. In contrast to the first type, it is stated at the end of the label. However, there are again several constructs which can be

<table>
<thead>
<tr>
<th>Name</th>
<th>Structure Pattern</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event-Participle</td>
<td>BO + SC + A (past participle) + (PP + ADD)</td>
<td>Asset to be created</td>
</tr>
<tr>
<td>Event-Noun</td>
<td>(BO) + A (nominalized) + SC + (PP + ADD)</td>
<td>Post Capitalization is carried out</td>
</tr>
<tr>
<td>Verb-Object</td>
<td>A (imperative) + BO + (PP + ADD)</td>
<td>Perform similar part search</td>
</tr>
</tbody>
</table>
found. Start events include constructs as *required* or *is necessary*. End events contain phrases such as *carried out*, *executed* or *was performed*.

Besides these two structures which are solely used amongst events, some events also make use of the verb-object style. As this style directly advises to perform an action and does not represent a state, it cannot be considered to be a proper event style. However, as it is also used amongst events, it is mentioned as well.

Summarizing the observations about the event types, Table 4 provides the general structural patterns for the event styles. In addition to the already introduced abbreviations for the structure definitions, the signal construct is represented by *SC*.

Utilizing the signal constructs and their position in the label, it is straightforward to identify to which style a given event belongs and to subsequently derive action and business object based on the structural pattern.

### 4.4. Phase 2: Information Extraction

Once the process model is annotated, we can employ different sub approaches to generate potentially useful labels. The results of these sub approaches are then bundled together to the final proposal list. As indicated in subsection 4.2, the presented sub approaches explicitly build on the manual naming strategies identified in the last section. Table 5 summarizes the mapping from the manual naming approaches to the automated

<table>
<thead>
<tr>
<th>Manual Naming Strategy</th>
<th>Automated Sub Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dominating Element</td>
<td>Dominating Element Extraction</td>
</tr>
<tr>
<td>Main Activity</td>
<td>Main Activity Extraction</td>
</tr>
<tr>
<td>Start/End Event</td>
<td>Event Extraction</td>
</tr>
<tr>
<td>Only Existing Activity</td>
<td>Main Activity Extraction</td>
</tr>
<tr>
<td>Conjunction</td>
<td>Logical/Lexical Conjunction</td>
</tr>
<tr>
<td>Semantic Naming</td>
<td>Label Repository</td>
</tr>
</tbody>
</table>
sub approaches which will be presented in the context of this subsection. The table furthermore illustrates that we aim for covering each of the manual approaches to provide a variety of potentially useful labels. Note, that the approach Subordinate Element Extraction is not listed in this table as it only provides input for other sub approaches, but does not compute label proposals itself.

The following subsections will introduce each sub approach depicted in Figure 14 and discuss its steps in detail.

4.4.1. Dominating Element Extraction

Building on the concept of the dominating action and dominating business object of section 3, this sub approach investigates whether the given process model includes such dominating elements. The steps are formalized in Algorithm 2. The algorithm requires two input parameters: A process model and the type, determining whether the approach searches for a dominating action or a dominating object. In order to identify a dominating element, the occurrence of each element, such as action or business object, among all activities of the model is investigated (lines 4-7). If one element has a higher occurrence than all other elements, it is saved as dominating element (lines 8-9). If both,

Algorithm 2 Dominating Element Extraction

1: $\text{extractDominatingElements(}\text{ProcessModel } \text{model, String } \text{type} )$
2: List $\text{elementCount} = \text{new List}()$
3: List $\text{elements} = \text{model.getElements}(\text{type})$
4: for all elements $\text{elem}$ in $\text{model}$ do
5: $\text{currentCount} = \text{elementCount}\cdot\text{get}(\text{elem})$
6: $\text{elementCount.set}(\text{elem, currentCount+1})$
7: $\text{maxCount} = \text{elementCount}\cdot\text{getMax}()$
8: if amount of elements $\text{elem}$ with count $\text{maxCount}=1$ then
9: $\text{return}$ $\text{elem}$ with count $= \text{maxCount}$
10: else
11: $\text{return } ""$
dominating action and dominating business object are found, these elements are used to construct a new activity. If only one dominating element has been identified, this element can be used for the **Subordinate Element Extraction**. In case no dominating element could be detected, the further steps are limited to the **Event Extraction** and the **Main Activity Extraction** as these sub approaches do not require the input of a dominating element.

### 4.4.2. Subordinate Element Extraction

If only one dominant element was detected, the **Subordinate Element Extraction** identifies those actions or business objects with whom the dominant element is connected in the given process model. The identification of these subordinate elements is straightforward. If for instance, the dominating action *analyze* was derived from the two activities *Order Analysis* and *Program Analysis*, the subordinate elements are given by the business objects *order* and *program*. Hence, all activities containing the dominating element are selected and the subordinate elements are derived.

Algorithm 3 illustrates this concept using pseudo code. The algorithm requires a list of all activities of the considered process model, the according dominating element and the type, determining whether the dominating element is an action or business object. As a result, it returns a list of subordinate elements which contains actions if the dominating element is a business object and vice versa. In order to implement the subordinate element extraction each label in the given activity list is inspected whether it contains the dominating element (line 4). If this is the case, the according subordinate element is added to the result list `subordinateElements` (line 6,8). After all activities have been investigated, the result list is returned (line 9). Based on the identified set of subordinate elements, we can use two approaches to combine the dominating element and its subordinates to a proper label: **Lexical Conjunction** and **Logical Conjunction**. Both strategies are introduced in the following subsections.
Algorithm 3 Subordinate Element Extraction

1: `extractSubordinateElements(List activityList, String dominatingElem, String type)`
2: List `subordinateElements = new List();`
3: for all activities `a` in `activityList` do
4:   if `a` contains `dominatingElem` then
5:     if `type` = "Action" then
6:       `subordinateElements.add(a.getBusinessObject());`
7:     else
8:       `subordinateElements.add(a.getAction());`
9:   return `subordinateElements;`

4.4.3. Lexical Conjunction

One option to create a label from the dominating element and a set of its subordinates is to perform a lexical conjunction. Thereby, the subordinate actions or business objects are replaced with a newly introduced element derived from the lexical relations between these actions or business objects. In particular, the lexical conjunction approach detects common holonyms and hypernyms of the subordinate elements. If a proper holonym or hypernym is found for the given set of subordinate elements, a label can be constructed using the dominating element and the according holonym or hypernym.

Algorithm 4 formalizes this approach. The algorithm requires three parameters as input: A list of the subordinate elements, the dominating element and the type, determining whether the dominating element is an action or a business object. As a result, the algorithm returns a list of activities created from the dominating element and the identified holonyms and hypernyms. In the first step (lines 4-6) the original list with all subordinate elements is reduced to those elements which have an entry in the WordNet dictionary. This is necessary, as for all other elements it is not possible to determine lexical relations using WordNet. If more than one element remains (line 7) the common hypernyms and holonyms are identified for those elements. This is accomplished by deriving all holonyms and hypernyms for the first element in the list `subordinateElementsClean`
Algorithm 4 Lexical Conjunction

1: `getLexicalConjunctions(List subordinateElems, String dominatingElem, String type)`
2: `lexicalConjunctions = new List();`
3: `subordinateElemsClean = new List();`
4: `for all elements elem in subordinateElems do`
5:   `if WordNet contains entry for elem as type then`
6:     `subordinateElemsClean.add(elem);`
7: `if subordinateElemsClean.length > 1 then`
8:   `holonyms = subordinateElemsClean.get(1).getHolonyms();`
9:   `hypernyms = subordinateElemsClean.get(1).getHypernyms();`
10: `for i=2 to subordinateElemsClean.getLength() do`
11:   `holonyms = getCommonElems(holonyms, subordinateElemsClean.get(i).getHolonyms());`
12:   `hypernyms = getCommonElems(hypernyms, subordinateElemsClean.get(i).getHypernyms());`
13: `lexicalConjunctions.add(holonyms);`
14: `lexicalConjunctions.add(hypernyms);`
15: `returnList = new List();`
16: `for all conjunctions c in lexicalConjunctions do`
17:   `returnList.add(new Activity(dominatingElem, c, type));`
18: `return returnList;`

(lines 8-9). Subsequently, by iterating over all other subordinates, the two lists `holonyms` and `hypernyms` are reduced to those elements which they share with the considered subordinate element (lines 10-12). As a result, at the end of the loop the two lists contain only these holonyms and hypernyms which are part of the intersection between all subordinate elements holonyms and hypernyms. Subsequently, for each identified holonym and hypernym an activity is constructed in combination with the dominating element and stored in a list (lines 15-17). Finally, this list is returned (line 18).

4.4.4. Logical Conjunction

If for instance a dominating action is performed on a small number of business objects, these business objects can be simply connected using the logical operators `and` or `or`.
Algorithm 5 Logical Conjunction

1: `getLogicalConjunctions(List subordinateElems, String dominatingElem, String type, String connector)`
2: `newElem = first element of subordinateElems;`
3: `for i=2 to subordinateElems.getLength() do`
4: `if i < subordinateElems.getLength() then`
5: `newElem.append("," + subordinateElems.get(i));`
6: `else`
7: `newElem.append(connector + subordinateElems.get(i));`
8: `newActivity = new Activity(dominatingElem, newElem, type)`
9: `return newActivity;`

Likewise, the same strategy can be applied for dominating business objects by connecting the actions which are performed on this business object. The advantage in comparison to the lexical conjunction is given by the fact that the logical connection is not depending on an existing relationship between the subordinate elements. For example, the business objects `Service Order` and `Information System` are not connected with a lexical relation such as holonymy or hypernymy. Nevertheless, they can still be connected logically, complemented by a dominating action.

Algorithm 5 illustrates the required steps using pseudo code. Similarly to the *Lexical Conjunction*, the algorithm requires four parameters as input: A list of subordinate elements, the dominating element, the type, determining whether the dominating element is an action or a business object and the connector, defining whether *and* or *or* is applied. The steps of the algorithm are concerned with adequately connecting the multiple actions. Therefore, the first subordinate element is saved to the variable `newElem` (line 2). Subsequently, the remaining elements are appended in the context of a loop (lines 3-7). In case there remain multiple elements, this is done by adding a comma and the subordinate element (line 5). In the case only one element is left, it is appended using the connector plus the according subordinate element (line 7). After the activity was constructed from this new action and the dominating element (line 8), the activity is returned (line 9).
4.4.5. Label Repository

As companies may already possess existing process models, the activities of these models can be used to build up a label repository. For our purposes such a label repository simply consists of a set of activity labels. Hence, if a dominating element was identified in the Dominating Element Extraction, a label repository can be consulted to find a corresponding element which is likely to be connected with the dominating element. This strategy utilizes the fact that some actions are frequently connected with the same business objects and vice versa. Moreover, this sub approach provides the possibility to obtain information which is not included in the considered model and hence also incorporates semantic aspects. If we create such a label repository using the activities of the SAP Reference Model, we attain a repository with 2433 activities. In order to demonstrate the concept, assume we are looking for a business object for the dominating action allocate. By looking up the action allocate, we obtain labels such as Allocate Budget or Allocate Costs. As a result, a complement for the dominating action allocate was identified, which may fit as an appropriate business object in a given context.

Algorithm 6 provides a formal description of the Label Repository approach. The algorithm requires four input parameters: a list of activities which is used as the repository, the dominating element of the considered process model, the type of the dominating element, whether it is an action or a business object and the maximum number of elements which may be returned. The output of the algorithm is a list of candidate activities which may fit for appropriately describing the process model. In the beginning of the algorithm two lists are created (lines 2-3). The list candidates is employed for saving the identified candidate activities and the list candidateCount is used for storing the count of each identified activity. Thereby, the indices of the activity and its count are always identical. Thus, the count for the second activity in candidates will be also found at the second position in candidateCount. In the following loop each activity comprised in the repository is analyzed whether it contains the dominating element (lines 4-5). If this
Algorithm 6 Derivation from Label Repository

1: deriveFromLabelRepository(List repository, String dominatingElem, String type, Integer maxCount)
2: List candidates = new List();
3: List candidateCount = new List();
4: for all activities a in repository do
5: if a.getElement(type) = dominatingElement then
6: if candidates contains a then
7: currentCount = candidateCount.get(a);
8: candidateCount.set(a, currentCount + 1);
9: else
10: candidates.add(a);
11: candidateCount.add(a, 1);
12: order candidates by candidateCount;
13: if candidates.getLength() > maxCount then
14: candidates = candidates.getFirstElements(maxCount);
15: return candidates;

is the case and the identified activity already exists in the candidate list, the index of
the activity in candidates is determined and the count in candidateCount is increased
accordingly (lines 7-8). If the activity was not found in previous iterations, the activity
is added to candidates and the count 1 is added to candidateCount (lines 10-11). After
all labels in repository have been investigated, the activities in the resulting candidate
list are ordered according to their count (line 12). Subsequently, it is checked whether
the number of candidates exceeds the maximum count given by maxCount. If this is the
case, candidates is shortened to the first and thus the most frequent elements given by
maxCount (lines 13-14). Finally the candidate list is returned (line 15).

4.4.6. Event Extraction

As shown in section 3, start and end events are quite frequently used for capturing
information about the overall contents of the process model. Utilizing the information
Algorithm 7 Event Extraction

1: extractFromEvents(List eventList, String eventType)
2: returnList = new List();
3: for all events e in eventList do
4:   if eventType = "Start Event" then
5:     if e contains "required" or "is necessary" or "must be" or "needs to be" or "to be" then
6:       returnList.add(e);
7:   if eventType = "End Event" then
8:     if e contains "executed" or "carried out" or "performed" or "was" or "were" then
9:       returnList.add(e);
10:  for all events e in eventList do
11:    returnList.add(e.toActivity());
12:  return returnList;

about action and business object from the annotation phase, potentially meaningful labels can be derived. However, especially the analyzed SAP Reference Model contains models having up to 29 start and 41 end events. In order to provide an appropriate number of label proposals some of these labels need to be excluded. Considering the event patterns and analysis results presented in section 4.3, it can be assumed that certain patterns are more likely to indicate an event which is actually presenting useful information about the process. For instance, it is not very probable that the start event Asset was found indicates what is going to be performed in the process, but may represent a state which is required for triggering the first activity. By contrast, the start event Asset is to be created captures the necessity for the execution of a certain action. Hence, the start and end events need to be reduced to those, where the signalizing construct indicates that the event actually contains information about what is going to happen or what already happened in the process.

Algorithm 7 formalizes this approach using pseudo code. It requires a list of events of the considered process model and the type of events which are handed over. As output, the algorithm provides a list of activities which might be suitable for labeling the process.
once they have been refactored. In order to obtain potentially appropriate events, all events in the \textit{eventList} are analyzed for signal constructs which may indicate whether the event is likely to contain information about the process. Therefore, it is differentiated between start and end events (lines 4, 7). Start events are checked for constructs indicating that something is going to be performed (line 5) and end events are inspected for constructs indicating that something was conducted during the process (line 8). Thereby, the constructs mentioned in the algorithm are derived from the analyzed process model collections. If one of these constructs was found, the event is added to the \textit{resultList} (lines 6, 9), which is subsequently used for constructing appropriate activities (lines 10-11). In the last step, these activities are returned (line 12).

4.4.7. Main Activity Extraction

Automatically deciding whether a considered activity represents a main activity for the given process model or not, is a real challenge. In order to be able to make this decision for an individual activity, we would have to automatically derive the context of the process and subsequently decide about the role of the activity. Hence, to circumvent this complex task, this approach utilizes the insight about the frequent position of main activities, which was acquired in the context of an analysis on all main activities. From this analysis we learned that approximately 85% of the main activities are either in the first or the last position of the process model. Thus, the main activity extraction presumes the existence of a main activity in the first or last position and returns the according labels.

Algorithm 8 illustrates the required steps. The algorithm expects a list of all activities of the process model and returns a list of potential main activities. As the algorithm assumes the existence of main activities, the core task is to identify the first and last activities. Therefore, all activities from \textit{activityList} are investigated in the following manner. First of all, the preceding events of the considered activity \textit{a} are derived (line 4) and the boolean variable \textit{firstActivity} is set to \textit{true} (line 5). Subsequently, all preceding events are checked for their predecessors (lines 6-7). If one of the events has any preceding
Algorithm 8 Main Activity Extraction

1: extractMainActivities(List activityList)
2: returnList = new List();
3: for all activities a in activityList do
4:   precedingEvents = a.getPredecessors();
5:   firstActivity = true;
6:   for all events e in precedingEvents do
7:     if e.getPredecessors().getLength() > 0 then
8:       firstActivity = false;
9:     if firstActivity = false then
10:    succeedingEvents = a.getSuccessors();
11:   lastActivity = true;
12:   for all events e in succeedingEvents do
13:     if e.getSuccessors().getLength() > 0 then
14:       lastActivity = false;
15:   if firstActivity = true or lastActivity = true then
16:     returnList.add(a);
17: return returnList;

activities, the originally considered activity $a$ cannot be the first activity within the process model. Hence, $firstActivity$ is set to $false$ (line 8). By contrast, if none of the events has any preceding activities, $firstActivity$ accordingly remains $true$. If the activity was not proven to be a first activity, the analogous steps are performed for determining whether $a$ is a last activity (lines 10-14). Finally, if the considered activity was identified as last or a first activity, it is accordingly added to the result list (line 16) which is subsequently returned (line 17).

4.4.8. Combining All Approaches

In order to obtain an all-encompassing approach, we combine all sub approaches as depicted in Figure 14. To some extent the order of the sub approaches is strict as some sub approaches like the Lexical Conjunction are relying on the input provided by other
Algorithm 9 Main Algorithm Phase 2

1: `performMainAnalysis(ProcessModel model)`
2: `returnList = new List();`
3: `dominatingAction = extractDominatingElements(model, "Action");`
4: `dominatingBusinessObject = extractDominatingElements(model, "Business Object");`
5: if `dominatingAction.hasValue() and dominatingBusinessObject.hasValue()` then
   6: `returnList.add(new Activity(dominatingAction, dominatingBusinessObject));`
7: if `dominatingAction.hasValue() or dominatingBusinessObject.hasValue()` then
   8: if `dominatingAction.hasValue()` then
      9: `type = "Action";`
     10: `dominatingElem = dominatingAction;`
   else
     12: `type = "Business Object";`
    13: `dominatingElem = dominatingBusinessObject;`
14: `subElems = extractSubordinateElements(model.getActivities(), dominatingElem, type);`
15: `returnList.add(performLexicalConjunction(subElems, dominatingElem, type));`
16: `returnList.add(performLogicalConjunction(subElems, dominatingElem, type, "and");)`
17: `returnList.add(deriveFromLabelRepository(repository, dominatingElem, type, 5));`
18: `returnList.add(extractFromEvents(model.getEvents(), "Start Event");)`
19: `returnList.add(extractFromEvents(model.getEvents(), "End Event");)`
20: `returnList.add(extractMainActivities(model.getActivities());)`
21: return `returnList;`

algorithms. However, algorithms such as the Main Activity Extraction are acting independently from other sub approaches and may be executed at any time.

Algorithm 9 illustrates how the main approach is composed from the introduced algorithms. As provided by Phase 1, it requires an annotated process model as input and returns a list of activities, which are then refactored to proper activity labels in Phase 3. The first step is the determination of the dominating action and dominating business object (lines 3-4). If the process model contains both, a new activity is created from the dominating elements (line 5-6). Otherwise, if only one dominating element was detected,
the type of the dominating element and the dominating element itself are saved to variables (lines 7-13). Subsequently, using the identified dominating element, the according subordinate elements are extracted (line 14). On the basis of the subordinate elements the approaches Lexical Conjunction, Logical Conjunction and Label Repository are performed (lines 15-17). Finally, the Event Extraction for start and end events and the Main Activity Extraction are executed (lines 18-20). However, as both do not rely on any input besides the annotated model from Phase 2, they could be performed at any time. Having saved all activities to the variable returnList, the creation of the label proposal list in phase 3 can be triggered.

4.5. Phase 3: Proposal List Creation

Once all information and proposal labels have been derived, they can be refactored to proper verb-object labels. Subsequently, the final proposal list which is presented to the user, can be constructed. Algorithm 10 illustrates the required steps using pseudo code. As all activities were annotated with their according action, business object and also additional information, the refactoring to the verb-object style is straightforward. First of all, action, business object and additional information are derived and saved to variables (lines 4-6). Thereby, not the action itself, but the lemma of the action is derived by consulting WordNet. This is necessary, as the resulting label is supposed to be a verb-object label. In order to obtain a proper verb-object label, the information for each label must correspond to the structure $A \text{ (imperative)} + BO + (PP + ADD).$ Hence, the imperative of the action has to be derived if not already given. As in the English language the imperative equals the infinitive, it is sufficient to derive the lemma of the action using WordNet. Afterwards, the label elements can be positioned according to the verb-object structure (line 7) and the label is added to the proposal list (line 8). Finally, the proposal list is returned (line 9). In order to complete the labeling task, the modeler simply has to select the most appropriate activity from this list.
4 STRATEGY DEVELOPMENT FOR AUTOMATED LABELING

Algorithm 10 Creation of Proposal List

1: getLabelProposalList(List activityList)
2: proposalList = new List();
3: for all activities a in activityList do
4:   action = WordNet.getLemma(a.getActivitiy(), "Verb");
5:   businessObject = a.getBusinessObject();
6:   addition = a.getAdditionalInfo();
7:   proposalLabel = action + businessObject + addition;
8:   proposalList.add(proposalLabel);
9: return proposalList;

4.6. Application on an Exemplary Process Model

In order to be able to automatically apply the introduced algorithms on process models, we created a prototype. This prototype implements the introduced algorithms in Java. For a description of the implementation, please refer to Appendix A. This section demonstrates the application of the labeling approach on a particular process model and discusses the resulting proposal list.

To illustrate the results of the labeling approach we make use of the process model Capacity Planning from the SAP Reference Model (see Figure 15). According to our classification, the model name was derived via a combination of the concept of Semantic Naming and Dominating Business Object.

Table 6 shows the label proposals the modeler will be provided with. In order to get better insights how the label proposals were created, we state the according derivation approach for each label. As a result, he proposal labels illustrate which algorithms were used to generate the labels: First of all, a dominating business object was identified, which could be subsequently used by the logical conjunction and the label repository to successfully compute activity labels. By contrast, the lexical conjunction approach did not find any holonyms or hypernyms among the set of the subordinate elements and hence, does not provide any result. However, also the approaches which are not relying
on the dominating element succeeded to provide label proposals. For example, the event
event extraction algorithm derived three labels from first events and one label from a last event.

As a result from all these approaches, we obtained a list containing ten verb-object
labels. However, the most interesting point is that the process model name, which was
partially determined by semantic means, is included in the proposal list. Based on the
identified business object capacity, the Label Repository approach computed the label
Plan capacity, which is the accurate verb-object version of the process model name. Con-
sequently, we assume the modeler to choose this label from the proposal list. As a result,
the labeling task is reduced to one single click on the most suitable label proposal.

Figure 15: SAP Model Capacity Planning
4 STRATEGY DEVELOPMENT FOR AUTOMATED LABELING

Table 6: Computed Proposal List for SAP Model *Capacity Planning*

<table>
<thead>
<tr>
<th>Label Proposal</th>
<th>Derivation Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>Dominating Business Object</td>
</tr>
<tr>
<td>Dispatch, level and evaluate capacity</td>
<td>Logical Conjunction</td>
</tr>
<tr>
<td>Level capacity</td>
<td>Label Repository</td>
</tr>
<tr>
<td><strong>Plan capacity</strong></td>
<td>Label Repository</td>
</tr>
<tr>
<td>Evaluate capacity situation</td>
<td>First Event</td>
</tr>
<tr>
<td>Release maintenance order</td>
<td>First Event</td>
</tr>
<tr>
<td>Carry out sequencing</td>
<td>First Event</td>
</tr>
<tr>
<td>Perform capacity leveling or planning</td>
<td>Last Event</td>
</tr>
<tr>
<td>Evaluate capacity</td>
<td>Main Activity (First Activity)</td>
</tr>
<tr>
<td>Dispatch and level capacity</td>
<td>Main Activity (Last Activity)</td>
</tr>
</tbody>
</table>

4.7. Summary

In this chapter we presented a semi-automatic approach for naming process models. The approach consists of three fully automated phases, while the core of the approach is composed from seven sub approaches. Each of these sub approaches pursues a different strategy to compute adequate label proposals. The advantage of this modular design is that other strategies could be easily implemented as additional sub approaches and may thus, enhance the overall appropriateness of the approach. As a result, the labeling approach provides a list of potentially suitable labels to the modeler. By choosing the most appropriate label proposal, the labeling task is reduced to one single click.

However, the most significant aspect which has to be proven, is the actual capability of our approach to successfully compute activities, which are suitable for labeling process models. In the last section we have shown that our approach provided a label which was actually similar to the process name determined by humans. Nevertheless, as we only presented one example, we still need to systematically validate our approach. Therefore, the next section will present a validation experiment, where we apply our prototype to
the SAP Reference Model.
5. Validation

In order to demonstrate the capability of our labeling approach to find appropriate names for process models, we conducted a validation experiment using our prototype. The challenge in the context of the validation was to choose an appropriate method. For instance, if we decided to assess the computed results in isolation, we would have to measure how plausibly a computed label describes a given process. However, as this is a highly subjective task, we decided to compare the computed results with process names which were created by humans. In order to include a significant number of process models, we employed the SAP Reference Model for our validation experiment. To finally assess the confidence of our approach we need to measure the similarity between the computed labels and the existing process names in an objective manner. Therefore, the next section introduces the backgrounds on how the similarity measurement is accomplished.

5.1. Methodology

As already pointed out earlier, we consider a process label to be characterized by mainly two elements: action and business object. Hence, it is of primary importance that a computed label proposal captures the according action and business object provided by original process name. Therefore, we assume a label proposal to be equal to the process name if action and business object are similar. In order to objectively determine this similarity for two actions and two business objects, we introduce the following functions:

- $\text{match}_A : A \times A \rightarrow \{1, 0\}$ is a function that assigns 1 or 0 to a pair of actions. Function $\text{match}_A$ returns 1 if the lemmata of both actions are equal or if both actions are elements of the same synset in WordNet. In the case of multiple actions all lemmata must fulfill the preceding requirements, otherwise the function returns 0. In case the original process name does not provide any action, $\text{match}_A$ returns 0 accordingly.
• $\text{match}_{BO} : BO \times BO \rightarrow \{1, 0\}$ is a function that assigns 1 or 0 to a pair of business objects. Accordingly to $\text{match}_A$ the function $\text{match}_{BO}$ returns 1 if the lemmata of both business objects are equal or if both business objects are part of the same WordNet synset. Analogously to $\text{match}_A$, in the case of multiple business objects all lemmata must fulfill the preceding requirements, otherwise the function returns 0. If the original process name does not include a business object, $\text{match}_{BO}$ returns 0 accordingly.

Based on these function definitions we can specify a function $c(l_c, l_n)$ for measuring the coverage of the label $l_n$ by the label $l_c$. Thereby $l_c$ represents the computed label and $l_n$ the existing name of the model. In order to refer to the action or business object of the label, the subscript $A$ or respectively $BO$ is added to the variables $l_n$ and $l_c$. The function $\text{numberOfComponents}(l)$ returns 1, if the parameter label contains either action or business object and returns 2, if the label contains both, action and business object.

The coverage function $c$ is defined as follows:

$$c(l_c, l_n) = \frac{\text{match}_A(l_{cA}, l_{nA}) + \text{match}_{BO}(l_{cBO}, l_{nBO})}{\text{numberOfComponents}(l_n)}$$

The concept behind the coverage function is that we assign a coverage score to each label proposal. If the given process name $l_n$ contains both, action and business object, the coverage score is 1 if both are included, it is $\frac{1}{2}$ if either action or business object are covered, and it is 0 if no similarities can be detected. If the given process name $l_n$ either contains action or business object, the coverage is 1 if this particular element is included, and it is 0 if it is not covered. This coverage score is then calculated for each label proposal associated with a given process model. The coverage $c_m$ which is assigned to a process model $m$ and its set of label proposals $L$ is then given by the maximum coverage value among all label proposals in $L$.

$$c_m(m) = \arg \max_{l_c \in L} c(l_c, l_n)$$
Consequently, the coverage $c_m$ is computed for all $n$ process models contained in the investigated process model collection $M$. Hence, by adding up all coverage results we receive a coverage sum. If we divide this coverage sum by the maximum coverage value, given by the number of process models in the collection $M$, we finally obtain a total coverage $c_{total}$, normalized to the interval $[0;1]$.

$$c_{total} = \frac{\sum_{i=1}^{n} c_m(m_i)}{n}$$  \hspace{1cm} (3)

After we introduced the methodology, the following subsection discusses the results of the validation experiment.

5.2. Results

Table 7 summarizes the results from the conducted experiment by clustering the frequency of the three coverage values 0, $\frac{1}{2}$ and 1 according to the different labeling approaches conducted by humans.

Considering the results, we need to address several aspects. Most important of all, we need to discuss the final result which is given by the total coverage $c_{total}$. The value of $c_{total}$ was determined with 0.734 or accordingly 73.4%. Interpreted in the context of the introduced labeling approach, this means that the approach covers, and was thus able to automatically compute, 73.4% of the names given by the process model collection. From the first impression, this result seems to be quite satisfying. However, as the total coverage also indicates potential for improvement, one significant question is, where the introduced algorithm failed to meet its objective. Therefore, the following paragraphs introduce the most remarkable limitations of the algorithm.

As Table 7 reveals, the biggest problem occurs among models which were labeled using the semantic naming approach. Almost all models with $c_m(m) = 0$ belong to this category. This result is caused by the fact that the introduced sub approaches mainly rely on information, which is explicitly given in the model. Hence, the algorithm frequently fails to match the process names, which were determined by semantic means. However,
Table 7: Results of the Validation Approach

<table>
<thead>
<tr>
<th>Manual Labeling Approach</th>
<th>$c_m(m) = 1$</th>
<th>$c_m(m) = \frac{1}{2}$</th>
<th>$c_m(m) = 0$</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dominating Business Object</td>
<td>64</td>
<td>0</td>
<td>0</td>
<td>64</td>
</tr>
<tr>
<td>Dominating Action</td>
<td>28</td>
<td>0</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>Main Activity</td>
<td>42</td>
<td>6</td>
<td>6</td>
<td>54</td>
</tr>
<tr>
<td>First Event</td>
<td>29</td>
<td>0</td>
<td>0</td>
<td>29</td>
</tr>
<tr>
<td>Last Event</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>26</td>
</tr>
<tr>
<td>Only existing Activity</td>
<td>76</td>
<td>0</td>
<td>0</td>
<td>76</td>
</tr>
<tr>
<td>Conjunction</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>Semantic Naming</td>
<td>51</td>
<td>76</td>
<td>84</td>
<td>211</td>
</tr>
<tr>
<td>Dominating Action + Dominating Business Object</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>Dominating Action + Semantic Naming</td>
<td>13</td>
<td>9</td>
<td>5</td>
<td>27</td>
</tr>
<tr>
<td>Dominating Business Object + Semantic Naming</td>
<td>21</td>
<td>20</td>
<td>7</td>
<td>48</td>
</tr>
<tr>
<td><strong>Sum</strong></td>
<td><strong>385</strong></td>
<td><strong>117</strong></td>
<td><strong>102</strong></td>
<td><strong>604</strong></td>
</tr>
</tbody>
</table>

although most of the uncovered cases occurred among models with semantic naming, for about 30% of these models the name was still determined correctly. The reason therefore can be found in the degree of the semantic abstraction. While some models mention the components of the model name, others do not refer to them at all. Especially these models, where at least some element parts of the process name were found, are labeled properly.

Another remarkable case is given by the models named with the combination of semantic means and the dominating activity. As stated in the result table, there are five cases where the coverage value is 0. However, from the intuition we would assume that at least the dominating action should have been found and the coverage value would accordingly be $\frac{1}{2}$. One particular example of these cases can be found in the model Transaction Processing which contains the dominating action process and the business object Transaction which was determined by semantic means. The reason why the prototype
failed to identify the dominating action is given by the poor annotation result of Phase 1. The considered process model contains three verb-object labels with the action process. Caused by a weakness of the annotation algorithm, the action was not determined correctly and the count of the action process was determined with 1 instead of 4. Hence, the imprecise annotation results from Phase 1 negatively affected the coverage achieved by the prototype.

Considering the results of the non-semantic approaches in isolation, we achieved a very satisfying result for these approaches. If we calculate $c_{total}$ for the non-semantic approaches, we obtain a total coverage of 96.4%. Remaining problems among these models are caused by mainly two reasons.

The first problem is illustrated by the SAP model Repair Order which was labeled using the Main Activity approach but contains a main activity in an intermediate position. As the main activity extraction approach assumes that the main activity can be either found in the first or last position of the model, it consequently fails if the main activity is positioned elsewhere. However, although in the given example the main activity Repair Order Processing was not used for deriving the process name, the last event Order is created was employed to propose the label Create Order. This label matches the given process name at least partially.

The second reason for non-semantic approaches to fail can be found among models using conjunction. For instance, the process name of the model Settlement and Completion was derived from the activities Order Settlement and Business Order Completion. From the name of the process, we may conclude that the terms Order and Business Order refer to the same business object. However, as WordNet does not contain an entry for Business Order, both terms are treated as totally different business objects. As a result, the dominating business object Order is not identified. Consequently, the logical conjunction approach is never triggered as it relies on a given dominating element. However, the labeling approach e.g. proposes the label Complete Business Order which at least partially
5 VALIDATION

covers the given process name.

All in all, we can state that the proposed algorithm achieves a satisfying coverage value. The prototype almost completely covers the process names which were created with non-semantic approaches. However, for the 47% of the process models where the name was completely or partially derived by semantic means, the approach only achieved a coverage value of 1 among 30% of the models. Hence, the incorporation of a sub approach identifying process names via part-of relationships represents a critical aspect for improving the overall result. Nevertheless, in order to accurately assess the results of our experiment and to conclude whether the presented results indicate that our approach is capable of accurately naming process models, we need to include a discussion about the validity of the experiment. Therefore, the next subsection investigates different validity aspects and accordingly assesses the value of our result.

5.3. Assessment of Validity

In order to precisely assess the results of our experiment, we need to focus on certain validity aspects. The importance of this field is highlighted by Basili et al. who state that experiments and drawing general conclusions from empirical studies in the field of software engineering is fairly difficult [BSL99]. They furthermore emphasize that the credibility of a study always depends on how the conclusions are drawn. Thereby, they refer to different classes of evaluation criteria. Especially relevant for our setting are the internal and external validity proposed by Campbell and Stanley [CS66, pp. 13-16] and the construct validity defined by Judd. et al. [JSK91, pp. 27-35]:

- **Internal validity**: In general terms, the internal validity reflects to what extent the cause-and-effect relationship between the factors of interest and the observed results is confident. In our experiment the internal validity refers to the value of the total coverage.

- **External validity**: The external validity expresses to what extent the results from
the internal validity can be generalized. In our context, the external validity consequently assesses to what extent our approach is applicable to other process model collections.

- **Construct validity**: The construct validity assesses to what extent the variable successfully measures the theoretical constructs of the hypothesis. Applied to our setting, the construct validity refers to the question whether our validation actually measures the suitability of our approach for labeling process model names.

The following paragraphs will investigate each of these validity classes for our validation experiment.

### 5.3.1. Internal Validity

As already pointed out, the internal validity is given by the actual results from our experiment. As reported, our approach yielded a total coverage of 73.4%. Hence, we can state that the approach satisfactorily matches the given process names. However, as intensively discussed in the last subsection, in especially two cases the proposed algorithm also fails to meet its objective. In the following paragraphs we shortly discuss these problems and indicate possible solutions to further increase the total coverage and hence, the internal validity.

The most significant drawback is that our approach fails to cover models which have been labeled with approaches that include semantic naming. In order to circumvent this problem, we could e.g. introduce ontologies which adequately represent part-of relationships. As a result a superordinate category could be derived for a given set of semantically related activities. As already mentioned in section 2.4, Smirnov et al [SDMW10] introduce a similar approach by using a meronymy tree of activities. As a result, a set of activities is replaced with their common holonym. However, this approach relies on the existence of such a meronymy tree, which can be considered to be a very heavy prerequisite in practice. Nevertheless, if a suitable ontology can be automatically constructed
from existing process models, this approach may serve as a solution to cover models where the semantic naming approach was applied.

The second aspect, we also already mentioned in the result section, is the problem of partially inaccurate annotation from Phase 1. It is important to notice that the precision of the annotation phase may distort the final result. As reported in [LSM10], the annotation has a precision of 85%. Hence, the improvement of this approach is also important for improving the coverage value.

After we have shortly discussed the results from an internal perspective, the following subsection discusses to what extent these results can be generalized.

5.3.2. External Validity

Concerning the external validity Basil el al. state that building knowledge in the domain of software engineering requires the implementation of a family of experiments [BSL99]. Based on these combined results general conclusions can be drawn. However, as we cannot provide a set of experiments in this work, we assess the external validity by focussing on the limitations of the validation experiment. The most important limitation of the experiment is the use of one single process model collection, the SAP Reference Model. As the SAP Reference Model cannot considered to be representative for any kind of real world process model collection, we cannot assume the results to be universally valid. Although the SAP Reference Model has been widely used for different experiments in the process model domain, also its quality issues have been discussed [Men08, DRvdAS08, MRR09]. Such quality issues concerned structural aspects as well as pragmatic factors such as the labeling quality. Knowing the limitations of the SAP Reference Model raises the question, how these quality issues impact the results of our validation experiment. Also it is quite hard to generally anticipate how the structural and pragmatic quality of a process model collection may influence the total coverage, some examples based on problems from the SAP Reference Model can illustrate the effects:

- **Label Quality**: A consistent labeling using activity-based labeling styles or even
only the verb-object style will lead to a 100 \% correct annotation result from Phase 1. Consequently, all errors based on inaccurately annotated labels would be avoided.

- **Consistent Labeling**: If start and events really provide information about the states before and after the process execution, the coverage may increase further.

- **Adequate use of Constructs**: Models of the SAP Reference Model contain up to 41 end events. An adequate use of events may increase the accuracy as it is hard to select relevant events from such a big set.

As a result, we can conclude that the quality issues of the SAP Reference Model influenced the total coverage in a negative manner. If other model collections avoid these problems and do not contain other quality issues we have not encountered so far, the total coverage can assumed to be higher. A last point we may take into consideration is that the other analyzed collections, the procurement models and the German IT Service Management models, did not exhibit any different labeling approaches or quality issues than the SAP Reference Model. Although we can still not generalize the results, we can state that the approach presents a valuable starting point for further research and that additional validation will be required to emphasize the results presented in this thesis.

### 5.3.3. Construct Validity

Considering the criterion of construct validity we have to return to our initial goal of the validation experiment. We intended to show the capability of the algorithm to find at least one appropriate label for a given process model by the means of comparing the computed labels with the given process names. This approach heavily relies on the assumption that the given process names are actually suitable for describing the process. As we already pointed out, this assessment is highly subjective. However, we cannot preclude that the SAP Reference Model contains process names, a majority of consulted people would consider to be inadequate for describing the processes. From our perspective, there are two classes of such process names.
The first group is concerned with those names only providing an action or a business object and are thus incomplete. As an example consider the process model Analysis depicted in Figure 16. The proposal list created by our approach includes, amongst others, the following labels:

- Analyze program,
- Analyze order,
- Analyze information
- and Analyze program and order.

When observing the process model, we can see that only two out of five activities are concerned with the task of analysis. The business objects affected by this analysis are
given by *program* and *order*. Assuming that these activities, in a comparable way as main activities, play an important role for the process, we may also assume that humans would probably prefer the more specific label *Analyze program and order*. The consequence for our validation is that if the algorithm comes up with the incomplete and maybe also not semantically suitable label *Analysis*, we would still conclude that our approach provides a good label proposal as it results in a match with the given process name. But still, humans would probably have judged differently. However, this problem cannot be considered to be very fatal as the difference in the label is only given by an improvement of the expressiveness of the basically useful label *Analysis*.

The second group includes process names which completely fail to describe the contents of the underlying process model and thus have a more serious impact on the results of the validation. As an example consider the process model *Information System* depicted in Figure 17. Apparently, the process name *Information System* does not convey any information about the contents of the underlying process besides the fact that an information system might be required or used. Amongst others, our labeling approach proposes the following labels:
• Search report using report information system,

• Search substance using substance information system,

• Carry out drilldown reporting,

• and Search report and substance.

The first and second proposal are derived from the activities of the model and consequently only describe parts of the process. By contrast, the third proposal, derived from the first event, and the second proposal, created using logical conjunction, may be used to properly describe the process and are apparently more suitable than the label Information System. However, neither the third nor the fourth label proposal will be considered to properly describe the process model by the validation approach, simply due to the fact that they do not match the given process name.

Concluding about the construct validity, we can state that we observed some cases which may cause doubts whether the validation allows us to conclude about the ability of the labeling algorithm to provide suitable labels. However, according to our assessment theses cases only add up to 10 till 15 cases among the whole process model collection. Hence, we can claim that they did not dramatically affect the result. Finally we can say, that in order to really assess whether the introduced algorithm provides suitable labels, the results need to be validated by experiments using humans. For now, we can judge the achieved coverage of the given information and thus can state that the results indicate that the algorithm provides useful labels.

5.4. Summary

In this section we presented a validation experiment in order to demonstrate the capability of the introduced algorithm to successfully provide useful labels. To achieve this, we computed label proposals for the models of the SAP Reference Model collection. The comparison with the given names of the SAP Reference Model resulted in an total
coverage of 73.4 %]. As discussed in the previous sections, this result needs to be assessed in the context of several validity criteria. Accordingly, we can say that the results of the validation are quite satisfying in regard to the coverage of the SAP Reference Model. In order to generalize the results and to finally prove that the computed labels have useful character, further validation is required. Nevertheless, a first approach addressing the problem of the labeling task was presented and the results indicate that the approach will remain promising.
6. Conclusions

This section discusses the results of this thesis. We start by summarizing the findings of this work. Subsequently, we discuss the implications of our results for the field of BPM research. Finally, we close this thesis by sketching the potential for future research.

6.1. Summary of Results

In this thesis we developed an approach for automatically computing a set of potentially suitable labels for adequately naming process models. The application areas of this approach reach from modularization and abstraction scenarios to naming whole process models in the context of process repositories. By referring to the contributions of this thesis, the results of this work can be summarized as follows:

- **Classification of Labeling Practices**: As a result from the analysis of three real-world process model collections in section 3, we identified different labeling practices for naming process models. We classified these naming strategies into six distinct approaches. While in five of them the model name completely relies on information which is explicitly given in the model, in the labeling approach which we refer to as Semantic Naming, a part-of relationship is utilized to derive the process name from the broader context of the model. By analyzing the frequency distribution of the naming approaches among the investigated process model collections, we further uncovered that the process names of 68% of the models were labeled using approaches which completely or partially rely on explicit information.

- **Extension of Label Structure Classification**: In order to be able to adequately derive action and business objects from start and end events, we extended the label structure classification provided in [LSM09]. We learned that the structure of the investigated events can be classified into three different types. While two of them clearly indicate a state in a process, the third structure is given by the verb-object style for activities. Although the outcome also indicates a quality limitation of the
SAP Reference Model, the analysis of the structure was a significant requirement for the labeling approach to accomplish the annotation of the analyzed models.

- **Labeling Approach:** Based on the strategies identified during the analysis, we successfully developed an approach for semi-automatically labeling process models. Our approach consists of three completely automated phases: annotation, information extraction and refactoring. The main part of the approach lies in the information extraction phase which is composed of seven sub approaches. Each of these sub approaches aims at computing adequate label proposals by pursuing a different strategy. As a result, the approach automatically provides the modeler with a set of potentially adequate labels. By simply choosing the best option, the labeling task is reduced to a single click.

- **Prototype of Labeling Approach:** Based on the developed labeling approach we provide a prototypical implementation in Java. For details about the classes please refer to Appendix A.

- **Validation Experiment:** In order to demonstrate the capability of our labeling approach, we conducted a validation experiment. In this experiment we showed that our labeling approach covered 73.4% of the SAP Reference Model names, which were created by humans. Assuming that the names in the SAP Reference Model reflect the general choice of humans, the experiment indicated a good capability of the presented labeling approach.

Having summarized the findings of this thesis, the next subsection discusses the implications of these results for the BPM research domain.

### 6.2. Discussion

As already intensively discussed in subsection 5.3.1, the key question which arises when considering the results of the validation experiment is whether these results can expected
to be achieved for any kind of process model collection. As our validation only contained one model collection and humans were not included in the experiment, we cannot claim that the results can be generalized. Nevertheless, we successfully analyzed a considerable amount of models and in spite of the quality limitations of the SAP Reference Model our approach has still proven to provide good results. Hence, we may at least interpret the validation result as an indicator for the capability of our approach. However, further validation remains one of the most important aspects.

Notwithstanding the many considerations, we can identify the following implications of our results for the domain of business process management:

- **Increased Modeling Speed and Modeler Support**: By incorporating the proposed approach in a modeling tool, as for instance the web based tool Oryx\(^1\), the modeler is assisted in three major scenarios. First of all, the modeler is supported in the use case of abstraction, when activities are aggregated and a label is required for newly created activities. Furthermore, if a modeler decides to split up a complex process, the approach also aids him in finding appropriate names for the resulting subprocess. Finally, the modeler is assisted in naming whole process models for instance for the purpose of building process model repositories. In each of the cases the naming task is reduced to a single click. Even if the algorithm fails to provide a completely suitable label, the modeler is provided with proposals indicating possible solutions. Hence, the effort of manually abstracting from the model content in order to determine a label, is significantly reduced.

- **Enhanced Model Quality**: Apart from the fact that the proposed approach aims at providing labels which are suitable from a semantic perspective, the computed labels additionally fulfill structural quality criteria. As discussed in section 4.1 real world process models often contain various label styles. By consequently proposing

\(^1\)http://bpt.hpi.uni-potsdam.de/Oryx
verb-object labels, which have been proven to be superior to other label styles in regard to clarity and understandability, our approach also contributes to the quality and understanding of process models.

- **Complementation of Abstraction Approaches:** As pointed out in section 2.4, the current approaches for automatically implementing abstraction do not address the computation of activity labels although they are frequently required. Even though the proposed algorithm is not fully automated, the existing abstraction approaches could still be complemented with the presented labeling approach. By automatically selecting one particular label proposal from the list, the abstraction of process models could be fully automated. If the modeler considers a preselected label to be inappropriate, he may exchange it by choosing a more suitable label from the proposal list. In this vein, e.g. the algorithm for the creation of a process quick view [BRB07] could be implemented as an operation in modeling tools.

Summarizing the considerations, we can state that the developed approach can be utilized in different use cases and supports modelers to create consistent and understandable labels and process names. As a result, this thesis provided novel and beneficial contributions to the field of business process management and in particular to the field of business process model modularization, by employing natural language techniques.

Before we close this thesis, the next subsection discusses opportunities for future research.

### 6.3. Future Research

There are several aspects which could not be addressed in the context of this thesis. The most important task for future research is to conduct further validation of the proposed approach. In this context, additional model collections as well as the involvement of humans in the validation experiment should be addressed. Especially the latter aspect
is of primary importance, as the proposed approach can only be considered to be valuable if humans assess the results to be useful.

Besides the validation, the proposed algorithm needs to be improved with respect to the coverage of the semantic naming approach. Therefore, additional strategies have to be implemented which e.g. aim at deriving information from an ontology which reflects the required part-of relationships. As such ontologies might be hard to acquire, it may also be valuable to generate such ontologies from existing process model repositories.

Another aspect, which was not addressed in detail, is the problem of information overhead. It is necessary to keep in mind that a proposal list can only accelerate the labeling task if the resulting list contains a manageable count of labels. Hence, it would be valuable to adjust the focus from achieving a good recall in the sense of generating valuable labels, to ensuring a certain value of precision in order to increase the share of useful labels in the proposal list. This becomes especially important, when new sub approaches are incorporated, as they also entail additional label proposals. As a solution, relations between the label proposals could be investigated in order decide that one label proposal is inferior to another one. Consequently, label proposals could be ranked according to their importance or even dropped if they are considered to not add any value. Another solution could be to incorporate learning mechanisms. Thus, if a modeler frequently prefers a label provided by a specific sub approach, the label proposal could be ranked according to his preference profile.
References


REFERENCES


support features in process-aware information systems. pages 574–588.
2007.


A. Java Classes of Prototype Implementation

As pointed out during this thesis, the implementation of the prototype was conducted using Java and the JWNL API for WordNet. The final prototype implementation includes five packages:

- **Package epcModel** includes all classes which are required to properly represent an EPC model collection with Java.

- **Package phase1** contains the classes which implement the annotation of an EPC process model collection.

- **Package phase2** represents the main package as it contains the implementation of the labeling approach itself.

- **Package phase3** provides functionality for refactoring the raw labels from phase 2.

- **Package main** contains the class Main which coordinates the steps for applying the complete prototype implementation on the SAP Reference Model.

In the following sections we will introduce the major contents of the prototype packages. We will therefore describe the comprised classes and the most important functions they contain.

A.1. Package epcModel

The classes in the package epcModel are used to map the concepts of EPCs to according Java classes. Using these classes, we can adequately build up the structure of the EPCs of the SAP Reference Model which are stored in an EPML file (Event-Driven-Process-Chain-Markup-Language, for details refer to [MN06]), a format based on the XML. For this purpose, we introduce the following classes:
• **Class EPC_Item** is a generic class which serves as a superclass for EPC_Elem and EPC_Gateway.

• **Class EPC_Elem** can represent EPC functions or EPC events. The most important attribute of this class is given by the string *label*.

• **Class EPC_Gateway** accordingly represents EPC gateways. The type of the gateway is determined by the attribute *gateway_type* which may carry the values *or*, *xor* or *and*.

• **Class EPC_Connector** represents the arcs in EPCs and hence establishes the connection between two objects belonging to the class EPC_Item. Therefore, an object instantiated from EPC_Connector contains two attributes, named *source* and *target*, which are derived from the class EPC_Item.

• **Class EPC** refers to a complete EPC process model. It therefore consists of four lists. The two lists *functions* and *events* contain objects instantiated from EPC_Elem. A further list, named *gateways*, includes objects instantiated from EPC_Gateway and a fourth list, called *connections*, defines the arcs between these elements by using objects from the class EPC_Connector. In addition, the class EPC provides several functions for deriving information from the represented EPC process model. The most important functions are given by the *getSuccessors* and *getPredecessors* which accordingly return the successors and predecessors of a given EPC_Item object. On the basis of these functions it is possible to derive the start and end events of a given EPC process model.

• **Class EPC_ModelCollection** represents a whole model collection and thus contains a list *epcModels* comprising a set of EPC objects. In addition, the class provides the function *createEPCs* which automatically builds EPC objects from a given EPML file.
A.2. Package phase1

The most significant class of package phase1, realizing the the process model annotation, is given by the class ModelAnalyzer. Its function annotateModels implements the annotation of the models in two phases. As a first step, all verb-object labels are identified. Subsequently, the remaining labels, which are consequently assumed to be action-noun or rest labels, are analyzed and annotated according to their structure definitions. For further details on the methodology, please refer to [LSM10].

A.3. Package phase2

The phase2 package solely consists of the class LabelingApproach which implements the algorithms presented during this thesis and can therefore considered to be the main part of the implementation. Hence, we will introduce each function of this class and we will emphasize how the functions correspond to the algorithms presented in chapter 4. The following functions are provided by the LabelingApproach class:

- **Function getAnnotatedActivities**: loads the data about the annotated activities from phase 1. Thereby, the variable fromCSV decides whether the annotation is loaded from a given CSV file or actually computed. Note, that the latter option is very expensive if applied on a whole model collection as the SAP Reference Model.

- **Function getDominatingAction** implements the determination of the dominating action and thus corresponds to Algorithm 2.

- **Function getDominatingBusinessObject** accordingly implements the determination of the dominating business object and hence corresponds to Algorithm 2 as well.

- **Function extractSubordinateElements** identifies the subordinate elements for a given combination of process model and dominating element and consequently represents the implementation of Algorithm 3.
• **Function** *getLexicalConjunctions* conducts the lexical conjunction of the identified subordinate elements according to Algorithm 4.

• **Function** *getLogicalConjunctions* performs the logical conjunction of the identified subordinate elements according to Algorithm 5.

• **Function** *deriveFromLabelRepository* determines the repository labels based on a given label repository and a dominating element in line with the steps of Algorithm 6.

• **Function** *extractFromEvents* returns activities derived from the start and end events of the given process model as illustrated in Algorithm 7.

• **Function** *extractMainActivities* derives potential main activities from the given process model according to Algorithm 8.

• **Function** *getLabelProposalList* combines to the introduced functions in conformance with the steps presented in Algorithm 9.

A.4. Package *phase3*

The package *phase3* only contains the class *RefactoringManager* which provides the function *getRefactoredList*. With the help of this function a given set of labels is refactored to proper verb-object labels. As illustrated in Algorithm 10, the function relies on the annotation from phase 1. As a consequence, if a label has been annotated incorrectly, this function will return an inaccurately organized label.

A.5. Package *main*

Like package *phase3*, *main* only contains a single class, which is accordingly called *Main*. The class *Main* coordinates the invocation of the introduced classes and functions in order to perform the labeling task on the models of the SAP Reference Model. Therefore, it loads the data from the given EPML file and calls the function *createEPCs* in order to build up the structure of the SAP models. Subsequently, the function *annotateModels*
is used to annotate the label of the activities comprised on the created EPC objects. Afterwards, the labeling approach itself is applied on each EPC object. Using the function \textit{getProposalList}, potentially suitable labels are computed. For the purpose of the validation, this result is stored in the file \textit{output.txt}. However, the results can be simply processed by other programs, as for instance modeling tools, which accordingly offer the computed results to the modeler.
B. Structure of the Enclosed CD

On the enclosed CD the most important results of this can be found. Table B8 provides an overview about the contents of the CD.

Table B8: Content of the Enclosed CD

<table>
<thead>
<tr>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Validation/Validation - Results.xls</td>
<td>Excel file containing the results of the validation experiment. The file contains a complete summary (as provided in the thesis), an overview about all analyzed models and the actual output of the prototype.</td>
</tr>
<tr>
<td>/Prototype/Prototype.jar</td>
<td>Java Archive containing the whole Java project \textit{LabelPrototype} including all required sources.</td>
</tr>
<tr>
<td>/Prototype/epcModel/</td>
<td>Directory containing all Java classes belonging to the package \textit{epcModel}.</td>
</tr>
<tr>
<td>/Prototype/main/</td>
<td>Directory containing the package \textit{main} and thus the according class \textit{Main}.</td>
</tr>
<tr>
<td>/Prototype/phase1/</td>
<td>Directory containing all Java classes belonging to the package \textit{phase1}.</td>
</tr>
<tr>
<td>/Prototype/phase2/</td>
<td>Directory containing all Java classes belonging to the package \textit{phase2}.</td>
</tr>
<tr>
<td>/Prototype/phase3/</td>
<td>Directory containing all Java classes belonging to the package \textit{phase3}.</td>
</tr>
</tbody>
</table>
Declaration of Authorship

I hereby confirm that I have authored this Master’s thesis independently and without use of others than the indicated sources. All passages which are literally or in general matter taken out of publications or other sources are marked as such. I am aware of the examination regulations. Until now, I did neither submit nor finally failed a Master’s thesis within my course of studies.

Berlin, September 15th 2010

Henrik Leopold