WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

EFMD EQUIS ACCREDITED

Titel der Dissertation:

Linguistic Refactoring of Business Process Models

Dissertation zur Erlangung des akademischen Grades

einer Doktorin/eines Doktors

der Sozial- und Wirtschaftswissenschaften an der Wirtschaftsuniversität Wien

eingereicht bei

1. Beurteilerin/1. Beurteiler: **Univ.Prof. Dipl.-Wirt.Inform.Dr. Jan Mendling**

2. Beurteilerin/2. Beurteiler: **Prof. Dr. Mathias Weske**

von **Fabian Pittke, M. Sc.**

Fachgebiet: **Informationswirtschaft**

Wien, im **November 2015**

To Laura and to my family

Abstract

In the past decades, organizations had to face numerous challenges due to intensifying globalization and internationalization, shorter innovation cycles and growing IT support for business. Business process management is seen as a comprehensive approach to align business strategy, organization, controlling, and business activities to react flexibly to market changes. For this purpose, business process models are increasingly utilized to document and redesign relevant parts of the organization's business operations. Since companies tend to have a growing number of business process models stored in a process model repository, analysis techniques are required that assess the quality of these process models in an automatic fashion. While available techniques can easily check the formal content of a process model, there are only a few techniques available that analyze the natural language content of a process model. Therefore, techniques are required that address linguistic issues caused by the actual use of natural language. In order to close this gap, this doctoral thesis explicitly targets inconsistencies caused by natural language and investigates the potential of automatically detecting and resolving them under a linguistic perspective. In particular, this doctoral thesis provides the following contributions. First, it defines a classification framework that structures existing work on process model analysis and refactoring. Second, it introduces the notion of atomicity, which implements a strict consistency condition between the formal content and the textual content of a process model. Based on an explorative investigation, we reveal several reoccurring violation patterns are not compliant with the notion of atomicity. Third, this thesis proposes an automatic refactoring technique that formalizes the identified patterns to transform a non-atomic process models into an atomic one. Fourth, this thesis defines an automatic technique for detecting and refactoring synonyms and homonyms in process models, which is eventually useful to unify the terminology used in an organization. Fifth and finally, this thesis proposes a recommendation-based refactoring approach that addresses process models suffering from incompleteness and leading to several possible interpretations. The efficiency and usefulness of the proposed techniques is further evaluated by real-world process model repositories from various industries.

Zusammenfassung

In den vergangenen Jahren mussten sich Unternehmen vielen Herausforderungen aufgrund von Globalisierung, kürzer werdenden Innovationszyklen und zunehmender IT-Unterstützung stellen. Geschäftsprozessmanagement wird als ein umfassendes Werkzeug zur Ausrichtung von Geschäftsstrategie, Organisation, Controlling und operativer Geschäftstätigkeiten gesehen, um auf diese Marktänderungen flexibel zu reagieren. Dazu werden Prozessmodelle eingesetzt, die den relevanten Ausschnitt der Geschäftstätigkeiten dokumentieren und als Grundlage für eine Umstrukturierung dienen. Da Unternehmen eine immer größer werdende Zahl von Prozessmodellen pflegen, wird die Nachfrage nach automatischer Qualitätskontrolle zunehmend größer. Von Forschunsseite wurden diesbezüglich zahlreiche Techniken zur automatischen Qualitätssicherung hervorgebracht. Während bereits zahlreiche Forschungsansätze zur Analyse formaler Eigenschaften eines Prozessmodells existieren, gibt es nur wenige Beiträge, die auf eine Analyse von natürlicher Sprache abzielen. Insbesondere wurde die spezifische Verwendung von natürlicher Sprache und das Erkennen und Auflösen von sprachlichen Inkonsistenzen nur unzureichend untersucht. Um diese Forschungslücke zu schließen leistet diese Dissertation die folgenden Beiträge: Zum Ersten stellt sie einen Klassifikationsrahmen zur Einodnung bestehender bestehender Forschungsarbeiten bereit. Zum Zweiten wird in dieser Arbeit die Idee der Atomarität für Prozessmodelle vorgestellt, welche eine Konsistenzbedingung für den formalen und textuellen Inhalt eines Prozessmodells vorsieht. Basierend auf einer explorativen Untersuchung von mehreren Prozessmodellsammlungen wurden wiederkehrende Muster identifiziert, die das Kriterium der Atomarität verletzen. An dritter Stelle wird eine automatische Technik definiert, welche die zuvor identifizierten Muster formalisiert und nicht-atomare Prozessmodelle in atomare Prozessmodelle transformiert. Als Viertes stellt diese Dissertation eine Technik zur Erkennung und Korrektur semantischer Inkonsistenzen vor, welche die Nutzer bei der Vereinheitlichung der verwendeten Terminologie unterstützt. Zuletzt wird eine Technik präsentiert, welche unvollständig benannte Modellelemente korrigiert und somit unterschiedliche Interpretationen eines Modells verhindert. Alle vorgeschlagenen Techniken wurden mit Hilfe realer Modelle aus der Praxis evaluiert, um deren Effizienz und Nutzen zu demonstrieren.

Acknowledgments

On these pages I would like to thank all people who were involved in the creation of this doctoral thesis. First of all, I would like to express my deep gratitude to my supervisor Prof. Dr. Jan Mendling. I thank him for his excellent support, his constant interest in my research, his fruitful and continuous feedback with regard to my research, and his trust in my skills. In particular, I want to thank him for the possibility to work with him on interesting research projects and for the chance to give lectures at WU Vienna after I quit with Humboldt University and SRH University of Applied Science. Looking back on my journey, it is safe to say that I would not be here without him. Second, I want to express my sincere thanks to my second supervisor Prof. Dr. Mathias Weske. I want to thank him for his support and his feedback regarding my research. In particular, I am very grateful for offering me the chance to stay in his group at HPI and to provide me with a quiet and creative working environment to elaborate my research and my doctoral thesis. Without his generous offer I would not be here writing these final lines.

Moreover, I was lucky to collaborate with many excellent people in various contexts. At this stage, I want to particularly thank Henrik Leopold for our joint collaboration on various language-related topics, for our fruitful discussions, and for his incredible support in many regards. Further, I want to thank Pedro Piccoli Richetti and Fernanda Baião. I really enjoyed our delightful collaboration and our time on this year's BPM conference in Innsbruck. Furthermore, I thank my colleagues in Vienna and at HPI for always having an open door for me and for providing a friendly and creative working environment. Special thanks go to Thomas Baier, Felix Barbre, Adriatik Nikaj, Johannes Prescher, Saimir Bala, and again Pedro Piccoli Richetti for proofreading my thesis and related discussions. A thousand thanks for everything.

Meine letzten Dankesworte möchte ich gerne meiner Familie und meiner Freundin auf Deutsch widmen. Meinen Eltern und meiner Schwester danke ich für ihre nimmerendende Unterstützung und ihr Vertrauen in meine Fähigkeiten. Ohne sie und ihren Rückhalt, den sie mir stets gegeben haben, wäre diese Arbeit nicht denkbar gewesen. Nicht zuletzt gebührt Laura meine tiefste Dankbarkeit.

Sie gab mir stets die Liebe, Aufmunterung und unglaubliche Unterstützung, die ich brauchte, um diese Arbeit zu einem guten Ende führen zu können. Dies schließt insbesondere Deine unglaubliche Mühen mit ein, diese Arbeit vollständig zu lesen und an vielen Stellen zu korrigieren. Vielen Dank für Dein Verständnis und Deine unglaubliche Geduld in den letzten Wochen. Ich freue mich auf unsere gemeinsame Zukunft und auf alle Dinge, die ich gemeinsam mit Dir erleben werde.

Contents

List of Figures

List of Tables

Acronyms

AdjP	Adjective Phrase
AI	Artificial Intelligence
AIC	Academic Initiative Model Collection
ANC	American National Corpus
API	Application Programming Interface
ARIS	Architecture of Integrated Information Systems
BDI	Belief, Desire, and Intentions
BNC	British National Corpus
BPEL	Business Process Execution Language
BPM	Business Process Management
BPMN	Business Process Model and Notation
COCA	Corpus of Contemporary American English
Det	Determiner Phrase
EPC	Event-driven Process Chains
EPML	EPC Markup Language
GoM	Guidelines of Modeling
IMC	Insurance Model Collection
LIMAS	Linguistik und Maschinelle Sprachbearbeitung
LoLa	Low Level Petri Net Analyzer
NLP	Natural Language Processing
NP	Noun Phrase
OANC	Open American National Corpus
POS	Part-of-Speech
PrepP	Prepositional Phrase
SESE	Single-Entry-Single-Exit
SPMT	Structured Process Model Theory
SpW	Senses per Word
SRM	SAP Reference Model Collection
TC	Telecommunication Model Collection
UML	Unified Modeling Language
UMLDI	UML Diagram Interchange

VP	Verb Phrase
Woflan	Workflow Analyzer
WpS	Words per Sense
WSD	Word Sense Disambiguation
XMI	XML Metadata Interchange
XML	Extensible Markup Language

1 Introduction

This chapter provides an introduction of this doctoral thesis. Section 1.1 motivates the research presented in this thesis and emphasizes the general need for analysis techniques for the textual content of process models. Section 1.2 clarifies the research contributions of this thesis in order to address this need. In Section 1.3, we discuss the methodological background of this thesis, before Section 1.4 provides an outlook of the thesis structure. Finally, Section 1.5 lists the papers that have been published in the course of this thesis.

1.1 Motivation

This doctoral thesis is located in the field of Business Process Management (BPM). BPM has evolved to a mature discipline that enables organizations to flexibly react to emerging changes within their business environment. The relevance of BPM in practice is, for example, reflected by the growth of BPM-related products and services. A market analysis of Forrester estimates the BPM market growth with 18.7% reaching a volume of 7.5 billion US Dollars in 2016 [207]. Academia emphasizes the relevance of BPM with a number of important text books (see e.g. [407, 193, 408, 55, 437, 108, 179]) and dedicated conferences and conference tracks (see e.g. the International Conference on BPM [375] or the International Conference on advanced Information Systems Engineering [450]). The range of topics goes from organizational BPM aspects, such as BPM adoption, success factors, as well as technical aspects, such as the creation, implementation, and discovery of business process models. Due to the importance of business process models for documenting and redesigning business processes as well as designing and implementing process-related information systems, researchers have focused on topics with regard to process model design and quality. However, there are still open problems, for which hardly any solution has been proposed.

A specific problem in this regard is how to properly handle natural language in process models. Prior research clearly demonstrated the importance of natural language in process models in terms of understanding and modeling quality [285] which has been manifested in many naming recommendations from research and practice [395, 283, 399]. Moreover, prior research has facilitated the analysis of the textual content by recognizing the syntactic structure of model element labels [249] and rearranging them according to desired naming guidelines [238]. Nevertheless, these approaches provide only little insight into linguistic issues that are caused by the actual use of natural language to describe model elements such as activities and events. These linguistic issues cannot be resolved by the help of available techniques since they assume regular linguistic structures of model element labels which do not hold in a real-world setting [275, 285]. Issues that relate to the specification of process behavior with natural language, the usage of ambiguous terminology, or the incompleteness and underspecification of model elements remain unresolved. In consequence, the results of available techniques, especially those that rely on textual elements of process models (e.g. [101, 240, 245]), are not reliable and may result in wrong conclusions [144]. This problem is further intensified by taking into account that many organizations create several hundreds or thousands of process models and organize them in large process model repositories [369]. The amount of process models makes the manual maintenance more and more unfeasible. Therefore, there is a general need to extend prior research on model analysis and refactoring techniques to successfully handle language-related problems in process models.

1.2 Research Contributions

The goal of this doctoral thesis is the extension of model analysis and refactoring techniques by integrating and applying concepts from the branches of *Linguistics*. It links linguistic concepts and ideas in order to develop analysis techniques with regard to the branches of syntax, semantics, and pragmatics. In particular, this doctoral thesis is providing the following contributions:

- **Framework of Process Model Analysis**: So far, available process model analysis techniques emphasize the correctness of the formal part of the process model. The formal part relates to the modeling language which provides a set of symbols and rules to combine them. Although the natural language part of process models equally contributes to the overall quality [236], there are only partial solutions available. Therefore, this doctoral thesis will complement the idea of process model analysis and apply natural language processing (NLP) techniques to the textual part of process models. In Chapter 3, a classification framework is provided that integrates the linguistic branches with the well established conceptualization of a modeling language. The framework is further used to review prior research on process model analysis and to identify research gaps and emerging requirements. Additionally, future research approaches of process models analysis may be distinguished based on the provided framework.

- **Conceptualization of Atomicity for Process Models**: Many modeling initiatives in practice include casual modelers that are not sufficiently trained [369, 419] and thus use an arbitrary style of naming model elements [285]. While specific naming styles are automatically detected and refactored [238], there is a lack of recognizing and reworking element names, when modelers use natural language to express control-flow logic, such as parallel execution or decisions. Expressing control-flow logic with natural language makes it impossible to draw conclusions from formal analysis techniques and at least difficult to develop process-related systems based on such ill-defined requirements. This thesis will introduce the notion of atomicity that ensures a strict separation between modeling and natural language. In Chapter 5, the notion of atomicity is introduced and language patterns that violate this notion are identified. The notion of atomicity serves as a baseline for available language analysis techniques in process models and increases the reliability of their results.
- **Detection and Refactoring of Syntactic Ambiguity**: An initial analysis of process model collections from practice revealed that many process models suffer from syntactic ambiguities caused by non-atomic process model elements. This thesis formalizes the notion of atomicity and uses the non-atomicity patterns as an input in order to facilitate the refactoring of non-atomic process models and to provide automatic support. To this end, Chapter 5 also introduces an approach that automatically detects and refactors non-atomicity issues. Furthermore, this approach is experimentally evaluated by employing four real-world process model collections.
- **Detection and Refactoring of Semantic Ambiguity**: A fixed terminology represents an important asset in an organization to avoid misunderstandings and to ensure a precise communication of business goals and requirements. Additionally, in situations when organizations merge or acquire another one, the terminology may differ with regard to their meaning. Chapter 6 proposes an automatic technique for synonym and homonym detection and refactoring to facilitate the detection and refactoring of terminology ambiguity. Both techniques are evaluated by the help of real-world process models and native speakers.
- **Detection and Refactoring of Pragmatic Ambiguity**: According to Stachowiak [410], models are characterized by relevance and pragmatism. A process model is used by modelers as a substitution of the original object by reducing this object on the relevant parts. However, in many cases, process models do not provide sufficient details to specify the relevant parts of the underlying business process. Process models thus appear to be incomplete with regard to the naming of process model elements which makes it hard for model readers to understand and make sense out of the process model. Since process model might allow for several alternatives, this thesis proposes a recommendation-based approach that explicitly considers the context of the incomplete model element. The approach to refactor

incomplete model elements as well as the evaluation of these techniques is discussed in Chapter 7. As an overall result, the refactoring of incomplete elements is reduced to the task of selecting a candidate on a ranked list.

1.3 Methodological Background

The goal of information systems research is the acquisition of knowledge on how to understand and improve how socio-technical systems gather, process and present data, information and knowledge to users, in particular with regard to the organizational workplace. In order to acquire such knowledge, there are two complementary but distinct paradigms, behavioral science and design science [162, 146, 322]. *Behavioral science* seeks to develop and verify theories that explain or predict human or organizational phenomena surrounding the analysis, design, implementation, management, or use of information systems [162]. The resulting theories inform researchers and practitioners of the interactions among people, technology, and organizations that must be managed, so that the information system can achieve its stated purpose. *Design science* has its roots in engineering and the sciences of the artificial and is primarily a problem-solving paradigm [400, 162]. It seeks to create innovative artifacts that define ideas, practices, technical capabilities, and products through which the analysis, design, implementation, management, and use of information systems can be effectively and efficiently accomplished [162]. Examples of design artifacts are constructs, models, methods, and instantiations [263].

This doctoral thesis will employ both design science as well as behavioral science methods. While the contributions represent novel design science artifacts and provide a solution to the problem at hand, the evaluation of these artifacts uses methods from behavioral science, such as the formulation of hypotheses, the collection of suitable evaluation data and the test of hypotheses with statistical methods. Nevertheless, the main contributions are the result of design science research and thus need to address unsolved problems in a unique or in a more efficient way. Therefore, the contributions of this thesis are discussed with regard to the design science guidelines to illustrate their scientific character.

Hevner et al. [162] propose seven guidelines to conduct, evaluate and present design science contributions. This set of seven guidelines represents a widely-accepted reference to assist researchers as well as reviewers to understand and assess the value of design science artifacts. In the following, the contributions are related to these guidelines to discuss in how far they meet information systems research standards.

Guideline 1: Design as an Artifact. The result of design science research is, by definition, a purposeful IT artifact which is created to address a relevant organizational problem. It must be described effectively to enable its implementation and application in the respective domain in the form of a construct, a model, a method, or an instantiation [162]. In this thesis, existing process models analysis techniques are enriched with novel techniques inspired by branches of linguistics. This thesis formalizes syntactical language patterns, semantic ambiguity conditions and pragmatic completeness criteria as constructs that enable the detection of linguistic issues in process models. Furthermore, we define methods for the refactoring of syntactical, semantic, and pragmatic issues in process models. Lastly, each of these methods is implemented as a Java prototype which instantiates the aforementioned methods on the one hand and shows the applicability and feasibility of these methods for realistic scenarios.

Guideline 2: Problem Relevance. The relevance of any design science contribution is constituted by developing technology-based solutions to important business problems. These problems are given by the practitioners who are concerned with the planning, management, design, implementation, operation, and evaluation of information systems. Thus, design science research must address those problems faced and the opportunities afforded by the interaction of people, organizations, and information technology [162]. The general need of this thesis stems from the increasing adoption of BPM in organizations (see e.g. [344, 401, 181]) and the associated creation of several hundreds and thousands of process models [369]. In particular, the amount of such a big number of process models makes the manual maintenance and quality assurance a labor-intensive and time-intensive task. As a result, the techniques of this thesis contribute to the quality maintenance of process model collections and to the time-savings related to this task.

Guideline 3: Design Evaluation. Evaluation is a crucial component of the design science research process as it needs to rigorously demonstrate the utility, quality, and efficacy of a design science artifact. In order to evaluate the artifact, it is necessary to employ well-executed evaluation methods and metrics that show the benefits of the artifact with regard to the problem [162]. In this thesis, the utility of the proposed techniques is shown by employing them onto several process model collections from practice and measuring their utility with established metrics from information retrieval, namely precision, recall, and the f-measure [21]. On the one hand, we use precision and recall to measure the utility of the syntactical and pragmatic refactoring approach. Regarding the semantic refactoring we use a combination of precision and recall and the ambiguity metrics *Words per Sense* and *Senses per Word* which reflect the overall degree of homonym and synonym ambiguity in a process model collection.

Guideline 4: Research Contributions. Effective design science research must provide contributions based on the novelty, generality, and significance of the designed artifact. These contributions may address the design artifact itself as well as the design construction knowledge or the design evaluation knowledge [162]. The contributions of this thesis have already been presented in Section 1.2. They include a new classification framework for process models analysis techniques, a formalization of the atomicity notion enabling a consistent analysis of process models. Furthermore, this thesis proposes novel refactoring techniques for the textual elements in process models with regard to the syntactical, semantic, and pragmatic perspective. Each of these contributions are considered to be a significant contribution to the body of knowledge, i.e. techniques for the automatic quality assurance of process models.

Guideline 5: Research Rigor. Rigor addresses the way in which research is conducted in order to build and design the artifact. An essential requirement in this context is the effective use of the knowledge base and the associated foundations and methodologies. Therefore, this thesis constantly refers to existing knowledge, techniques and artifacts to build and evaluate the artifact. Among others, the techniques of this thesis make use of prior research on existing natural language parsers [199, 200, 409], process model parsers [249], as well as computational lexicons [292, 291, 304].

Guideline 6: Design as a Search Process. Design is essentially a search process for an effective solution of a problem that uses available means (actions and resources) to reach a set of desired goals by respecting laws within the environment [400]. Given the nature of many information systems design problems, it may however not possible to consider all relevant means, ends, or laws [424]. There might be several solutions to a problem which leads to a big and complex solution space with regard to one particular problem [162]. In such situations, it is recommended to search for satisfactory solutions without explicitly specifying all possible solutions [400]. The the design task involves the construction of an artifact that works well for a specified class of problems considering specific assumptions. Against this background, this thesis focuses on problems with regard to syntactical, semantic and pragmatic issues. For example, the semantic refactoring techniques concentrate on particular problems of semantic inconsistencies, namely synonymy and homonymy, and provides a working solution to reliably detect and resolve them. Nonetheless, it has to be noted that there might exist other semantic problems, for example with regard to hyponyms or meronyms. Therefore, it is important to identify and design one solution for which its utility and adequateness is demonstrated by a meaningful evaluation.

Guideline 7: Communication of Research. The results of the design science research process need to be be presented to both, technology-oriented as well as management-oriented audiences. Technology-oriented audiences need sufficient detail to enable the described artifact to be constructed and used within an appropriate organizational context. Management-oriented audiences need sufficient detail to determine if the organizational resources should be committed to constructing and using the artifact within their specific organizational context [162]. The results of this thesis have lead to one journal publication and five workshop and conference publications are thus already publicly available as part of the body of knowledge on process model analysis.

Relating the contributions of this thesis to the design science guidelines emphasizes that the thesis complies with international research standards and

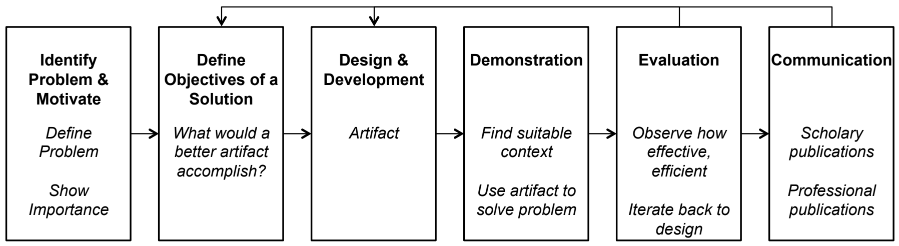

Identify Problem & Motivate
Define Problem
Show Importance
Define Objectives of a Solution
What would a better artifact accomplish?
Design & Development
Artifact
Demonstration
Find suitable context
Use artifact to solve problem
Evaluation
Observe how effective, efficient
Iterate back to design
Communication
Scholary publications
Professional publications

Fig. 1.1: Design Science Process, adapted from Peffers et al. [321, 322]

complements the body of knowledge of the information systems discipline with regard to innovative process model analysis techniques.

1.4 Thesis Structure

Before looking deeper into the chapters, it has to be noted that the presentation of the design artifacts follows an acknowledged procedure for design science research. In particular, the presentation uses the framework of Peffers et al. [321, 322] that provides a methodology for design science research and that meets the stated objectives, processes, and outputs. Figure 1.1 depicts the general process and its core activities. Accordingly, each of the design artifacts will further discuss the following activities:

1. **Problem identification and motivation:** This activity addresses the identification of the specific research problem and the justification of the value of its solution. This activity mainly accomplishes two things: first, it motivates the researcher and the audience of the research to strive for a solution, to accept the results, and to understand the reasoning associated with the researchers understanding of the problem [321]. Each technique of this paper implements this activity by carefully reviewing the body of knowledge with regard to the problem and the importance of its solution according to standard guidelines [435, 56].
2. **Objectives of a solution:** This activity is concerned with the definition of the objectives and requirements of a solution from the problem definition. In general, the objectives can be quantitative, e.g., terms in which a desirable solution would be better than current ones, or qualitative, e.g., where a new artifact is expected to support solutions to problems not addressed [321]. This activity is realized by identifying essential requirements of a solution which arise after the identification of the problem and the review of current body of knowledge.
3. **Design and development:** This activity involves the creation of the artifact, such as constructs, models, methods, or instantiations [162]. It also includes the specification of desired functionality, of underlying concepts,

and of the architecture [321]. This presentation of the artifacts realizes this activity by formalizing the essential concepts, methods, and algorithms of the respective artifact in detail which leverages the reconstruction of their functionality.

4. **Demonstration:** This activity demonstrates the efficacy of the artifact to solve the problem. It involves the deployment of the artifact in experiments, simulations, case studies, or other appropriate activities [321]. In many cases, demonstrating the artifact's capabilities is linked with the subsequent evaluation with the help of performance metrics. In this thesis, the developed artifacts are applied to real world process model collections to demonstrate their capabilities in realistic scenarios.
5. **Evaluation:** The evaluation observes and measures how well the artifact supports a solution to the problem. This activity involves comparing the objectives of a solution to the actual observed results of the demonstration. It further involves the definition of relevant metrics and analysis techniques that quantify the performance and the utility of the artifact or that compare it with existing solutions [321]. This thesis realizes this activity with the help of performance metrics from information retrieval and a comparison with existing solutions, if applicable.
6. **Communication:** The final activity is concerned with the communication of the relevant results of the preceding activities to researcher's and other relevant audiences, such as practicing professionals [321]. As already in the guideline discussion, the results of this thesis have already been published in journals as well as conference and workshop proceedings. Section 1.5 lists the respective publications.

This thesis is subdivided into nine chapters in total. It begins with an overview of the field of business process management and a discussion of prior research in process model analysis. Afterwards, it provides an introduction to the field of linguistics and elaborates on the employed ideas and concepts to address the identified research gaps. In the subsequent chapters, the thesis introduces the novel design artifacts to refactor process models according to the syntax, semantics, and pragmatics of their textual elements.

- **Chapter 1: Introduction**. In this chapter, we motivate the research topic of semiotic process model refactoring, highlight the specific research contributions, discuss the research design and methods, and present the related publications of this thesis.
- **Chapter 2: Business Process Management**. The next chapter provides an overview of the discipline of business process management and process modeling. Afterwards, we particularly focus on process models as the central concept of this thesis and operationalize them for the design artifacts.
- **Chapter 3: Analysis of Business Process Models**. In this chapter, we introduce the concept of process model refactoring and elaborate on different dimensions that may be target of such actions. Afterwards, the

chapter provides an extensive overview of recent analysis approaches and which dimension they address. Finally, we identify requirements, which arise from the review of recent approaches.

- **Chapter 4: Overview of Linguistics**. This chapter provides a short overview of the discipline of linguistics and the relevant subfields, namely syntax, semantics and pragmatics. Afterwards, we turn towards the field of natural language processing and introduce relevant techniques of this thesis, i.e. natural language text corpora, parsers, taggers, lexical databases, and finally word sense disambiguation.
- **Chapter 5: Syntactical Refactoring of Process Models**. This chapter presents an artifact for the syntactical refactoring of process models. As a basis, we introduce the notion of atomicity and identify several patterns that violate this notion by using three large process model collections from practice. Building on these insights, we define an approach that can detect and rework non-atomic process models. The capabilities of this technique are evaluated with four real-world process model collections containing more than 2800 process models.
- **Chapter 6: Semantic Refactoring of Process Models**. This chapter introduces the artifact for the semantic refactoring of ambiguities in process models. To this end, we operationalize the concept of a word sense and formulate reliable conditions for synonyms and homonyms. Afterwards, we present the techniques to detect and refactor ambiguous textual elements in process models. Again, we apply the techniques on process model collections from practice and evaluate their performance by comparing the performance of the novel artifact with a naive approach of ambiguity detection and refactoring.
- **Chapter 7: Pragmatic Refactoring of Process Models**. This chapter presents the artifact for the pragmatic refactoring. This artifact is built upon on the insight that syntactically and semantically refactored process model elements may still be imprecise with regard to the overall context of the process model. We propose a recommendation-based approach that supports the detection and refactoring of imprecisely labeled model elements which is again evaluated with available process model collections from practice.
- **Chapter 8: Conclusion**. Finally, this chapter summarizes the results of this thesis and gives an outlook of future research activities. Further, it discusses the implications of the findings for the linguistic analysis in conceptual models and for the quality of business process models.

1.5 Related Publications

This thesis has lead to several publications that are related to the analysis and the refactoring of process models. The following list provides an overview of accepted publications and submitted work.

Publications related to Syntactical Refactoring

- Fabian Pittke, Henrik Leopold, Jan Mendling: When Language meets Language: Anti Patterns Resulting from Mixing Natural and Modeling Language. In: 5th International Workshop on Process Model Collections: Management and Reuse (PMC-MR 2014), 118-129, 2014. [330]
- Fabian Pittke, Benjamin Nagel, Gregor Engels, Jan Mendling: Linguistic Consistency of Goal Models. In Proceedings of the 19th International EMMSAD Conference, 393-407, 2014. [333]

Publications related to Semantic Refactoring

- Fabian Pittke, Henrik Leopold, Jan Mendling: Automatic Detection and Resolution of Lexical Ambiguity in Process Models. IEEE Transactions on Software Engineering. 41(6): 526-544, 2015. [331]
- Fabian Pittke, Henrik Leopold, Jan Mendling: Spotting Terminology Deficiencies in Process Model Repositories. In: BPMDS13 Working Conference (BPMDS 2013), 292-307, 2013. [329]

Publications related to Pragmatic Refactoring

- Fabian Pittke, Pedro Henrique Piccoli Richetti, Jan Mendling, Fernanda Araujo Baião: Context-Sensitive Textual Recommendations for Incomplete Process Model Elements. In: International Conference on Business Process Management (BPM 2015), 189-197, 2015. [334]

Publications related to Process Modeling and Language Analysis

- Henrik Leopold, Fabian Pittke, Jan Mendling: Automatic Service Derivation from Business Process Model Repositories via Semantic Technology. Journal of Systems and Software. 108(1): 134147, 2015. [247]
- Henrik Leopold, Christian Meilicke, Michael Fellmann, Fabian Pittke, Heiner Stuckenschmidt, Jan Mendling: Towards the Automated Annotation of Process Models. In 27th International Conference on Advanced Information Systems Engineering (CAISE 2015), 401-416, 2015. [239]
- Jan Mendling, Henrik Leopold, Fabian Pittke: 25 Challenges of Semantic Process Modeling. International Journal of Information Systems and Software Engineering for Big Companies 1(1): 78-94, 2015. [278]
- Norbert Ahrend, Henrik Leopold, Fabian Pittke: Barriers and Strategies of Process Knowledge Sharing in Public Sector Organizations. In: Multikonferenz Wirtschaftsinformatik (MKWI 2014), February 26-28, 2014, Paderborn, Germany. [14]
- Henrik Leopold, Fabian Pittke, Jan Mendling: Towards Measuring Process Model Granularity via Natural Language Analysis. In: 4th International Workshop on Process Model Collections: Management and Reuse (PMC-MR 2013), 417-429, 2013. [246]

- Fabian Pittke, Henrik Leopold, Jan Mendling, Gerrit Tamm: Enabling Reuse of Process Models trough the Detection of Similar Process Parts. In: 3rd International Workshop on Reuse in Business Process Management (rBPM 2012), 586-597, 2012. [332]

2

Business Process Management

This chapter provides an introduction to business process management and business process models. First, Section 2.1 provides an overview of business process management, its core concepts, and recognizes business process models as being a central artifact. Section 2.2 particularly elaborates on the modeling process and on the building blocks of process models. Based on these insights, Section 2.3 provides a precise characterization of process models. Finally, Section 2.4 summarizes the most important things of this chapter.

2.1 Overview of Business Process Management

The roots of BPM date back to the 18th century. As one of the very first persons, Adam Smith illustrated the benefits of a process-oriented manufacturing of products in his example of a pin factory [406]. The output of the manufacturing process could be increased by subdividing it into several steps that can be done by the most appropriate workers. Following this idea, Frederick Winslow Taylor proposed a systematic analysis to identify the best way to perform these steps [415, p. 117] by "selecting the quickest way [..], eliminating all false movements, slow movements, and useless movements [..], and collecting into one series the quickest and best movements". As an implementation of this type of organization, Henry Ford invented the *assembly line*. He reports on a notable reduction of the assembly time of manufacturing a flywheel magneto in an initial experiment. As a result, the overall time was reduced by more than 75% and the idea was applied to all aspects of car manufacturing [130, pp. 80-90].

From an academic perspective, Nordsieck was one of the first researchers who pointed out the general necessity of a process-oriented organizational design [309, p. 9]. He described several types of workflow diagrams for the subdivision of labor, the identification of activity sequences and the assignment of tasks to people. However, the adoption of the proposed design took place much later since the focus of organizational research put more emphasis on the structural aspect (see e.g. [213, 148]).

With increasing relevance of office automation, a stronger focus has been put on the flow of information and the need of understanding the underlying processes [438, 116] and arrange them in a value-creating chain that delivers a product to the market as advocated by Porter [342]. The value chain model enables the identification activity sequences, which consequently lead to reengineering approaches to achieve productivity, competitiveness and process innovation by technology [87]. In this context, Kaizen [175] and Business Process Reengineering [154] have been proposed as specific process reengineering methodologies. Building on these management concepts and the insights on the role of integrated information systems by Scheer [382], companies widely started to adopt process orientation.

Today, business process management has become a well-established approach in research and practice alike. Due to its roots in organizational theories and due to the increasing technology adoption, it has evolved to a inter-disciplinary field that combines techniques and approaches from business administration, organizational theory, computer science and information systems research. These insights have been adopted by numerous companies and established a market for BPM software solutions and consultancy.

Before we elaborate on the modern understanding of BPM, it is useful to have a look on business processes first, i.e. the core artifact of BPM. The different influences of business process management are also reflected by the various definitions of *business processes*. Thereby, these definitions emphasize different facets of business processes. Nordsieck introduces a business process as being a sequence of activities to produce an output. In his definition, an activity is the smallest unit of work that is performed by people [309, pp. 27-29]. Similarly, Hammer and Champy understand a business process as a collection of activities that transforms several inputs into a defined output. Additionally, the authors explicitly emphasize the customer value of the products [154, p. 38]. Davenport [87, p. 5] extends the previous definitions and includes a particular order and specific conditions of the activities as being necessary to produce the output.

Other authors also emphasize the relevance of the business strategy [386, pp. 4-5] and the use of information and communication technology [137] to perform the process tasks. A comprehensive definition has been provided by Weske [437, p. 5] who combines the aforementioned characteristics:

Definition 2.1. (**Business Process**). A *business process* consists of a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations.

Based on these definitions, BPM can be described as summarizing all managerial activities that are related to business processes. It integrates

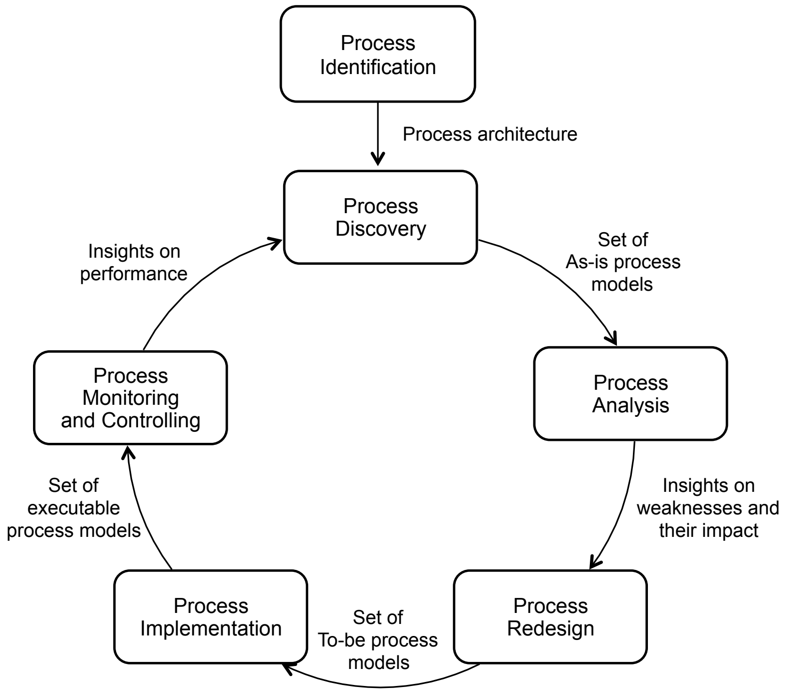

Process Identification
Process architecture
Process Discovery
Set of As-is process models
Process Analysis
Insights on weaknesses and their impact
Process Redesign
Set of To-be process models
Process Implementation
Set of executable process models
Process Monitoring and Controlling
Insights on performance

Fig. 2.1: Business Process Management Life Cycle, adapted from [108, p. 21]

management, organization, controlling and improvement of business processes to fulfill customer needs and to contribute to the strategic and operational goals of the organization [386, pp. 4-5]. Ideally, these management activities related to business processes are arranged in a life cycle as proposed by many authors (e.g. [5, 298, 55, 437, 108]). This doctoral thesis takes the life cycle of Dumas et al. [108, p. 21] into account, because it is particularly concerned with business process models as an artifact. Figure 2.1 depicts the aforementioned life cycle including all artifacts as results of each life cycle activity. The life cycle comprises the management activities of identification, discovery, analysis, redesign, implementation, as well as monitoring and controlling:

- *Process Identification*: The life cycle is entered by an initial analysis activity. In this phase, a business problem is posed and affected processes are identified and related to each other. This phase results in a process architecture that provides a comprehensive view of the organization's processes and their interrelationships.
- *Process Discovery*: This phase is also termed *as-is modeling* or process design and describes the activity of documenting the current state of all affected processes. In most of the cases, a set of as-is process models are created for the subsequent analysis.

- *Process Analysis*: In this phase, the previously developed process models are inquired with regard to specific issues causing the business problem. These issues are then documented and quantified by performance measures, if possible. As a result, this phase produces a structured overview of all issues and weaknesses that can be prioritized in terms of impact and resolution efforts.
- *Process Redesign*: The process redesign phase identifies feasible changes to the current processes that solve the identified problems on the one hand and that allow the organization to fulfill its goals on the other hand. Therefore, several scenarios are elaborated and compared according to the performance measures. This implies that process redesign and process analysis complement each other as the scenarios are evaluated based on different analysis techniques. The output of this phase is a set of improved to-be process models.
- *Process Implementation*: This phase prepares and conducts necessary changes in order to transform as-is processes into to-be processes. Typically, process implementation involves organizational and infrastructural change. Organizational change covers all activities that are related to change management, while infrastructural change is concerned with the development and deployment of IT systems that support the execution of to-be processes. This phase thus results in a set of executable process models.
- *Process Monitoring and Controlling*: Finally, after the implementation of the new processes, process-related data is collected and analyzed to quantify the process performance with regard to measures and objectives. Hence, it is possible to identify and correct recurrent errors or minor deviations. In case of major issues, the life cycle needs to be repeated again with the performance insights gained from monitoring the processes.

The life cycle shows that process models play an important role in several phases, such as discovery, analysis, and redesign. These process models serve as a valuable artifact to document the current process status within the organization, to analyze the processes with regard to issues and problems, and to design an improved version of the processes. Therefore, the next section further elaborates on business process models and business process modeling.

2.2 Overview of Business Process Models

Before we turn to the business process model as the central artifact of this thesis, we need to discuss the principles of general models. Based on these principles, it is possible to extend the discussion to business process models and to provide a definition (Section 2.2.1). Afterwards, we turn to the creation of process models (Section 2.2.2) and their essential building blocks (Section 2.2.3).

2.2.1 Principles of Models

From a general viewpoint, a business process model is a specific type of a *model*. The word *model* originates from the Latin word *modulus* and was frequently used in the Renaissance to describe an architectural benchmark. In modern times, the meaning of the term changed and refers to a scheme or blueprint of real objects [371]. Thereby, models have three characteristics as identified by Stachowiak [410, pp. 131-132]:

- *Mapping*: Models are mappings of naturally existing or artificial originals, which can be models by themselves.
- *Reduction*: Generally, a model does not include all attributes and characteristics of the originals. Instead, it only considers those that are relevant for model users.
- *Pragmatism*: The model is used by a modeler as a substitution of the original for a certain time and for a certain purpose.

With these characteristics, a model is defined as a simplified mapping of an original, which was created at a specific point of time and for a specific purpose. Moreover, this conceptualization has been widely applied by numerous seminal works, such as (organizational) system engineering [216, 215], enterprise architectures [314], meta-models and ontologies [160], and software engineering [258, 272].

However, it needs to be mentioned that this conceptualization is not free of criticism. Authors criticize that Stachowiak's conceptualization of a model abstracts from the subjective perception of the modeler [390, 355, 274]. Among those, Schuette and Rotthowe [390, p. 242] argue that, if a model had some similarity with the reality, we would have to assume that there is a reality independent from the observer. In consequence, we would also have to assume single objectivity of reality that is inherent to the epistomological position of positivism. Hence, only if each subjective observation of the reality corresponded exactly to reality itself, a mapping-driven definition of the term model would be acceptable. Instead, Schuette and Rotthowe propose an alternative position that regards a model as the result of a construct of a modeler who examines the elements of a system for a specific purpose at a given point of time with a specific language [390, p. 243]. The authors propose a shift from the formal mapping as proposed by Stachowiak towards a construct of the modeler. Considering a model to be a mapping of an object as subjectively perceived by a person and taking the definition of Mendling [274] into account, a *business process model* is defined as follows:

Definition 2.2. (**Business Process Model**). A business process model is the result of creating a mapping of a business process as perceived by a modeler. The business process can be a real-world process, or a process which has been conceptualized by the modeler.

The definition shows the dependency of a business process model on the perception of a modeler. In consequence, different modelers will create different models of the same underlying process which makes model creation a complex task [365]. The complexity of the modeling task arises due to the fact that the model itself has to comply with certain rules and that each perception has to be reconciled to represent the original object [134]. Hence, there is a demand for model creation rules to achieve comparable results.

In order to achieve comparability and inter-subjectivity among process models, it is necessary to require specific modeling rules and guidelines. These guidelines facilitate the comprehensibility of the model creation process and leverage a simplified integration of different models into an enterprise model. Ultimately, the danger of a defective integration can be reduced [390]. For that purpose, we can apply the Guidelines of Modeling (GoM) [33, 390, 34] which contain six principles to enhance the quality of process models. These are the principles of correctness, relevance, economic efficiency, clarity, comparability, and systematic design. While the first three principles are mandatory for model quality, the remaining three principles are considered to be optional.

- **Correctness**: The guideline of correctness involves two aspects. First, the model needs to be correct in terms of syntax, i.e. the consistent use of constructs from the employed modeling notation. Second, the model needs to be correct with regard to semantics, i.e. the structure and the behavior of the model is consistent with the underlying business process.
- **Relevance**: The guideline of relevance demands that only those objects are represented in the model that are crucial to the universe of discourse and that they are presented by a suitable modeling notation.
- **Economic Efficiency**: This guideline imposes an economic restriction on process models and tries to prevent an over-estimation of the other guidelines. Hence, it directly relates to the benefits and costs of including specific aspects of the universe of discourse into the model. For example, it might be acceptable to not include all elements of the universe of discourse (and to violate correctness or clarity), if time and costs exceed defined limits.
- **Clarity**: The guideline of clarity is concerned with the readability and the understandability of the process model by different persons. Clarity is extremely subjective since the understandability depends on the person's background and knowledge. For these reasons, clarity is mainly achieved by layout or naming conventions.
- **Comparability**: As its name implies, comparability demands the consistent use of all guidelines within a modeling project. This guideline may refer to layout and naming conventions that are consistently applied on all process models of such a modeling project. As a result, it is possible to use process models for inter-model comparisons and draw meaningful conclusions from them.
- **Systematic Design**: The guideline of systematic design requires well-defined relationships between models that belong to different views, for instance process models and data models. A prominent example that

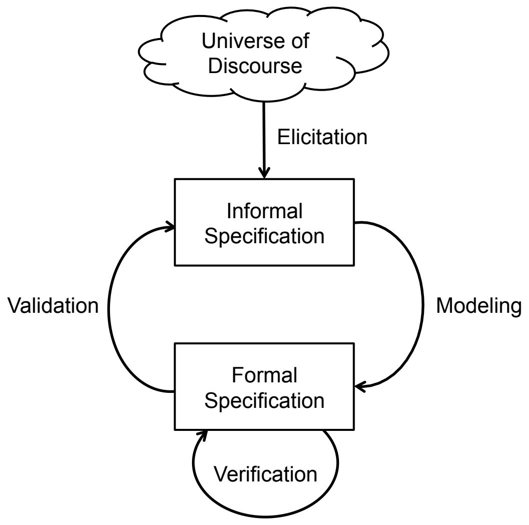


Fig. 2.2: Information Modeling Process, adapted from [134, p. 9]

implements this guideline is the ARIS approach [383, 384, 385] which provides models for specific views as well as a model that presents the integration of these views.

Despite the guidance that is provided by the GoM and specific process modeling guidelines (see e.g. [283, 395, 261]), a large number of process models, in particular in practice, still do not comply with these guidelines [275, 285]. Thus, it is necessary to rework these models in order to ensure the overall model quality as intended by the GoM. This is of particular importance, if the process models are used for analysis and simulation scenarios [98, 108] or for process implementation [109, 37]. Hence, we need to discuss the process of business process modeling and their building blocks in order to understand how the guidelines affect the reworking of process models.

2.2.2 Process of Business Process Modeling

A prominent example of creating process models is given by Frederiks and van der Weide [134] who investigate the process of information modeling, the required competencies of its participants and determinants of its quality. The authors propose four core activities that are arranged in a cycle. This procedure also applies to the creation of process models since they are a subclass of information models. Moreover, this procedure is also employed by prior research approaches related to process models (see e.g. [275, 248, 61, 327, 328, 432, 360]). Figure 2.2 illustrates the four activities, i.e. elicitation, modeling, verification and validation.

The cycle is initiated by the *elicitation* phase. In this phase, the modeler collects all relevant information objects from the universe of the discourse, which typically refers to the application domain. Afterwards, the identified objects need to be verbalized in a common language. In the next step, the initial specifications are rewritten into a unifying format which results in a semi-formal specification of the application domain. Typically, the informal specification is a natural language text which is understandable by all stakeholders [10]. The *modeling* phase is concerned with the transformation of the informal specification into a formal specification. Hence, the modelers need to discover the significant modeling concepts and their relationships. Afterwards, they need to match sentence structure on modeling concepts, i.e. the elements that are provided by the selected modeling notation. This phase results in a formal specification of the domain. In the phase of *verification*, the formal specification is checked for internal consistency. In case of process models, this may include a check for syntactical errors of the modeling notation [4, 274, 286], a control flow check for soundness and other properties [41, 122, 397], or a check for naming conventions [238]. In the final *validation* step, the model is assessed with regard to the consistency with the universe of discourse which often requires the consultation and the discussion with process stakeholders. Hence, a textual description of the conceptual model is created by paraphrasing the model with the help of information grammars and a lexicon. The textual description is then compared with the informal specification. However, the validation remains a very ambiguous task which requires the stakeholders to find a mapping of the paraphrased text and the informal specification. Therefore, this step cannot be solved by algorithms [394].

The process of modeling has been further investigated by current research and provided further insights. The modeling process has been studied by using specific tools that track the interaction of a modeler with a modeling tool [326]. This tracking produces event logs that can be analyzed using process mining techniques [79]. It has been found that good models are correlated with certain behavioral patterns of modeling [78]. Moreover, the structured process modeling theory (SPMT) also identifies individual fitting and serialization to positively influence the resulting process model [77]. In consequence, a good process model reduces the efforts for the subsequent activities of verification and validation.

However, process modeling is often conducted in an unstructured way [78] and requires a rework of the final models, such that verification and validation of process models is still necessary to achieve high quality models. Closing the circle to the GoM, verification leverages the guidelines of correctness, systematic design, and comparability, while validation techniques are inline with clarity and relevance. Based on the guidelines, verification and validation techniques will uncover inconsistencies in process models. However, these inconsistencies may be rooted in different aspects of process models which we discuss as the building blocks of process models in the next section.

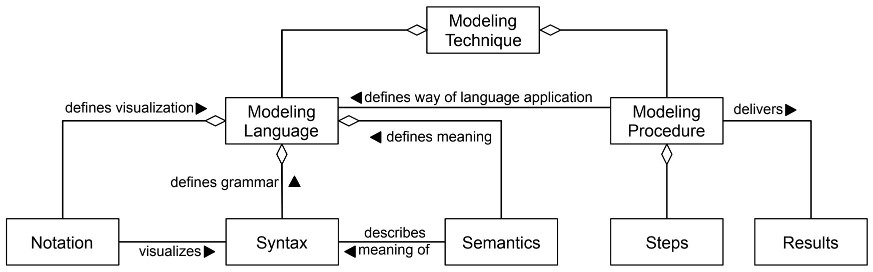


Fig. 2.3: Conceptualization of a Modeling Technique, adapted from [189]

2.2.3 Building Blocks of Process Models

In order to understand the building blocks, it is helpful to start with the notion of a *modeling technique* as depicted in Figure 2.3. Karagiannis and Kühn [189] identify two major parts of a modeling technique, i.e. a modeling procedure and a modeling language. On the one hand, the *modeling procedure* describes the necessary steps of using the modeling language to create conceptual models. As an example of a modeling technique, the framework of Frederiks and van der Weide [134] has been discussed in the previous section. On the other hand, the *modeling language* defines the elements with which the model is built. Generally, any graphical modeling language is described by its syntax, semantics, and notation. The syntax provides a set of constructs and rules to combine the model constructs and to create models according to the model grammar. The semantics describe the meaning of a modeling language and its constructs. Finally, the notation defines the visualization of a modeling language and provides a set of graphical symbols that represent the constructs along with their meaning. Examples of process model languages are the Business Process Model and Notation (BPMN) [312], the Event-driven Process Chains (EPC) [384], and Petri Nets [324]. Each of these modeling languages provides graphical symbols with a specific syntax and semantics to be combined according to specific rules. Therefore, we consider the notation, the syntax and the semantics as the building blocks of any process modeling language.

One aspect that is not covered by this framework is the textual annotation of process model constructs. The task of *labeling* is, however, important as it determines the intuitive understanding of process models [285] (principle of clarity) and thus contributes to the overall quality [220]. Moreover, these labels are further required for several searching [110, 222, 444, 445] or matching approaches [436, 245, 202]. It does not make much sense to base a mapping of process models only on behavioral characteristics of the process models because they might not necessarily stem from the same business domain and depict a completely different business process [203]. Indeed, recent research has shown that the labels of model constructs convey a large share of the overall

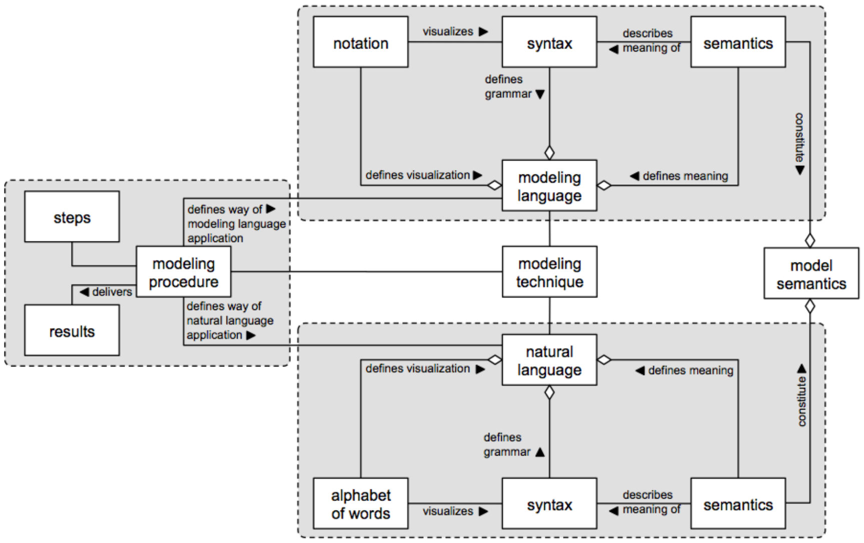


Fig. 2.4: Conceptualization of a Modeling Technique Including Natural Language [238]

model semantics [238, 203] which requires an extension of the Karagiannis' and Kühn's framework with natural language.

This extension has been conceptualized by Leopold [238, p. 12] who defines the overall semantics of a process model to be the combination of a modeling language and a natural language. Hence, modeling language and natural language are independent concepts which are combined for the purpose of process modeling. In consequence, the modeling procedure does not only define rules and guidelines to properly use the modeling language, but also to properly describe model constructs with natural language. A *natural language* is based on an alphabet which comprises a finite set of words to express a specific scenario. *Syntax* defines rules on how these words should be combined to form proper sentences. *Semantics* defines the meaning of words. Reconsidering the building blocks of process models, it is apparent that natural language contributes equally to the overall semantics of a process model and that it also needs to be part of process models. Consequently, verification and validation also need to take the natural language into account in order to improve the overall quality of process models.

2.3 Conceptualization of Business Process Models

Prior conceptualizations of process models describe their characteristics from a structural and behavioral perspective. Structural formalizations are concerned with the symbols of a notation and describe formal rules and relations on how to combine these symbols. The behavioral formalizations of process models refer to the execution semantics and describe the behavior of the process model in an information system. In this context, the behavior of process models is described by the help of Petri Nets [2]. While this thesis will focus on the structural dimension of process models, this section will only elaborate a structural conceptualization which corresponds to the syntax of a modeling language according to Karagiannis and Kühn [189].

Prior research has introduced syntactical process model definitions that are dependent on specific modeling notations, such as BPMN or EPC. For the EPC notation, van der Aalst [3], La Rosa et al. [224], Mendling [274], and van Dongen et al. [106] give formal definitions of EPCs. Likewise, Dijkman et al. [104] formally describe the syntax of BPMN process models. In general, all these definitions agree on the fact that a process model consists of a set of activities, events, and control flow gateways as well as a binary relation which is a directed arc between these elements. These common characteristics have been aggregated into an abstract syntactical definition of process models by several authors, such as La Rosa et al. [225], Ouyang et al. [316, 317], Polyvyanyy et al. [339, 340], Weske [437], and Leopold [238]. Indeed, the abstract definition is more useful in the context of this thesis because it presents techniques that are independent from particular modeling languages.

The abstract definitions of process models share common properties that are beneficial for a definition of process models in an operational sense. La Rosa et al. [225] provide a meta model of process models to leverage the advanced functionality of the APROMORE model repository. The meta model covers all necessary elements of a process model, but is not specific enough to define a process model in an operational sense. The proposed definitions of Ouyang et al. [316, 317], Polyvyanyy et al. [339, 340], and Weske [437] are, by far, more specific and propose a format that distinguishes between activities, events, gateways, and control flow relations. Among the first authors, Dijkman et al. [102, 103] introduce a process model definition that also considers the labeling of elements to be an essential part. These ideas have been adopted by Leopold [238] who provides a comprehensive definition with explicit consideration of element labeling. This thesis makes use of the definitions given by Leopold [238] as it aims for an explicit analysis of the natural language labels. Accordingly, a process model is defined as follows:

Definition 2.3. (**Process Model**). A process model P = (A, E, G, F, R, P, L, ρ, π, λ, γ, τ) consists of six finite sets A, E, G, R, P, L, a binary relation $F \subseteq (A \cup E \cup G) \times (A \cup E \cup G)$, a surjective function $\rho : A \mapsto R$, a surjective function $\pi : R \mapsto P$, a partial function $\lambda : (A \cup E \cup G \cup F \cup R) \mapsto L$, a function $\gamma : G \mapsto \{and_S, and_J, or_S, or_J, xor_S^D, xor_S^E, xor_J\}$, and a function $\tau : E_{int} \mapsto A$, such that

- A is a finite non-empty set of activities.
- E is a finite set of events.
- G is a finite set of gateways.
- N denotes all nodes of the process model, i.e., $N = A \cup E \cup G$.
- R is a finite set of resources.
- P is a finite set of pools.
- L is a finite set of text labels.
- F is a finite set of sequence flows. Each sequence flow $f \in F$ represents a directed edge between element types.
- U represents all units of the process model which can carry a label, such that $U = N \cup F \cup R \cup P$.
- The surjective function ρ specifies the assignment of a resource $r \in R$ to an activity $a \in A$.
- The surjective function π specifies the assignment of a resource $r \in R$ to a pool $p \in P$.
- The partial function λ defines the assignment of a label $l \in L$ to a process model unit $u \in U$.
- The function γ specifies the type of a gateway $g \in G$ as and_S, and_J, or_S, or_J, xor_S^D, xor_S^E, xor_J. The subscripts $_J$ and $_S$ denote joins and splits, and the superscripts ^{D} and ^{E} denote the distinction between data and event-based split gateways.
- The function τ assigns an intermediate event $e_{int} \in E_{int}$ to an activity (attached event). If such an event occurs, the execution of the respective activity is interrupted.

We illustrate the previous definition for an exemplary process model, before we define further properties of process models. Consider the depicted job application process of Figure 2.5. The process is started after an application was received by a company. In the subsequent step, the application is checked for completeness. In case that the application documents are incomplete, the missing documents are requested from the applicant. Afterwards, these documents are evaluated by the help of a particular software application. Finally, a decision has to be taken which either results in accepting or rejecting the applicant. This last step finishes the application process.

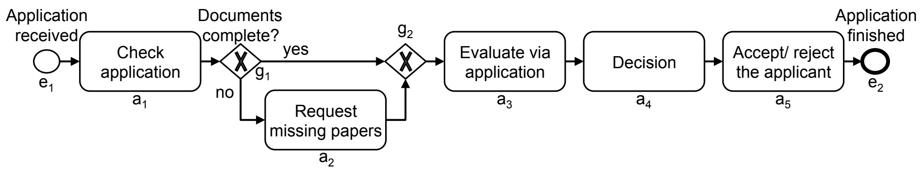


Fig. 2.5: An Example for a Job Application Process

According to definition 2.3, the job application process is described by P = $(A, E, G, F, R, P, L, \rho, \pi, \lambda, \gamma, \tau)$ such that

$$
\begin{aligned}
A &= \{a_1, a_2, a_3, a_4, a_5\}\\
E &= \{e_1, e_2\}\\
G &= \{g_1, g_2\}\\
F &= \{(e_1,a_1),(a_1,g_1),(g_1,g_2),(g_1,a_2),(a_2,g_2),(g_2,a_3),(a_3,a_4),\\
&\quad (a_4,a_5),(a_5,e_2)\}\\
R &= \emptyset\\
P &= \emptyset\\
L &= \{\text{Check application, Request missing papers, Evaluate via}\\
&\quad \text{application, Decision, Accept/reject the applicant, Application}\\
&\quad \text{received, Application finished, Documents complete?}\}\\
\lambda(a_1) &= \text{Check application, } \lambda(a_2) = \text{ Request missing papers}\\
\lambda(a_3) &= \text{Evaluate via application, } \lambda(a_4) = \text{ Decision}\\
\lambda(a_5) &= \text{Accept/ reject the applicant}\\
\lambda(e_1) &= \text{Application received, } \lambda(e_2) = \text{ Application finished}\\
\lambda(g_1) &= \text{Documents complete?}\\
\lambda((g_1,g_2)) &= \text{yes}\\
\lambda((g_1,a_2)) &= \text{no}\\
\gamma(g_1) &= xor_S^E,\ \gamma(g_2) = xor_J
\end{aligned}
$$

This thesis also uses existing notations for predecessors and successors of nodes, incoming and outgoing flows, paths [237] in order to allow for a more concise specification of process models and to facilitate the conceptualization of the techniques.

Definition 2.4. (**Predecessors and Successors of Nodes**). Let N be a set of nodes of a process model P and $F \subseteq N \times N$ a binary relation over N representing the sequence flows. For each node $n \in N$, the set of *preceding nodes* is given by $\bullet n = \{x \in N \mid (x,n) \in F\}$, and accordingly the set of *successing nodes* is given by $n\bullet = \{x \in N \mid (n,x) \in F\}$.

Definition 2.5. (**Incoming and Outgoing Flows**). Let N be a set of nodes of a process model P and $F \subseteq N \times N$ a binary relation over N representing the sequence flows. For each node $n \in N$, the set of *incoming flows* is given by $n^{in} = \{(x, n) \mid x \in N \wedge (x, n) \in F\}$, and accordingly the set of *outgoing flows* is given by $n^{out} = \{(n, x) \mid x \in N \wedge (n, x) \in F\}$.

Definition 2.6. (**Path**). Let P = (A, E, G, F, R, P, L, ρ, π, λ, γ, τ) be a process model, and N a set of nodes. There is a *path* between two nodes $x \in N$ and $y \in N$, denoted with $x \rightsquigarrow y$, if there exists a sequence of nodes $n_1, ..., n_k \in N$ with $x = n_1$ and $y = n_k$ such that for all $i \in 1, ..., k-1$ holds: $(n_i, n_{i+1}) \in F$.

Once again, the Definitions 2.4 and 2.5 are illustrated by referring to the job application process depicted in Figure 2.5. Consider the gateway $g_1 \in G \subseteq N$. Then, the set of predecessors ($\bullet g_1$) and successors ($g_1 \bullet$) as well as the sets of incoming flows (g_1^{in}) and outgoing flows (g_1^{out}) is given as follows:

- $\bullet g_1 = \{a_1\}$
- $g_1 \bullet = \{a_2, g_2\}$
- $g_1^{in} = \{(a_1, g_2)\}$
- $g_1^{out} = \{(g_1, a_2), (g_1, g_2)\}$

Further, we consider the activities a_1 and a_3. There is a path that connects these two nodes $a_1 \rightsquigarrow a_3$, since there is a sequence of nodes $\langle a_1, g_1, g_2, a_3 \rangle$ and a set of flow relations $(a_1, g_1), (g_1, g_2), (g_2, a_3)$.

Moreover, we define several subsets, which are of particular interest for characterizing process models and specifying the techniques of this thesis [237].

Definition 2.7. (**Subsets**). For a process model P = (A, E, G, F, R, P, L, ρ, π, λ, γ, τ) the following subsets are defined for events and labeled model elements:

- $E_{start} = \{e \in E \mid \bullet e = \emptyset\}$, being the set of *start events*.
  $E_{int} = \{e \in E \mid \bullet e \neq \emptyset \wedge e \bullet \neq \emptyset\}$, being the set of *intermediate events*.
  $E_{end} = \{e \in E \mid e \bullet = \emptyset\}$, being the set of *end events*
- $S = \{g \in G \mid \gamma(g) = and_S \vee \gamma(g) = xor_S^D \vee \gamma(g) = xor_S^E \vee \gamma(g) = or_S\}$, being the set of *split gateways*.
- $J = \{g \in G \mid \gamma(g) = and_J \vee \gamma(g) = xor_J^D \vee \gamma(g) = xor_J^E \vee \gamma(g) = or_J\}$, being the set of *join gateways*.

– $A^{\mathrm{P}}_{\lambda} = \{a \in A \mid a \in dom(\lambda)\},$ being the set of *labeled activities*.
$E^{\mathrm{P}}_{\lambda} = \{e \in E \mid e \in dom(\lambda)\},$ being the set of *labeled events*.
$G^{\mathrm{P}}_{\lambda} = \{g \in G \mid g \in dom(\lambda)\},$ being the set of *labeled gateways*.
$F^{\mathrm{P}}_{\lambda} = \{f \in F \mid f \in dom(\lambda)\},$ being the set of *labeled sequences flows*.
$R^{\mathrm{P}}_{\lambda} = \{r \in R \mid r \in dom(\lambda)\},$ being the set of *labeled resources*.
$P^{\mathrm{P}}_{\lambda} = \{p \in P \mid p \in dom(\lambda)\},$ being the set of *labeled pools*.

We consider again the example process models from Figure 2.5. Definition 2.7 reveals the following subsets for the job application process:

- $E_{start} = \{e_1\}$
- $E_{int} = \emptyset$
- $E_{end} = \{e_2\}$
- $S = \{g_1\}$
- $J = \{g_2\}$
- $A^{\mathrm{P}}_{\lambda} = \{a_1, a_2, a_3, a_4, a_5\}$
- $E^{\mathrm{P}}_{\lambda} = \{e_1, e_2\}$
- $G^{\mathrm{P}}_{\lambda} = \{g_1\}$
- $F^{\mathrm{P}}_{\lambda} = \{(g_1, g_2), (g_1, a_2)\}$
- $R^{\mathrm{P}}_{\lambda} = P^{\mathrm{P}}_{\lambda} = \emptyset$

Finally, we also introduce the notion of a process model repository as being the set of several process models [431, 350]. Along with that, we define useful subsets for a process model repository $\mathcal{P}$.

Definition 2.8. (**Process Model Repository**). Let $\mathrm{P}_i = (A_i, E_i, G_i, F_i, R_i, P_i, L_i, \rho_i, \pi_i, \lambda_i, \gamma_i, \tau_i)$ (i = 1, $\cdots$, n) be a number of n different process models. A process model repository $\mathcal{P}$ is defined as being the set of all n process models:

$$\mathcal{P} = \bigcup_{i=1,\ldots,n} \mathrm{P}_i$$

Moreover, let $A^{\mathrm{P}_i}_{\lambda}$ be the set of all labeled activities, $E^{\mathrm{P}_i}_{\lambda}$ the set of all labeled events, $G^{\mathrm{P}_i}_{\lambda}$ the set of all labeled gateways of a given process model P_i of a repository $\mathcal{P}$. For a process model repository, we define the following subsets:

- $A^{\mathcal{P}}_{\lambda} = \bigcup_{i=1,\ldots,n} A^{\mathrm{P}_i}_{\lambda}$ the set of all labeled activities of the repository.
- $E^{\mathcal{P}}_{\lambda} = \bigcup_{i=1,\ldots,n} E^{\mathrm{P}_i}_{\lambda}$ the set of all labeled events of the repository.
- $G^{\mathcal{P}}_{\lambda} = \bigcup_{i=1,\ldots,n} G^{\mathrm{P}_i}_{\lambda}$ the set of all labeled gateways of the repository.

A process model has to fulfill a set of specific requirements in order to be syntactically correct. Accordingly, the following definition summarizes the rules for syntactic correct process models [311, 277, 238].

Definition 2.9. (**Syntactical Correctness of Process Models**). A process model P = (A, E, G, F, R, P, L, ρ, π, λ, γ, τ) is called *syntactically correct*, if it fulfills the following requirements:

1. A process model contains at least one activity: $|A| \geq 1$.
2. Each node, that is not a start or end event, is on a path from a start to an end event: $\forall n \in (N \setminus (E_{start} \cup E_{end})) : \exists e_{start} \in E_{start}, e_{end} \in E_{end}$, such that $e_{start} \rightsquigarrow n \rightsquigarrow e_{end}$.
3. The latter rule implies that the sets of start and end events are never empty: $E_{start} \neq \emptyset \wedge E_{end} \neq \emptyset$.
4. Start events and attached intermediate events have no incoming and exactly one outgoing flow: $\forall e \in E_{start} \cup dom(\tau) : |\bullet e| = 0 \wedge |e\bullet| = 1$.
5. Non-attached intermediate events have exactly one incoming and one outgoing flow: $\forall e_{int} \in (E_{int} \cup dom(\tau)) : |\bullet e_{int}| = 1 \wedge |e_{int}\bullet| = 1$.
6. End events have exactly one incoming and no outgoing flow: $\forall e_{end} \in E_{end} : |\bullet e_{end}| = 1 \wedge |e_{end}\bullet| = 0$.
7. Activities have exactly one incoming and one outgoing flow: $\forall a \in A : |\bullet a| = 1 \wedge |a\bullet| = 1$.
8. Split gateways have exactly one incoming and at least one outgoing flow: $\forall g \in S : (|\bullet g| = 1 \wedge |g\bullet| > 1)$.
9. Join gateways have at least one incoming and exactly one outgoing flow: $\forall g \in J : (|\bullet g| > 1 \wedge |g\bullet| = 1)$.
10. The process model may not contain unlabeled activities: $|A_\lambda| = |A|$.

In the following, we discuss the syntactical correctness of the example process model from Figure 2.5. Accordingly, the process is checked against each of the requirements from definition 2.9. The job application process is correct with regard to the model syntax as it fulfills the following requirements:

1. The job application process has five activities in total: $|A| = 5$.
2. All nodes of the process model are on a path from the start event to the end event, because there is no isolated node in the process model. Moreover, the defined flow relations always link the start event to a specific node $n \in N$ and also link n with the end event.
3. The process has at least one start event and at least one end event: $|E_{start}| = 1 \wedge |E_{end}| = 1$.
4. The start event of the job application process has no incoming flow ($\bullet e_1| = 0$) and exactly one outgoing flow ($|e_1\bullet| = 1$). Further, it does not specify any intermediate events such that the requirements on the number of incoming and outgoing flows does not apply here.
5. The job application does not specify any non-attached intermediate events such that the requirements on the number of incoming and outgoing flows for non-attached intermediate events does not apply here.

6. The end event of the process has exactly one incoming flow ($|\bullet e_2| = 1$) and no outgoing flow ($|e_2\bullet| = 1$).
7. Each activity of the job application process has exactly one incoming and one outgoing flow. The process meets the condition $\forall a \in A : |\bullet a| = 1 \wedge |a\bullet| = 1$.
8. The process model specifies exactly one split gateway $g_1 \in G$. This gateway has one incoming flow ($—\bullet g_1| = 1$) and two outgoing control flows ($|g_1\bullet| = 2$) which fulfill the respective requirement.
9. The process model specifies exactly one join gateway $g_2 \in G$. This gateway has two incoming flows ($—\bullet g_2| = 2$) and one outgoing control flow ($|g_2\bullet| = 1$) which fulfill this requirement.
10. All activities of the process model are labeled: $|A| = |A_\lambda| = 5$

The previous definitions have provided a baseline to assess the correctness from a syntactical perspective. However, it has to be noted that these criteria for syntactical correctness only apply to the modeling constructs and do not take the quality of the text labels into account. The only requirement associated with labeling refers to the labeling of activities. Theoretically, this requirement would allow to label each activity with an abstract letter that does not convey any particular meaning. Apparently, such process models have serious issues with regard to the clarity principle and are not of much use when analyzing them. Thus, deficient models need to be reworked in a subsequent step which restores the quality of process models and textual labels alike. In the next section, we will therefore investigate particular analysis techniques to improve process model quality.

2.4 Summary

This chapter has provided a general overview to business process management. The chapter explains the main concepts and highlights the importance of business process models within the life cycle of BPM. A deeper investigation into the nature of process models has revealed the subjective nature of process models and motivates the need for modeling principles to facilitate inter-subjectivity and comparability of related process models. Since a considerable number of process models does not comply with these principles, it is necessary to rework them in order to remove clarity and consistency issues. These issues, however, may affect different aspects of the process model. We have argued that these aspects either relate to the formal content of the modeling language or the textual content of the natural language. We have further identified that both modeling and natural language require syntax and semantics in order to describe process models. For a modeling language, the syntax defines how modeling symbols are combined in a meaningful way while the semantics provide an interpretation of these symbols. For a natural language, the syntax describes grammatical rules on how to combine words to sentences and the semantics defines the meaning of words and sentences. Based on these building blocks, we have provided a formal conceptualization, the basic characteristics, as well as a formal definition of syntactical correctness of process models. If we apply the correctness definition on an example process model, we have observed observe that the correctness criteria mainly focus on the modeling language aspect, but neglect model element labels. In order to assess the quality of these element labels, we require a further insights into different refactoring and analysis techniques.

3

Refactoring of Business Process Models

This chapter is concerned with the refactoring of process models that do not comply with the syntactical and semantic rules of the modeling or the natural language. To this end, Section 3.1 presents the current state of process modeling in practice . Afterwards, Section 3.2 discusses correction and refactoring approaches of process models which have been developed by prior research. Section 3.3 will then summarize the main results of the literature review and identify open research gaps. Finally, Section 3.4 gives a conclusion of the main insights of the chapter.

3.1 Process Model Refactoring

This section is dedicated to process model refactoring, which aims for improving the quality of process models with regard to understandability and maintainability. Thus, we begin with a motivation of refactoring by discussing empirical insights on the quality of process models in Section 3.1.1. Afterwards, Section 3.1.2 develops a framework which structures prior research on process model refactoring along with several categories.

3.1.1 Empirical Insights on Process Model Correctness

Several researchers have investigated correctness aspects of process models, its determining factors and consequences thereof. In general, we can distinguish between empirical studies that provide insights on the usage of modeling language and on specific inconsistencies affecting the process model quality. In the first group, Recker [357] investigated the actual BPMN usage by conducting a world-wide survey with 590 BPMN users. Among others, his findings provide insights into the problems and desires practitioners have with respect to the notation. For example, practitioners wish for more possibilities to model the organization in terms of roles and business areas, while decreasing the number of different event types at the same time [357], which would significantly increase the ease of use and the correctness of process models [356].

In the second group of studies, Mendling et al. [279, 284, 286] extensively analyze the connection between formal errors and a set of metrics that capture various structural and behavioral aspects of a process model. Their findings demonstrate that errors do not occur by chance and that certain characteristics like structuredness are desirable to avoid errors in process models. Moreover, it is desirable to restrict the number of model elements and arcs between them to decrease the error probability of the process model and to support the model comprehension task. Leopold et al. [235] enrich these insights by identifying inconsistencies rooted in the structure, layout, and labeling of process models from the perspective of practitioners. The authors report that particularly representational choices for splits and joins, the correct use of message flow, the proper decomposition of models, and the consistent labeling appear to be connected with quality issues and give five specific recommendations how these issues can be avoided in the future and how the comprehension of process models can be increased. Specifically, adjusting the model granularity to user preferences [211] and using an imperative style of labeling [285] are further examples.

Despite these rich insights into model correctness and comprehension, the knowledge has not been transferred to industry yet. In the beginning, studies report error rates between 10% to 20% caused by the erroneous combination of elements in process models [286, 275]. Similarly, Weber et al. [431] identify a rate of unsound models that ranges from 3.3% up to 37.5%. Most recently, Leopold et al. [235] investigate 585 process models from six companies for quality issues. The authors revealed that, although the models comply with syntactical modeling and layout rules, there are inconsistencies, such as inconsistency among process model hierarchies (80% of models affected), process model size (around 48% of models affected) as well as the labeling (40% to 47%), that affect the correctness of process models. Moreover, these numbers are reported in situations in which organizations tend to model all business process [369] and to create several hundreds or thousands of separate process models in a process model repository [370]. Since process model repository technology is increasingly emerging [115, 114, 113, 225], there is a general need to maintain process model quality.

For this purpose, verification and validation techniques are two complementary steps [174, p. 452]. *Verification* addresses the general properties of a process model and ensures that the formal representation of the process model is correct. *Validation* is concerned with the consistency of the model and the universe of discourse and ensures that the model resembles the object it is supposed to. While verification typically can be supported by formal analysis techniques, validation requires the consultation and the discussion with involved business experts. Thus, it is nearly impossible to perform validation steps in a fully automatic way [242].

Another possibility to maintain process model quality is given by process model refactoring. Process model refactoring stems from software engineering [313] and refers to the process of changing a software system in such a way that it does not alter the external behavior of the code, yet improves its internal structure [131]. Analogously, process model refactoring improves the internal quality of a model such that it becomes easier to read and maintain without affecting the semantics or behavior of the respective process model.

Research has proposed a considerable number of approaches to assess the quality of process models. For these reasons, the next section elaborates a structure to categorize such approaches.

3.1.2 Refactoring Categories

We use the two essential building blocks as elaborated in the framework by Leopold [237, p. 12] in order to derive different structure-given categories of analysis approaches. The framework distinguished between the modeling language and the natural language which are combined for the purpose of creating process models (cf. Figure 2.4). The *formal content* of a process model relates to the modeling language, which provides symbols and rules to combine these symbols. Typically, the formal content describes the structure and the execution behavior of the underlying business process. The *textual content* refers to the natural language part of process models and enriches the formal description of the process model with additional semantics of the business domain. Hence, we use the formal content and the textual content as the main categories to classify the existing approaches. Additionally, we further elaborate suitable sub categories for each of the main categories.

With regard to the *formal content* of a process model, we consider the framework of Karagianis and Kühn [189] as well as from Leopold [237, p. 12]. All of these authors argue that the formal content of a process model can be discussed in terms of syntax and semantics. The *syntax* defines a set of constructs and rules on how to combine these constructs. *Semantics* describes the meaning of these constructs. Similarly, Lindland et al. [255] as well as Krogstie et al. [219, 220] also recognize syntax and semantics to contribute to the overall quality of a process model. In addition to that, the authors propose *pragmatics* to be part of the formal content which refers to the purposefulness and comprehensibility of a process model. Therefore, we do not only consider syntax and semantics but equally include pragmatics to be a sub category to structure the model analysis techniques.

With regard to the *textual content* of a process model, we use again the framework of Leopold [237, p. 12] to develop suitable sub categories. The author discusses the textual content in terms of syntax and semantics. The *syntax* inquires the formal and structural relations between words, while *semantics* investigate the meaning of words and the relationship that arise from the various meanings of words. In addition to that, we can extend the discussion of the textual content with the studies of pragmatics. Ferdinand de Saussure, the

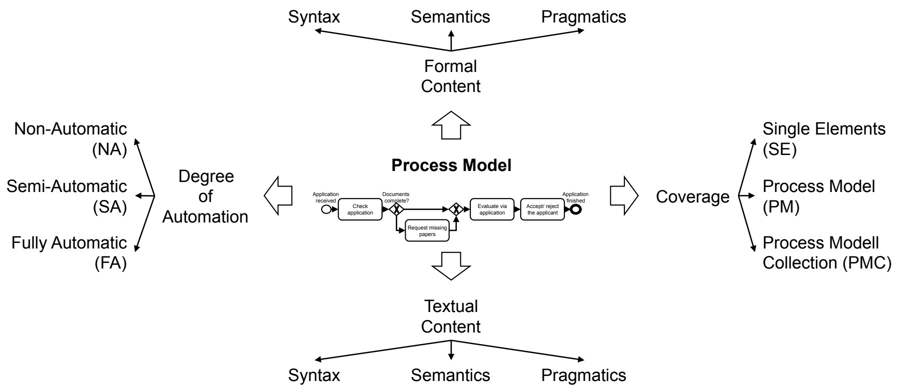


Fig. 3.1: Categorization Framework for Process Model Analysis Techniques

founder of our modern understanding of linguistics, views natural language as a system of signs and relations between these signs [381, p. 16] and proposes to further investigate language from the pragmatic perspective. *Pragmatics* is concerned with the relation of words to interpreters [296, pp. 6-7] and how they understand these words in a given context or discourse [43, p. 152]. Thus, we can also include the dimensions syntax, semantics, and pragmatics into the categorization structure.

In addition to the distinction of the formal content and the textual content, we further enrich the framework with a consideration of coverage and the degree of automation. The *coverage* (Cov) is motivated by the work of Mendling et al. [278], who identify several challenges with regard to semantic process modeling. The authors define the coverage as being the extent of process model elements that are covered by analysis approaches. Accordingly, we differentiate between approaches that focus on single model elements (SE), on the entire process model (PM), or on a repository of several process models (PMC). The extension of the categories with regard to the *degree of automation* (DoA) is particularly interesting from the perspective of a model repository [370]. Since the assessment of such a large repository can be hardly done in a manual way [35], we are interested to what extent the user is involved when the approaches are applied. Thus, we distinguish between non-automatic approaches (NA), semi-automatic approaches (SA) and fully automatic approaches (FA). *Non-automatic techniques* require the user to perform an extensive amount of manual work to apply the proposed approach. *Semi-automatic techniques* perform a considerable amount of work in an automatic way and involve the user for important decisions. *Fully automatic techniques* autonomously perform all necessary steps and present the final result to the user according to given input parameters. Typically, these approaches do not require any further user assistance.

Figure 3.1 gives an overview of the framework and its criteria. This framework is used to categorize existing work and to structure the following discussion of relevant approaches and techniques of process model refactoring.

3.2 Classification of Refactoring Approaches

In the subsequent sections, we discuss available approaches according to the categorization structure. Section 3.2.1 presents relevant approaches that are concerned with the refactoring of the flaws of the formal content of process models. Section 3.2.2 discusses those approaches that improve the textual content of process models.

3.2.1 Refactoring of the Formal Content

Table 3.1 provides an overview of analysis and refactoring approaches that address the *syntax* of the formal content of a process model. In this way, we roughly distinguish between approaches that either provide fundamental formalization of syntactical properties or technical capabilities to detect and refactor violations of these properties. Note that some of the discussed approaches are not refactoring techniques in the aforementioned sense. Nevertheless, they are considered in the discussion because they may trigger refactoring initiatives to improve the quality of the process model. Regarding the *formalizations of syntactical properties*, van der Aalst [1] introduces the notion of a workflow net which imposes specific conditions on Petri net-based process models to be sound. It requires that a process model has exactly one initial node and one final node. Moreover, it demands that each node in a Petri Net should be on a directed path from initial node to a final node. Following this idea, Puhlmann and Weske [347] as well as Weske [437, pp. 270-271] introduce structural soundness to business process models in any modeling language. Nüttgens and Rump [311], Mendling et al. [277], and Laue and Mendling [230] provide a set of criteria for syntactically correct EPCs. The introduced criteria have been adapted by Definition 2.9. Moreover, Mendling and Nüttgens [280] introduce the concept of implicit element and arc types, which fosters the automatic evaluation of syntactical correctness and connector validity by looking at the implicit arc type group of the ancestor and descendant arcs.

Regarding the *detection of errors*, Mendling and Nüttgens [281] provide an implementation of the given EPC syntax definition in XML and use Schematron to automatically verify the syntactical correctness of EPCs. Puhlmann and Weske [347] also provide technical means to verify structural soundness of process models. Sadiq and Orlowska [377, 378] propose a syntactical verification technique that relies on reduction. The technique is available via the modeling and verification tool FlowMake. Gruhn and Laue [149, 150] use logical programming with Prolog to verify structural properties of process models. The approach identifies those models that violate the given properties

Table 3.1: Overview of Syntactical Refactoring Techniques for the Formal Content of Process Models

Author	Approach	Cov.	DoA
Formalizing Syntactical Properties			
van der Aalst [1]	Concept of Workflow Nets	PM	NA
Puhlmann and Weske [347], Weske [437, pp. 270-271]	Definition of Structural Soundness	PM	NA
Nüttgens and Rump [311], Mendling et al. [277], Laue and Mendling [230]	Definition of Syntactically Correct EPCs	PM	NA
Mendling and Nüttgens [280]	Definition of Implicit Arcs for Syntactical Correctness of EPCs	PM	NA
Error Detection and Syntax Refactoring			
Mendling and Nüttgens [281]	Syntactical Verification of EPCs with Implicit Arcs	PM	FA
Puhlmann and Weske [347]	Structural Soundness Verification with π-Calculus	PM	FA
Sadiq and Orlowska [377, 378]	Syntactical Verification with Graph Reduction Techniques	PM	FA
Gruhn and Laue [149, 150]	Verification of Structural Properties with Prolog	PM	FA
Gruhn and Laue [152]	Error Detection with Label Analysis	PM	FA
Awad and Puhlmann [20]	Syntactical Error Detection with Queries	PM	FA
Verbeek et al. [423], Wolf [442]	Petri Net Editors with Syntactical Check	PM	FA
Mendling et al. [286, 274]	Metric-based Error Detection and Prediction	PM	FA
La Rosa et al. [223, 226]	Syntax Refactoring Patterns	PM	NA
Gambini et al. [138]	Error Correction with Simulated Annealing	PM	FA

by translating process models and structural properties into Prolog facts. An enhanced version of their approach [152] uses text label analysis of activities and events to detect logical errors in the model structure. For example, an application cannot be rejected and accepted at the same time. Awad and Puhlmann [20] use queries to detect a broad range of common structural error patterns leading to deadlocks. Verbeek et al. [423] develop a workflow verification tool called Woflan. Woflan uses Petri net-based analysis techniques, which ensure, among others, the free-choice or the well-structuredness property of Petri nets. Similarly, the LoLA tool [442] detects syntactical errors in Petri nets and notifies the user about them. Mendling et al. [274, 286] have developed a set of metrics that assess different syntactical aspects of process models, such as density, partitionability, cyclicity or concurrency, which aim for the detection and prediction of formal errors in process models. By employing a

Table 3.2: Overview of Semantic Refactoring Techniques for the Formal Content of Process Models

Author	Approach	Cov.	DoA
Formalizing Semantic Properties			
Murata [299]	Formalization of Liveness and Boundedness for Petri Nets	PM	NA
Desel and Esparza [99]	Introduction and Characterization of Free-Choice Petri Nets	PM	NA
van der Aalst [1]	Definition of Soundness Criterion for Workflow Nets	PM	NA
Martens [267, 269]	Definition of Weak Soundness	PM	NA
Dehnert and Rittgen [94]	Definition of Relaxed Soundness	PM	NA
Mendling [273]	Formalization of EPC soundness	PM	NA
Detection and Refactoring of Semantic Violations			
Kemper and Bause [191]	Verification Algorithm for Liveness and Boundedness of Petri Nets	PM	NA
Verbeek et al. [423], Wolf [442]	Petri Net Editor with Semantic Checks	PM	FA
Fahland et al. [122]	Verifying Soundness with SESE decompositions	PM	FA
Puhlmann and Weske [347] Puhlmann [346]	Soundness Verification with π-Calculus	PM	FA
Martens [268]	Verification of Weak Soundness	PM	FA
Mendling and van der Aalst [276]	Verification of EPC Soundness	PM	FA
Gruhn and Laue [151]	Detection of Control Flow Errors with Prolog	PM	FA
Fahland and van der Aalst [124, 123]	Repairing Process Models according to Observed Behavior	PM	FA
Buijs et al. [63]	Adjusting Models on the Basis of Event Logs	PM	FA

logistic regression, these error metrics then predict the error probability of a business process model without any deeper analysis. La Rosa et al. [223, 226] introduce a set of modification patterns that improve the syntax of a process model in order to increase its understandability. Among others, the authors propose to use block-structuring or duplication measures, which involves a modification of the model syntax and an improvement of the internal structure of the model. The technique of Gambini et al. [138] uses simulated annealing on Petri nets to compute a number of change-minimal corrections that transform a syntactically erroneous net into a correct one.

In Table 3.2, we summarize those approaches that evaluate the *semantics* of the formal content and detect problems that relate to the general behavior and the execution semantics of process models. We also discuss approaches which put a stronger focus on semantic correctness because they may trigger refactoring initiatives to improve the quality of the process model. From a general perspective, there are semantic approaches that formalize semantic properties or provide technical support for the detection and refactoring of semantic violations.

Among the approaches that *formalize semantic properties*, Murata [299] summarizes prior work on semantic properties of Petri nets. Among others, he discusses the notion of liveness and boundedness and their relevance for Petri Nets. While liveness ensures that Petri Net transitions may fire in every reachable marking, boundedness ensures that a Petri net does not contain more than a specific number of tokens in all reachable markings. Desel and Esparza [99] introduce free-choice Petri nets as a specific class of Petri nets and further conceptualize desirable properties of these nets, such as the liveness and boundedness property. Van der Aalst [1] builds upon these works and introduces the criterion of soundness to Petri nets used to model workflow systems. The soundness criterion demands that for any case, a process instance will terminate eventually and that at the moment the instance terminates there is a token in the final node and all the other nodes are empty. Besides the classical notion of soundness, several authors introduce less strict definitions of soundness, such as weak soundness [267, 269], relaxed soundness [94], or lazy soundness [346]. There is also the notion of EPC soundness as formalized by Mendling [273].

Among the approaches that *detect and refactor semantic violations*, Kemper and Bause [191] introduce an efficient algorithm to assess the liveness and boundedness property of Petri nets in polynomial time. However, since their approach provides only a theoretical construct to assess these properties, the tools Woflan [423] and LoLA [442] offer technical means for semantic checks. Woflan analyzes process models in Petri net notation with regard to several desirable properties, such as boundedness, liveness, and soundness. Similarly, LoLA also analyzes Petri nets with regard to these properties by creating and exploring a reduced state space. With regard to the soundness property, Fahland et al. [122] use single-entry-single-exit (SESE) decompositions. In contrast to the aforementioned approaches, the authors' approach scales well with the model size and is capable to determine the soundness property within milliseconds, even for large process models from industry. Martens [268] also proposes a technical implementation to check the weak soundness criterion. Puhlmann and Weske [347] as well as Puhlmann [346] use the π-calculus to formalize the aforementioned soundness properties and describe technical means for their automatic assessment. With regard to EPC soundness, Mendling and van der Aalst [276] employ a set of reduction rules for the automatic verification of the EPC-specific soundness property. An approach based on logical programming is proposed by Gruhn and Laue [151]. The

Table 3.3: Overview of Pragmatic Refactoring Techniques for the Formal Content of Process Models

Author	**Approach**	**Cov**	**DoA**
Rework of Model Layout			
Effinger et al. [111]	Analysis of Layout Preferences in Modeling Tools	PM	NA
Effinger and Siebenhaller [112]	Auto-Layout Algorithm for Process Models	PM	FA
Gschwind et al. [153]	Auto-Layout Algorithm embedded in Modeling Editor	PM	FA
Malesevic et al. [259]	Auto-Layout Algorithm for Process Models in UML activity diagram notation	PM	FA
Visualization			
Mendling and Recker [282]	Enriching Activities with Graphical Icons	SE	NA
La Rosa et al. [223, 226]	Modification of Process Model Layout	PM	NA
Reijers et al. [358]	Symbol Highlighting	SE	NA
Detection and Refactoring of Comprehensibility Issues			
Gruhn and Laue [151]	Detection of Bad Modeling Style in Process Models with Prolog	PM	FA
Polyvyanyy et al. [338, 337]	Block Structuring of Unstructured Process Models	PM	FA
Reijers et al. [359]	Identifying Suitable Fragments for Building Modular Sub Processes	SE	FA

authors develop a heuristic approach that identifies several behavioral violations in process models by translating EPC process models and desirable properties into Prolog facts. The Prolog inferences are then capable to detect behavioral errors, such as deadlocks and livelocks. The approaches of Fahland and van der Aalst [124, 123] and Buijs et al. [63] go one step further and provide technical means to automatically correct semantic inconsistencies. While the approach of Fahland and van der Aalst [124, 123] investigates the problem of adjusting and aligning a process model with regard to the process that is taking place in reality, Buijs et al. [63] present a technique that automatically improves the process model on the basis of the observed behavior as recorded in the event logs.

Table 3.3 gives an overview of recent approaches that address the pragmatic aspect in a process modeling language. As mentioned, pragmatics is concerned with the comprehensibility of a process model in a given context. Thus, most cases of pragmatic issues are related to the layout of process models which complicates its comprehensibility. Accordingly, we distinguish approaches that rework the layout of a process model, improve the visualization of specific aspects, and detect or refactor comprehensibility issues.

In order to *rework the layout of a process model*, several researchers have investigated automatic layout capabilities for process models. Effinger et al. [111] analyze layout preferences of user groups when modeling with BPMN. The authors present a set of layout criteria that are formalized and then confirmed by a user study. The study reveals preferences of single user groups with respect to secondary notation and layout aesthetics. From their results, proposals for adaptions of software tools towards different BPMN users can be derived. Moreover, Effinger and Siebenhaller [112] propose an approach to increase the readability of process visualizations. The approach is based on a graph-geometric algorithm that performs constrained cuts on given model visualizations. Malesevic et al. [259] develop a software tool for the automatic visualization of process model represented by UML activity diagrams. The implemented tool takes the XMI-based representation of the activity diagram as input and automatically generates its layout in accordance with the UMLDI specification. Gschwind et al. [153] propose a modeling editor that composes process models with larger process fragments that foster the creation of business process models of higher quality. Their approach also offers auto-layout capabilities that organize the process model in a structured way.

Approaches that improve the *visualization* of process models provide empirical evidence on the effects of highlighting relevant elements in process models such that the sense-making of process models is supported. For that purpose, Mendling et al. [282] develop a systematic approach to graphically represent verb classes through the use of graphical icons such that the resulting process models are easier and more readily understandable by end users. As a result, the authors enrich the activities that convey a particular verb class, e.g. to process or to modify, with icons that graphically represent the action that has to be carried out. La Rosa et al. [223, 226] propose and introduce a set of layout patterns to handle the complexity of process models and to improve model comprehension. They also provide empirical evidence how the introduced measures positively impact model comprehension. Reijers et al. [358] adapt the concept of syntax highlighting to process models in order to support human sense making of these models. In particular, the approach employs coloring of matching gateways to emphasize the beginning and the end of a gateway block. The experimental evaluation shows that users with colored models performed significantly better than those with regular models and were able to draw correct conclusions from the model.

Regarding the *detection and refactoring of comprehensibility issues*, Gruhn and Laue [151] make use of logical programming to detect bad modeling practice. Here, the authors formalize patterns that hinder the understandability of process models. For example, these rules point to instances where process models excessively use OR gateways that could be replaced by a simple XOR gateway. Moreover, they identify cases when an XOR gateway splits the control flow and results in two or more alternative events which do not affect the future execution of the process. Polyvyanyy et al. [338, 337] propose techniques that transform unstructured graph-based process models into block-structured ones. The advantage of such block-structured models is that every split gateway has a corresponding join gateway resulting in a self-contained block. This block is then easier to comprehend and less error-prone [229, 231]. Reijers et al. [359] explore different criteria to automatically derive process fragments that are worth capturing as sub processes. These criteria implement the concepts of block-structuredness, connectedness, and label similarity to identify sub process fragments.

3.2.2 Refactoring of the Textual Content

Similarly to the previous section, we will discuss existing approaches to assess the correctness of the textual content of process model, i.e. the correctness and the consistency of the process model elements labels.

Table 3.4 provides an overview of the relevant approaches that address the quality and correctness of text labels in process models. *Syntactical approaches* investigate the grammatical structure of text labels and provide means to improve the labels with regard to understandability. First of all, there exist several guidelines that particularly address the labeling of process model elements [283, 395, 261]. For example, these guidelines suggest that activities in process models should begin with an action verb which is followed by an object. Leopold et al. [248, 249] propose a refactoring technique that reworks the syntactical structure of labels in order to correct text labels that do not comply with these guidelines. The authors employ different layers of label context for an accurate classification of the labeling style. After the label style has been recognized, the label is parsed and transformed according to the aforementioned naming guidelines. This approach has also been extended to the languages German and Portuguese [238]. Becker et al. [31, 32] develop a tool that provides modeling support for the naming of process model elements. The tool already enforces specific naming conventions during the process of modeling and ensures that they are automatically fulfilled. Thus, it is not necessary to correct the labels afterwards. Similarly, Delfmann et al. [96] as well as Havel et al. [157] introduce a prototype which guides users through the process of resolving naming violations by providing an automatic list of correct phrases.

Semantic approaches investigate the meaning of labels and the relationship of the concepts they refer to. Van der Vos [425] uses a semantically based lexicon to check the quality of text elements. It ensures that words and phrases of model element labels are used in a linguistically meaningful and sense-making way. For that purpose, the technique checks if the label correctly refers to a concept or an object of the real world and if the relation between several objects makes sense in the context of a model. Weber et al. [433, 434] propose a semantic annotation approach that enriches process model activities with preconditions and effects and propagate those for semantic consistency

Table 3.4: Overview of Refactoring Techniques for the Textual Content of Process Models

Author	Approach	Cov.	DoA
Syntax			
Sharp and McDermott [395], Mendling et al. [283], Malone et al. [261]	Naming Conventions for Process Model Elements	SE	NA
Leopold et al. [248, 249]	Parsing and Refactoring of Activity Labels	SE	FA
Leopold et al. [238]	Detecting and Correcting Naming Violations	SE	FA
Becker et al. [31, 32]	Enforcing of Naming Conventions during Modeling	SE	SA
Delfmann et al. [96], Havel et al. [157]	Prototype for Naming Convention Enforcement	E	SA
Semantics			
Van der Vos [425]	Verifying Concept Relationships with a Semantic Lexicon	SE	NA
Weber et al. [433, 434]	Semantic Annotation and Verification with Preconditions and Effects	PM	SA
Friedrich [135]	Measuring Ambiguity of Process Model Labels	SE	FA
Becker et al. [31, 32]	Preventing Ambiguity with a Domain Thesaurus	SE	SA
Havel et al. [157]	Resolving Synonyms with User-defined Dominant Synonyms	SE	SA
Pragmatics			
Friedrich [135]	Metrics for the Specificity of Labels	SE	FA
Leopold et al. [246]	Label Granularity Measurement with Language Analysis	PM	FA
Koschmider and Blanchard [208]	Identification of Non-Uniformly Specified Elements	SE	SA
Leopold et al. [243, 244]	Generating Abstract Model Element Names	PM	FA
Smirnov et al. [403, 404, 405] Richetti et al. [363]	Activity Aggregation based on Meronymy Relations	SE	FA

verification. For example, it is only possible to send a cancellation of a purchase order if it has not been conformed yet. Friedrich [135] proposes semantic metrics that quantify the understandability and integratability of text labels in process models. In particular, the author develops a consistency metric that measures the degree of ambiguity of a particular word. The metric considers the number of occurrences of the given word divided by the total number of occurrences of its synonyms in a model repository. This metric thus points to labels with a high chance of ambiguity. The approach of Becker et al. [31, 32] was already introduced and enforces specific naming conventions in process models. Due to the linkage with a domain thesaurus, the tool is also capable of proposing alternative terms and preventing naming conflicts and ambiguity in advance. The research prototype of Havel et al. [157] is also corrects synonym terminology by selecting the dominant synonym among a set of ex-ante defined synonyms.

Pragmatic approaches are concerned with the interpretation and understandability of process models in a given context. Another metric of Friedrich [135] quantifies the specificity of a text label in terms of depth within the WordNet hyponym tree. Apparently, the deeper the word is located in the hyponym tree, the higher is its specificity. The metric serves as an indicator to maintain a consistent level of detail in process models. Similarly, Leopold et al. [246] propose and evaluate a set of syntactic and semantic metrics that measures the granularity of a single process model. Their evaluation results suggest that these metrics are suitable to quantify the granularity of a process model within a process hierarchy or process architecture. Koschmider and Blanchard [208] propose a semi-automatic approach to detect non-uniformly specified process elements. The approach analyzes text labels with regard to their level of detail and highlights process model elements that strongly deviate from that level. Finally, it obliges the user to align the deficient elements and improve process model consistency. Leopold et al. [243, 244] propose a technique to infer suitable names for process models to facilitate simplification and abstraction of process models. The technique helps in understanding process models as it abstracts from details and reduces the complexity of the process model repository. Similarly, Smirnov et al. [403, 404, 405] have proposed techniques on process model abstraction. The idea is to aggregate fine-grained activities or model fragments into a coarse-grained one. This is achieved by employing the meronymy-relation, i.e., part-of relation, defined between activities. Finally, Richetti et al. [363] introduce a technique that reduces complexity of declarative process models by aggregating activities according to inclusion and hierarchy semantic relations.

3.3 Discussion and Problem Statement

The review of prior research illustrates that errors that relate to the textual dimension of process models have only been addressed to a limited extend. Prior research approaches focus on the modeling language and offer help to correct errors with regard to the syntactic, semantic and pragmatic dimension of the modeling language.

Nevertheless, it has also been shown that a large share of the overall semantics of a process model is conveyed by natural language. As illustrated in Leopold [237, pp. 9–10], it is not possible to infer any conclusions from

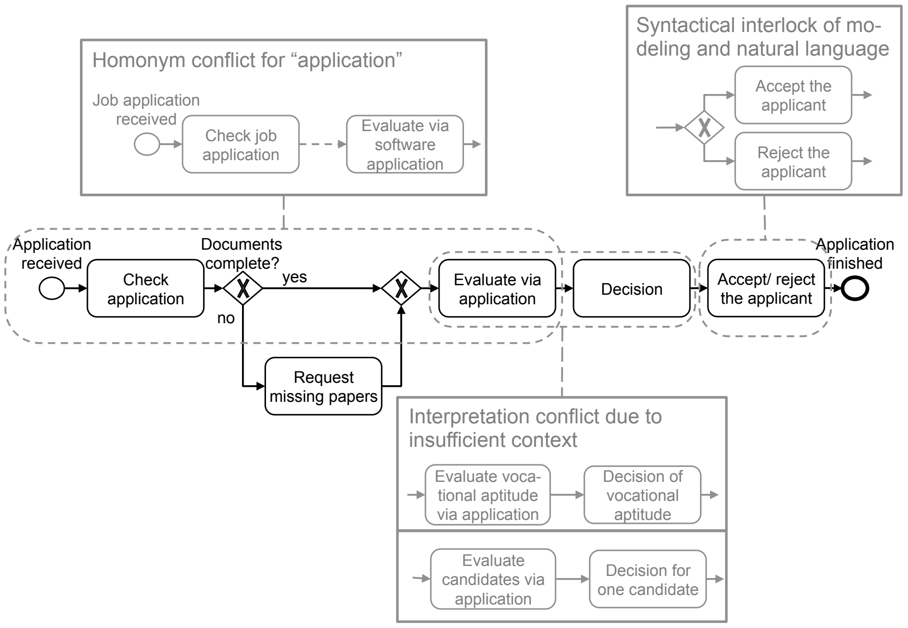

Homonym conflict for "application"
Job application received
Check job application
Evaluate via software application
Syntactical interlock of modeling and natural language
Accept the applicant
Reject the applicant
Application received
Check application
Documents complete?
yes
no
Evaluate via application
Decision
Accept/ reject the applicant
Application finished
Request missing papers
Interpretation conflict due to insufficient context
Evaluate vocational aptitude via application
Decision of vocational aptitude
Evaluate candidates via application
Decision for one candidate

Fig. 3.2: Job Application Process with Linguistic Inconsistencies

a process model containing no labels or only abstract element labels. Consequently, it is hardly possible to use such process models for meaningful applications such as model matching [436, 245, 202], model analysis [98], or legal compliance checking [54, 140, 86]. In general, it can be stated that natural language substantially affects the quality of a process model. If a model contains ambiguous or non-intuitive terms, the model reader might not be able to properly infer its full semantics [339]. It is therefore important to assure the consistency of text labels and to elaborate techniques that improve these labels. We use the example process model as depicted in Figure 3.2 to illustrate the necessity for such techniques. In contrast to the previous section, the application process highlights particular linguistic issues that cannot be addressed by prior research.

Concerning the importance of labeling quality, prior research has also proposed various label analysis techniques. As discussed before, these techniques ensure that the element labels comply with naming guidelines and rework them if these guidelines are violated. Additionally, prior approaches provide possibilities to measure the degree of ambiguity and specificity of text labels. Despite these efforts, there are still notable gaps that have not been addressed so far. With regard to the syntax, the available techniques are capable of reworking the syntactical structure labels if they describe only one single stream of action (see e.g. activity *Check application*). However, they only provide parts of a solution if labels are built with complex phrases, such as conjunctions, conditions, and hidden decisions. These phrases then contain information about the order of performing particular tasks and influence the control flow of the entire process. As an example, consider the activity *Accept/reject the applicant* which describes two different activities, i.e. the rejection and the acceptance of an applicant. We also observe that this activity implies a decision, since it is absurd to reject and accept the same applicant at the same time. This decision, however, is not expressed in the process model and completely hidden to available model checking techniques. With regard to the semantics, we discussed several research approaches that use semantic metrics and knowledge sources to handle ambiguous terminology. However, they are of limited use to reliably detect an ambiguous term and to provide support for its resolution. For example, the ambiguity metrics might correctly consider the word *application* as an ambiguous one, but do not provide sufficient support to correct this case, e.g. replacing it with *job application* in the first place and with *software application* in the second place. Moreover, it has to be considered that not every word that has the potential to be ambiguous is used in an ambiguous way. Thus, we require additional means to distinguish between true and false ambiguities. With regard to pragmatics, prior research focuses on term specificity and the identification of cases in which the specificity of a label is violated, as in the activities *Decision*. However, these approaches are of little help to correct such issues. As example, consider the issue of underspecification arising from missing information within a model element, e.g. the labels *Evaluate via application* and *Decision*. In this case, the interpretation of the activities might be affected since we do not know what the evaluation and the decision aim for. Possible interpretations might be that the vocational aptitude of the candidate is evaluated and finally decided or that all application candidates are evaluated together followed by a decision for the best one. Consequently, our understanding of the process is still limited, since this necessary information is not provided in the process model. Therefore, approaches are required that employ concepts and techniques from the field of linguistics to address the aforementioned issues.

The review of related approaches also shows two important shortcomings. First, most of the textual analysis techniques only cover issues on the level of process model elements. Overall, only the pragmatic dimension revealed approaches that address the process model as a single unit. However, these approaches are of limited use if linguistic issues affect the process model repository on a global level. For example, if we consider the activities *Documents complete?* and *Request missing papers* from Figure 3.2. On a local level, these labels might not conflict with each other, since the labels clearly refer to a particular object of interest. On the level of a process model however, we might encounter a semantic inconsistency. Since both terms *documents* and *papers* may refer to the same object, i.e. a piece of writing holding relevant information, it is unclear whether these terms actually refer to the same object or not. This problem is intensified if the number of process models increases.

Table 3.5: Overview of Requirements

Requirement	Description	Relevant Chapters
Requirement 1 (R1)	Identification of Applicable Linguistic Concepts	4
Requirement 2 (R2)	Formalizing Detection Rules for Linguistic Ambiguities	5, 6, 7
Requirement 3 (R3)	Formalizing Refactoring Measures for Detected Linguistic Ambiguities	5, 6, 7
Requirement 4 (R4)	Ensuring a Sufficient Degree of Automation	5, 6, 7

Hence, process models and the model repository should also be covered by such techniques.

Second, the review also reveals an insufficient degree of automation for most of the techniques. While the syntactic reworking can be handled in an automatic way, this is much harder for semantic and pragmatic approaches. In many cases, the user or the model creator need to be involved in order to refactor the model. The reason is that most semantic and pragmatic problems require additional knowledge about the environment and domain in order to remove an ambiguous term or to align the specificity of an activity. This, however, cannot be achieved by current algorithms and is still an unsolved problem in artificial intelligence [300, 49]. Therefore, a fully automatic approach is not feasible at this stage. Nevertheless, as the trend is towards large process model repositories, it is further necessary to provide (semi-)automatic support to assess a large number of process models and to point users to the critical cases. Moreover, the users should be supported as good as possible, e.g. by providing meaningful recommendations for rework, which requires a good performance and a high reliability of the techniques.

In order to address the identified gaps, we require a deeper insight into linguistic issues that are caused by the actual use of natural language in process models. These linguistic issues cannot be resolved with the help of available techniques as they only address parts of the aforementioned inconsistencies. As examples, consider the label refactoring approach of Leopold et al. [238] assuming regular label structures without textually incorporated process logic, or the approach of Havel et al. [157] reworking only user-defined synonyms but no homonyms. Accordingly, we formulate the following requirements (see Table 3.5):

Requirement 1 (R1): Identification of Applicable Linguistic Concepts. Requirement 1 emphasizes the necessity to acquire knowledge from the science of linguistics and to identify relevant concepts, ideas, and technologies. The identified concepts and technologies are then adapted to process models

and provide an understanding why particular text labels are inconsistent. Moreover, they provide means to detect and rework existing inconsistencies.
Requirement 2 (R2): Formalizing Detection Rules for Linguistic Ambiguities. The insights from Requirement 1 are an important asset for Requirement 2. In general, a reliable technical support for addressing the identified problems needs to consider knowledge about the phenomenon. This knowledge facilitates the definition and formalization of reliable detection rules that help users in detecting linguistic issues in process models.
Requirement 3 (R3): Formalizing Refactoring Techniques for Detected Linguistic Ambiguities. It is necessary to conceptualize refactoring techniques that resolve linguistic issues after having detected them. Similar to the previous requirement, the insights from Requirement 1 also foster the definition and formalization of reliable refactoring techniques that support users in refactoring these issues.
Requirement 4 (R4): Ensuring a Sufficient Degree of Automation. Finally, we need to consider the situation in which the proposed detection and refactoring techniques are applied. In particular, Requirement 4 considers process model repositories which hinders the application of non-automatic techniques. Therefore, Requirement 4 demands a sufficient degree of automation as well as a reliable performance of the techniques.

In this doctoral thesis, requirements 2 to 4 should hold for each of the proposed techniques. Therefore, we show for each of the proposed techniques how the requirements are fulfilled and under which assumptions they are valid. Since all these requirements rely on linguistic concepts, the next sections introduce to linguistic studies and identifies relevant concepts and technology from the field of linguistics and natural language processing.

3.4 Summary

This chapter has provided an overview of prior research that is concerned with the refactoring of process models. The overview has been divided into approaches that consider the formal dimension and the textual dimension of process models. Moreover, the discussion has distinguished between the sub-dimensions syntax, semantics, and pragmatics which is appropriate to discuss the quality of conceptual models in general and to investigate natural language. Since this thesis is concerned with a large number of process models maintained in a repository, it is beneficial to further classify these approaches depending on their degree of automation and their coverage. The overview has revealed a notable gap of approaches addressing the use of natural language in process models and the phenomena that arise from it. Therefore, this doctoral thesis aims at closing this gap and providing techniques that support users in maintaining language consistency within process model repositories.

4

Concepts of Linguistics and Natural Language Processing

This chapter introduces the research field of linguistics and natural language processing and discusses the most relevant concepts in order to analyze and improve the labels of process model elements. Section 4.1 starts with a short overview of linguistics and its most important schools of thought and their contribution to the different branches of linguistics. Section 4.2 continues with an elaboration on the branches syntax, semantics, and pragmatics in more detail. Afterwards, Section 4.3 introduces to the field of natural language processing (NLP) and gives a summary of available techniques to perform syntactical, semantic and pragmatic analyses. Afterwards, we investigate how natural language processing leverages language analysis with specific technology. To this end, Section 4.4 discusses technology to analyze the syntactic structure of sentences and further explains the necessary steps of syntactically processing short natural language fragments of process models. Section 4.5 gives an overview of technology that facilitate the semantic analysis of words, before Section 4.6 focuses on techniques for pragmatic analysis. Finally, Section 4.7 summarizes the most important points of this chapter.

4.1 Overview of Linguistics

In general, linguistics is concerned with the scientific study of all phenomena involved with language, including its structure, its use and the implications of these [30, p.11]. In this context, language does not refer to a particular language, such as English or German, but to a system of symbols [366]. Linguistics attempts to study general principles of language organization and language behavior, often by referring to an actual language. This typically involves an investigation of the language mechanisms, its parts, and how all these parts fit together to perform particular functions [414, pp. 12-13]. There are, however, several possibilities and angles to study language-related phenomena.

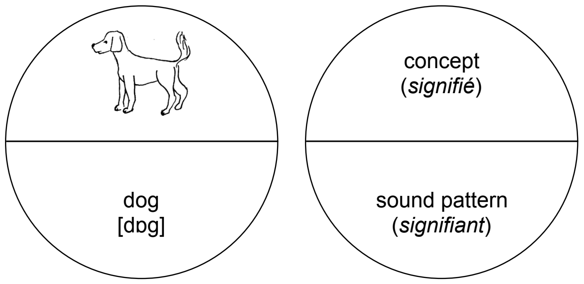


Fig. 4.1: Linguistic Sign Model by Saussure, adapted from [381, pp. 65-67]

On the one hand, linguistic research may be distinguished by its main purpose. We can distinguish between descriptive, comparative, and historical linguistics [414, pp. 16-18]. The main concern of *descriptive linguistics* is to describe a language system, to study its nature, and to establish a theory of it. It seeks to study the components of a language system and to provide explanatory statements on the system's mechanics. *Historical linguistics* studies the change of a language system over a specific period of time and develops theories about how and why it has changed. If the history is also taken into account to study the relatedness and to identify particular language families, we are in the field of *comparative linguistics.* On the other hand, linguistics may also be divided into theoretical and applied linguistics. *Theoretical linguistics*, which is still considered to be the core of linguistic research, focuses on the description of language and the development of models that map the linguistic knowledge [64, pp. 27-28]. *Applied linguistics* is using what theoretical linguistics knows about language, how language is learned and used in order to achieve some purpose or to solve problems of the real world [388, p. 1].

In linguistics, there exist three different schools of thought that shaped the field up until now. The modern understanding of linguistics is significantly based on the ideas of the Swiss linguist Ferdinand de Saussure who was responsible for a dramatic change of direction in the early 20th century. He proposed to focus on the structure of a language system by abstracting from the concrete individual use of language (*parole*) to generic elements of a language system (*langue*). His ideas founded the linguistic school of *structuralism.* Structuralism aims at the description of the elements of the language system as well as their interrelations. The elements and their interrelations are investigated at different structural levels. These levels are reflected by Saussure's model of the *linguistic sign* [381, pp. 65-67] (see Figure 4.1). He conceptualizes the linguistic sign to consist of two inseparable parts, i.e. the concept (*signifié*) and the sound pattern (*signifiant*). The sound pattern determines how the sign is expressed in a language, while the concept determines the meaning of the sign. He further emphasizes that there is no natural connection between the sound and the concept and that this link has been created arbitrarily by convention. Considering the animal that is expressed by the word *dog* as an example, we refer to the same animal as *Hund* in German or *chien* in French.

Another important school of linguistics is commonly referred to as *functionalism* or the *Prague School of functionalism.* This school of thought partly extended the structuralism ideas, but it puts emphasis on the function of language and individual linguistic features. It argues that the structures of a language system are best understood with reference to the functions they carry out. In this context, Karl Bühler [62] has to be mentioned as one of the most popular functionalist scholars. In his work, Bühler introduced the *organon model* and identified three main functions of language: an *expressive* function that allows the adressers to express their own beliefs and feelings, a *representative* function that allows talking about the world, and an *appelative* function that allows the request or issue of a command [62, pp. 34-36]. In contrast to the structuralist approach, functional theories pay attention to the way language is actually used in communicative context, and not just to the structural relations between linguistic elements and thus emphasizes purpose and pragmatics of language [307].

Finally, by the mid 1950s, the linguistic school of *formalism* or *generative linguistics* has become increasingly influential. The term generative has been introduced by Noam Chomsky in his famous work *Syntactic Structures* [75]. He considers grammar to consist of a system of rules that is intended to generate exactly those combinations of words which form grammatical correct sentences in a given language. Thus, it is possible to produce a theoretically unlimited number of sentences from a limited number of means, i.e. words and grammar rules. Due to the strong grounding in well formed rules and expressions, generative linguistics has proposed several grammatical structures (see e.g. relational grammar [44], the generalized phrase structure grammar [139], or the head-driven phrase structure grammar [336]). The field of generative linguistics is most influential in the sub field of syntax, which focuses on the analysis of phrases and sentences.

In modern linguistics, the three schools of linguistics exist side by side. Moreover, the insights from theoretical linguistics are now applied within many other disciplines such as psychology, sociology, philosophy, and literature studies [414, pp. 21-26] which also lead to new branches within applied linguistics. Additionally, linguistics has found its way to computer science (*computational linguistics*) and artificial intelligence and resulted in promising application scenarios, such as machine translation [205, 170, 59], speech recognition [351, 178], or speech understanding [117]. To this end, computational linguistics makes use of concepts and models of theoretical linguistics to apply them on real world problems related to language ([388, p. 1],[414, p. 26]). However, these problems might also point to new issues or problems that are a concern for theoretical linguistics. Figure 4.2 depicts the various branches of linguistics and shows the interrelations between related areas of study.

As far as this thesis is concerned, the presented linguistic streams are complementary to each other and equally used to improve different aspects of

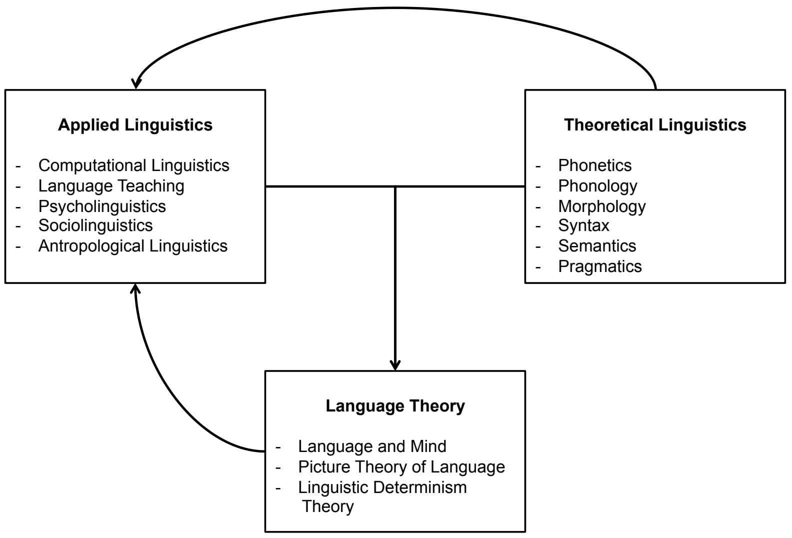


Fig. 4.2: Overview of Linguistic Fields, adapted from [414, p. 26]

process model labels. In general, this thesis applies available constructs and models from linguistics and adopts them in process models. We apply concepts from the branch of syntax in order to improve the text labels by studying and describing the structure of text labels. As far as the interpretation of labels is of interest, this thesis employs applicable concepts from the field of semantics. Finally, this thesis also investigates the text labels with regard to their comprehensibility in particular contexts, which refers to the functional aspect of linguistics and is particularly related to the field of pragmatics. The next section introduces the essential branches of linguistics and the essential underlying concepts and technologies in order to provide the means to analyze the text labels in process models adequately.

4.2 Fields of Theoretical Linguistics

According to Saussure, a language of any sort represents a system of signs and relations between these signs [381, p. 16]. He proposes to investigate the language with regard to the three branches of semiotics, i.e. syntactics, semantics, and pragmatics [296]. The next subsections are dedicated to these branches and discuss the main concepts and models. Section 4.2.1 will focus on the branch of syntax, Section 4.2.2 discusses the main ideas with regard to semantics, and finally, Section 4.2.3 introduces the branch of pragmatics.

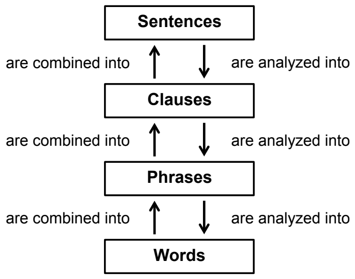


Fig. 4.3: Structure of Words into Phrases, Clauses, and Sentences [43, p. 100]

4.2.1 Syntax

The linguistic field of syntax is concerned with the combination of words into larger grammatical units, such as phrases, clauses, and sentences [43, p. 100]. Thereby, words are combined into phrases, phrases are combined into clauses and clauses may either be sentences by themselves or combined into complex sentences. We thus observe a strict hierarchy of combining words into complex grammatical units as depicted in Figure 4.3.

In the following, the concepts of words, phrases, etc. are discussed based on a running example. Consider the exemplary sentence:

The user creates a new business process model.

On the level of *words*, syntax is concerned with classifying the words of a sentence into different syntactical categories, also referred to as *parts of speech* [184, pp. 137-138]. On a very general level, the English language categorizes words into the following syntactic categories ([217, p. 38], [184, pp. 138-142]):

- Noun (N): Subjects or Objects of a sentence, such as user, business, process, model.
- Verb (V): Predicate of a sentence, such as create
- Adjective (Adj): Modifier of a noun, such as new.
- Adverb (Adv): Modifier of a verb, adjective, or adverbs, e.g. thoroughly.
- Preposition (Prep): Expressing spatial or temporal relations, e.g. towards, in or before.
- Determiners (Det): Expressing the reference of a noun in a context, e.g. the, a.
- Conjunctions (Con): Joining two or more clauses, e.g. and, or.

The next level of the sentence model involves *phrases*. Phrases are groups of at least one word that function as a constituent within a sentence [217, p. 26]. This means that they build a single unit of several words that are closely related with each other ([217, p. 26],[43, p. 109]). Usually, such phrases contain a head word which is defining the type of the phrase. Similar to the different syntactical categories of a word, we can distinguish several types of phrases. The most important phrases of the English language are:

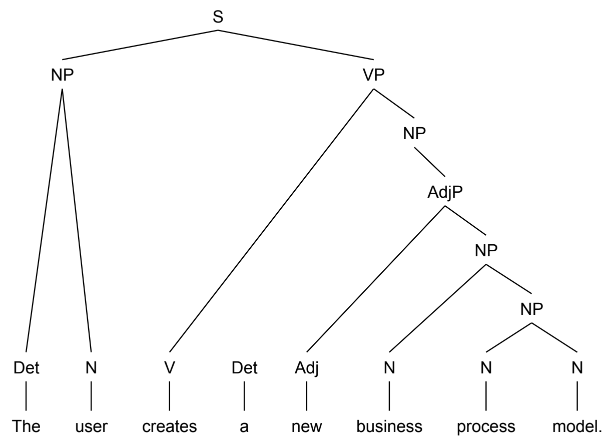


Fig. 4.4: Phrase Structure Tree of the Example Sentence

- Noun phrase (NP): Contains a single proper noun or a combination of several nouns, a determiner, and/or a adjective.
- Verb phrase (VP): Contains a single head verb or a combination of verb and noun phrase.
- Adjective phrase (AdjP): Contains a single adjective or a combination of these.
- Prepositional phrase (PrepP): Contains one preposition together with a noun phrase.
- Determiner phrase (Det): Contains a single determiner or a combination of these (e.g. all the, every)

For visualization purposes, the different phrases are depicted in a phrase structure tree. A *phrase structure tree* illustrates the linguistic structure of a grammatical unit and provides the syntactical category of the unit (noun, verb, etc.) as well as its size (word, phrase, clause, sentence) [217, p. 39]. As an example, Figure 4.4 depicts the phrase structure tree of the example sentence introduced earlier. The example sentence consists of two phrases, a noun phrase and a verb phrase. The noun phrase is given by the noun *user* and the verb phrase is given by the verb *creates*. The verb phrase further consists of a complex noun phrase with the head word *model*. Since the head word is further specified by the two nouns *business* and *process* as well as the adjective *new*, it is necessary to create additional phrase layers of adjective and noun phrases to capture all of the words.

Finally, words and word phrases are used to build clauses and sentences. A *clause* is the smallest grammatical unit which can express a complete proposition [217, p. 26]. A *sentence* on the other hand is understood as a complete thought and thus might consist of several clauses. Depending on the number of clauses, a sentence is either *simple*, *or compound*, *complex* [43, pp.

107-108]. A simple sentence can stand on its own and only contains one clause. It is thus called a main clause. The example sentence above is an example of a main clause. However, sentences may also consist of more than one clause. A *compound sentence* is a combination of main clauses that are linked by a coordinating conjunction, such as *and*, *or*, *but* etc. As an example for a compound sentence, consider the sentence:

The user creates a process model and analyzes it for weaknesses.

A *complex sentence* contains only one main clause with at least one subordinate clause. A subordinate clause cannot stand on its own and is always dependent on the main clause. Typically, subordinate clauses are introduced by specific subordinate conjunctions, such as *although*, *because*, or *when*. The following sentence is an example of a complex sentence:

The user creates a process model, because his boss told him to.

The introduced concepts from the branch of syntax leverage the description and analysis of any natural language sentence. Although process models labels do not represent proper English sentences with regard to the syntax [285], it is still possible to use these concepts and describe them in terms of words, phrases, and clauses.

4.2.2 Semantics

The branch of semantics deals with the systematic study of meaning in human language. While the philosophical perspective is more interested in the way and the circumstances under which objects are interpreted to convey specific meanings [443, 440], the linguistic branch investigates meaning that is conveyed by words, phrases, and sentences [43, p. 128]. Accordingly, the field of linguistics distinguishes between the lexical semantics (the meaning of words) and the sentential semantics (the meaning of phrases, clauses, and sentences). Since process models do not include proper English sentences [285], this thesis puts more emphasis on the lexical semantics and presents the most important concepts of this sub field of semantics.

Lexical semantics investigates the meaning of words in terms of meaning relations (or sense relations). These relations express a connection between two words of a certain language that semantically links these words with each other via a common concept. For example, consider the words *dog* and *barker* and the animal they refer to. While both terms refer to the same animal, the word *barker* is far more informal and negatively connotated expression to point to the animal. Nevertheless, these words are semantically related with each other since they both describe the same animal. Lexical semantics is concerned with a number of these sense relations. The most important sense relations, also in this thesis, are synonymy, homonymy, hyponomy, and meronymy [163, pp. 150 - 160].

In general, the *synonym* relation of two words is characterized by the fact that these two words may share the same or nearly the same meaning. Traditionally, the synonym relation between two words is defined as words that have exactly the same meaning (perfect synonymy) [163, pp. 157]. In practice, the share of words that have exactly the same meaning is, however, very rare. Some authors also argue that the phenomenon of perfect synonymy is theoretical, since there is no such pair of words that is interchangeable in every context of use [136, p. 181]. Hence, linguists have proposed to define synonymy in terms of *semantic similarity*, which has lead to a non-perfect synonymy relation [43, p. 129]. Accordingly, non-perfect synonyms share a common set of contexts in which they are mutually exchangeable. Examples for such synonyms are:

– bill $\leftrightarrow$ invoice: Sharing the meaning of an *itemized statement of money*
– goods $\leftrightarrow$ products: Sharing the meaning of *commodities offered for sale*
– buy $\leftrightarrow$ purchase: Sharing the meaning of *acquiring something for payment*

We already observed that the meaning of a word might change depending on the context of its use. It is frequently the case that a word has a number of different meanings and thus refers to different concepts of the real world. Considering Saussure's model, this is either caused by the sound sequence that referring to different concepts (*Homophony*) or by one equally spelled word that refers to different concepts of the real world (*Homonymy*). Since this thesis is mainly concerned with written words, it considers only the homonymy relation. Homonym examples include:

– application
  • In the sense of a *job application*
  • In the sense of a *software program*
– bank
  • In the sense of a *financial institution*
  • In the sense of a *river bank*
– bill
  • In the sense of an *itemized statement of money*
  • In the sense of a *banknote*

The concept of *Hyponomy* defines hierarchies between sense relations and describes the phenomenon when the meaning of one word is included in the meaning of another one [43, p. 135]. In case of nouns, the subordinate words are referred to as hyponyms and the superordinate is called a hypernym. For verbs, the respective verbs are called troponyms. Hyponym and troponym examples are given as follows:

– order is a hypernym of
  • credit order
  • production order
  • purchase order

- software is a hypernym of
  - upgrade
  - freeware
  - groupware
- to control is a troponym of
  - to manage
  - to handle

A specific hierarchy relation is given by the *meronomy* relation as it refers to words which are part of a whole concept [43, p. 135]. In this case, the parts are referred to as meronyms, while the whole is called holonym. For verbs, this relation is called entailment [292]. Meronym examples may be taken from the following list:

- tree is a holonym of
  - trunk
  - branch
  - leave
- car is a holonym of
  - car engine
  - tire
  - vehicle body
- to apply is a holonym of the verbs
  - to snore
  - to rest

This thesis is particularly concerned with semantic relations. These relations associate a word and its phonetic representation with a concept of the real world and defines a particular meaning of this word [381, pp. 65-67]. In many cases, these associations allow several words to refer to one concept (synonymy) or allow one word to be associated with several concepts (homonymy). For these reasons, the associated meanings of a word are best represented by so called *word senses* which implement them in a technical context [300]. There are, however, different approaches on how these word senses are organized, i.e. the generative approach and the enumerative approach.

The *generative approach* by Pustejovsky [348, 349] creates related word senses based on rules which capture regularities and properties of the referred concept. Word senses are expressed in terms of *qualia roles* that structure the basic knowledge about this concept. These qualia roles go back to Aristoteles basic elements to describe the meaning of lexical items, i.e. formal (identification of the object), constitutive (parts and constitution of an object), telic (the purpose of the object), and agentive (involved factors of the objects origin) [180]. The combination of qualia roles allows the creation of a particular word sense.

As an example for this representation, consider the following possible generative entry for the word *job application* in the notation of Pustejovsky

$$\begin{bmatrix} \textbf{job application} \\ \text{ARGSTR} = \begin{bmatrix} \text{ARG1} = \mathbf{x}\text{: physical object} \\ \text{ARG2} = \mathbf{y}\text{: request} \\ \text{ARG3} = \mathbf{z}\text{: job} \\ \text{ARG4} = \mathbf{v}\text{: applicant} \\ \text{ARG5} = \mathbf{w}\text{: company} \end{bmatrix} \\ \text{EVENTSTR} = \begin{bmatrix} \text{E}_1 = \mathbf{e_1}\text{: process} \\ \text{E}_2 = \mathbf{e_2}\text{: transition} \end{bmatrix} \\ \text{QUALIA} = \begin{bmatrix} \text{FORMAL} & = \text{hold}(\mathbf{x},\mathbf{y}) \\ \text{CONSTITUTIVE} & = \{\text{Motivation Letter, CV,} \\ & \quad \text{Attachments, ...}\} \\ \text{TELIC} & = \text{employment}(\mathbf{e_1},\mathbf{v},\mathbf{z}) \\ \text{AGENTIVE} & = \text{hire\_act}(\mathbf{e_2},\mathbf{v},\mathbf{w}) \end{bmatrix} \end{bmatrix}$$

Fig. 4.5: Example Qualia Roles of a Job Application

[348]. As depicted in Figure 4.5, the definition of the qualia roles of a job application first requires the definition of several arguments (the physical object, the request, the job, the applicant and the company) as well as two events (transition and process). Afterwards, the four qualia are defined. The formal qualia is given by a relation of the physical object holding all information regarding the request. The parts of the job application are given by the constitutive parts, such as motivation letter, CV, diploma, etc. The purpose of a job application is defined as an employment act that is described at a process which brings the applicant into the job. Finally, the agent qualia of a job application is given by a hire act that involves a transition of the applicant conducted by the company.

As an alternative to the generative approach, the *enumerative approach* lists all objective word senses in a sense inventory. If possible, the sense inventory partitions should only contain discrete word senses that cannot be reduced any further [300]. This approach is frequently adopted by several paper-based and machine-readable dictionaries. Due to the widespread adoption, this thesis will also follow the enumerative approach to refer to the word senses of a specific word. The association of discrete word senses is defined as follows:

Definition 4.1. (**Word Senses**). Let D be a dictionary of words, the senses of a word are defined by the function $Senses_D : W \times POS \rightarrow 2^S$, such that

– W is the set of words denoted in dictionary D.
– $POS = \{n, v, a, r\}$ is the set of open-class parts of speech (nouns, verbs, adjectives, and adverbs).
– S is the set of word senses that are encoded in the dictionary D. 2^S denotes the powerset of senses.

To illustrate the enumerative approach, consider again the word application. Among others, the following distinct word senses s_i of the noun application may be distinguished, i.e. $s_i \in Senses_D$(application,n):

- s_1: a written request for employment
- s_2: the action of using something for a particular purpose
- s_3: a software program that is used by humans to accomplish a task

An essential component of Definition 4.1 is the sources of *word senses*. In general, word senses are provided by dictionaries that contain conceptual knowledge about the world. There is a plethora of such dictionaries, such as thesauri, machine-readable dictionaries, and computational lexicons. *Thesauri* provide information about the words and their interrelationships, such as synonymy or antonymy [195]. Consequently, the word senses of a word are defined via the semantic relations to other words. A frequently employed thesaurus is the Rogets International Thesaurus [72] which contains 250,000 word entries organized in six classes and 1000 categories. *Machine-readable dictionaries* are dictionaries that provide various information about words in a particular language. Among others, this information includes possible meanings, etymologies, or phonetics. Examples are the Oxford Advanced Learners Dictionary of Current English [165], the Oxford Dictionary of English [412], and the Longman Dictionary of Contemporary English [345]. Finally, *computational lexicons* encode a rich semantic network of concepts, similar to ontologies. The most popular computational lexicon is WordNet, which captures various semantic relationships in a structured way [291, 292]. A similar resource is the BabelNet lexicon [302], which combines WordNet senses with the web encyclopedia Wikipedia and which provides support for different languages.

4.2.3 Pragmatics

The branch of pragmatics is concerned with the systematic study of how people understand and communicate more than the literal meaning of words or sentences when they interpret or produce utterances. Utterances are understood as spoken or written contributions that derive their meaning partly from the context of their appearance [43, p. 152]. There might not be a big difference between semantics and pragmatics, since both branches are interested in the original meaning of words and sentences in a particular context. While semantics is more interested in the meaning of words and sentences themselves, pragmatics brings the sender and recipient of the sentences into the focus and emphasizes the meaning of the sentence in the specific situation [176, p. 1]. Thus, pragmatics may be regarded as the study of *meaning in context*. Accordingly, it also takes the context into account, i.e. the discourse that surrounds a language unit. Moreover, pragmatics acknowledges that the context determines the interpretation of a language unit and that the context may be subject to change during the interaction with other people. In order to correctly interpret the meaning of a sentence according to its context, it is necessary that it complies with two essential concepts of pragmatics, i.e. deixis and the cooperative principle.

The term *deixis* corresponds to the Greek verb *deiknynai* and translates to *to point* or *to show*. In pragmatics, it refers to all linguistic means that point to a specific context in which a sentence or statement has been uttered and within which it has to be interpreted. These deictic expressions point to those things that build the context of a statement and may involve persons, places, objects, actions, or time ([176, p. 191-194],[163, p. 319-323]). All deictic expressions have in common that they require a larger context, since they cannot be understood independently of it. Understanding the context of a statement involves the identification of the perspective from which something has been communicated. This perspective is called the *deictic center* [43, p. 154]. If the deictic center cannot be found, it is hard to interpret and understand a statement and act accordingly. This is illustrated by the following example.

Imagine to find yourself in the situation in which the usual BPM lecture room is empty and the following notice is found on the door:

Introduction to Business Process Management
We are here today:
Guest Lecture of Prof. Dr. Marlon Dumas
D2.105

Since you are a participant in the lecture, you will know how to interpret this notice although you are not in the same context as the authors of the notice. This is achieved by some deictic expressions that clearly point to a specific group of persons and to a specific place. The expression *We* refers to all persons that are related to the lecture and indicates that only a specific group of people is addressed by the notice. Moreover, the expression *here* is usually used for locations close to the speaker and thus indicates a specific room at the university (D2.105). Altogether, it is possible to identify the deictic center as among the participants of the lecture and being reminded on the guest lecture. However, if you are external to the lecture, you might have difficulties in interpreting this notice since you might not be aware that the last line refers to a room or that the first line refers to a particular group of persons. In particular, you would not have any information about the time of the guest lecture, since you do not attend the lecture as part of your studies.

In order to avoid such interpretation bias, it is helpful to structure any written or spoken communication according to the *cooperative principle* by H. Paul Grice [147, p. 45]:

> "Make your conversational contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged."

From this principle, four maxims have been derived to structure communication [147, p. 45-46]:

1. The Maxim of Quantity: Make the contribution as informative as required, not more.
2. The Maxim of Quality: Make the contribution true (no false or unproven statements).
3. The Maxim of Relation: Make the contribution relevant.
4. The Maxim of Manner: Make the contribution clear, i.e. free of obscurity or ambiguity.

Applying these maxims to the example reveals that the notice itself is already in a good shape. It fulfills the maxims of quality, (the guest lecture is taking place in the respective room and at the time of the lecture), relation (the room is empty and the note gives an explanation why), and manner (it is free of ambiguity). Nevertheless, there might be some problems with the maxim of quantity since the note does not specify any time at which the guest lecture is scheduled. The readers need to know about the scheduled date and time are infer it from their regular lecture schedule. However, an arbitrary, yet interested reader may not be able to infer time and date because he or she is not part of the regular BPM lecture.

This thesis also investigates the pragmatic dimension of the textual labels of process models. In this setting, it is important that the textual information in process models complies with the cooperative principle and the derived maxims. It is important that the text labels of process models meet the maxims of quantity and manner, i.e. that they provide sufficient information to understand the process model and to derive a set of actions from it. The maxims of quality and relevance are hard to assess for process models. The reason is that such an assessment would require additional information from the modeling domain and context. If this information is missing, it is not possible to check the truth of text labels and their relevance for the underlying business process.

4.3 Overview of Natural Language Processing Techniques

Natural language processing (NLP) is concerned with the interactions between computers and human natural language [221, p. 1]. NLP is an interdisciplinary field of computer science, linguistics, and psychology [289] and is applied to various language tasks, such as machine translation [205, 170, 59], speech recognition [351, 178], speech understanding [117], text generation and summarization [262, 169, 306], or word sense disambiguation [172, 11, 300]. Since the field of natural language processing is considerably big, we will first provide a general overview of the most relevant techniques leveraging syntactical, semantic, and pragmatic analysis.

Table 4.1: Overview of Syntactical NLP Techniques

Purpose	Approach	Author
Text Analysis with Corpora	American National Corpus (ANC)	Ide and Suderman [171]
	British National Corpus (BNC)	BNC Consortium [83]
	Brown Corpus	Francis and Kucera [133]
	Corpus of Contemporary American English (COCA)	Davies [89]
Natural Language Tagging	Rule-based POS Taggers	Voutilainen [427, 428]
	Statistical POS Taggers	Chang and Manning [70]
		Brants [53]
		Schmid [387]
		Ratnaparkhi [352]
		Toutanova and Manning [418]
Natural Language Parsing	Rule-based Parsing	Stahl et al. [411]
		Yngve [448]
	Probabilistic Parsing	Dowding et al. [107]
		Collins [82]
		Klein and Manning [200, 199]
		Charniak [74]

Table 4.1 gives an overview of syntactical NLP techniques. In general, we distinguish between the analysis of natural language corpora, natural language tagging, and natural language parsing. *Natural language corpora* are large collections of natural language texts consisting of thousands and millions of words. Corpora contain a representative sample of a particular type of naturally occurring language to investigate several quantitative and qualitative characteristics, such as frequencies or collocations of words [23, p. 2]. In many cases, these corpora have been further enriched by additional information to foster corpus-based linguistic research. This processing is also known as corpus annotation [23, p. 2].

Natural language tagging implements the process of assigning a part-of-speech (POS) or another syntactic word class to each word in a sentence, paragraph, text, or corpus [184, p. 147]. These techniques typically get a string of words as well as a specified tagset as an input. The result is a single POS tag for each word that best reflects the syntactic class of the word in the given text string. As shown in the table, there are mainly two classes of tagging techniques, i.e., rule-based taggers and stochastic taggers [184, p. 137]. *Rule-based taggers* generally employ a large database of hand-written grammar rules to resolve conflicts of ambiguous tags. Opposed to that, *stochastic taggers* use statistical methods to determine the best POS tag. Therefore, stochastic parsers require training on large text corpora.

Table 4.2: Overview of Semantic NLP Techniques

Purpose	Approach	Author
Lexical Semantic Analysis	WordNet	Miller et al. [293, 292, 291]
	EuroWordNet	Vossen [426]
	BabelNet	Navigli and Ponzetto [302]
	VerbNet	Schuler [389]
	Proposition Bank	Palmer et al. [318]
Semantic Field Analysis	FrameNet	Baker et al. [22]
	VerbNet	Schuler [389]
	ConceptNet	Liu and Singh [256]
	PhraseNet	Li et al. [251]
Word Sense Disambiguation	Supervised WSD	Cai et al. [65]
		Agirre and Lopez de Lacalle [12]
		Niu et al. [308]
	Unsupervised WSD	Pederson [320]
		Kern et al. [192]
		Koeling and McCarthy [206]
	Knowledge-based WSD	Navigli and Velardi [305]
		Chan et al. [69]
		Novischi [310]

Natural language parsing maps a sentence consisting of several tagged words to its phrase structure tree which represents the internal structure of the sentence. This syntactical sentence structure then serves as an important input for semantic analysis techniques, such as question answering [354, 449] or information extraction [128, 118]. In general, the literature distinguishes between approaches that use a context-free grammar and that use statistical methods. Examples for such parsers are given in the table.

Table 4.2 distinguishes semantic NLP techniques that enable the analysis of lexical semantics, semantic fields, as well as techniques for word sense disambiguation (WSD). We already mentioned that *lexical semantics* investigates the meaning of words in terms of sense relations, such as synonymy or hyponymy. For the analysis of lexical semantics, computational lexicons implement a traditional lexicons and contain a collection of words with information regarding usage, definitions, etymologies, phonetics, pronunciations, or translation. Moreover, they also encode a rich semantic network among the words and concepts to which these words refer [300]. Examples include WordNet [293, 292, 291], EuroWordNet [426], VerbNet [389], the Proposition Bank [318], and BabelNet [302].

Semantic fields comprise a set of words that are semantically close or related to each other [120, p. 14]. This means that words in a semantic field are not necessarily synonyms, but are all used to talk about the same general phenomenon [15, p. 239]. For this type of analysis, one can employ the systems VerbNet [389], FrameNet [22], ConceptNet [256], or PhraseNet [251].

Word sense disambiguation describes the task of automatically identifying the word senses of each word in a sentence or text. WSD is comparable to POS tagging [23] or text categorization [393] where to each element is assigned a predefined tag or a category. However, the WSD task is more complex since the set of classes, i.e. the word senses, typically changes depending on the word to be assigned. Consequently, each word has its own set of word senses among which the most suiting one has to be selected. For these reasons, WSD is regarded as an AI-complete problem [260], which refers to problems whose difficulty is equivalent to solving central problems of artificial intelligence such as the Turing Test [421]. There are several methodological approaches to perform WSD, i.e. supervised WSD, unsupervised WSD, and knowledge-based WSD. Supervised methods use machine-learning to train a classifier from sense-annotated text corpora. Afterwards, this classifier is used to assign the correct word sense. Among the most successful supervised WSD systems, we mention the approaches by Cai et al. [65], Agirre and Lopez de Lacalle [12], and Niu et al. [308] achieving an accuracy of more than 86%. Unsupervised WSD methods are based on unlabeled corpora and do not exploit any manually sense-tagged corpus to select a suitable word sense. Among the approaches, those of Pederson [320], Kern et al. [192], and Koeling and McCarthy [206] are the most successful. Knowledge-based WSD systems make use of external lexical resources, such as machine-readable dictionaries, thesauri, and computational lexicons to assign the appropriate word sense. With regard to performance, the systems by Navigli and Velardi [305], Chan et al. [69], and Novischi [310] achieve a F1-score of at least 81.45%. Most recently, BabelNet also includes a knowledge-based disambiguation approach that can compete with these state-of-the-art systems. On average, it achieves a F1-score of 82.5%.

Finally, Table 4.3 gives an overview of important techniques for pragmatic purposes. In general, the literature identifies discourse interpretation, language generation, and machine translation as major areas of pragmatic techniques [184]. *Discourse interpretation* is concerned with the analysis of a group of collocated and related sentences, which builds the discourse. For the automatic interpretation of such discourses, we distinguish between plan-inference interpretation and cue-based interpretation approaches. The *plan-inference* approaches make use of the belief-desire-intention model (BDI) for intelligent agents. This model represents the agent's beliefs, desires and intentions and uses these concepts to solve a particular problem, i.e. in our case the interpretation and comprehension of the contextual meaning of the discourse [16]. The cue-based approaches use of different sources of knowledge (cues) for the discourse interpretation task, such as lexical, collocational, syntactic, or conversational cues and combine them with machine-learning algorithms, trained on a corpus of hand-labeled discourses. Examples of these approaches may be taken from the table.

Table 4.3: Overview of Pragmatic NLP Techniques

Purpose	Approach	Author
Discourse Interpretation	Plan-Inferential Interpretation	Allen and Perrault [16]
		Raux and Eskenazi [353]
		Loisel et al. [257]
	Cue-based Interpretation	Hinkelmann and Allen [164]
		Jurafsky et al. [183]
		Webb [430]
Language Generation	KPML-System	Bateman [28]
	Amalgam	Corston-Oliver et al. [84]
	RealPro	Lavoie and Rambow [232]
	Atlas.txt	Thomas and Sripada [416]
	Artequakt	Kim et al. [196]
Machine Translation	Statistical Machine Translation	Brown et al. [60]
	Supervised Machine Translation	Koehn et al. [205]
	Rule-based Machine Translation	Forcada et al. [129]
	Machine Translation with Neural Networks	Devlin et al. [100]

Language Generation is the process of constructing natural language outputs from non-linguistic inputs. The goal of this process is to map from meaning to understandable natural language text by applying the steps of selecting the relevant content and words, structuring a single sentence, and finally structuring the entire discourse [184, pp. 761-762]. Language generation is frequently employed in application areas where the output texts should comply with a desirable format, such as weather forecasts, stock markets, documentations, and technical manuals. Accordingly, the depicted systems of the table may be used to generate natural language texts for these purposes.

Finally, *machine translation* is concerned with the automatic translation of texts from one language into another [184, pp. 797]. Research in this field has brought forth a number of translation techniques that make use of statistics [60], machine-learning techniques [205, 100], or specific translation rules for word pairs [129].

After this general overview of language processing techniques, we will now further detail those techniques that are most relevant for this thesis.

4.4 Relevant Techniques for Syntactical Analysis

In this section, we focus on NLP techniques that leverage the analysis of natural language syntax and that are used in the course of this thesis. In particular, we will provide a deeper understanding of available text corpora in

Section 4.4.1. Afterwards, we turn the focus on tagging and parsing of natural language texts in Section 4.4.2 and of process models in Section 4.4.3.

4.4.1 Natural Language Corpora

We already mentioned that a natural language corpus is a large collection of texts consisting of thousands and millions of words that are used for several quantitative and qualitative investigations of texts and words, such as frequencies or co-occurrence [23, p. 2]. Moreover, corpora are frequently enriched by additional linguistic information [23, p. 2] to make them accessible for various NLP approaches. This typically involves the assignment of syntactical categories (verb, noun, adjective, etc.) to each word of the corpus in the form of POS tags, which is referred to as part-of-speech tagging (POS-tagging). In some cases corpora may also incorporate information about the syntactic sentence structure in form of a phrase structure tree or about the meaning of words and sentences. A big achievement in this field was the creation of the so called *Penn Treebank* [265]. The Penn Treebank consists of over 4.5 million words of American English and includes POS information as well as a parsing of sentence structures and semantic annotations [197]. The Penn Treebank also introduced a POS tagset which has been widely adopted by state of the art tagging and parsing software, such as the Stanford Tagger and Parser [200, 199]. Appendix A provides an overview of the Penn Treebank POS tags.

Table 4.4 summarizes the main characteristics of English and German text corpora. Moreover, nearly all of them have been annotated with syntactical categories. In some cases, these corpora have also been parsed manually by linguists which ensures a high quality of these information. However, this is not usual since the human annotation is costly in terms of human efforts and time. Thus, it is common to parse a subset of a corpus by hand, while a machine is parsing the whole corpus.[4] Additionally, the table reveals a notable difference in terms of size and sources. Frequently, the size of the corpus is directly related to the corpus sources from which texts have been sampled. For example, the ANC corpus is one of the largest corpora in the table and has been built from newspapers, research articles, fiction and non-fiction books, other documents such as brochures, booklets, and flyers. However, this does not imply that this corpus is the best choice for each linguistic analysis. Considering a domain-specific language analysis, it might be more appropriate to use texts from books, articles, or documents of the respective domain. Therefore, the language analyst needs to pay careful attention when selecting a corpus to develop tools that rely on such data sources.

This thesis proposes refactoring techniques for the textual content of English process models. For this reason, the selection of corpora is restricted to

[4] This procedure has been applied for the OANC corpus which was parsed by a machine. A subset of the OANC, the MASC (Manually Annotated Sub-Corpus), has been annotated and parsed by human annotators.

Table 4.4: Overview of Text Corpora

	Corpus	No. Words	Features	Text Sources
English	Google N-Gram Corpus[1]	155 billion	-	Books retrieved from Google Books
	American National Corpus (ANC) [171]	40 – 50 million	T+P	newspapers, research articles, fiction and non-fiction books, other documents
	Open ANC (OANC)[2]	15 million	T+P	Subset of ANC
	British National Corpus (BNC) [83]	100 million	T	newspapers, research articles, fiction and non-fiction books, other documents
	Brown Corpus [133]	1 million	T	books, newspapers, scientific articles, other documents
	Corpus of Contemporary American English (COCA) [89]	450 million	T	academic journals, newspapers, fiction books, and magazines
German	TIGER Corpus [52]	900,000	T+P	newspapers
	LIMAS Corpus [141]	1 million	-	books, newspapers, other documents
	Leipzig Corpora Collection [3]	425 million	-	newspapers

Legend: T: Corpus has been tagged, T+P: Corpus has been tagged and parsed

English corpora only. Moreover, since most of the process models cover a large spectrum of fields and interests, we require a corpus that also includes texts from various areas. Therefore, we use the Open American National Corpus (ANC) for this thesis since it offers a sufficient subset of texts stemming from various sources of knowledge, such as newspapers, research articles, fiction and non-fiction books, as well as other documents. Moreover, the corpus has also been tagged and parsed in advance, which saves valuable preprocessing time.

4.4.2 Tagging and Parsing of Natural Language Text

We also introduced *tagging* as being the process of assigning a POS tag or other syntactic word classes to each word in a sentence, paragraph, text, or corpus [184, p. 147] as well as several available taggers. To illustrate the basic idea of tagging, we use the example sentence of section 4.2.1. Depending on the underlying tagset, the single tags vary with regard to specific tags. In the example, we use the Stanford tagger and parser and get the following result of the tagging process. We recognize the aforementioned word classes in the tagging example, i.e. determiners (DT), nouns (NN), verbs (VBZ), adjectives (JJ):

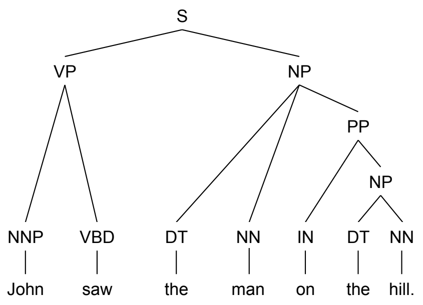
> The/DT user/NN creates/VBZ a/DT new/JJ business/NN process/NN model/NN ./.

It has to be noted that the automatic assignment of tags is not a trivial task, even for small and simple examples. For humans, it is relatively easy to recognize nouns and verbs and to resolve ambiguous cases which have more than one possible POS tags. For instance, the word *process* might be ambiguous as it can either be a noun (a set of activities in a particular order) or a verb (to deal with something in a routine way). Therefore, a main problem of POS tagging is to resolve these ambiguities and choose the appropriate tag for the given context [184, p.149].

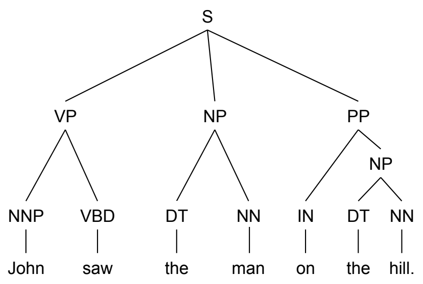
There are different possibilities to resolve ambiguities on word level, i.e. rule-based taggers and stochastic taggers. As already mentioned, *rule-based taggers* generally employ a large database of hand-written grammar rules to resolve conflicts of ambiguous tags. This means that rule-based taggers do not rely on statistical inferences, but employ predetermined rules to eliminate ambiguous cases. For example, these rules specify that a word is a noun rather than a verb, if it follows a determiner. The works on rule-based tagging approaches mainly build on approaches from the sixties and seventies [156, 201, 145]. Today, the capabilities of these approaches are enhanced by integrating bigger rule sets and comprehensive dictionaries [184, pp. 137-139]. The basic architecture of a rule-based tagging algorithm is a two-stage approach. At the first stage, a dictionary is consulted to derive a list of potential POS tags for each word. At the second stage, the rule set is used to reduce the set of potential parts of speech to a single tag. Examples of rule-based taggers involve the EngCG tagger [427, 428] or the Stanford Temporal Tagger for temporal expressions [70]. *Stochastic taggers* use statistical methods to determine the best POS tag. Therefore, stochastic parsers are trained on large corpora. Popular algorithms for POS-tagging make use of the Hidden Markov Model (HMM) [184, pp. 182-185] or the Maximum Entropy model (MaxEnt) [184, pp. 199-207]. The goal of statistical tagging is finding the best sequence of tags that corresponds to a particular input sequence. Prominent examples for stochastic taggers involve the Trigrams'n'Tags [53], the TreeTagger [387], or MaxEnt taggers as proposed by Ratnaparkhi [352] and Toutanova and Manning [418].

Parsing addresses the problem of mapping a sentence consisting of several tagged words to its phrase structure tree which represents the structure of the sentence. This syntactical sentence structure then serves as an important input for further linguistic analysis techniques. In general, the literature distinguishes between approaches that use a context-free grammar and that use statistical methods.

Context-free Grammar parsers try to build a parse tree by applying declarative rules that are given by context-free grammars. The creation of the phrase structure tree thus resembles a search problem. The search space is only limited by the input itself and the grammar. The former guarantees that leaves of such a tree equal the input sentence, while the latter guarantees that the root must be a sentence. Accordingly, we may distinguish between parsers that build the sentence structure by beginning from sentence level (the root), as presented in the approach by Stahl et al. [411], and those who start with the input words



(a) Phrase Structure Tree A



(b) Phrase Structure Tree B

Fig. 4.6: Example of Ambiguous Phrase Structure Trees

(the leaves) to identify the sentence structure. Yngve [448] and Dowding et al. [107] have proposed parsers that apply the bottom-up approach.

A particular challenge for context-free grammar parser is structural ambiguity, which arises when the parsing algorithm needs to choose among several possible phrase structure trees [217, pp. 51-52]. To illustrate the problem of structural ambiguity, we use the example sentence *John saw the man on the hill* which leads to the two phrase structure trees as depicted in Figure 4.6. Both trees are valid interpretations of the sentence with, however, slightly different meanings. In Tree 4.6(a), the sentence tells us that a particular man is seen, i.e. the man on the hill, whereas, in tree 4.6(b), the observer John was on a hill when he saw the man. The fact that there are many, also unreasonable, phrase structure trees for sentences is a common problem that affects all parsers. Parsers need to be able to choose the correct phrase structure tree among all possible ones. Unfortunately, this task requires additional knowledge which is not readily available during syntactic processing.

Probabilistic parsers are used in order to solve the problem of structural ambiguity. A probabilistic parser offers a solution to the ambiguity problem by employing a statistical syntax model. This model assigns probabilities to each phrase structure tree. These probabilities are learned by supervised training of a statistical model from tree-banks and parse trees provided by human linguists, such as the Penn Treebank [265]. Afterwards, the tree having the highest probability is chosen as the most-probable interpretation of a given input sentence. In fact, due to the prevalence of ambiguity, most modern parsers are of necessity probabilistic. The simplest augmentation of the context-free grammar with probabilistic elements is the Probabilistic Context-Free Grammar (PCFG), first proposed by Booth [48]. Other prominent examples include the parsers developed by Collins [82], by Charniak [74], and by Klein and Manning [200, 199].

In this thesis, we will use the statistical tagger and parser which have been developed by Toutanova and Manning [418] as well as Klein and Manning

Table 4.5: Typed Dependencies for the given Example Sentence

Type	**Governor**	**Dependent**	**Explanation**
det	(user-2	, The-1)	determiner relation
nsubj	(creates-3	, user-2)	subject-verb relation
det	(model-6	, a-4)	determiner relation
compound	(model-6	, process-5)	compound words relation
dobj	(creates-3	, model-6)	verb-object relation
cc	(creates-3	, and-7)	and-conjunction relation
conj	(creates-3	, analyzes-8)	relation of conjuncted elements
dobj	(analyzes-8	, it-9)	verb-object relation
case	(weaknesses-11	, for-10)	object of the preposition
nmod	(analyzes-8	, weaknesses-11)	noun modifier relation

[200, 199] at Stanford University. The main advantage of this technology is that both techniques have been integrated into one self-contained piece of NLP software that performs tagging and parsing with high quality and performance. Moreover, it has been extended by additional features which leverage additional syntactical analyses based on the parsing result. One of these features is capable to recognize syntactical elements of a given sentence, e.g. subjects and predicates. For that purpose, this parser also supports the extraction of syntactical elements and grammatical relationships from natural language texts [90] and represents them in specific data structures, i.e. typed dependencies [92].

Typed dependencies consist of a triplet that specifies the name of the relation, the governor (the head of the dependency) and dependent (the word that is dependent from the head word) [91]. We consider the following example sentence to illustrate the concept of typed dependencies: *The user creates a process model and analyzes it for weaknesses*. We receive the output shown in Table 4.5 by using the Stanford Parser for dependency determination [68]. Among others, the sentence contains one subject-predicate dependency (nsubj) and two verb-object dependencies (dobj). The subject-predicate dependency is formed between the noun *user* and the verb *creates*, while the verb *creates* and the noun *model* are in a direct-object dependency. Furthermore, we also have information about the position of the single words within a sentence. For instance, the noun *model* in the direct-object dependency is found on position 6 of the input sentence. Finally, we conceptualize the typed dependency relations in order to correctly refer to correctly refer to the set of typed dependencies.

Definition 4.2. (**Set of Typed Dependencies**) Let C be a corpus of natural language text. The set of typed dependencies Dep_C of a corpus C is a triple $(T \times W \times W)$, such that

– T contains the different kinds of typed dependencies as defined by Marneffe et al. [91].
– W is the set of all words, such as nouns, verbs, determiners, prepositions, conjunctions, etc.

Using the syntactical information in natural language sentences is essential for further analysis with regard to semantics and pragmatics. This also counts for process models which, however, only have fragments of natural language that do not represent proper English sentences. Since tagging and parsing technology require appropriate sentence structures, the performance of these technologies is limited when applied to process models [249]. For these reasons, we have a closer look on parsing the text fragments of process models in the next section.

4.4.3 Parsing of Process Model Elements

There is one major challenge that prevents tagging and parsing techniques to be directly applicable on process models and to perform with reliable results [249],[237, p. 52], i.e. the shortness of model element labels. The shortness of element labels limits the quality of tagging and parsing results, because the introduced parsers require a certain degree of context in form of a sentence or a paragraph as well as a tagged training corpus to correctly derive phrase structure trees and typed dependencies. Such a resource is, however, not available for process models, such that a tagger and parser cannot be trained as usual. Moreover, the process model labels do not even specify a regular grammatical sentence, but use different grammatical styles to describe the model element [285]. These styles are not covered in available linguistic corpora which makes it even more difficult for taggers and parsers to identify the correct word classes and phrases. In consequence, we require alternative approaches to use the linguistic information in process models.

A comprehensive solution of parsing process models and their labels has been proposed by Leopold [237, pp. 49–80]. The author has developed a parsing and annotation technique particularly tailored for process models that outperforms current natural language parsers. In order to deal with the challenge of textual shortness of labels, the technique makes use of specific naming styles of modeling elements to parse the element labels. These parsing rules are grounded on prior study on activity labeling styles by Mendling et al. [285]. In this study, the authors identified different activity labeling styles and investigated if the labeling style influences model users in intuitively understanding the labels assigned to the construct. This was confirmed in an empirical questionnaire in which participants were required to answer model-related comprehension questions. Moreover, the authors also revealed

Table 4.6: Labeling Styles of Process Model Elements, adapted from [237, pp. 53-57]

Labeling Style	**Syntactical Structure**	**Example**
Verb-Object	VP NP	Check application, Request missing papers
Action-Noun (regular)	NP	Application check, Missing papers request
Action-Noun (of)	NP `of` NP	Check of application, Request of missing papers
Action-Noun (gerund)	VBG NP	Checking application, Requesting missing papers
Descriptive	NP VBZ NP	Manager checks application, Missing papers are requested
Participle	NP VBN	Application checked, Missing papers requested
Modal	NP MD VB VBN	Application must be checked, Missing papers can be requested
Adjective	NP ADJ	Application is correct, Documents are complete
Categorization	NP VB NP	Customer is member
Participle-Question	NP VBN .	Application checked?, Missing documents requested?
Infinitive-Question	VP NP .	Check application?, Request missing papers?
Adjective-Question	NP ADJ .	Application is correct?, Documents are complete?

Legend: NP: Noun Phrase (e.g. data, application, customer documents) VP: Verb Phrase (e.g. check, requests, requested)

components that an activity label should contain, i.e. an action, a business object, and an additional information fragment. The *action* specifies a specific stream of tasks that needs to be carried out in a process. The *business object* describes a work item on which the action is applied. The *additional fragment* brings further information to the model reader and may help to improve the understandability of the construct. As an example, consider the activity *Check application* from the process model in Figure 2.5. This activity contains the action *to check* and the business object *application*. However, it does not describe any additional information fragment. Alternatively, the same activity could also be described with the label *Application check*. In this case, the grammatical style of the activity has changed capturing the action as a noun. These different components and the style in which they are communicated are the foundation of the parsing rules.

In order to derive the parsing rules, several process model collections have been manually annotated to gain a deeper understanding of the labeling practice [237, p. 53]. This annotation process included process modeling collections that varied along several dimensions, such as modeling notation, model size, business domain, and labeling quality. In this work, 1,450 process models with almost 30,000 labels have been annotated and aggregated to representative labeling styles for activities, events, and gateways. Table 4.6 provides an overview of these regular labeling styles, including their syntactical core structure and examples. For the description of the syntactical structure, the syntactic categories of the Penn Treebank are used (see Appendix A).

In case of activity labels, there are three main classes of labeling styles [237, pp. 53-54]. In *verb-object* labels, the action is given as an imperative verb in the beginning of the label, followed by the business object. The business object either consists of a single noun or a compound of several nouns. The activity labels *Check application* or *Request missing papers* are examples of the verb-object style. *Action-noun* labels express the action in form of a noun and use different grammatical structures to link it with the business object. Frequently, the action of an activity is nominalized and provided after the business object (Action-noun (regular)), such as in *Application check* or *Missing paper request.* Alternatively, the preposition *of* is used to link the nominalized action with the business object (Action-noun (of)). As examples, consider *Check of application* or *Request of missing papers.* The action-noun style (gerund) contains a gerund in the beginning of the label which is then followed by the business object. Examples are *Checking application* or *Requesting missing papers.* Finally, *descriptive* labels provide a text fragment that is most similar to a comprehensive sentence, as it contains a subject-predicate-object structure. In many cases, this style also specifies the role executing this activity. Examples of descriptive labels are *Manager checks application* or *Missing papers are requested.*

Events represent a specific state within the process. Obviously, events need to be expressed in different ways than activities [237, p. 54]. In total, there are four classes of event labeling styles. The first event labeling style (Participle Event) combines a business object with a past participle construction of a verb. The labels *Application checked* or *Missing papers requested* are instances of this labeling style. Similar to that, the modal event style incorporates an additional modal verb such as `can`, `must`, or `shall` into the label. Then, the model verb is followed by an auxiliary and a participle verb, such as in *Application must be checked* or *Missing papers can be requested.* Event labels that fall into the adjective event style use an adjective construction at the end of the label. As examples, consider the events *Application is correct* or *Documents are complete.* The categorization event style combines two nouns with each other, such that the second noun represents a category or class for further processing. The event label *Customer is member* is an example of this labeling style.

Gateways indicate that a particular decision needs to be taken in order to continue a particular path in a process model. Hence, all gateway labels usually

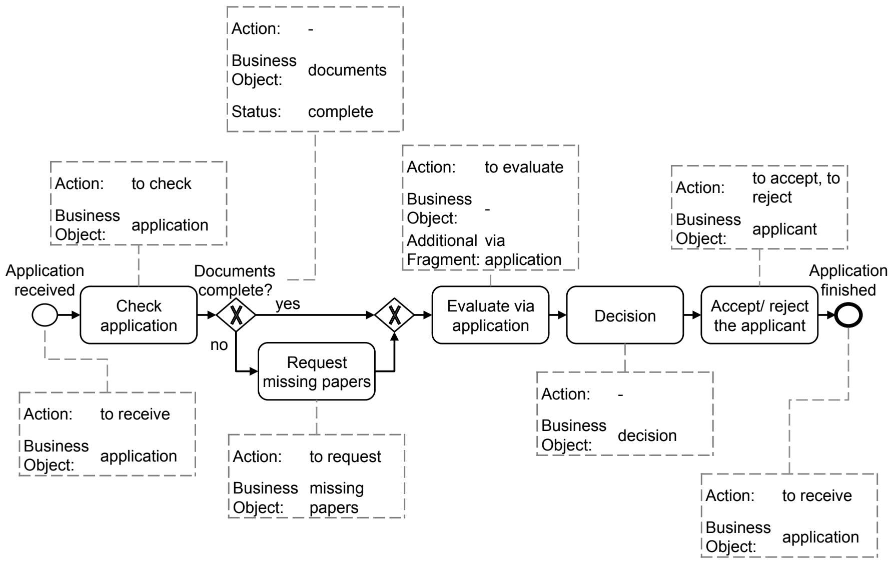

Action: -
Business Object: documents
Status: complete
Action: to check
Business Object: application
Action: to evaluate
Business Object: -
Additional Fragment: via application
Action: to accept, to reject
Business Object: applicant
Application received
Check application
Documents complete?
yes
no
Request missing papers
Evaluate via application
Decision
Accept/ reject the applicant
Application finished
Action: to receive
Business Object: application
Action: to request
Business Object: missing papers
Action: -
Business Object: decision
Action: to receive
Business Object: application

Fig. 4.7: Job Application Process with Annotated Elements

end with a question mark to explicitly point to this decision. Again, three main classes of gateway labels have been derived by Leopold [237, pp. 54-55]. The first class of gateway labels combines a business object with a past participle construction of a verb. The labels *Application checked?* or *Missing documents requested?* are examples of the participle-question style. The infinitive-question style combines a verb-object construct with a question mark. Generally, the infinitive verb is positioned in the first position of the label followed by the business object and a question mark. *Check application?* or *Request missing papers?* are instances of this gateway style. The adjective-question style is similar to the adjective-event style. It uses a business object in the beginning and an adjective construction at the end of a label, such as in *Application is correct?* or *Documents are complete?*.

Based on these general labeling styles, a process model parser has been developed by Leopold [237, p. 58–70] that annotates a model element with the relevant components. A comprehensive discussion and evaluation of the model parser is provided in [249] and [238, p. 71-78]. For these reasons, we focus on the application of the model parser to an example process and provide a formalism to refer to the different label components. Consider again the example process model as depicted in Figure 4.7. The figure shows the job application process from section 2.4, Moreover, all of the labeled elements have been annotated with the respective components by the model parser. For example, the activity *Request missing papers* is annotated with the action *to request* and the business object *missing papers*. The additional information component is left empty, because the label does not specify any further information. Next to it, the activity *Evaluate via application* contains the action *to evaluate* and the additional fragment *via application*. However, a business object is missing. Both of these activities are instances of the verb-object style. Looking at the start event *Application received*, the model parser classifies this label as a participle-event and accordingly identifies the object *application* and the past participle of the verb *to receive*.

As this thesis further investigates the label components of the process model elements, the parsing and annotation technique of Leopold is used to access the textual information in a standardized way. We use the following definitions to refer to the components of activities, events, and gateways [247] in order to consistently refer to the respective component of a process model element. These definitions are explained by taking the process model in Figure 4.7 as an example.

Definition 4.3. (**Label Components of Process Model Activities**). Let $\mathrm{P} = (A, E, G, F, R, P, L, \rho, \pi, \lambda, \gamma, \tau)$ be a process model, and $a \in A_{\lambda}^{\mathrm{P}}$ a labeled activity of process model P. Further, let W_V be the set of all verbs and W_N be the set of all nouns. Accordingly, the label components of an activity label $l = \lambda(a)$ are given by the following functions:

- $\alpha_A : L \mapsto W_V$ refers to the action of the activity a.
- $\beta_A : L \mapsto W_N$ refers to the business object of the activity a.
- $\gamma_A : L \mapsto W_N$ refers to the additional information fragment of the activity a.

As an example, consider the activity $a_1 \in A_{\lambda}$ from the process model in Figure 4.7. The label of this activity is given with $l_{a_1} = \lambda(a_1) =$ *Check application*. According to definition 4.3, the components are given as follows:

- $\alpha_A(l_{a_1}) =$ *to check*
- $\beta_A(l_{a_1}) =$ *application*
- $\gamma_A(l_{a_1}) = \emptyset$

Definition 4.4. (**Label Components of Process Model Events**). Let $\mathrm{P} = (A, E, G, F, R, P, L, \rho, \pi, \lambda, \gamma, \tau)$ be a process model, and $e \in E_{\lambda}^{\mathrm{P}}$ a labeled event of process model P. Further, let W_V be the set of all verbs, W_N be the set of all nouns, and W_{ADJ} be the set of all adjectives. Accordingly, the label components of an event label $l = \lambda(e)$ are given by the following functions:

- $\alpha_E : L \mapsto W_V$ refers to the action of the event e as given by the participle event style.
- $\beta_E : L \mapsto W_N$ refers to the business object of the event e.
- $\sigma_E : L \mapsto W_{ADJ}$ refers to the status of the event e as given by the adjective-event style.

In case of the event annotation, we consider the end event $e_2 \in E_{\lambda}$ from the process model in Figure 4.7. The label of the event is given with $\lambda(e_2) =$

Application finished. According to definition 4.4, the components are given as follows:

- $\alpha_E(l_{e_2}) =$ *to finish*
- $\beta_E(l_{e_2}) =$ *application*
- $\sigma_E(l_{e_2}) = \emptyset$

Definition 4.5. (**Label Components of Process Model Gateways**). Let $\mathrm{P} = (A, E, G, F, R, P, L, \rho, \pi, \lambda, \gamma, \tau)$ be a process model, and $g \in G^{\mathrm{P}}_{\lambda}$ a labeled gateway of a process model P. Further, let W_V be the set of all verbs, W_N be the set of all nouns, and W_{ADJ} be the set of all adjectives. Accordingly, the label components of a gateway label $l = \lambda(g)$ are given by the following functions:

- $\alpha_G : L \mapsto W_V$ refers to the action of the gateway g as given by the participle and infinitive question style.
- $\beta_G : L \mapsto W_N$ refers to the business object of the gateway g.
- $\sigma_G : L \mapsto W_{ADJ}$ refers to the status of the gateway g as given by the adjective-question style.

Finally, consider the gateway $g_1 \in G_{\lambda}$ and its label $\lambda(g_1) =$ *Documents complete?* of the process model in Figure 4.7. According to definition 4.5, the following gateway components are retrieved:

- $\alpha_G(l_{g_1}) = \emptyset$
- $\beta_G(l_{g_1}) =$ *documents*
- $\sigma_G(l_{g_1}) =$ *complete*

By using the discussed model parser, it is possible to access the text information from labels in a structured way and to make them available for further semantic analysis. The relevant techniques for semantic analysis are discussed in the next chapter.

4.5 Relevant Techniques for Semantic Analysis

This section is concerned with NLP technologies that leverage the analysis of semantics in the course of this thesis. In particular, we will provide a deeper understanding of computational lexicons and their capabilities to implement word senses and semantic relations in Section 4.5.1. Afterwards, we turn the focus on word sense disambiguation and illustrate the main idea of these techniques in Section 4.5.2.

4.5.1 Computational Lexicons

Due to the importance of computational lexicons for semantic analysis [173, 13], we further investigate how computational lexicons operationalize word senses and which semantic relations they cover. We use WordNet and BabelNet as illustrative examples in order to explain the concept of computational lexicons. On the one hand, WordNet is a large lexical database for the English language [293, 292, 291]. It organizes nouns, verbs, adjectives and adverbs into sets of synonym words, i.e., synsets, to express a distinct concept. The synsets are interlinked by means of semantic relations and span a network of meaningfully related words and concepts. In total, WordNet contains 155,287 words which are grouped into 117,659 synsets. WordNet also provides short definitions and usage examples for each synset which clarifies the intended word sense. It can be regarded as a extended thesaurus since it groups words based on their word sense and supports users to disambiguate closely related words in the network. On the other hand, BabelNet [302] is a multilingual encyclopedic knowledge source which combines all features of a dictionary and a semantic network. Accordingly BabelNet covers a broad range of lexicographic and encyclopedic concepts and connects these concepts with semantic relations similar to WordNet. In total, BabelNet contains more than 13 million concepts, called Babel synsets, which represent a given meaning and which contain all their synonyms. Concepts and their relations have been harvested from WordNet, Wikipedia, and most recently, from OmegaWiki, Wiktionary, and Wikidata and integrated into one unified knowledge source by applying intelligent mappings between these sources. Additionally, it contains translations obtained from sense-annotated sentences such that it incorporates more than 270 languages. Both lexicons offer a web interface to explore word senses and semantic relations as well as a Java API to integrate these sense extraction functionality and semantic analysis into software tools.

Obviously, the concept of synsets play an important role in both lexicons. A *synset* typically represents a particular word sense by grouping together those words, that express approximately the same meaning. As an example of the word sense representation with synsets, consider the synsets that are retrieved for the noun application. Depending on the lexicon, we may use application in either 7 or 10 different contexts as depicted in Table 4.7. The table also lists the members of the respective synsets together with a description when employing WordNet or BabelNet. For instance, we learn that we can use the word application in the sense of a job application (Synset 2) and in the sense of a software program (Synset 4). Besides these obvious senses, we also learn that the word application describes the action of bringing something to bear, i.e. to use something for a particular purpose (Synset 1). We also learn synonymous words which can be used interchangeably for the word application. For example, we can also use the word *lotion* if we refer to a liquid in a medical treatment (Synset 5). Besides, BabelNet adds three more synsets which have been discovered in the other knowledge sources, such as the sense of *determination to perform a task* (Synset 8) or of *an application for sole rights to an invention* (Synset 10).

Computational lexicons also provide several sense relations between synsets and words, such as synonymy, homonymy, hyponomy, and meronymy. The synset representation of word senses directly implements the synonym relation

Table 4.7: Synsets Retrieved from WordNet and BabelNet for the Noun Application

	Description	**WordNet**	**BabelNet**
1	the act of bringing something to bear	{application, practical application}	{application, practical application}
2	a verbal or written request for assistance or employment or admission to a school	{application}	{application}
3	the work of applying something	{application, coating, covering}	{application, coating, covering}
4	a program that gives a computer instructions that provide the user with tools to accomplish a task	{application, application program, applications programme}	{application, application program, application software }
5	liquid preparation having a soothing or antiseptic or medicinal action when applied to the skin	{lotion, application}	{application, lotion}
6	a diligent effort	{application, diligence}	{application, diligence}
7	the action of putting something into operation	{application}	{application}
8	persevering determination to perform a task	N/A	{diligence, industriousness, industry, assiduity, application (virtue)}
9	in United States law, a motion is a procedural device for decision	N/A	{Motion in United States law, application (legal)}
10	an application for sole rights to an invention	N/A	{patent application, application (patent)}

because it describes a particular word sense in terms of interchangeable words combined with a short description. Moreover, this representation also points to homonym words, as it lists all possible interpretations of a given word. However, in order to capture the other semantic relations, it is necessary to explicitly define these relations between the synsets. Thus, the synsets of a computational lexicon are also linked via lexical and semantic relations [293]. *Semantic relations* apply to all members of a synsets, while lexical relations are defined between single words of different synsets. As an example of semantic relations, we already introduced the hyponym and meronym relation for nouns and the troponym relation for verbs. The antonomy relation would be an example for a lexical relation between words. This relation expresses the opposite of concept of a word, such as the word *good* would be the opposite to the word *bad*.

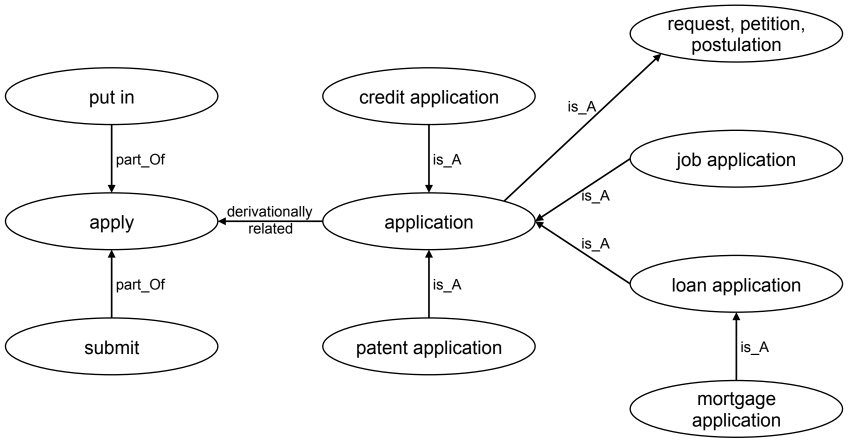


Fig. 4.8: Exerpt of the BabelNet Semantic Graph for Application

Both lexicons, WordNet and BabelNet, may be interpreted as a graph. Nodes of the graph refer to synsets and edges represent lexical and semantic relations between the synsets or words. Figure 4.8 shows a fragment of the graph for the word *application* and Synset 2. The figure also depicts several hyponym relations, two hypernym relations, one derivation relation as well as two meronym relations. The derivation relation is a lexical relation and is thus only defined for one word in the respective synset. In this case, the derivation relation expresses that the verb *to apply* verbalizes the noun application. The hyponym relations expresses that one synset is subsumed by another less specific synset. For example, the synsets *credit application*, *job application*, *loan application*, and *patent application* are all hyponyms of the synset *application*. The hypernym relation is the inverse relation to the hyponym relation. Accordingly, the synset *application* is a hypernym of the synset *loan application*, while *loan application* is a hypernym of *mortgage application*. Finally, the entailment relation describes that the verb *to apply* consists of several parts. In the figure, we identify that the verbs *to put in* and *to submit* are both entailed in the action of applying.

We define several functions to describe the concept of synsets as well as the required semantic relations in order to precisely refer to the specific concepts of a computational lexicon. These functions focus on the BabelNet lexicon. This choice is motivated by the fact that BabelNet significantly outperforms WordNet in terms of knowledge coverage and its fitness for word sense disambiguation [302]. We adapt definition 4.1 and express word senses with the help of synsets in order to refer to the synsets of the BabelNet dictionary. Accordingly, BabelNet synsets are defined as follows [300]:

Definition 4.6. (**BabelNet Synsets**). Let BN be the BabelNet dictionary, the senses of a word are defined by the function $Senses_{BN} : W \times POS \rightarrow 2^{Synsets}$, such that

- W is the set of words denoted in the BabelNet dictionary BN.
- $POS = \{n, v, a, r\}$ is the set of open-class parts of speech (nouns, verbs, adjectives, and adverbs).
- $Synset$ is the entire set of synsets encoded in the BabelNet dictionary BN. 2^{Synset} denotes the powerset of synsets.

As an example of this definition, consider Table 4.7 which shows the synsets for the word *application.* The function $Senses_{BN}$(application,n) would describe the set of synsets which is shown in the right column of the table. In order to refer to the i-th synset of the set of synsets, we will use an index $i \in N$, such that $s_i \in Senses_{BN}$(application,n). Accordingly, the fourth synset is refered to as $s_4 = \{$application, application program, application software$\}$.

Finally, we also specify functions to refer to the hyponyms and hypernnyms of a specific synset. Since the hyponym and hypernym relation are defined between synsets, it is not sufficient to provide only the word and its POS tag. Instead, this function requires a specific word sense which is given by a particular synset. Accordingly, the hyponym and hypernym relations can be defined as follows:

Definition 4.7. (**Hypomym and Hypernym Relation**). Let BN be the BabelNet dictionary. The hyponym and hypernym relations are defined by the functions $hypo_{BN} : Synset \rightarrow 2^{Synset}$ and $hyper_{BN} : Synset \rightarrow Synset$, such that

- $Synset$ denotes the entire set of synsets encoded in the BabelNet dictionary BN
- 2^{Synset} denotes the powerset of synsets.
- $hypo_{BN}$ retrieves the set of all hyponym synsets.
- $hyper_{BN}$ retrieves the set of all hypernym synsets.

As an example, consider again the graph fragment of BabelNets hyponymy tree for the word application as shown in Figure 4.8. It illustrates that BabelNet provides four different hyponym synsets for the synset $s_2 = \{application\} \in Senses_{BN}$(application,n)$\}$. Accordingly, these hyponyms are given by $hypo_{BN}(s_2) = \{\{$job application$\}, \{$patent application$\}, \{$credit application$\}, \{$loan application$\}\}$. Moreover, is also shows one hypernym synset which is given by $hyper_{BN}(s_2) = \{$request, petition, postulation$\}$.

So far, we paid considerable attention to represent semantic concepts and relations with technical means. However, we still miss a link between the semantic processing of text with these semantic relations. To this end, it is necessary to annotate words with their respective meaning in a given sentence. The task of assigning the most likely word sense to a word is known as *word sense disambiguation* (WSD) and is comparable to syntactic tagging and parsing. The next section is dedicated to a short discussion of word sense disambiguation.

4.5.2 Word Sense Disambiguation

In general, WSD describes the task of automatically assigning the appropriate word senses to all or some of the words of a given input text. If we consider a text as a sequence of words $(w_1, w_2, ..., w_n) \in (W \times W \times ... \times W)$, WSD needs to identify a mapping from a word w_i to its word senses, i.e.:

$$A : W \times (W \times W \times ... \times W) \mapsto Senses_D, \text{ such that}$$

- $A(w_i, (w_1, w_2, ..., w_n)) \subseteq Senses_D(w_i, p)$
- $A(w_i)$ is the subset of the word senses of w_i which are appropriate for the word sequence

The mapping can assign more than one word sense to each word $w_i \in T$, although typically only the most appropriate word sense is selected. As discussed earlier, there are a number of WSD approaches and systems to identify the correct word sense. In this thesis, we will employ the multilingual knowledge-based WSD approach of Navigli and Ponzetto [302, 303, 304] because it is integrated into the BabelNet lexicon and because it can compete with available systems by achieving a F1-score of 82.5%.

The BabelNet WSD approach aims for the construction of a directed graph $G = (V, E)$ for a target word w' and an input word sequence $(w_1, ..., w_n)$ representing the context. In this case, the node set contains all word senses of the input word sequence, while the edges are given by semantic relations. The graph is constructed by employing a depth-first-search for one node to another and adding all intermediate nodes if BabelNet specifies a path of semantic relations between these nodes [301]. The result of this procedure is a subgraph of BabelNet which contains all senses of the words in the input word sequence, as well as all edges and intermediate senses found in BabelNet. Given that representation, WSD is then regarded as a ranking problem [304]. Each node of the subgraph G representing a word sense is ranked by applying a connectivity measure given by the function $score(.)$. The most appropriate word sense for w' and the POS tag p is then given by

$$s_i^* = \underset{s \in Senses_{BN}(w',p)}{\operatorname{argmax}} \; score(s) \tag{4.1}$$

Experiments of different centrality concepts revealed that the *degree centrality* performs best for the WSD task [301, 341]. Degree centrality relies on the notion of vertex degree and ranks the senses of a given word in the subgraph G based on the number of their outgoing edges. The connectivity measure assumes that the most appropriate sense is always involved when comparing other senses of the graph and accordingly assigns a higher degree to it. This scoring method is described as follows:

$$score(s) = |\{(s, v) \in E \mid v \in V\}| \tag{4.2}$$

In this thesis, we will employ the BabelNet system to learn about word senses and the integrated graph-based disambiguation approach to assign the respective word sense to a target word. The relevant techniques for pragmatic analysis are discussed in the next section.

4.6 Relevant Techniques for Pragmatic Analysis

Finally, we discuss relevant techniques for the pragmatic analysis. With this regard, we already discussed techniques from the area of discourse interpretation, language generation, and machine translation.

Discourse interpretation is concerned with the analysis of with a group of collocated and related sentences, i.e. a discourse [184, p. 663-664]. For the automatic interpretation of such discourses, we mentioned several approaches that make use of the BDI model or other sources of knowledge for the discourse interpretation task. Unfortunately, the adoption of such techniques for process model labels is rather limited which mainly has two reasons. First, a process model does not represent a proper discourse of sentences leading to the fact that it cannot easily be analyzed with regard to its beliefs, desires, or intentions. The second reason is that these approaches provide only very little help for the aforementioned pragmatic issues where the context of a process model and its activities is insufficiently specified. They might be helpful to identify process models without a clear intention, but do not provide help in refactoring such issues.

Language Generation is the process of constructing natural language outputs from non-linguistic inputs. The goal of this process to map from meaning to understandable natural language text by applying the steps of selecting the relevant content and words, structuring a single sentence, and finally structuring the entire discourse [184, pp. 761-762]. Indeed, there are also approaches available that generate natural language texts from process models (see e.g. [241, 242]) and other conceptual models (see e.g. [93, 143, 295]). Although these approaches support users, which are not familiar with a specific modeling notation, to understand the process model, the approaches themselves do not directly help in refactoring pragmatic issues in process models where essential information in the process model is missing. It would be up to the user to read through the generated process description and improve it wit hregard to specificity. A similar argument applies to approaches of *machine translation* that are is concerned with the automatic translation of texts and process models from one language into another [29].

For these reasons, this thesis does not employ any of the aforementioned techniques belonging to the pragmatic NLP techniques and seeks alternatives to address the issue of pragmatic inconsistencies caused by insufficient context information in a process model.

4.7 Summary

This chapter has provided a general introduction into the field of linguistics, the different schools of thought, and the basic ideas. In order to analyze the textual information in process models elements, this chapter has also discussed the involved branches of theoretical linguistics, namely syntax, semantics, and pragmatics. We have also provided a short introduction to the field of natural language processing and available techniques to analyze the syntactical, semantic, and pragmatic aspects of natural language.

Moreover, this chapter has further presented the most important concepts of each linguistic branch which are required for the model analysis techniques of this thesis. From a syntactical perspective, this chapter has discussed available technical means to annotate process model elements with their components which enable a deeper investigation and serve as an input for semantic technology. Regarding semantics, this chapter has given an introduction to the most relevant semantic technologies, i.e. computational lexicons and the approaches of word sense disambiguation. Moreover, it has motivated the choice for the computational lexicon BabelNet and its integrated WSD algorithms. Equipped with these linguistic and technical concepts, this thesis will now present approaches that further analyze the model element labels and correct them, if desired.

5

Refactoring of Syntactical Ambiguity

This chapter discusses the problem of detecting and refactoring syntactical ambiguity in process models. Syntactical ambiguity refers to such cases where text labels encode other aspects of the process model, such as the control flow or resources. Section 5.1 further details the problem of complex label phrases and discusses the consequences thereof. It motivates the notion of atomicity as a new correctness criterion for process models, which is then operationalized in Section 5.2. Since a considerable number of industrial process models do not comply with the atomicity notion, it is necessary to refactor deficient process models in an automated way. For this purpose, Sections 5.3 describes nine atomicity violation patterns that have been discovered in process model repositories from industry. Based on these violation patterns, Section 5.4 presents an approach that automatically detects and refactors non-atomic process models. In Section 5.5, this approach is then evaluated by four process model collections from practice. Finally, Section 5.6 discusses the results and implications of the evaluation, before Section 5.7 summarizes the main achievements of this chapter.

5.1 Syntactical Ambiguity in Process Models

The text labels assign an element of a process model to a name and thus contribute to the overall semantics of a process model. These labels have in common that they are subject to linguistic variation in terms of structure and wording. There exist several ways to express the same semantics with different text labels. In order to unify the structure and to avoid misinterpretation of text labels, modeling guidelines recommend specific labeling styles for different process model elements [283, 395, 261], such as the verb-object style for activities. However, these guidelines only cover those labels that refer to one single stream of action. If, however, the text label describes complex phrases incorporating several streams of action or control flow information,

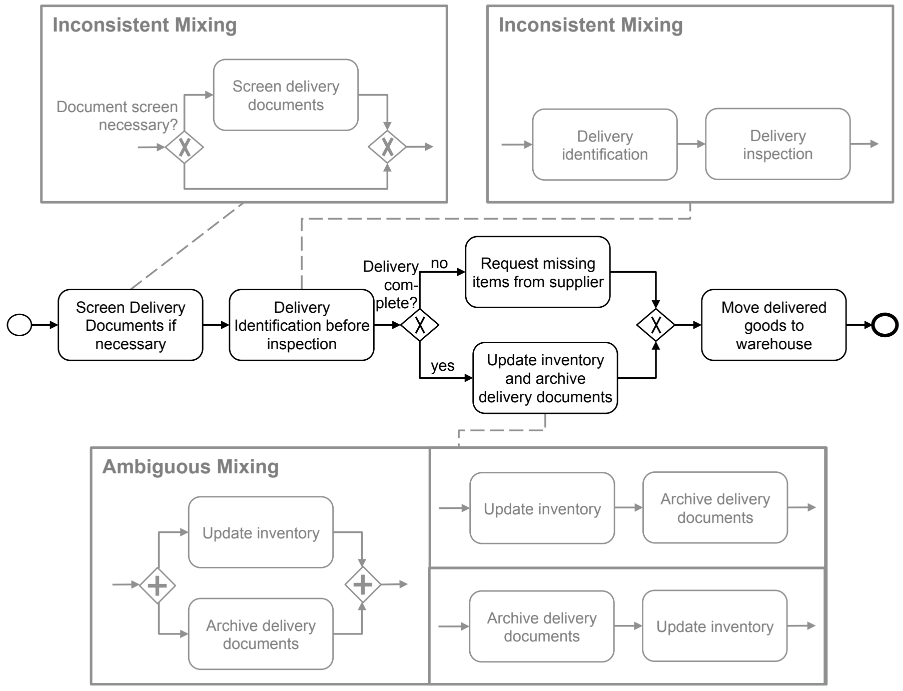

Inconsistent Mixing
Document screen necessary?
Screen delivery documents
Inconsistent Mixing
Delivery identification
Delivery inspection
Screen Delivery Documents if necessary
Delivery Identification before inspection
Delivery complete?
no
yes
Request missing items from supplier
Update inventory and archive delivery documents
Move delivered goods to warehouse
Ambiguous Mixing
Update inventory
Archive delivery documents
Update inventory
Archive delivery documents
Archive delivery documents
Update inventory

Fig. 5.1: Process Model with Mixing Natural Language and Modeling Language

these guidelines and the available correction techniques address only parts of the problem.

We use an exemplary BPMN process model as depicted in Figure 5.1 in order to elaborate on the problem of complex label phrases. It shows a process model of a company receiving new goods for its inventory. The process starts by screening the documents of the delivery. Subsequently, the delivery is identified within the company before the inspection of the delivery begins. Afterwards, it needs to be determined if the delivery is complete or not. Depending on the result, the missing items are either requested from the supplier or the inventory is updated and the delivery documents are archived. Finally, the delivered goods are moved to the warehouse which resembles the end of the process.

Figure 5.1 depicts a number of process model elements that go beyond the simple verb-object style. For example, the activity *Screen delivery documents if necessary* instructs the model quickly check the delivery documents. Implicitly, it further instructs the user to make a decision. Since the phrase *if necessary* adds optionality to the activity, the user has to decide depending on the situation, if he checks these documents or not. In consequence, it would be more consistent to explicate this optional activity by using an exclusive choice gateway with an empty path on the one hand and the respective activity on the other hand. The next example involves the activity *Delivery identification before inspection.* This activity describes a sequence of two activities with a particular order. Indeed, the delivery is first identified within the company and then inspected accurately. In this case, it would be more consistent to create two separate activities in a direct sequence. Although both cases represent an inconsistent mix of natural and modeling language, it has to be noted that the interpretation of these text labels is clear. Nevertheless, there are also less understandable cases. For example, consider the ambiguous label of the activity *Update inventory and archive delivery documents.* Although the activity tells us to update the inventory and to archive the delivery documents, it is unclear about the sequence of these activities, because the intention of the and-conjunction is not obvious. In the example, it may be interpreted as a parallel sequence of activities or in a direct sequence. These complex labeling phrases do not only affect the ability of a reader to properly understand the model and to develop a solid understanding of the underlying process, but also complicates the application of the process model for other purposes, such as model compliance, process model analysis, as well as system design and analysis.

Process model compliance ensures that business processes, operations and practice are in accordance with a prescribed agreed set of norms [376, 140]. Frequently, compliance requirements may stem from legislature and regulatory bodies (see e.g. Sarbanes-Oxeley Act [380] or Basel III [25]. In such a setting, process models are annotated with additional compliance conditions and assessed by inference techniques to verify the compliance requirements [376, 86]. However, if model elements are non-atomic and incorporate several activities, the annotation of a such an element with compliance conditions does not reveal to which activity it refers. In consequence, an inference on these conditions might evaluate the process as compliant although it violates specific compliance requirements. Ultimately, the process model cannot be used to prove the implementation of compliance requirements.

Process analysis and monitoring is concerned with a quantitative and continuous assessment of processes and evaluating them according to defined performance measures and objectives [108]. Alternatively, processes may also be simulated to detect bottlenecks or resources with high workload [420, 98]. In order to conduct such analyses, the process models need to be enriched by additional information, such as activity costs, resources, roles, or time, typically specified for each activity [98, 8]. Similar to the aforementioned case, the analysis approaches consider each activity to be atomic and do not detect non-atomic activities. In consequence, the annotated information is not broken down to the incorporated activities, but remains unchanged. For example, if a process is analyzed for its execution time, a non-atomic activity would mingle waiting time and processing time of its nested activities instead of correctly distributing these times on the nested activities. Thus, the analysis results and the key performance indicators are imprecise and do not reflect the current

Table 5.1: Refined Overview of Textual Model Refactoring Approaches and their Focus on Atomicity

Authors	**Approach**	**Atomicity**
Sharp and McDermott [395], Mendling et al. [283], Malone et al. [261]	Naming Conventions for Process Model Elements	None
Leopold et al. [248, 249]	Parsing and Refactoring of Activity Labels	Partly
Leopold et al. [238]	Detecting and Correcting Naming Violations	Partly
Becker et al. [31, 32]	Enforcing of Naming Conventions during Modeling	Partly
Delfmann et al. [96], Havel et al. [157]	Prototype for Naming Convention Enforcement	Partly

state within the process which may ultimately lead to wrong business decisions [144].

System design and analysis is concerned with the study of organizations and systems in order to identify their goals and purposes and to create innovative systems that will achieve new requirements in an efficient way [37]. In this context, process models help to document relevant parts of the system [88] and to design systems [109]. In particular, process models may be transformed into executable process-aware systems by using established standards such as BPMN [399] and BPEL [182]. Prior research has also proposed techniques to automatically generate executable BPEL code from process models in BPMN and UML notation [315, 317]. The weak point of these approaches is that they consider each model activity to refer to a single system function that needs to be implemented. Thus, the translation approaches are not capable of detecting nested functions and of correctly mapping them to executable code. Therefore, the final system might be incomplete with regard to the required functionality.

Apparently, the implications of complex labeling phrases in process models are potentially severe. It is beneficial to verify a process model with regard to this property in order to avoid the aforementioned issues. Accordingly, we investigate whether the previously discussed approaches are capable to deal with complex label phrases and to resolve them. Table 5.1 refines the overview of existing syntax verification approaches. Moreover, the table also investigates approaches from other domains and how they implement the notion of atomicity.

We already discussed the naming guidelines for process model elements which recommend the verb-object style of labeling [283, 395, 261]. However, the guidelines do not provide any restriction on the quantity of verbs or actions of a single text label and are not fully applicable to prevent this issue. Leopold et al. [248, 249, 238] propose a refactoring technique that reworks the syntactical structure of English and German labels. Their approach marks complex element labels as a guideline violation, but does not rework these labels comprehensively. The tools by Becker et al. [31, 32], Delfmann et al. [96], and Havel et al. [157] provide modeling support for the naming of process model elements. These tools enforce specific linguistic naming conventions during the process of modeling and ensures that particular naming guidelines are fulfilled. In this way, the tool indirectly keeps model element labels simple and prevents the specification of complex labels. Overall, these approaches do not explicitly consider the problem of complex labels and provide only building blocks to tackle it. A comprehensive solution to that problem has not been developed so far. Therefore, this thesis proposes a theoretical concept, denoted as *atomicity*, which imposes an additional condition on process model elements. It demands that process model elements should not incorporate multiple actions and business objects at once. Hence, they clearly refer to one distinct stream of action without specifying additional information of other perspectives of the process, such as the control flow or the data and resources perspective. In this way, atomicity defines a correctness criterion for process models that, if fulfilled, provides the basis for their meaningful analysis.

In order to establish the atomicity concept in process models, we need to address the following requirements:

1. RQ1: Operationalization of the atomicity concept
2. RQ2: Automatic detection of non-atomic instances
3. RQ3: Automatic resolution of non-atomic instances

The first requirement involves the completeness of cases that correctly and incorrectly apply the notion of atomicity in process models. Generally, the first requirement has to be approached in an inductive way by generalizing reoccurring patterns. Based on these patterns, we can provide an operationalization of atomicity and of the problematic cases. The other two requirements are concerned with the adoption of the atomicity concept into modeling practice. They involve a deductive step with which deficient process models are detected and refactored. The second requirement particularly addresses the necessity of detecting non-atomic instances in a reliable way which requires a formal specification of the basic characteristics of the non-atomicity patterns. The third requirement is concerned with the automatic rework of detected non-atomicity instances and makes use of the general structure of violation patterns.

5.2 Operationalizing the Concept of Atomicity

As aforementioned, the notion of *atomicity* represents a correctness criterion on process models and its element labels. It demands that all process model elements should not incorporate several actions and business objects at once. This condition ensures that the elements clearly describe one specific stream of action that is not impacted by any additional information relating to other process perspectives. Moreover, atomicity ensures that both building blocks, i.e. the modeling language and the natural language, are distinctly separated from each other, such that process model analysis techniques can focus on the respective block.

We refer to the definition of a process model as given by Definition 2.3 in order to formulate the notion of atomicity. This definition allows the assignment of arbitrary labels to process models elements which may involve several actions, business objects, or control flow restrictions. As shown in Fig. 5.1, the activity *Update inventory and archive delivery documents* comprises two actions (to update, to archive) as well as two business objects (inventory, delivery documents) resulting in two individual activities combined with either a direct or a parallel sequence. According to the discussion from the previous section, the text label should be free of such information. Relying on the Definition 4.3 that introduced the different label components of a process model parser, the notion of *atomicity* is defined as follows:

Definition 5.1. (**Atomicity**). A process model $\mathrm{P} = (A, E, G, F, R, P, L, \rho, \pi, \lambda, \gamma, \tau)$ is *atomic* if each labeled activity has exactly one action, one business object, and no more than one addition. Formally:

$$\forall a \in A^{\mathrm{P}}_{\lambda} : |\alpha(\lambda_A(a))| = 1 \wedge |\beta_A(\lambda(a))| = 1 \wedge |\gamma_A(\lambda(a))| \leq 1.$$

Applying this definition to the process model in Fig. 5.1 reveals that the model does not comply with this the notion of atomicity. One of the reasons is the activity label *Update inventory and archive delivery documents* which comprises several business objects, i.e. $|\beta(l)| = |\{$*inventory*, *delivery documents*$\}| \neq 1$, and several actions $|\alpha(l)| = |\{$*to update*, *to archive*$\}|$. A similar argument applies to the activity label *Delivery identification before inspection*, which comprises two actions, i.e. $|\alpha(l)| = |\{$*to identify*, *to inspect*$\}| \neq 1$. In consequence, both violations mark the example process model to be non-atomic and to violate the notion of atomicity.

Indeed, the number of atomicity violations are worth considerable. In order to quantify the extent of non-atomic process models, Definition 5.1 has been checked for several process model collections from various industries. Table 5.2 gives an overview of affected process models that have at least one non-atomic element as well as the average number of non-atomic elements per model. The selected process model collections are rather heterogeneous with regard to the different characteristics and show notable differences with regard to size, standardization, the expected degree of modeling quality, and the modeling domain:

- **SAP Reference Model**: The *SAP Reference Model* (SRM) contains 604 Event-Driven Process Chains organized in 29 different functional branches [190]. Examples are procurement, sales, and financial accounting. The model collection includes 2,432 activity labels. Since the SAP Reference

Table 5.2: Atomicity Violations in Process Model Repositories from Industry

Characteristic	**SRM**	**IMC**	**AIC**	**TC**
No. of Models	604	349	1,091	979
No. of Labels	2,432	1,840	8,339	12,088
Modeling Language	EPC	EPC	BPMN	EPC
Domain	Independent	Insurance	Academic Training	Communication
Standardization	High	Medium	Low	Medium
Total No. of Violations	160 (26%)	232 (67%)	444 (41%)	445 (45%)
Avg. No. of Non-Atomic Elements	2.44	2.59	2.31	3.98

Legend: SRM: SAP Reference Model Collection, IMC: Insurance Model Collection, AIC: Academic Initiative Collection, TC: Telecommunication Model Collection

Model was designed as an industry-independent recommendation, we expect a high degree of standardization and a high model quality which should result in smaller numbers of non-atomic pattern occurrences.

- **Insurance Model Collection**: The Insurance Model Collection (IMC) contains 349 EPCs dealing with the claims handling activities of a large insurance company. In total, the models include 1,840 activities and hence are slightly smaller than the models from the SRM. Compared to SRM, we expect a bigger number of pattern occurrences due to the low level of competence of casual modelers [369, 419] and higher error rates in industry model collections [274].
- **Academic Initiative Collection**: The models from the Academic Initiative Collection (AIC) stem from academic training and cover diverse domains[1]. From the available models, we filtered those with proper English labels. The resulting subset includes 1,091 process models with 8,339 activity labels. Since the collection targets no specific industry and has been mainly created by students, the number of pattern occurrences is expected to be the highest among all considered collections.
- **TC Collection**: The TC collection contains the processes from an international telecommunication company. It comprises 803 process models with 12,088 activities in total. Thus, the TC collection is the biggest model collection which we employ for our experiments. Similar to the IMC collection, the TC collection was also created by casual and semi-professional modelers which leads to the assumption that the techniques will detect pattern occurrences between the SRM and the AIC collection.

Apparently, the issue of non-atomic process models affects a notable number of process models in the model repositories. It has been revealed that between 26% and 67% of process models specify elements containing non-atomic element

[1] Please refer to: http://bpmai.org

labels. Within the affected models, we identified on average 2.31 to 3.98 model elements that violate the atomicity criterion. These numbers also reflect standardization and experience of the modelers who created the respective models. In the SRM collection, we have found the smallest number of non-atomic process models. Surprisingly, the AIC collection having the smallest degree of standardization has less problems with atomicity than the process models stemming from the insurance and communication domain.

It is, however, necessary to increase our current understanding of non-atomic cases in order to leverage the automatic refactoring of the affected process models. Therefore, we discuss our explorative methodology as well as specific non-atomicity patterns in the next section.

5.3 Atomicity Violation Patterns

In order to increase the current understanding of non-atomicity patterns, we adopt the explorative approach by Weber et al. [431] to identify refactoring opportunities in process models. The manual analysis of industry process models includes the following steps to derive a list of generic patterns that violate the notion of atomicity.

1. **Pattern Extraction**: The first step consists of two iterations. In the first iteration, each non-atomic instance has been explored for interesting phrases and annotated with an initial code. This procedure has been continued until a point of saturation has been reached [73], which describes the situation when no new and interesting construct was emerging. The first iteration ended by creating an initial list of violation patterns. In the second iteration, the initial set of violation patterns has been further refined. These patterns have been characterized according to their disjunctive properties. As a result, we aggregated those patterns which turned out to be special cases of others. Afterwards, the entire set of patterns has been annotated again with the refined pattern set.
2. **Pattern Interpretation**: In the second step, we further investigated the refined set of violation patterns and analyzed how the atomicity notion may be restored. We thus considered each violation pattern and interpreted it with respect to the natural language and the modeling language it implies. The second step resulted in a set of violation patterns with their possible interpretation as well as their core structure.
3. **Pattern Classification**: In the last step, we further investigated these patterns and classified them according to the process model perspective they refer. We classified each pattern as either affecting the *control flow* or *data and resources* as well as describing *implicit objects*. All three classes describe patterns which mix natural language and modeling language in an inconsistent way, but still have only one possible interpretation. Moreover, there are also patterns that are ambiguous in such a way that two or more interpretations are possible.

As a result of this process, a set of nine atomicity violation patterns has been identified. We discuss the identified patterns and represent their structure with the help of syntactical concepts in order to leverage the forthcoming automatic refactoring of these patterns. We distinguish between patterns that concern the control flow (Section 5.3.1), the resources and data perspective (Section 5.3.2), and implicit model elements (Section 5.3.3).

5.3.1 Non-Atomic Control Flow Patterns

In total, we found five non-atomicity patterns that relate to the control flow of process models. Table 5.3 provides an overview of these patterns, their syntactical structure, and their interpretation.

Violation Pattern 1. Activities suffering from the *Sequence* pattern incorporate sequential text information. This information imposes additional conditions on the task by stating that the task has to be executed in preparation for or as a consequence of another task. The pattern directly relates to the basic sequential flow of activities [7, 317, 437]. Typically, this pattern links two distinct activities by using temporal prepositions, such as `before` or `after`, to express the order of activities. The atomicity of these labels is violated due to the inclusion of several actions, business objects, or additional information fragments. Hence, we describe the set of labels of a process model P that fall into the Sequence pattern as follows:

$$\begin{aligned} V^{\mathrm{P}}_{Sequence} =& \{a \in A^{\mathrm{P}}_{\lambda} : l = \lambda(a) \wedge m(l, \texttt{*(before|after)*}) \\ & \wedge (|\alpha_A(l)| \neq 1 \vee |\beta_A(l)| \neq 1 \vee |\gamma_A(l)| > 1)\}, \end{aligned} \tag{5.1}$$

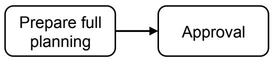
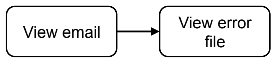
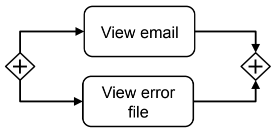
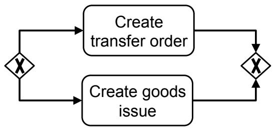
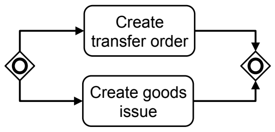
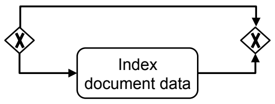
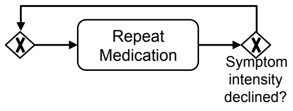
where $m(l,\texttt{regex})$ denotes a logical predicate that evaluates to true if the label matches the regular expression `regex`.

Violation Pattern 2. Elements suffering from the *Parallel Activities* pattern combine several actions, business objects or combinations of these in a single activity element. Hence, a single activity element instructs people to perform multiple individual streams of action. Typically, this pattern includes the conjunction `and` or special characters such as `+`, `&`, or `/` to indicate several distinct activities. Moreover, the interpretation of this pattern is not obvious since the meaning of the conjunctions is not clearly defined. Consequently, the label may refer to a sequence of activities as well as to a parallel execution of multiple activities, which is another basic control flow pattern [7, 317, 437]. The atomicity condition is violated since these labels include several actions and business objects. Accordingly, we formalize the set of labels in a process model P that violate the Parallel Activities pattern as follows:

$$\begin{aligned} V^{\mathrm{P}}_{Parallel} =& \{a \in A^{\mathrm{P}}_{\lambda} : l = \lambda(a) \wedge m(l, \texttt{*(and|+|\&|/)*})) \\ & \wedge (|\alpha_A(l)| \neq 1 \vee |\beta_A(l)| \neq 1 \vee |\gamma_A(l)| > 1)\}, \end{aligned} \tag{5.2}$$

where $m(l,\texttt{regex})$ denotes a logical predicate that evaluates to true if the label matches the regular expression `regex`.

Table 5.3: Overview and Interpretation of Control Flow Patterns

	Syntax Structure	Example	Interpretation
Sequence	A_1 O_1 `before` A_2	Prepare full planning before approval	
	A_1 O_1 `after` A_2	Note requirement to collect excess after lodgement	
Parallel	A_1 O_1 `and` A_2 O_2	Start quick scan and clarify low level requirements	
	A_1 O_1 `and` O_2	View email and error file	
	A_1 `and` A_2 O_1	Create and submit quote	
Decision	A_1 O_1 `or` A_2 O_2	Enter pin or choose random pin option	
	A_1 O_1 `or` O_2	Create transfer order or goods issue	
	A_1 `or` A_2 O_1	Cancel or vary fulfillment	
Skip	A O `as (required|necessary)`	Update claim exposure estimate as required	
	A O `if (required|necessary)`	Index document data if necessary	
Iteration	A O `until` C	Repeat medication until symptom intensity declines	
	A O `per` O	Check SLA per client	
	A O `for each` O	Notify updated invoice for each order	

Legend: A: Action, O: Business Object, C: Condition or Status of a Business Object

Violation Pattern 3. The *Decision* pattern implies a control flow split and may lead to several exclusive or inclusive paths in a process model. Similarly to the previous pattern, this pattern is using multiple actions, business objects, or combinations of these and thus violates atomicity. Typically, this pattern occurs when two alternatives are linked by the conjunction `or`. Alternatively, the special character `/` may also point to this pattern. The interpretation of the decision pattern is also ambiguous. On the one hand, it might refer to a regular exclusive decision as presented in [7, 317, 437]. On the other hand, the interpretation as an *inclusive or* gateway, which gives the possibility of choosing among multiple alternatives [7, 76], is also valid. We formalize the set of labels that violate the Decision pattern as follows:

$$\begin{aligned} V^{\mathrm{P}}_{Decision} =& \{a \in A^{\mathrm{P}}_{\lambda} : l = \lambda(a) \wedge m(l, \texttt{*(or|/)*}) \\ & \wedge (|\alpha_A(l)| \neq 1 \vee |\beta_A(l)| \neq 1 \vee |\gamma_A(l)| > 1)\}, \end{aligned} \tag{5.3}$$

where $m(l,\texttt{regex})$ denotes a logical predicate that evaluates to true if the label matches the regular expression `regex`.

Violation Pattern 4. In general, the *Skip* pattern implies a decision about executing an optional activity. This activity is only conducted under specific conditions. If the conditions are not met, the activity is skipped and the process continues without executing this activity. Typically, this pattern combines prepositions, such as `if` or `as`, together with the past participle of the verb `to require` or adjectives that express optionality, such as `necessary` or `applicable`. The interpretation of the Skip pattern corresponds to a consistent solution that is similar to the switch pattern for conditional routing [317] in which the respective activity is executed or not. In many cases, element labels of this pattern correctly specify a single action, a single business object, and an optional addition. Hence, this pattern is regarded as an exception and does not violate atomicity. However, the optional character of this pattern still has significant impact on the control flow and encodes relevant information in the element label. We formalize the set of labels that violate the Skip pattern as follows:

$$\begin{aligned} V^{\mathrm{P}}_{Skip} =& \{a \in A^{\mathrm{P}}_{\lambda} : l = \lambda(a) \\ & \wedge m(l, (\texttt{*(if|as) (necessary|required)}))\}, \end{aligned} \tag{5.4}$$

where $m(l,\texttt{regex})$ denotes a logical predicate that evaluates to true if the label matches the regular expression `regex`.

Violation Pattern 5. The *Iteration* pattern is arranged in such a way that the natural language fragment asks for an iteration or a loop construct. In most of the cases, the iteration is expressed by the language pattern `repeat ... until ...` or a statement such as `per` or `for each`. In these cases, the label also contains the iteration condition. The interpretation of the Iteration pattern resembles the arbitrary cycle pattern [7, 437] or, more specifically, the while and the repeat pattern [317]. Similar to the Skip pattern, the atomicity condition is not violated in general because most of the element labels correctly specify action, business object and an addition. Accordingly, we formalize the set of labels that match the Iteration pattern as follows:

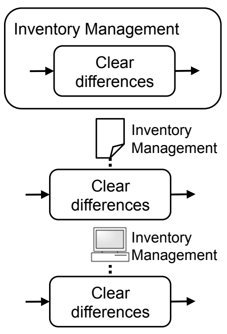
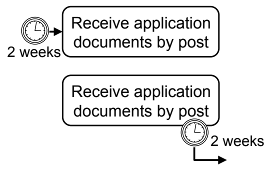
$$\begin{aligned} V^{\mathrm{P}}_{Iteration} =& \{a \in A^{\mathrm{P}}_{\lambda} : l = \lambda(a) \\ & \wedge m(l, (\texttt{repeat * until *|* per *}))\}, \end{aligned} \tag{5.5}$$

where $m(l,\texttt{regex})$ denotes a logical predicate that evaluates to true if the label matches the regular expression `regex`.

5.3.2 Non-Atomic Resource and Data Patterns

Table 5.4 provides an overview of patterns that hide resource and data information in text labels. Furthermore, we provide a core structure and interpretation of the patterns.

Table 5.4: Overview and Interpretation of Resource and Data Patterns

	Core Structure	Example	Interpretation
Extra Information	$A\ O_1\ (O_2)$	Clear differences (Inventory Management)	 Inventory Management Clear differences Inventory Management Clear differences Inventory Management Clear differences
	$A\ O_1\ [O_2]$	Updating [Investment Projects]	
	$O_2\ :\ A\ O_1$	Sales quotes: Budgeting	
	O_2 - $A\ O_1$	Rental Unit - Assign Application	
Time Information	$A\ O\ (CD$ (min|h|day|week))	Receive application documents by post (2 weeks)	 2 weeks Receive application documents by post Receive application documents by post 2 weeks

Legend: A: Action, O: Business Object, CD: Cardinal Number

Violation Pattern 6. The *Extra Information* pattern refers to activities that ambiguously incorporate additional information fragments into the element label. The element label violates the atomicity condition because of having more than one addition. The additional information themselves may include a specification of business objects, the refinement of entire activities into subprocesses, or the specification of process resources. The most prominent examples make use of brackets or dashes to indicate additional specifications. The interpretation of such labels is unclear and strongly dependent on the context. On the one hand, it may refer to multiple activities in form of a subprocess that are specified elsewhere [105]. On the other hand, it may refer to resources that are used by the activity, such as data inputs or processing systems [374, 373]. Accordingly, we formalize labels that fall into the Extra Information pattern as follows:

$$\begin{aligned} V^{\mathrm{P}}_{Extra} = \{ & a \in A^{\mathrm{P}}_{\lambda} : l = \lambda(a) \wedge m(l, \texttt{(*(*)|*[*]|*:*|*-*)}) \\ & \wedge (|\alpha_A(l)| \neq 1 \vee |\beta_A(l)| \neq 1 \vee |\gamma_A(l)| > 1)\}, \end{aligned} \tag{5.6}$$

where $m(l,\texttt{regex})$ denotes a logical predicate that evaluates to true if the label matches the regular expression `regex`.

Violation Pattern 7. The *Time Exception* pattern is similar to the latter one. It, however, incorporates temporal information by including temporal prepositions that clarify the duration (e.g. in minutes, hours, or days) or other time-related constraints of an activity. Typically, the time information is provided in brackets and, in many cases, unclear. The temporal information may represent waiting time, i.e., time that must pass before the process continues normally [228], or the temporal information could be interpreted in the sense of an attached intermediate event. The latter implies that the execution of the activity is canceled as soon as the time limit is reached triggering a new stream of actions that is not specified [372]. Altogether, labels belonging to the Time Exception pattern are described as follows:

$$\begin{aligned} V^{\mathrm{P}}_{Time} = \{ & a \in A^{\mathrm{P}}_{\lambda} : l = \lambda(a) \wedge m(l, \texttt{*((0-9)(min|h|day|week))}) \\ & \wedge (|\alpha_A(l)| \neq 1 \vee |\beta_A(l)| \neq 1 \vee |\gamma_A(l)| > 1)\}, \end{aligned} \tag{5.7}$$

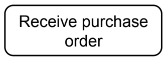
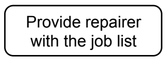
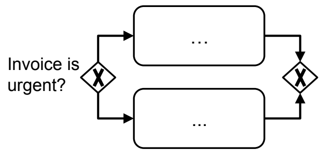
where $m(l,\texttt{regex})$ denotes a logical predicate that evaluates to true if the label matches the regular expression `regex`.

5.3.3 Implicit Element Patterns

Finally, we discuss patterns that implicitly refer to decisions and actions and hide these constructs from the process model. Table 5.5 shows the syntactical pattern structure and the consistent interpretations of these patterns.

Violation Pattern 8. Process model elements suffering from the *Implicit Action* pattern erroneously combine labeling style and modeling construct, i.e. activity, event or gateway. As an example, consider an activity that is described as an event or vice versa. As shown in the examples of the table, the activity *Purchase order received* might refer to the respective event or the activity *Receive purchase order*. Moreover, they also violate the atomicity criterion because they specify a particular state or condition of an object rather than an action that needs to be applied on the object. As far as the interpretation is concerned, we assume that the modeling construct takes precedence over the text label. Following this assumption, the text label would imply an activity in the process. Hence, we interpret these cases as a regular activity with a correct style of labeling [283]. Accordingly, we define the set of all Implicit Action labels as follows:

Table 5.5: Overview and Interpretation of Implicit Object Patterns

	Core Structure	Example	Interpretation
Implicit Action	O A	Purchase order received	
	O A ?	Repairer provided with the job list?	
Implicit Decision	`determine if` O C	Determine if personal message is required	
	`check if` O C	Check if invoice is urgent	

Legend: A: Action, O: Business Object, C: Condition or Status of a Business Object

$$\begin{aligned} V^{\mathrm{P}}_{IAction} =&\{a \in A^{\mathrm{P}}_{\lambda} : l = \lambda(a) \wedge \\ &((|\alpha_E(l)| \geq 1 \vee |\sigma_E(l)| \geq 1) \wedge |\beta_E(l)| \geq 1) \vee \\ &(|\alpha_G(l)| \geq 1 \vee |\sigma_G(l)| \geq 1) \wedge |\beta_G(l)| \geq 1))\}, \end{aligned} \tag{5.8}$$

where $m(l,$`regex`$)$ denotes a logical predicate that evaluates to true if the label matches the regular expression `regex`.

Violation Pattern 9. The *Implicit Decision* pattern entails a decision in the process flow. It includes a specific condition that has to be checked in order to proceed. Many instances of this pattern contain a verb asking for the verification or investigation of the conditions and the conditional word `if`. Regular language patterns include `determine if`, `validate if`, `check if`, and `confirm if`. However, the respective activities only contain an action, but further need a business object. Therefore, these activities are not atomic. The interpretation is equal to the basic construct of an exclusive choice [7, 317, 437]. All labels that fall into the Implicit Decision pattern are formalized as follows:

$$\begin{aligned} V^{\mathrm{P}}_{IDecision} =&\{a \in A^{\mathrm{P}}_{\lambda} : l = \lambda(a) \\ &\wedge m(l, \texttt{(check|determine|validate|confirm) if*}), \end{aligned} \tag{5.9}$$

where $m(l,$`regex`$)$ denotes a logical predicate that evaluates to true if the label matches the regular expression `regex`.

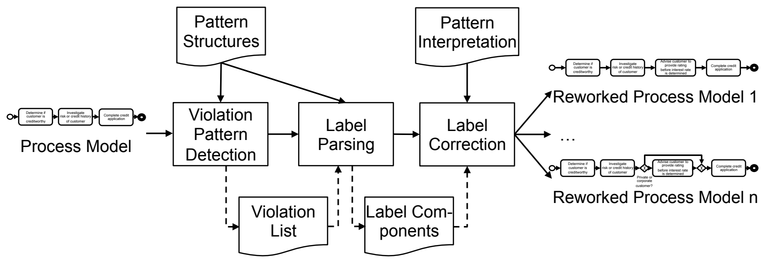


Fig. 5.2: Overview of Violation Pattern Refactoring

5.4 Refactoring of Atomicity Violation Patterns

For those process models that do not comply with the notion of atomicity, we introduce a technique to automatically refactor these instances. Figure 5.2 depicts the necessary refactoring steps. The technique takes the deficient process model as an input. First, the violation pattern detection step investigates the labels of the process model elements and tries to identify violations of non-atomicity. For this purpose, it uses the violation pattern structures which have been introduced in the previous section. The detected violations are then added to a list which is also input for the next step, the label parsing. This step also uses the pattern structures and extracts the necessary label components from the deficient labels. As a result, this step provides all label components of the deficient labels, such as actions, business objects and specific conditions. Finally, the *label correction* step creates a new model fragment according to the pattern interpretations. Since these fragments are not specified by a label yet, it also uses the extracted label components and assigns these components to the respective model fragment and its elements. Finally, the deficient element is replaced by the newly created fragment. Since some violation patterns allow several interpretations, the label corrector outputs a list of possible solutions among which the user may select.

The following sections explain these three steps in more detail. The violation detection component is introduced in Section 5.4.1. Afterwards, the sections 5.4.2 and 5.4.3 explain the label parsing component as well as the label correction component.

5.4.1 Detection of Non-Atomicity Patterns

In general, the detection technique finds non-atomic element labels by verifying the pattern formalizations of Section 5.3. For that purpose, it also makes use of the extracted components from the model parsing which has been introduced in Section 4.4.3. The model parser of serves as a preprocessing and retrieves the actions and business objects of the element labels, which are necessary the verify Definition 5.1. As a result, the detection technique provides a list that

Algorithm 1: Pattern Detection from Process Model Element Label

```
 1: detectViolationPatterns(Label $l$)
 2: if $l$.matches(*(before|after)*) $\wedge$ isNotAtomic($l$) then
 3:     $l$.violatesSequencePattern(true)
 4: if $l$.matches((and|+|&|/)) $\wedge$ isNotAtomic($l$) then
 5:     $l$.violatesMultiplePattern(true)
 6: if $l$.matches(*(or|/)*) $\wedge$ isNotAtomic($l$) then
 7:     $l$.violatesDecisionPattern(true)
 8: if $l$.matches(*(if|as) (necessary|required)) then
 9:     $l$.violatesSkipPattern(true)
10: if $l$.matches((repeat * until *|* per *)) then
11:     $l$.violatesIterationPattern(true)
12: if $l$.matches((*(*)|*[*]|*:*)) $\wedge$ isNotAtomic($l$) then
13:     $l$.violatesExtraInformationPattern(true)
14: if $l$.matches(*((0-9)(min|h|day|week))) $\wedge$ isNotAtomic($l$) then
15:     $l$.violatesTimePattern(true)
16: if isEventLabel($l$) $\vee$ isGatewayLabel($l$) then
17:     $l$.violatesImplicitActionPattern(true)
18: if $l$.matches((check|determine) if*) then
19:     $l$.violatesImplicitDecisionPattern(true)
20: return $l$
```

contains all non-atomic labels of the respective process model including the patterns that are violated by the labels.

From a technical perspective, the pattern detection approach analyzes whether the input label matches a particular non-atomicity pattern and marks the detected pattern in case of a positive match. For this purpose, the approach uses the formalization presented in the previous sections. The detection itself works in two steps. The first step evaluates if the activity label contains specific keywords that point towards a certain pattern. In a second step, the approach checks whether the activity label violates the criterion of atomicity by extracting actions, business objects as well as additions and counting the appearances of the respective component. In particular, this step is required for the Violation Patterns 1, 2, 3, 6 and 7. In case of positive evaluations, the label is marked with the respective pattern. Algorithm 1 summarizes the pattern detection for each pattern.

5.4.2 Label Parsing

We first need to parse the deficient element label in order to restructure the detected violation pattern. The goal of the label parser is the extraction of relevant information from the deficient model element. Depending on the respective pattern, we need to remove specific keywords and extract relevant text fragments, such as conditions and resources. Moreover, we also retrieve actions and business objects by using the presented model parser from Section

Algorithm 2: Component Extraction from Process Model Element Label

```
1: parseNonatomicLabel(Label $l$)
2: LabelComponents $lc$ = new LabelComponents()
3: List $AtomicLabels \leftarrow \emptyset$
4: List $conditions \leftarrow \emptyset$
5: List $extraInfo \leftarrow \emptyset$
6: if $l$.matchesSequencePattern() $\vee$ $l$.matchesMultiplePattern() $\vee$
   $l$.matchesDecisionPattern() then
7:     $AtomicLabels \cup l$.splitByPhrase()
8: if $l$.matchesSkipPattern() then
9:     $AtomicLabels \cup l$.removePhrase()
10: if $l$.matchesIterationPattern() then
11:     $AtomicLabels \cup l$.splitByPhrase()[1]
12:     $conditions \cup l$.splitByPhrase()[2]
13: if $l$.matchesExtraPattern()$\vee$ $l$.matchesTimePattern() then
14:     $AtomicLabels \cup l$.removeExtraInfos()
15:     $extraInfo \cup l$.extractInfoByPhrase()
16: if $l$.matchesImplicitActionPattern() then
17:     $l \leftarrow l$.removePhrase()
18:     $\alpha(l) \leftarrow$ getInfinitiveOfAction($\alpha_E(l)$)
19: if $l$.matchesImplicitDecisionPattern() then
20:     $conditions \cup l$.removePhrase()
21: $lc \leftarrow AtomicLabels \cup condition \cup extraInfo$
22: return $lc$
```

4.4.3. Reconsider the activity label *Update inventory and archive delivery documents* from Figure 5.1. Based on the detection technique, we recognize the parallel activities pattern, because the activity label uses the conjunction `and` and entails two different actions and business objects indicating two different streams of action. Accordingly, the label parsing technique removes the `and`-conjunction and extracts the actions *to update* and *to archive* as well as the business objects *inventory* and *delivery documents*.

This procedure is formalized by Algorithm 2. The algorithm takes a label l as an input, which was marked with the respective pattern violations. In the beginning, we initialize a label component data structure lc to store the extracted components (Step 2) as well as three lists for the atomic activity labels, the conditions and the extra information that may be specified in a non-atomic label (Steps 3-5). In the following steps, we extract the relevant parts depending on the detected pattern. If the sequence pattern, the multiple activity pattern, or the decision pattern have been detected, we split the label using the respective keyword and add the parts to the atomic label list (Steps 6–7). If the label matches the skip pattern, the parsing algorithm solely removes the text fragment that indicates the optionality of the label and adds the atomic activity to the list (Steps 8–9). In case of the iteration pattern, the algorithm splits the label by the keywords which results in two parts. According to the pattern, the first part specifies the activity and is stored in the activity list. The second part, which relates to the iteration condition, is stored in the condition list (Steps 10–12). Step 13 processes labels that violate the extra information and the time exception pattern. On the one hand, the algorithm removes the additional information and stores the atomic activity in the activity list (Step 14). On the other hand, it also stores the extra information in a separate list (Step 15). If the label matches the implicit decision pattern, the algorithm begins with removing specific keywords (Step 17), before it assigns the missing action the label by transforming the state or condition of a business object into an action (Step 18). Finally, the algorithm parses implicit decision labels by removing the keywords and adding the remaining part to the set of conditions (Steps 19–20). The algorithm terminates by adding the activity, the condition, and the extra information list to the set of the label component (Step 21) and returning the label component (Step 22).

5.4.3 Label Correction

The second step involves the correction of the deficient model elements. In general, the label corrector has to remove the deficient element, insert a blank model fragment based on the possible interpretations of the patterns, and instantiate all fragments with proper element names. The element names have been extracted in the previous step and also serve as an input to this technique. Hence, the label correction takes the label component data structure of the label parsing as an input and uses these components to assign labels to model fragments. In the following, we describe the necessary transformation steps for each pattern separately.

In case of the sequence pattern, the deficient activity label needs to be replaced by two distinct activities. We further need to specify a directed sequence flow between the two added activities to establish a strong order relation between these two activities. Finally, the activities are assigned by the respective label from the atomic activity list of the label component. Each of these steps is described in the following definition.

Definition 5.2. (**Sequence Pattern Correction**). Let P be a process model as defined in Def. 2.3 and $a \in V^{\mathrm{P}}_{Sequence}$ be an activity of the sequence non-atomicity pattern. Further, let $x = \bullet a$ and $y = a \bullet$ denote the predecessor and successor of a, and a^{in} and a^{out} the incoming and outgoing sequence flows. Finally, let lc the parsed label component. The activity a is corrected with the following steps:

- $A \setminus \{a\} \cup \{a_1, a_2\}$
- $F \setminus (a^{in} \cup a^{out}) \cup \{(x, a_1), (a_1, a_2), (a_2, y)\}$
- $\lambda(a_1) = lc.AL[1]$, $\lambda(a_2) = lc.AL[2]$

The correction of the multiple activity and the decision pattern are very similar. In both cases, the correction involves the removal of the deficient activity and the insertion of a splitting gateway, a joining gateway and a number of blank activities depending on the number of atomic labels that are stored in the atomic activity list. Afterwards, control flow edges are added to implement a parallel or interleaving order of activities. Finally, each blank activity is assigned a label from the list of atomic activities. Please note, that parallel gateways are used in case of the multiple activity pattern and inclusive or exclusive gateways in case of the decision pattern. These steps are again described by the following definitions:

Definition 5.3. (**Parallel Activities Pattern Correction**). Let P be a process model as defined in Def. 2.3 and $a \in V^{\mathrm{P}}_{Parallel}$ be an activity of the parallel activities pattern. Further, let $x = \bullet a$ and $y = a\bullet$ denote the predecessor and successor of a, and a^{in} and a^{out} the incoming and outgoing sequence flows. Finally, let lc the parsed label component. Accordingly, the activity a may be corrected by the following alternatives:

1. Correction with parallel activities
   - $A \setminus \{a\} \cup \{a_1, ..., a_n\}$
   - $G \cup \{g^S_{and}, g^J_{and}\}$
   - $F \setminus (a^{in} \cup a^{out}) \cup \{(x, g^S_{and}), (g^S_{and}, a_1), ..., (g^S_{and}, a_n), (a_1, g^J_{and}), ..., (a_n, g^J_{and}), (g^J_{and}, y)\}$
   - $\lambda(a_1) = lc.AL[1]$, ..., $\lambda(a_n) = lc.AL[n]$
2. Correction with a sequence of activities
   - $A \setminus \{a\} \cup \{a_1, ..., a_n\}$
   - $F \setminus (a^{in} \cup a^{out}) \cup \{(x, a_1), (a_1, a_2), ..., (a_{n-1}, a_n), (a_n, y)\}$
   - $\lambda(a_1) = lc.AL[1]$, ..., $\lambda(a_n) = lc.AL[n]$

Definition 5.4. (**Decision Pattern Correction**). Let P be a process model as defined in Def. 2.3 and $a \in V^{\mathrm{P}}_{Decision}$ be an activity of the decision non-atomicity pattern. Further, let $x = \bullet a$ and $y = a\bullet$ denote the predecessor and successor of a, and a^{in} and a^{out} the incoming and outgoing sequence flows. Finally, let lc the parsed label component. Depending on the interpretation, the activity a is corrected by the following alternatives:

1. Refactoring with exclusive decision
   - $A \setminus \{a\} \cup \{a_1, ..., a_n\}$
   - $G \cup \{g^S_{xor}, g^J_{xor}\}$
   - $F \setminus (a^{in} \cup a^{out}) \cup \{(x, g^S_{xor}), (g^S_{xor}, a_1), ..., (g^S_{xor}, a_n), (a_1, g^J_{xor}), ..., (a_n, g^J_{xor}), (g^J_{xor}, y)\}$
   - $\lambda(a_1) = lc.AL[1]$, ..., $\lambda(a_n) = lc.AL[n]$
2. Refactoring with inclusive decision
   - $A \setminus \{a\} \cup \{a_1, ..., a_n\}$
   - $G \cup \{g^S_{or}, g^J_{or}\}$
   - $F \setminus (a^{in} \cup a^{out}) \cup \{(x, g^S_{or}), (g^S_{or}, a_1), ..., (g^S_{or}, a_n), (a_1, g^J_{or}), ..., (a_n, g^J_{or}), (g^J_{or}, y)\}$
   - $\lambda(a_1) = lc.AL[1]$, ..., $\lambda(a_n) = lc.AL[n]$

The correction of the skip pattern mainly involves the same steps as the decision pattern. Accordingly, the defective activity is removed from the process model and a model fragment is added that specifies an exclusive decision. In contrast to the decision pattern, one path of this decision is empty in order to skip the activity if necessary. The following definition summarizes the necessary steps:

Definition 5.5. (**Skip Pattern Correction**). Let P be a process model as defined in Def. 2.3 and $a \in V^{\mathrm{P}}_{Skip}$ be an activity of the skip non-atomicity pattern. Further, let $x = \bullet a$ and $y = a\bullet$ denote the predecessor and successor of a, and a^{in} and a^{out} the incoming and outgoing sequence flows. Finally, let lc the parsed label component. The activity a is corrected by the following steps:

- $G \cup \{g^S_{xor}, g^J_{xor}\}$
- $F \setminus (a^{in} \cup a^{out}) \cup \{(x, g^S_{xor}), (g^S_{xor}, a), (a, g^J_{xor}), (g^S_{xor}, g^J_{xor}), (g^J_{xor}, y)\}$
- $\lambda(a) = lc.AL[1]$

The correction of the iteration patterns needs to replace the deficient activity with an arbitrary loop construct. Consequently, we insert two gateways together with a blank activity to the process model. In contrast to the previously discussed patterns, we need to alter the positions of the exclusive join and the exclusive split gateway to implement the loop. Afterwards, we can assign the activity label and the iteration condition to the respective constructs which concludes all necessary refactoring steps.

Definition 5.6. (**Iteration Pattern Correction**). Let P be a process model as defined in Def. 2.3 and $a \in V^{\mathrm{P}}_{Iteration}$ be an activity of the iteration non-atomicity pattern. Further, let $x = \bullet a$ and $y = a\bullet$ denote the predecessor and successor of a, and a^{in} and a^{out} the incoming and outgoing sequence flows. Finally, let lc the parsed label component. The activity a is corrected by the following steps:

- $G \cup \{g^S_{xor}, g^J_{xor}\}$
- $F \setminus (a^{in} \cup a^{out}) \cup \{(x, g^J_{xor}), (g^J_{xor}, a), (a, g^S_{xor}), (g^S_{xor}, g^J_{xor}), (g^S_{xor}, y)\}$
- $\lambda(a) = lc.AL[1]$
- $\lambda(g^S_{xor}) = lc.conditions[1]$

The correction of the extra information pattern is not straightforward. We cannot propose a comprehensive correction of this pattern, because the additional information may refer to different aspects of a process model, such as resources, roles or even sub-process activities. In general, we may replace the original label with an atomic one and set the control flow edges accordingly. For the extra information fragment, we require the decision of a business analyst to correctly specify this information with available modeling syntax.

Definition 5.7. (**Extra Information Pattern Correction**). Let P be a process model as defined in Def. 2.3 and $a \in V^{\mathrm{P}}_{Extra}$ be an activity of the extra information non-atomicity pattern. Further, let $x = \bullet a$ and $y = a \bullet$ denote the predecessor and successor of a, and a^{in} and a^{out} the incoming and outgoing sequence flows. Finally, let lc the parsed label component. The activity a is corrected by the following steps:

1. Correction with subprocess activity a^S
   - $A \setminus \{a\} \cup \{a^S\}$
   - $F \setminus (a^{in} \cup a^{out}) \cup \{(x, a^S), (a^S, y)\}$
   - $\lambda(a^S) = lc.extraInfo[1]$
2. Correction with resources
   - $R \cup r$
   - $\rho(a) = r$
   - $\lambda(a) = lc.AL[1]$
   - $\lambda(r) = lc.extraInfo[1]$

The correction of the time information pattern requires the insertion of a new activity and a new event. The activity replaces the deficient one and is labeled as an atomic activity. The new event precedes the activity and is labeled by the respective time information of the original label. These steps are formally described by the following definition:

Definition 5.8. (**Time Information Pattern Correction**). Let P be a process model as defined in Def. 2.3and $a \in V^{\mathrm{P}}_{Time}$ be an activity of the time information non-atomicity pattern. Further, let $x = \bullet a$ and $y = a \bullet$ denote the predecessor and successor of a, and a^{in} and a^{out} the incoming and outgoing sequence flows. Finally, let lc the parsed label component. The activity a is corrected by the following steps:

- $E \cup \{e'\}$
- $F \setminus (a^{in} \cup a^{out}) \cup \{(x, e'), (e', a), (a, y)\}$
- $\lambda(a) = lc.AL[1]$
- $\lambda(e') = lc.extraInfo[1]$

The correction of an implicit action pattern is the most simple one as it only requires the transformation of the condition or the status into an imperative action. This means that we only need to reassign the activity with the correctly parsed label without inserting a new activity.

Definition 5.9. (**Implicit Action Pattern Correction**). Let P be a process model as defined in Def. 2.3and $a \in V^{\mathrm{P}}_{IAction}$ be an activity of the implicit action non-atomicity pattern. Further, let lc be the parsed label component. The activity a is corrected by the following steps:

- $\lambda(a) = lc.AL[1]$

The correction of the implicit decision pattern is again similar to the decision pattern. We also need to remove the deficient activity and replace it with an exclusive choice model fragment. However, as the activities falling into that pattern do not specify the number or the name of subsequent activities, we cannot fill the blank activities with meaningful labels and may only assign a gateway label to the split gateway at the beginning of the fragment.

Definition 5.10. (**Implicit Decision Pattern Correction**). Let P be a process model as defined in Def. 2.3 and $a \in V^{\mathrm{P}}_{IDecision}$ be an activity of the implicit decision non-atomicity pattern. Further, let $x = \bullet a$ and $y = a\bullet$ denote the predecessor and successor of a, and a^{in} and a^{out} the incoming and outgoing sequence flows. Finally, let lc the parsed label component. The activity a is corrected by the following steps:

- $A \setminus \{a\} \cup \{a_1, ..., a_n\}$
- $G \cup \{g^S_{xor}, g^J_{xor}\}$
- $F \setminus (a^{in} \cup a^{out}) \cup \{(x, g^S_{xor}), (g^S_{xor}, a_1), ..., (g^S_{xor}, a_n), (a_1, g^J_{xor}), ..., (a_n, g^J_{xor}), (g^J_{xor}, y)\}$
- $\lambda(g^S_{xor}) = lc.conditions[1]$

5.5 Evaluation

In this section, the detection and correction techniques are evaluated by industrial process models in order to assess the performance of the presented techniques. The evaluation is structured as follows. Section 5.5.1 describes the test data for the evaluation experiments, while Section 5.5.2 discusses the employed evaluation strategy and setup. Sections 5.5.3 and 5.5.4 present the experimental results of the detection and the correction techniques.

5.5.1 Test Data

Considering experiments to be an integral part of the evaluation, the choice of the test data sets may highly influence the outcome of the techniques and the conclusions drawn from it. In general, the proposed techniques should be applicable to a large number of real-world process models. Moreover, a broader scope allows us to draw generalizable conclusions from the results and to give valid performance guarantees for other sets of process models. We require process models from different input sources and with different characteristics because we aim for a high external validity. As shown in Table 5.2, the selected process model collections are rather heterogeneous with regard to the different characteristics. The four process model collections show notable differences with regard to size, standardization, the expected degree of modeling quality, and the modeling domain. For these reasons, the test collections are well suited for the evaluation of both techniques.

5.5.2 Evaluation Goal and Setup

The overall evaluation goal is the performance assessment of the presented techniques with the help of defined metrics. The presented techniques aim for the implementation of atomicity in process models from practice and thus for the detection and refactoring of non-atomic instances in process models. For that purpose, the underlying strategy is outlined in terms of human benchmark, performance metrics, latent factors, and implementation.

The *human benchmark* represents the point of reference by which the algorithmic results are evaluated and the performance of the techniques is measured. For that purpose, building this benchmark involved two researchers inspecting the activities of process models and classifying them according to the patterns. The researchers identified non-atomic labels and marked any pattern violation independently from each other. Contradicting or inconsistent cases have been resolved in a subsequent step. Afterwards, both researchers resolved the detected instances by using the possible interpretations of the non-atomic activities.

The human benchmark facilitates the comparison of human and algorithmic results which can be quantified by suitable *performance metrics*. In particular, the comparison of human and algorithmic results classifies each label as either true-positive, true-negative, false-positive, or false-negative. These items provide the basis to calculate the metrics precision and recall [21]. In our context, the precision value is the number of correctly detected pattern instances divided by the total number of pattern instances retrieved by the algorithms. The recall is the number of correctly detected pattern instances divided by the total number of manually retrieved pattern instances. As it is important that each metric yields sufficiently high values, we also take the f-measure, i.e. the harmonic mean of precision and recall, into account. The performance of the refactoring is measured by the metric refactoring precision. The metric reflects the number of correctly reworked cases divided by the total number of reworked cases.

We then define acceptable ranges for these metrics in order to compare the performance with existing approaches. As an acceptable range, we follow the performance results of available techniques from activity label parsing and refactoring. In [238], the evaluation of the violation detection technique revealed an average recall and precision of 95% while the refactoring precision amounted to approximately 75%. Using these numbers as a lower bound, we hypothesize that the proposed techniques should perform at least as good as the related techniques. Accordingly, we formulate the following hypothesis for the detection algorithm of a pattern i:

- H_a^i: The average precision $\mu(P^i)$ and the average recall $\mu(R^i)$ of detecting the pattern i amount to at least 0.95.
- H_0^i: The average precision $\mu(P^i)$ and the average recall $\mu(R^i)$ of detecting the pattern i does not amount to at least 0.95.

In case of the refactoring algorithm, we formulate the following hypothesis:

- H_a^i: The average refactoring precision $\mu(RP^i)$ amounts to at least 0.75.
- H_0^i: The average refactoring precision $\mu(RP^i)$ does not amount to at least 0.75.

The performance metrics are also influenced by *latent factors* that are inherent to the data set. In particular, pattern variations and pattern combinations influence these metrics. Pattern variations refer to the linguistic variety to name activities and activity patterns. For example, the label *Determine if customer is creditworthy* may also be formulated as *Check if* or *Validate if* which would equally resemble an implicit decision. However, specific pattern variations may not be detected by the techniques because the formalization relies on general pattern characteristics. Accordingly, these variations influence the numbers of precision and recall. Pattern combinations refer to the incorporation of more than one non-atomicity pattern. As an example, consider the activity *Investigate risk and credit history if necessary.* Obviously, this activity violates the decision and the skip non-atomicity pattern. While this does not impose a problem for the detection of patterns, the refactoring might be affected since the order of refactoring steps may propose different refactoring solutions which is also reflected by the performance metrics.

With regard to the *implementation*, the detection and refactoring techniques have been implemented in a research prototype using Java 1.7 and existing language and model parsers, such as the Stanford Parser and the model annotation techniques of Leopold et al. [249] as well as the Stanford Tagger and Parser (see http://nlp.stanford.edu/software/) to detect and rework the deficient activities. For running the evaluation, each technique has been deployed on a MacBook Pro with a 2.4 GHz Intel Core Duo processor and 4 GB RAM, running on Mac OS X 10.7.5 and Java Virtual Machine 1.7.

5.5.3 Evaluation of Detection

The numbers of precision, recall, and f-measure are shown in Table 5.6. In general, these numbers show that the pattern detection technique works satisfactory. In most of the patterns, the f-measure amounts to more than 85% which ensures a reliable detection of relevant deficient model elements. However, we can also observe notable deviations among the process model collections of our test set and within the patterns.

Regarding the deviations among the *process model collections*, we observe that modeling experience and standardization is also reflected by the results. In SRM, the algorithm did not find any instances of particular patterns, such as the sequence pattern (P1), the skip pattern (P4), or the implicit decision pattern (P9). This observation confirms our initial assumption of SRM as being a reference model collection for several industries and thus requiring a consistent and correct modeling style. Opposite to that, a considerable number of pattern instances has been detected for the TC and the AIC collection. While

Table 5.6: Results of Violation Pattern Detection

		P1	P2	P3	P4	P5	P6	P7	P8	P9
SRM	Precision	-	0.94	0.98	-	-	1	-	0.2	-
	Recall	-	0.97	0.94	-	-	0.99	-	1	-
	F-measure	-	0.95	0.96	-	-	0.99	-	0.33	-
IMC	Precision	1	0.99	0.96	1	-	0.94	-	0.65	0.98
	Recall	1	1	1	1	-	0.89	-	0.75	0.98
	F-measure	1	0.99	0.98	1	-	0.91	-	0.70	0.98
AIC	Precision	1	0.96	0.93	1	1	0.99	1	0.92	0.98
	Recall	0.80	1	1	1	0.71	0.98	0.42	0.72	0.86
	F-measure	0.89	0.98	0.96	1	0.83	0.98	0.59	0.81	0.92
TC	Precision	1	0.98	0.99	1	1	0.97	-	0.74	1
	Recall	1	0.95	1	0.89	0.95	0.56	-	0.73	0.98
	F-measure	1	0.96	0.99	0.94	0.97	0.71	-	0.73	0.99
Avg.	Precision	1	0.97	0.97	1	1	0.98	1	0.63	0.99
	Recall	0.79	0.98	0.99	0.89	0.83	0.86	0.42	0.80	0.95
	F-measure	0.87	0.97	0.97	0.98	0.90	0.90	0.59	0.64	0.97

Legend: SRM: SAP Reference Model Collection, IMC: Insurance Model Collection, AIC: Academic Initiative Collection, TC: Telecommunication Model Collection

we expected a notable number of pattern occurrences of the AIC collection, i.e. a collection of process models from academic training, we have been surprised that most of the patterns are also present in the TC collection. Moreover, the result further confirms the validity and generalizability of the patterns to other process model collections.

Within the *patterns*, the performance of the technique is stable and does not deviate from the average score of the respective metric. Most notably, we observe a significant deviation of performance for the implicit action pattern (P8) in the SRM collection and the extra information pattern (P6) in TC. In the former case, the precision only amounts to 0.2 which results in a large number of false positives. A deeper investigation revealed that these labels incorporate complex business objects combined with status information. For instance, consider the labels *Parked Document Posting* or *Earned value calculation.* The detection algorithm suffers from an interpretation ambiguity of the first word. Despite of recognizing it as being part of the real business object (parked document), the algorithm interprets the word as a status of another business object (document posting) which does not resemble the original intention (post a parked document). In the latter case, the technique suffers from a low recall of 0.56 only. Again, we further investigated the reasons and found that many of the model elements in the TC collection miss a signal phrase to indicate this pattern. For example, the label *PLC 231 Create detailed configuration* *system* incorporates the resource statement *PLC 231*. However, this statement is not separated by a dash or a colon which does not result in a violation of the respective pattern. Despite these deviations, we still consider the algorithm to be suitable to detect the patterns in process models.

5.5.4 Evaluation of Refactoring

In order to evaluate the pattern refactoring technique, we consider only the result of the parsing technique, i.e. the extracted components from the target label. This is sufficient, because the refactoring technique uses a generic structure depending on the extracted label components and violation patterns and instantiates it with the extracted label components. Hence, we determine whether or not the defective label has been parsed properly and whether it would label the respective modeling fragment correctly. As a baseline, we consider all elements that violate a particular pattern as the set of all labels that should be refactored. Within this set, a label is either appropriately or erroneously corrected. Therefore, we define the metric *refactoring precision* as the share of appropriately corrected elements compared to all elements. Let L_c be the set of all labels that have been parsed correctly and L_e the set of labels that have been parsed with errors. Then, the refactoring precision (RP) is given as follows:

$$RP = \frac{|L_c|}{|L_c| + |L_e|}$$

Table 5.7 summarizes the result of appropriately and erroneously refactored labels as well as the corresponding refactoring precision. The results show that the refactoring technique is working satisfactory (RP score of 0.8 on average) and it is fairly stable among the different patterns. However, we also observe notable differences depending on the pattern type. If we consider the parallel activity (P2) and the decision pattern (P3), the RP score is stable around 0.7 to 0.75, yet rather small in comparison to other patterns. An analysis of erroneous refactorings revealed the *referential ambiguity* as the main reason. For instance, consider the element label *Create and submit quote and photos using fax or Email*. In this case, the relation between actions and business objects is more complicated because the two actions (create and submit) might either refer to one or two business objects (quote and photos). Ultimately, this might result in three or four distinct activities. Another example involves the refactoring of the implicit action pattern (P8) with an RP score of only around 0.50. A deeper investigation revealed that a corrected label has not been created because the business object was not detected (e.g. see the label *rcs software update embedded*) or the state could not be detected and reworked into an appropriate action (e.g. see the label *created detailed implementation plan*). Additionally, these effects are worsened if a label violates several patterns at the same time, as for instance in the label *implemented offer change or removal*, which makes

Table 5.7: Pattern Refactoring Results

		P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**
	L_c	-	102	47	-	-	168	-	2	-
SRM	L_e	-	30	19	-	-	22	-	3	-
	RP	-	0.77	0.71	-	-	0.88	-	0.40	-
	L_c	5	297	101	44	-	60	-	17	156
IMC	L_e	0	94	52	4	-	5	-	14	3
	RP	1	0.76	0.66	0.92	-	0.92	-	0.55	0.98
	L_c	7	327	58	1	23	119	31	215	42
AIC	L_e	3	110	30	0	9	13	0	107	3
	RP	0.7	0.75	0.66	1	0.72	0.90	1	0.67	0.93
	L_c	5	1002	283	2	15	60	-	120	64
TC	L_e	0	293	117	2	2	31	-	95	1
	RP	1	0.77	0.71	0.5	0.88	0.66	-	0.56	0.98
	L_c	5.33	306.25	15.50	9.00	16.50	81.25	15.50	77.75	85.67
Avg	L_e	1.00	76.25	5.75	0.00	4.00	14.00	0.00	46.50	2.33
	RP	0.89	0.81	0.81	1.00	0.82	0.85	1.00	0.57	0.97

Legend: SRM: SAP Reference Model Collection, IMC: Insurance Model Collection, AIC: Academic Initiative Collection, TC: Telecommunication Model Collection

the refactoring of these patterns quite challenging. Despite these deviations, we regard the performance of the refactoring technique as satisfactory given the various possibilities of natural language to combine several patterns in one element label.

5.6 Discussion

This section discusses results and implications of the techniques and their evaluation. Section 5.6.1 summarizes the results of the evaluation with regard to the given hypotheses. Section 5.6.2 reflects upon threats to validity and further limitations of the conceptual approach. Sections 5.6.3 and 5.6.4 identify implications of the proposed approach for research and for practice.

5.6.1 Summary of Results

The experimental evaluation of the presented algorithms provides support for a number of hypothesized performance expectations. Table 5.8 shows an overview of the hypotheses results with regard to the detection and refactoring algorithm. Turning to the detection hypothesis, the evaluation with the metrics precision and recall reveal support and partial support for four patterns in each

Table 5.8: Summary of Evaluation Results for Pattern Detection and Refactoring

Detection Hypothesis		**Result**	**Refactoring Hypothesis**	**Result**
$\mu(P^1) = 1$	$\mu(R^1) = 0.79$	partly supported	$\mu(RP^1) = 0.89$	supported
$\mu(P^2) = 0.97$	$\mu(R^2) = 0.98$	supported	$\mu(RP^2) = 0.81$	supported
$\mu(P^3) = 0.97$	$\mu(R^3) = 0.99$	supported	$\mu(RP^3) = 0.81$	supported
$\mu(P^4) = 1$	$\mu(R^4) = 0.96$	supported	$\mu(RP^4) = 1$	supported
$\mu(P^5) = 1$	$\mu(R^5) = 0.83$	partly supported	$\mu(RP^5) = 0.82$	supported
$\mu(P^6) = 0.98$	$\mu(R^6) = 0.86$	partly supported	$\mu(RP^6) = 0.85$	supported
$\mu(P^7) = 1$	$\mu(R^7) = 0.42$	partly supported	$\mu(RP^7) = 1$	supported
$\mu(P^8) = 0.63$	$\mu(R^8) = 0.80$	not supported	$\mu(RP^8) = 0.57$	not supported
$\mu(P^9) = 0.99$	$\mu(R^9) = 0.95$	supported	$\mu(RP^9) = 0.97$	supported

case. In particular, the experimental results suggest that the implementation achieves its goal of finding a relevant share of activities suffering from the parallel activities pattern (P2), the decision pattern (P3), the skip pattern (P4), and the implicit decision pattern (P9). In case of the sequence pattern (P1), the iteration pattern (P5), the extra information pattern (P6), and time exception pattern (P7), the implementation performs excellent in finding relevant pattern occurrences. However, we also learn that a large share of these occurrences is not retrieved due to the linguistic variations across the text collections. Therefore, the experiments provide only partial support for the detection hypothesis due to the smaller recall numbers. Finally, we need to reject the hypothesis in case of the implicit action pattern (P8) as both precision and recall do not overcome the critical value of 95%. The results of the refactoring technique suggest that eight out of nine hypotheses should be confirmed. These findings emphasize the capabilities of the refactoring technique to replace the deficient label with an atomic alternative requiring hardly any manual rework. However, due to the low performance in case of the implicit activity pattern (P8), we need to reject the corresponding hypothesis.

5.6.2 Limitations

The results of the evaluation also have to be discussed with regard to different threats to validity and other limitations. In particular, we discuss those threats that impact the generalizability of the techniques (external validity), the completeness of the non-atomicity patterns (internal validity), and the validity of the techniques themselves (construct validity).

External Validity. Threats to external validity are conditions that limit the generalization of the techniques to other cases [441, p. 110]. In our setting, external validity is mostly affected by the selection of test data, the selection of the target language, and the modeling purpose. In terms of *selecting the test* *data*, the four process model collections can hardly be seen as representative in a statistical sense. Therefore, we cannot completely rule out the fact that other model collections would yield different results or falsify the evaluation hypotheses. For the evaluation, we tried to mitigate this risk by selecting process model collections that vary along different dimensions such as degree of standardization, domain, and size and that thus reflect the complexity of process model collections from industry. Hence, it is valid to claim that the successful application of the proposed techniques is not limited to a particular set of models and that it will detect and refactor a large share of non-atomic process model elements. Moreover, this argument shares some perspectives with considerations of theoretical sampling and saturation as it is discussed for qualitative inductive studies [73]. This research methodology emphasizes the collection of deviant cases with different sizes and characteristics such that a broad spectrum of phenomena may be revealed.

Moreover, the presented techniques have been applied to process models with English labels only. The *selection of English as a target language* raises the question of adapting the techniques to different target languages. In general, the detection technique may be easily adapted to different target languages by adjusting key phrases of the respective pattern. Moreover, we may use localized parsers for process models [238] or natural language text [199, 200, 409]. For the refactoring technique, we require further adaptions with regard to label parsing and the restructuring of text labels. The parsing of model elements is facilitated by using insights on labeling styles of the target language [237]. In contrast to that, the refactoring step does not require further adaptions since it builds only the model fragment and uses the labels from the parsing component. Hence, we do not consider the adaption to different languages as a threat to external validity because the adaptions do not require much effort.

The external validity might also be affected by the *purpose* of the respective process model. As discussed before, one main characteristic of models is pragmatism which demands that the model is as a substitution of the original for a certain time and for a certain purpose. As a consequence, it might not be desirable to resolve the non-atomic instances and to introduce additional complexity because the purpose of the model is to provide a higher level of abstraction [363] or to give a quick overview of the most important steps [402]. Admittedly, enforcing the notion of atomicity is bound to the purpose of the process model and not suitable for applications that require a general overview of the process. Nevertheless, there are other purposes for which process models are employed, such as compliance to regulation rules, process model analysis, as well as system design and analysis. In these settings, the purpose of the process model is different and shifted from a general to a fine-grained perspective. For this perspective, it is, however, desirable that the process model correctly reflects the underlying business process and does not hide relevant resource or control flow information in element labels. In such a setting, the notion of atomicity comes into play and ensures that all relevant information is detected, explicated and available for further analyses based on the process model.

Internal Validity. In terms of internal validity, the *completeness of non-atomicity patterns* might also be affected because only three process model collections have been employed to derive the respective violation patterns [330]. However, the inclusion of several other process model collections might expand the set of the non-atomicity patterns. We tried to mitigate this risk by conducting a theoretical sampling methodology and selecting process model collections that vary along different characteristics. Furthermore, we also investigated an independent repository, i.e. the TC collection, and checked if it reveals further violation patterns. However, no new non-atomicity pattern emerged during the investigation, which indicates a point of saturation with regard to the completeness of the patterns [73]. Moreover, the human benchmark against which the techniques have been compared might also be affected by *selection of raters* [441, p. 107]. Depending on the motivation of the raters, the human benchmark might contain cases which have been erroneously marked to be non-atomic or which have been overlooked. In consequence, the metrics precision and recall might not reflect the true performance of the technique. We tried to mitigate this threat by involving several raters in the creation of the human benchmark and by integrating the results of both. Moreover, contentious cases have been discussed and resolved in an interactive way.
Construct Validity. Construct validity is affected when the construct of the study is not correctly reflected or implemented [441, p. 108]. In this case, the construct validity needs to be questioned if the formalization of the violation patterns does not cover their occurrence in the process models and thus impede their detection and resolution. Indeed, this is the case if activities linguistically vary the way the pattern is expressed or if they combine several of the aforementioned patterns. Consequently, deficient activities are not detected correctly and cannot be refactored by our techniques. The formalization also covers several specific cases to capture linguistic variance in the patterns in order to minimize this threat. This is also emphasized by the evaluation metrics precision and recall which have been used to distinguish between instances that are relevant and retrieved by the techniques and those that are either not relevant or not retrieved. Consequently, the greater the value of these metrics, the more accurate the respective techniques work. Since the experiments in the evaluation showed considerably big numbers for precision and recall, we conclude that the techniques do what they are supposed to and refactor the problematic instances in process models.

5.6.3 Implications for Research

As a major implication for research, the notion of label atomicity has been introduced for process model elements. Atomicity enables a unique specification of model elements such that each model element refers to only one individual aspect. This concept has notable impacts on process and workflow model analysis techniques since it unfolds hidden aspects of process models and thus addresses one challenge of semantic process modeling [278]. For example, atomicity explicates hidden control flow or data objects within process models which are then analyzable with existing approaches focusing on particular building blocks of process models (see e.g. [26, 6, 413]). Moreover, it contributes to the reliable applicability of process models in different scenarios, such as the design of workflow systems requiring a flawless specification of the underlying process [27] or the quantitative analysis of processes [108].

In addition, the notion of atomicity and its implementation also stimulate an extension of existing process modeling and naming guidelines, as presented by Mendling et al. [283], Sharp and McDermott [395], or Malone et al. [261]. In their current state, the naming guidelines demand elements to specify particular components thereof, such as an object and a verb in case of an activity. However, the number of objects and verbs is not restricted and hence allows the description of non-atomic elements. The results of this work suggest to incorporate the notion of atomicity in existing guidelines. In particular, the operationalization of atomicity is highly beneficial for several process model analysis techniques and for applications that rely on syntactically and semantically sound process models, such as the generation of process descriptions [242], automatic execution [121], or process model merging [245]. The notion of atomicity can assure that each activity contains one piece of information which is not subject to additional conditions and that the analysis of process models is efficient and accurate.

Finally, the notion of atomicity unfolds the semantics of the text label to structural elements of the process models. Thereby, it leverages a realistic understanding of the underlying process and leverages the application of process models in several application scenarios, such as *process model compliance*, *process analysis and monitoring*, and *system design and analysis*. With regard to compliance, the refactoring of non-atomic elements enables the precise annotation with specific compliance conditions, which can be assessed by available compliance checking techniques [376, 86]. A similar argument applies to the analysis and monitoring of process models. Due to the unfolding of labels to the underlying structure, it is possible to precisely annotate each activity with additional information, such as operational costs or execution time. In consequence, the analysis has a better chance to resemble the business activities and to take the right measures. Also various approaches to system design and analysis benefit from the notion of atomicity. In this case, process models are used to generate executable BPEL processes [315, 317]. The refactoring technique ensures that all relevant model activities are correctly represented by modeling constructs such that the translation approaches can correctly map into BPEL syntax.

5.6.4 Implications for Practice

The notion of atomicity also has important implications for practice. First, the explication of the non-atomicity patterns can be used to extend the capabilities of process modeling tools. In such a context, tools can be used to look for patterns during the modeling process and notify the modelers that they are probably specifying a decision or an arbitrary cycle. At the same time, the modeling tool suggests a resolved fragment containing all the information that have been specified so far. Hence, modeling errors could be prevented right from the beginning, which would save cost and time-intensive rework in later stages [46, 9, 17].

Second, the patterns can also be used with regard to organization-specific modeling conventions and quality controls. In general, modeling conventions aim to maintain a consistent modeling practice within an organization in terms of used modeling elements and naming of these elements [390, 35, 283, 31]. In this setting, the patterns may extend existing conventions with frequently used model fragments to facilitate the description of complex process behavior. Additionally, the violation patterns provide counterexamples, which can be avoided by using the respective model fragment.

5.7 Summary

This chapter has addressed the problem of detecting and refactoring syntactical issues in process models. These issues typically arise when natural language is used to describe information that relates to other aspects of the process model, such as the control flow or resources. In consequence, this information is not available for further analysis leading to unreliable results. To this end, the notion of atomicity has been introduced, which demands that each model element clearly refers to one single stream of action. Recognizing the number of affected process models from industry, we have identified reoccurring patterns that violate the notion of atomicity. Afterwards, we have developed techniques for the automatic detection and refactoring of process models that contain non-atomic elements. The experimental evaluation has confirmed that the proposed techniques are capable of detecting and refactoring the discussed patterns within industrial process model collections. Therefore, both techniques have provided a reliable baseline for further analysis of process models by clearly separating modeling language and natural language from each other.

6

Refactoring of Semantic Ambiguity

This chapter discusses the problem of managing semantic ambiguity in process models. Semantic ambiguity refers to the meaning of text labels and can be caused by several words that point to one particular meaning or that are overloaded with different meanings. Prior research in requirements engineering has brought forth several approaches to manage such issues. Despite the efforts, there is currently no approach available that addresses this problem of semantic ambiguity in process models. In Section 6.1, we give an overview of related approaches from ry equirements engineering and emphasize the necessity to address semantic ambiguity in process models. Section 6.2 explains how the linguistic concept of word senses is operationalized and used to describe characteristics of ambiguous words. The Sections 6.3 and 6.4 then present the techniques to detect and resolve ambiguous words. Section 6.5 continues with a discussion of the user evaluation where these techniques have been applied to process model collections from industry. Afterwards, Section 6.6 gives a summary of the results and gives important implications of the evaluation. Finally, Section 6.7 gives an overview of the main points to conclude the chapter. The main results of the presented techniques have lead to two publications [329, 331].

6.1 Lexical Ambiguity in Process Models

Process models are often used to support the specification of requirements of information systems [188] or to analyze and redesign the business processes of an organization [109]. It is essential that these models use a consistent terminology to describe the underlying business process, because these models are used by people from different organizational units with different knowledge and background. Still, several authors emphasize that this is a major concern in real-world settings [186, 40]. Bolloju and Leung [47] find ambiguous descriptions and inconsistent element names in 64% of the UML models they evaluated. In the field of automatic assessment of UML activity diagrams, Jayal and

Sheppard [177] also face the challenge of ambiguous label names when matching them to semantically similar ones. The most tangible manifestation of lexical ambiguity is the usage of homonyms and synonyms. Homonyms are words that have more than a single meaning, e.g. *application*. Synonyms are different words which refer to the same meaning, e.g. *bill* and *invoice*. Although the problem of synonyms and homonyms is well understood in various areas of system analysis and design tasks, such as requirements engineering [40, 186], use case descriptions [85, 417], model transformations [439], code and design smells [294], matching code and documentation [264], or system components reuse [290, 234], there has been only limited research on addressing such problems in process models. Section 6.1.1 discusses approaches and techniques that are used to manage ambiguity in models and natural language text in order to address the problem of ambiguous terminology. Then, Section 6.1.2 discusses notable challenges of ambiguity detection and resolution thereof.

6.1.1 Managing Ambiguity in Process Models

Despite the fact that semi-formal languages (e.g. BPMN or UML activity diagrams) are meant to restrict ambiguity [188], the elements of process models still entail small fragments of natural language text to specify the semantics of the business process. These language fragments are a reason of ambiguity which is similar to natural language text (see e.g. [119, 71, 142]). Moreover, the ambiguity problem is even more serious in process models since the process model gives an abstract view of the business process and provides only limited context to detect and resolve ambiguities [285, 249]. Finally, models often focus on isolated aspects of the system. Once a system integrates several parts of the organization, a number of models has to be considered before inconsistencies can be detected [343].

In order to improve process models and to ensure the quality of process-aware information systems, several authors propose an integration of process modeling with requirements engineering [19, 24]. This is particularly promising since approaches to ambiguity management have been discussed in the latter discipline. In requirements engineering, two classes of approaches can be distinguished. The first class includes techniques that *detect ambiguities* in requirements documents and explain them to the user. The second class encompasses techniques that attempt to *resolve ambiguities*. Each class can be subdivided into approaches focusing on detecting ambiguity in *requirements documents* and in *models* (see Table 6.1).

Ambiguity detection approaches aim at the reduction of ambiguity by identifying terms that can be interpreted in multiple ways. For that purpose, various techniques have been proposed to assess requirements documents of different input types. For *requirements in text*, i.e. requirements specifications written in natural language, different reading techniques may be employed. Inspection-based reading [188, 187] uses a detailed checklist of ambiguous

Table 6.1: Approaches for Ambiguity Management in Natural Language Text and Models

Technique	**Author**
Ambiguity Detection in Textual Requirements	
Reading Techniques	Kamsties et al. [188, 187, 186]
	Shull et al. [396]
Natural Language Patterns	Denger et al. [97],
	Gleich et al. [142]
	Rolland and Ben Achour [368]
Ambiguity Metrics	Fantechi et al. [125],
	Ceccato et al. [67],
	Fabbrini et al. [119],
	Wang et al. [429]
Ambiguity Heuristics	Chantree et al. [71]
Classifier Construction	Yang et al. [447, 446]
Ambiguity Resolution in Textual Requirements	
Notification	Chantree et al. [71]
	Gleich et al. [142]
Pattern-based rephrasing	Rolland and Ben Achour [368]
Ambiguity Detection in Models	
Reading Techniques	Anda and Sjøberg [18]
Rule-based Detection	Mens et al. [287]
Ambiguity Measurement Metrics	Friedrich [135]
Verifying Concept Relations with a Lexicon	Van der Vos [425]
Ambiguity Resolution in Models	
Rule-based Resolution	Mens et al. [287, 288]
Preventing Ambiguity with a Domain Thesaurus	Becker et al. [31, 32]
	Delfmann et al. [96]
	Havel et al. [157]

words in requirements engineering such that the document is carefully inspected for potential ambiguities. Scenario-based reading [186] provides an inspector with an operational scenario, which requires the inspector to first create an abstraction of the requirements document and then answer questions based on the document. If the questions cannot be answered consistently, the inspector has found an inconsistency. Object-oriented inspections [396] are a specific reading technique to align textual requirement descriptions with models of object-oriented components. This technique proposes to find associations of the textual description with regard to the model and to check the description for completeness of attributes, methods, or conditions. Besides the reading techniques, there are also technical approaches to detect ambiguities in requirements documents. Denger et al. [97] as well as Gleich et al. [142] use natural language patterns to manage ambiguity in requirements documents. While Denger et al. [97] proposes generic sentence patterns to describe events or reactions of a software, Gleich et al. [142] use regular expressions and keywords to detect ambiguities. For example, the expressions *many* or *few* might point to vaguely formulated requirements. Moreover, several authors develop metrics that indicate ambiguities in requirements documents. Among them, Fantechi et al. [125] and Ceccato et al. [67] define metrics for the degree of readability and understandability (average number per sentence) as well as the ambiguity of natural language text (average number of sentences containing vague expressions). Fabbrini et al. [119] also provide measures for underspecification (number of sentences containing words without modifier) and implicit sentence structures (number of sentences containing expressions such as *they* or *above*). Wang et al. [429] use metrics to create a ranking of ambiguities according to ordering criteria, such as concept frequency (occurrence of a word in all requirements documents) or context diversity (average cosine similarity of target word and co-occurring words). Chantree et al. [71] propose heuristics to automatically replicate human judgments of ambiguous requirements documents and their interpretation. For example, the authors introduce the distributional-similarity heuristic which measures how frequent two words occur in the same context. If the measure reveals a weak distributional similarity, an ambiguity is more likely. The previous approach has been extended by Yang et al. [447, 446]. The authors employ a machine learning approach and build an ambiguity classifier based on the human judgments.

For *requirements in models*, Anda and Sjøberg [18] adapt the idea of reading techniques to use case models and propose a checklist-based approach to detect inconsistencies and ambiguities. Mens et al. [287] propose a detection approach by using graph transformation rules to support the ambiguity detection task in UML class models and state machine diagrams. For example, the authors identify methods in a class diagram that have undefined types and resolve this inconsistency by either removing the respective parameters or assigning an existing type to the undefined one. Friedrich [135] proposes semantic metrics that reflect the degree of ambiguity of a particular word and that provide a baseline to automatically detect text fragments with a high chance of ambiguity. The metric considers the number of occurrences of a word divided by a user defined minimum frequency. Van der Vos [425] uses a semantic lexicon to check the quality of model elements. It ensures that words and phrases of model element labels are used in a linguistically meaningful and sense-making way. For that purpose, the technique checks if the label correctly refers to a specific concept or object of the real world and if the combination between several objects makes sense in the context of a model.

Once ambiguities have been detected, approaches for the *systematic resolution* are employed to restore the consistency of concepts and terminology. The resolution of ambiguous concepts needs to identify the correct interpretation of the respective item and to rewrite it according to the modeler's intention. For *textual requirements*, several authors, such as Chantree et al. [71], acknowledge mental capabilities of humans to resolve these ambiguities. Similarly, Gleich et al. [142] propose a notification approach that highlights an ambiguous concepts and explains why the concept is ambiguous. Afterwards, it is up to the users to handle the detected ambiguity by themselves. Going one step further, Rolland and Ben Achour [368] propose a rephrasing technique that makes use of precise natural language patterns to replace the ambiguous statement. For instance, the authors propose a clarification rule that changes the wording of anaphoric references, such as he, she, it, his or him.

In order to resolve such ambiguity in *models*, Mens et al. [287, 288] enrich the previously mentioned detection technique with transformation rules that automatically rework the defects in class and state machine diagrams. The approaches of Becker et al. [31, 32], Delfmann et al. [96] as well as Havel et al. [157] enforce specific naming conventions in process models. These tools also implement a user-created domain thesaurus, which replaces synonyms with a previously defined dominant alternative. For example, the user may specify that the words *invoice* and *bill* are synonyms and that *invoice* is the dominant synonym. Then, the tool automatically replaces all occurrences of *bill* with *invoice.*

Despite their merits, the presented techniques for the detection and resolution of lexical ambiguity suffer from three main shortcomings that impede their application to process model repositories: the required manual effort, the missing focus on process model text fragments, and their focus on single documents.

The first shortcoming, the *required manual effort*, refers to the extensive amount of manual work that is required to detect and resolve ambiguities in process models. Since companies tend to maintain several hundreds or even thousands of process models [369], the human effort can be tremendous. In particular, the previously discussed reading techniques can hardly be applied to such a number of models. Similarly, this also applies for the resolution of the detected ambiguities, where the requirements engineers need to understand the context of the detected ambiguity and come up with a more precise alternative. The large manual effort also impedes the application of heuristic-based or machine learning approaches because each of these approaches requires a manually created data set from which heuristics can be derived or from which machine-learning approaches can be trained.

The second shortcoming, the missing *focus on process model text fragments*, refers to the fact that many approaches are tailored to deal with sentences and phrases taken from a grammatically and syntactically correct natural language text. However, the elements of process models contain only short textual fragments that do not exhibit a complete or a correct sentence structure [249, 285]. As a result, the discussed approaches, as the ones from Denger et al. [97], Gleich et al. [142], or Rolland and Achour [368], are hardly applicable to process models. The few approaches that look deeper into the semantics of the textual fragments in models are however limited and do not provide means for ambiguity detection or resolution. While van der Vos [425] focuses on meaningful combinations of terms, the semantic metrics of Friedrich's ambiguity metrics [135] only point to possible ambiguous terms without considering the context of the process model.

The third shortcoming, i.e. the *focus on single documents*, relates to the observation that most of the proposed techniques are applicable only to single documents, single models, or smaller units thereof. Hence, these techniques only address ambiguities within a single document or process model. However, since we assume a repository of several process models, the correction of ambiguities on document level might introduce an inconsistency in another document or model. In this context, the approaches of Becker et al. [31, 32], Delfmann et al. [96] as well as Havel et al. [157] are beneficial as it enforces specific naming conventions based on a domain thesaurus, which prevents the introduction of conflicting terms during model creation. Unfortunately, their approach assumes the existence of such a domain thesaurus and the creation of a process model from scratch which impedes the application to already existing repositories.

Despite the findings and the diversity of approaches, there is no technique available that can address the detection and resolution of lexical ambiguity in process models on the level of a model collection. The following section addresses this gap by emphasizing particular challenges to meet this objective.

6.1.2 Challenges for Ambiguity Detection and Resolution

There are several challenges associated with the automatic detection and resolution of lexical ambiguity in process model repositories. We use BPMN process models as depicted in Figure 6.1 in order to illustrate these challenges. Scenario A describes a job application process. The process starts with the receipt of an application followed by a check for completeness. In case the documents are not complete, the missing documents are requested and checked again. Otherwise, the documents are evaluated and it is decided whether the application is rejected or the applicant is invited to an interview. Scenario B depicts the general procedure to update software programs. The process starts with necessary preparations of the software program the upcoming maintenance. Then, all necessary update files are received from the software vendor. If the update files are incomplete, the responsible person calls for the missing files and receives the missing files from the vendor. If everything is complete, the application is updated and restarted afterwards.

These models are subject to lexical ambiguity. First, each process model uses the word *application*. Scenario A utilizes it in the sense of a job application as well as in the sense of a software program. We can infer the meaning of the first instance from the context in which documents are checked and an interview takes place. The second instance refers more likely to a software program since it is involved in the evaluation of the candidate. In scenario B, the word *application* is also used as a software application. Apparently, this small

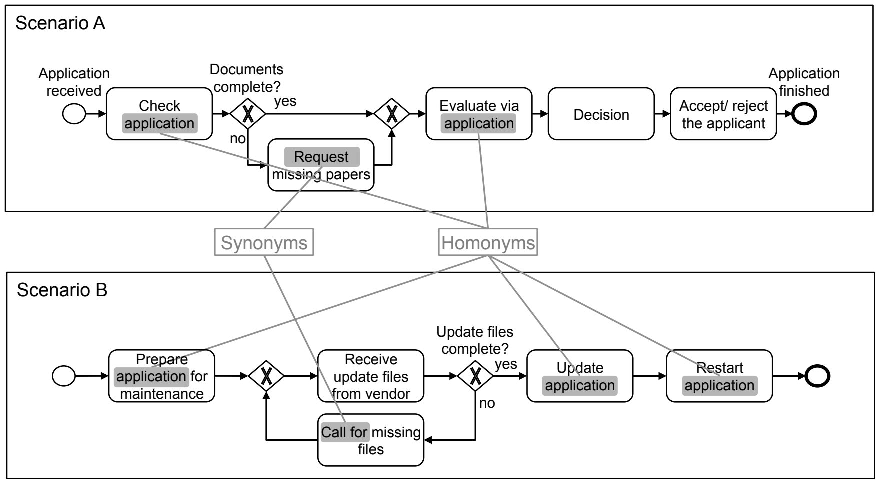


Fig. 6.1: Example for Lexical Ambiguities in Process Models

sample of two models uses the word *application* with two different meanings, which let us conclude that *application* is a homonym. Moreover, we observe another inconsistency in scenarios A and B. The word *to request* expresses a need for a particular object. The same semantics can be assigned to the word *to call for*. Thus, the two words are used to express the same concept and are hence considered to be synonyms. Apparently, these semantic inconsistencies are not restricted on a single process model, but also affect several models at the same time pointing to specific challenges to detect homonyms and synonyms. In general, we face three challenges for the detection and resolution of lexical ambiguities: sense disambiguation, sense operationalization, and ambiguity resolution. Table 6.2 provides an overview of techniques addressing these challenges.

The challenge of *ambiguity operationalization* is concerned with representing semantic meaning of a word in such a way that ambiguities can be detected. Such a representation would need to capture various aspects of a particular word. One prominent example of such a representation is the vector space model, which was introduced by Salton et al. [379] to describe documents in an information retrieval system. The model expresses indexing terms of a document as dimensions of the vector space. The degree to which a document is associated with a specific index term is expressed by numerical values. A similar approach, the *word space model* by Schütze [391], applies this concept to the linguistic field. The word space model interprets the vector dimensions as words. Following Schütze, a word in a corpus is represented as a vector whose components count the number of occurrences within a fixed context (for example a sentence or a larger context). As a result, the distributional profile of words implicitly expresses the meaning of the word in a given

Table 6.2: Challenges for Ambiguity Detection and Resolution

Challenge	**Applicable Concepts**
Ambiguity Operationalization	
Vector Space Model	Salton et al. [379]
Word Space Dimensions	Schütze [391]
Ambiguity Definitions	Deissenboeck and Pizka [95]
Sense Detection	
Supervised Methods	Cai et al. [65]
	Agirre and Lopez de Lacalle [12]
	Niu et al. [308]
Unsupervised Methods	Pederson [320]
	Kern et al. [192]
	Koeling and McCarthy [206]
Knowledge-based Methods	Navigli and Ponzetto [303, 304]
	Chan et al. [69]
	Novischi [310]
Word and Context Clustering	Schütze [392]
	Lin [252]
	Pantel and Lin [319]
Ambiguity Resolution	
Additional Information	Frakes and Pole [132]
	Lin and Chan [254]
Guided Interaction	Bouzeghoub et al. [50]
Alternative Selection	Hayne and Ram [158]
Automatic Rephrasing	Ben-Ari et al. [36]

context. Deissenboeck and Pizka [95] introduce a formalization of ambiguities to deal with synonyms and homonyms in identifier naming. Although their formalization captures essential characteristics of ambiguity, it is not sufficiently precise to automatically detect synonyms and homonyms. The mere existence of a word with multiple senses does not necessarily imply that it also represents an ambiguous case. To this end, we need to redefine the vector space model, to transfer it to word senses, and to refine the conditions that allow to confirm or reject the hypothesis of ambiguity.

The challenge of *sense detection* refers to the field of word sense disambiguation (WSD) and relates to determining the sense of a word in a given context [300]. In general, we distinguish between WSD techniques that employ supervised machine-learning techniques (e.g. [65, 12, 308]), unsupervised techniques (see. e.g. [320, 192, 206]), and knowledge-based methods (e.g. [303, 304, 69, 310]). Moreover, several clustering approaches have been used to identify context similar words (e.g. [392, 252, 319]). However, each existing approach requires an extensive amount of natural language text as input. As the textual information of process model elements is typically very short, it provides limited context to identify the correct sense of a word. This insight is also confirmed by Zong and Ng [451] who report worse WSD results for short queries in information retrieval. Referring to the example, available WSD algorithms may only employ the activity and gateway labels, such as *Receive application* or *Documents complete?* in order to find the correct meaning of the word *application.* An initial try with the technique of Navigli and Ponzetto [303] identified the software application as the most suitable meaning (score: 11), followed by job application (score: 8). It is not recommended to rule out the other possible meaning, because these scores are relatively close to each other and the process model does not provide much textual information as context. Therefore, the idea of sense determination needs to be extended and to consider the imprecision given by the short textual information of process models.

After the detection, the challenge of *resolution* needs to be addressed. Ideally, the detected ambiguities can be removed in order to achieve consistency and correctness among a set of process models. Reconsidering the example from Figure 6.1, it is desirable to replace the homonym *application* with *job application* in scenario A and with *software program* in scenario B.

In the literature, there are four different strategies to resolve ambiguities. Frakes and Pole [132] as well as Lin and Chan [254] propose to append additional information to the ambiguous word which explains the context, such as *application (job, employment).* Bouzeghoub et al. [50] suggest the incorporation of user-interaction in order to manually resolve ambiguity. A third strategy was developed by Hayne and Ram [158]. Their approach provides the user with the possibility to choose from automatically created recommendations. The last strategy suggests the automatic rephrasing or replacement of ambiguous words with more precise alternatives. An exemplary approach has been developed by Ben-Ari et al. [36] who propose an interactive approach to rephrase ambiguities. However, these strategies either require manual efforts ([132, 254]), have been applied in specific scenarios ([50, 158, 254]), or have been developed for correct natural language sentences ([36]). Hence, we require strategies that can deal with short text fragments that are common in process models.

Building on these challenges, this thesis proposes techniques to automatically detect and resolve lexical ambiguities in process models in the following sections.

6.2 Operationalizing Word Senses and Lexical Ambiguity

In the subsequent sections, we focus on the operatinalization of semantic concepts and explain how they are used to detect lexical ambiguity. Therefore, Section 6.2.1 introduces semantic vectors as a central concept to represent all word senses of a potentially ambiguous word. Based on this concept, Section

6.2.2 elaborates ambiguity conditions which leverage the automatic detection of homonyms and synonyms.

6.2.1 The Notion of Semantic Vectors

It is necessary to operationalize lexical ambiguity in order to address the first challenge. The operationalization mainly involves a formal description of the basic characteristics of lexical ambiguity, namely homonyms and synonyms. As already mentioned, a homonym is a word with multiple overlapping word senses. We regard the word *application* as a homonym as it may refer to multiple senses such as a *job application*, a *software program*, an *application in the medical sense*, or *diligence and effort*. Synonyms are words that have one word sense in a common. The words *bill*, *invoice*, and *account* are synonyms since they share a common meaning, i.e. the itemized statement of owed money. It is necessary to investigate all word senses of a given word in order to take a decision on homonymy and synonymy.

A suitable representation of all word senses is achieved by using a vector-based representation of its senses. This representation is inline with the concepts of the vector space model by Salton et al. [379] and the word space model by Schütze [391]. The vector space model describes a document with the help of indexing terms and treats them as vector dimensions. The degree to which a document is associated with a specific index term is expressed by numerical values in the vector. The word space model interprets the vector dimensions as words and counts the occurrences of a word within a fixed context. Transferring these concepts to word senses, we introduce the notion of a semantic vector SV_D^w of a word w. A semantic vector represents word senses as vector dimensions which correspond to distinct word senses of a given dictionary D. If the word is used with a specific meaning, its value in the semantic vector is non-zero. The higher the value, the more prevailing the respective meaning in a given context. Accordingly, the concept of semantic vectors is defined as follows:

Definition 6.1. (**Semantic Vector**). Let w be a word, $p \in POS$ its part of speech, and $Senses_D(w, p) = \{s_1, \cdots, s_i, \cdots, s_n\}$ its word senses denoted in a dictionary D. The semantic vector $SV_D^w$is given by

$$SV_D^w = \begin{array}{c} \\ \end{array}\begin{array}{ccccc} s_1 & \cdots & s_i & \cdots & s_n \\ \big(sv_1 & \cdots & sv_i & \cdots & sv_n\big) \end{array}, \text{ such that}$$

- The i^{th} dimension of SV_D^w corresponds to the i^{th} sense $s_i \in Senses_D(w, p)$.
- The value sv_i indicates to which degree the word is used with the i^{th} sense in a given context.[1]

To illustrate these concepts, we refer to the word *application* from Scenario A of Figure 6.1 and use the BabelNet word senses as shown in Table 4.7.

[1] In this thesis, the supporting scores are calculated by using the Multilingual WSD algorithm as presented in Section 4.4.3.

Accordingly, the word *application* has ten distinct word senses, such as the sense of a *job application* (s_2) or a *software program* (s_4). Taking the process model of Scenario A as context and using the WSD approach of Navigli and Ponzetto [303], the following semantic vector reflects the meaning of the word *application* in scenario A:

$$SV_{BabelNet}^{application} = \begin{array}{c} \\ \end{array}\begin{array}{cccccccccc} s_1 & s_2 & s_3 & s_4 & s_5 & s_6 & s_7 & s_8 & s_9 & s_{10} \\ \left(3\right. & 8 & 6 & 11 & 3 & 2 & 1 & 3 & 0 & \left.0\right) \end{array}$$

Apparently, the context of Scenario A provides sufficient support that the word *application* may refer to a software program (s_4-value: 11), a job application (s_2-value: 8) or the work of covering something (s_3-value: 6). Although it might be obvious for humans that the job application is the most appropriate word sense in the process model, we cannot completely rule out the fact that other interpretations are meaningful, even if only to a small extent. In some cases, it might happen that several word senses are more likely resulting in bigger s_i-values. In this case, we might face an ambiguity or a homonym to be precise. We can however use these regularities to describe conditions to detect ambiguous usage of words, as shown in the next section.

6.2.2 Ambiguity Detection Conditions

The specification of ambiguity detection conditions aims at operationalizing the basic characteristics of lexical ambiguity. These conditions rest upon the concept of a semantic vector and the formalization of homonyms and synonyms by Deissenboeck and Pizka [95]. The following conditions adapt their ideas to semantic vectors and describe necessary conditions for the homonyms and synonyms:

Definition 6.2. (**Homonyms**). Given a word $w \in W$ denoted in a lexicon D, its part of speech $p \in POS$, and its semantic vector SV_D^w. Then, the word w is a *homonym* if:

$$\exists s_i, s_j \in Senses_D(w,p) : s_i \neq s_j \wedge sv_i > 0 \wedge sv_j > 0$$

As an example consider the semantic vector $SV_{BabelNet}^{application}$ from the previous section. Since the semantic vector entails several values with a score bigger than 0, Definition 6.2 identifies the word *application* to be a homonym. Opposite to that, a synonym is defined as follows:

Definition 6.3. (**Synonyms**). Given two words $w_1, w_2 \in W$ denoted in a lexicon D, their parts of speech $p_1, p_2 \in POS$, and their semantic vectors $SV_D^{w_1}$ and $SV_D^{w_2}$. Then, the words w_1 and w_2 are synonyms if:

$$\exists s_i \in Senses_D(w_1,p_1), \exists s_j \in Senses(w_2,p_2) : s_i = s_j \wedge sv_i > 0 \wedge sv_j > 0$$

To illustrate Definition 6.3, consider the words *sales* and *revenue* in Scenario B from Figure 6.1. Moreover, consider Table 6.3 which depicts all elements of

Table 6.3: Example for Synonym Semantic Vectors

	Word Sense s_i	**Description**	**Value** sv_i
sales	s_1	Income received for goods and services	35
	s_2	Value of sales generated by a company	7

	Word Sense s_j	**Description**	**Value** sv_j
revenue	s_1	Income available to the government	4
	s_2	Income returned by an investment	5
	s_3	Income received for goods and services	35
	s_4	Government income due to taxation	18

the semantic vector for these two words, i.e. their word senses, a description of the meaning, and their value in the semantic vector:

The table shows that both words may refer to *Income received for goods and services*, i.e. $s_1 \in Senses_D(\text{sales}, n) = s_3 \in Senses_D(\text{revenue}, n)$. Moreover, each of these word senses have a value bigger than 0. According to Definition 6.3, the words *sales* and *revenue* are synonyms.

One weakness of homonyms and synonyms is that their definitions only capture the basic characteristics of lexical ambiguity. In particular, they do not sufficiently consider the context of a given word. The implications of the missing context are explained best by reconsidering the previously discussed word *application*. According to BabelNet, this word has ten different senses among which eight achieve a s_i-value bigger than 0. Hence, Definition 6.2 identifies the word *application* to be a homonym. However, only because a word *may* refer to multiple senses, this does not necessarily mean that it is actually harmful. A word is ambiguous only, if the context of the word is not sufficient to infer the correct sense. This is the case when the vector contains s_i-values that are close to each other. Therefore, the definitions 6.2 and 6.3 constitute necessary conditions only.

We then refine the notion of ambiguity with two consecutive steps in order to finally decide about the homonymous or synonymous character of a word. The first step involves the determination of the *dominant word sense* which is inline with standard WSD approaches, that typically assign a supporting score to an input target word given a set of words as a context [271, 270]. The word sense with the highest supporting score is typically assumed to be the most appropriate one and thus dominating all other word senses. Therefore, we formally state this condition as follows.

Definition 6.4. (**Dominant Word Sense**). Given a word $w \in W$, its part of speech $p \in POS$, and its semantic vector SV_D^w , a sense $s_i \in Sense_D(w, p)$ is *dominant* if there is no other sense $s_j \in Sense_D(w, n)$ with a higher value in the semantic vector: $\forall s_i, s_j \in Sense_D(w, p), s_i \neq s_j : \; sv_i \; > \; sv_j$. We denote the dominant sense with s_{max}.

Let us again consider the word *application* in Fig. 6.1 and its semantic vector with regard to scenario A, i.e.

$$
\begin{array}{ccccccccccccc}
 & & s_1 & s_2 & s_3 & \mathbf{s_4} & s_5 & s_6 & s_7 & s_8 & s_9 & s_{10} & \\
SV_{BabelNet}^{application} & = & (\; 3 & 8 & 6 & \mathbf{11} & 3 & 2 & 1 & 3 & 0 & 0 &)
\end{array}
$$

According to the definition of the dominant word sense, the 4^{th} sense is considered to be dominant since its value $sv_4 = 11$ is the highest among all supporting scores.

In the second step, we need to consider those cases where a clearly dominating word sense is not present. Although the 4^{th} sense is dominating due to the biggest vector value, the 2^{nd} sense is very close. Hence, the conclusion is valid that this particular sense is also appropriate for the given context and that it cannot be excluded from further consideration. In order to also capture such senses, we introduce a user defined confidence score $\epsilon \in [0..1]$ that defines a spectrum of appropriate word senses around the dominating one. Therefore, we introduce the notion of quasi-dominant word senses as follows.

Definition 6.5. (**Quasi-Dominant Word Senses**). Given a word $w \in W$, its part of speech $p \in POS$, its semantic vector SV_D^w, and its dominating sense s_{max}. Then, a sense $s_i \in Senses_D(w, p)$ is *quasi-dominant* if: $sv_i \;\geq\; \epsilon \cdot sv_{max}$.

Definition 6.6. (**Subsets for Dominant Word Senses**). Given a word $w \in W$, its part of speech $p \in POS$, its semantic vector SV_D^w. We define the following subsets for its dominating sense s_{max} and its quasi-dominant word senses given by the Definition 6.5:

– $senses_{QD}^w = \{s_i \in Senses_D(w,p) | \; sv_i \;\geq\; \epsilon \cdot sv_{max}\}$, as being the set of all quasi-dominant word senses.
– $domSense^w = \{s_{max}\} \cup senses_{QD}^w$, as being the set containing all dominant and quasi dominant word senses.

Reconsidering the example of the word *application*, we already identified the 4^{th} sense to be dominant. If we set the confidence score ϵ to 0.7 and apply the Definition 6.5 to the remaining values of the semantic vector $SV_{BabelNet}^{application}$, we also find the 2^{nd} sense as quasi-dominant with regard to the other vector values. Obviously, since more word senses appear to be appropriate, the word *application* actually represents a homonym. We thus extend the necessary conditions of homonyms and synonyms with these concepts. The extended definitions formulate an extended condition for synonyms and homonyms that build upon the necessary conditions in the definitions 6.2 and 6.3. They consider the dominant senses of the word in the respective context and thus indicate if the respective synonym or homonym candidate should be confirmed or rejected.

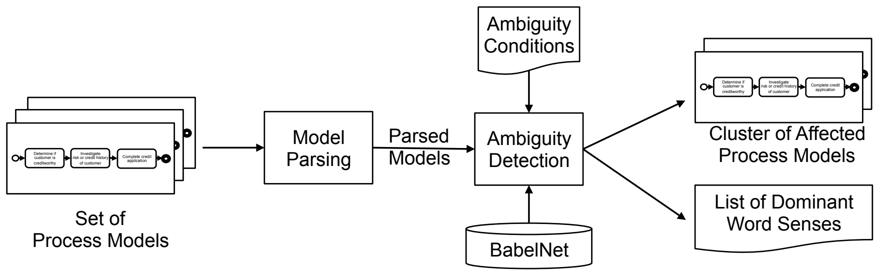


Fig. 6.2: Overview of Detection Approach

Definition 6.7. (**Extended Homonym Condition**). Given a word $w \in W$, its part of speech $p \in POS$, its semantic vector SV_D^w, and its set of dominant and quasi-dominant word senses $domSense^w$. Then, the word w is a *confirmed homonym* if Definition 6.2 holds and $|domSense^w| > 1$. Otherwise, it is a *rejected homonym*.

Definition 6.8. (**Extended Synonym Condition**). Let $w_1, w_2 \in W$ be two words, $p_1, p_2 \in POS$ their parts of speech, $SV_D^{w_1}$ and $SV_D^{w_2}$ their semantic vectors. Further, let $domSense^{w_1} \subseteq Senses_D(w_1, p_1)$ and $domSense^{w_2} \subseteq Senses_D(w_2, p_2)$ their sets of dominant and quasi-dominant word senses. Then, the words w_1 and w_2 are *confirmed synonyms* if Definition 6.3 holds and if $domSense^{w_1} \cap domSense^{w_2} \neq \emptyset$. Otherwise, the words are *rejected synonyms*.

This section provided the formal concepts and extended conditions for homonym and synonym ambiguity. Building on these definitions, we introduce a technique that identifies and resolves lexical ambiguities in process model repositories in the following section.

6.3 Detection of Ambiguity

This section discusses the general steps to detect homonym and synonym ambiguities. Figure 6.2 shows the general architecture of the detection approach. The input of the approach is a set of process models. The approach begins with an extraction of the label components. Then, the ambiguity detection component analyzes the process models with the help of the discussed ambiguity conditions and a computational lexicon, such as BabelNet. As a result, the component produces a list of detected ambiguities and affected models.

In the following sections, we will specify the ambiguity detection approach that detects lexical ambiguities in process models repositories. Section 6.3.1 focuses on homonym detection, while Section 6.3.2 explains the necessary steps for synonym detection.

6.3.1 Homonym Detection

Building on the arguments from the previous sections, we further require the detection technique to preserve the quality and consistency of the considered model collection, i.e., process models with a similar contexts need to be treated equally. In particular, we need to consider the fact that a word occurs in several process models and that the word can have a different sense in another model. This means that we need to find those models in which the target word is used with similar senses. In this cases, the semantic vectors are close to each other. In a subsequent step, we decide for a group of several process models if a target term is used ambiguously or not. his procedure ensures that the ambiguity is resolved consistently through all process models within a group. For this purpose, we employ the XMeans clustering approach by Pelleg and Moore [323] since it efficiently estimates the number of clusters automatically and is capable of finding smaller groups of vectors with similar characteristics. We then check the necessary and the extended condition for each cluster center in order to confirm or reject the homonym condition.

These steps are formalized in Algorithm 3, which requires the process model collection $\mathcal{P}$ and a target word w as input and starts with basic initializations (Steps 2-4). The set $SemanticVectors$ stores the semantic vectors that are created from disambiguating the word w with respect to the process model P which serves as the context. The set *homonymCandidates* stores the final results of the algorithm and contains the detected homonyms. After the initializations, the necessary condition for homonyms is checked. For this purpose, we employ the BabelNet database and retrieve the synsets of the word w (Step 5). If this check does not evaluate to false, the semantic vectors for each process model P and word w are calculated using the graph-based WSD approach with BabelNet. We denote this with the function $babel(w, P)$ that takes the word w and a process model P as input and returns the semantic vector according to definition 6.1. Each semantic vector is then added to the set $SemanticVectors$ (Steps 5-7). In Step 9, the semantic vectors are clustered to identify groups of context similar vectors. We denote this with the function $xmeans(semanticVectors)$ that takes the set of all semantic vectors of word w as input and returns a set of identified clusters. Afterwards, the extended homonym condition is checked according to Definition 6.7 for each cluster (Steps 10-11). The extended homonym evaluation makes use of the function $DominantSense(sc)$ that returns all dominant and quasi-dominant senses. If it evaluates to true, the word and the respective cluster are confirmed to be a homonym candidate and added to the result set (Step 12). At the end, the algorithm returns the final result set and terminates (Step 16).

6.3.2 Synonym Detection

Regarding the detection of synonyms, the technique has to consider pairs of words that have at least one meaning in common. Analogously to the

Algorithm 3: Detection of Homonyms for a Set of Process Models

```
 1: detectHomonyms(Word $w$, ProcessModels $\mathcal{P}$)
 2: $semanticVectors \leftarrow \emptyset$
 3: $semanticClusters \leftarrow \emptyset$
 4: $homonymCandidates \leftarrow \emptyset$
 5: if BabelNet.RetrieveSynsets($w$) $\geq 2$ then
 6:    for all P $\in \mathcal{P}$ do
 7:       $semanticVectors \leftarrow semanticVectors \cup$ babel($w$,P)
 8:    $semanticClusters \leftarrow$ xmeans($semanticVectors$)
 9:    for all $sc \in semanticClusters$ do
10:       if |DominantSenses($sc$)| $\geq 2$ then
11:          $homonymCandidates \leftarrow homonymCandidates \cup (w, sc)$
12: return $homonymCandidates$
```

homonym problem, these two words can only be synonyms if their context is approximately identical. For this purpose, we again need to learn about the word senses of two words and have to decide if they fulfill definition 6.3. If so, we use a WSD approach and create the semantic vectors. Afterwards, we employ the XMeans clustering approach for each semantic word space and identify the dominant and quasi-dominant word senses. In contrast to the homonym approach, we have to compare the resulting clusters of each candidate word with each other to check the extended synonym condition. If two clusters share at least one meaning, i.e., the intersection of dominant and quasi dominant sense is not empty, we confirm the two words in the respective clusters as synonym candidates.

The synonym detection approach is formalized in Algorithm 4. It takes two words w_1 and w_2 as well as the set of process models $\mathcal{P}$ as input. Afterwards, it starts with the basic initializations, i.e., the initialization of the semantic vector and cluster set for w_1 and w_2 and the result set (Steps 2-5). In Step 6, the necessary synonym condition is checked according to definition 6.8. If true, the semantic vectors of each word are calculated using the BabelNet WSD approach which we denote again with the function $babel(w, P)$ (Steps 8-11). The algorithm continues with clustering the vectors of each semantic vector space denoted by the function $xmeans(semanticVectors)$ (Steps 12-13). Finally, the algorithm checks the extended synonym condition for each cluster (Step 15). If the dominant and quasi-dominant senses of one cluster of word w_1 intersect with the dominant senses of one cluster of word w_2, the extended condition evaluates to true and the two words as well as the respective clusters are stored in the result map (Steps 15-16). The algorithm terminates by returning the result set *synonymClusters* (Step 20).

Algorithm 4: Detection of Synonyms in a Set of Process Models

```
1: **detectSynonyms**(**Word** $w_1$, $w_2$, **ProcessModels** $\mathcal{P}$)
2: $semanticVectors_1 \leftarrow \emptyset$
3: $semanticVectors_2 \leftarrow \emptyset$
4: $semanticClusters_1 \leftarrow \emptyset$
5: $semanticClusters_2 \leftarrow \emptyset$
6: $synonymCandidates \leftarrow \emptyset$
7: if $(Senses_D(w1,*) \cap Senses_D(w2,*)) \neq \emptyset$ then
8:   for all $\mathrm{P}_1, \mathrm{P}_2 \in \mathcal{P}$ do
9:     $semanticVectors_1 \leftarrow semanticVectors_1 \cup$ babel($w_1$,$\mathrm{P}_1$)
10:    $semanticVectors_2 \leftarrow semanticVectors_2 \cup$ babel($w_2$,$\mathrm{P}_2$)
11:  $semanticClusters_1 \leftarrow$ xmeans($semanticVectors_1$)
12:  $semanticClusters_2 \leftarrow$ xmeans($semanticVectors_2$)
13:  for all $sc_1 \in semanticClusters_1$,
     $sc_2 \in semanticClusters_2$ do
14:    if (DominantSenses($sc_1$)
       $\cap$ DominantSenses($sc_2$)) $\neq \emptyset$ then
15:      $synonymCandidates \leftarrow synonymCandidates \cup (w_1, sc_1, w_2, sc_2)$
16: return $synonymCandidates$
```

6.4 Resolution of Ambiguity

A fully automatic resolution of lexical ambiguity is associated with considerable challenges and thus far away from trivial. Therefore, it is often more appropriate to rely on human resolution instead and to notify them about detected ambiguity cases [159, 71]. In this line, this thesis proposes a resolution technique that aims at providing context-sensitive refactoring suggestions for the detected synonyms and homonyms. In Section 6.4.1, we first discuss strategies to generate resolution suggestions for homonyms. Then, Section 6.4.2 presents a strategy to resolve synonyms.

6.4.1 Homonym Resolution

The resolution of homonyms is addressed using different perspectives. We propose three resolution strategies. The first strategy exploits the semantic relationship of *hyponomy*. The hyponymy relation describes a transitive relation between word senses that organizes them into a sub-word hierarchy. It starts with a general word and becomes more and more specific. In order to resolve the homonym conflict, one can choose a suitable hyponym offered by the dictionary. As an example, consider the fragment of BabelNet's hyponymy tree for the word *application* in Figure 6.3. It illustrates that BabelNet provides four different hyponyms that can be used to resolve the ambiguity conflict of the word application. Accordingly, this resolution strategy would suggest the depicted hyponyms, i.e., *job application*, *credit application*, *loan application*,

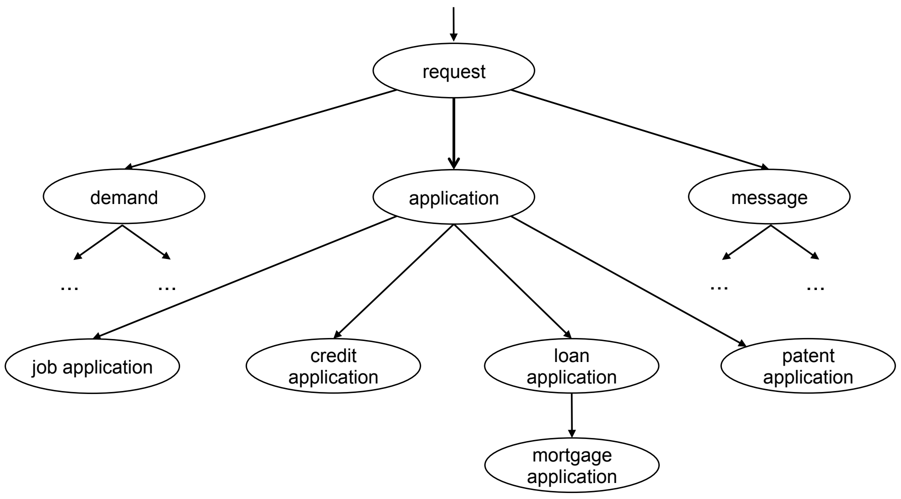


Fig. 6.3: Example Fragment of the Hyponymy Tree

and *patent application*. Building on this concepts, the hyponym resolution strategy is formalized as follows:

$$Suggestion_{Hypo} = hypo_{BabelNet}(s'),$$

where $hypo_D(w, s')$ denotes a function that returns the set of all hyponyms of the word sense $s' \in Senses_D(w, p)$ in the BabelNet dictionary as given by Definition 4.7.

The second homonym resolution strategy employs the hyponym relation as well. However, instead of replacing the ambiguous term with a hyponym (a more specific word), a hypernym, i.e. a more general word from the hyponym tree, is chosen as a qualifier [254]. A qualifier appends additional information to the ambiguous word pointing to the specific context. Reconsidering the hyponym tree in Figure 6.3, we pick the hypernym *request* and create the suggestion *application (request)* as a suggestion. Formally, the hypernym strategy is described as follows:

$$Suggestion_{Hyper} = hyper_{BabelNet}(s'),$$

where $hyper_{BabelNet}(s')$ denotes a function that returns the set of all hypernyms of the word sense $s' \in Senses_D(w, p)$ in the BabelNet dictionary as given by Definition 4.7.

The third strategy also implements the qualifier concept. In particular, it aims at identifying qualifiers in dictionaries and glosses of the respective word. The approach picks the word that is most similar to the original word in order to automatically select an appropriate candidate. To measure the closeness between the words, we employ the Lin metric [253] as is it has been shown to best correlate with the human judgments. For the word *application*, we can retrieve words such as *employment*, *patent*, *job*, or *admission*. According to the Lin measure, the word *patent* has the highest similarity score which subsequently leads to the suggestion *application (patent)*. Formally, this strategy can be described as follows:

$$Suggestion_{Lin} = \{w| \underset{w \in gloss_D(w',s')}{argmax} \{sim_{Lin}(w', w)\}\},$$

where $gloss_D(w', s')$ denotes a function that retrieves all gloss words of the original word w' and the word sense s' in a dictionary D as well as $sim_{Lin}(w_1, w_2)$ denoting a function that calculates the similarity score of two words arbitrary words $w_1, w_2 \in W$.

6.4.2 Synonym Resolution

The synonym resolution strategy is motivated by the research of Plachouras et al. [335] and Rijsbergen [364], who measured the generality of a search query in an information retrieval system. The authors argue that the generality of a search query is determined by the number of documents that are retrieved by this query. We adapt this concept in the following way. The search query is interpreted as a word that is looked up in the BabelNet database, while the query results represent the word senses. Accordingly, the more synsets the query word retrieves, the more general it is. The rationale is that a general word can be used in several contexts with a broader set of meanings. The approach determines the word with the minimal number of word senses in order to finally suggest an alternative word for replacement. For example, consider the synonyms *sales* and *revenue* from Fig. 6.1. The approach retrieves four senses for sales and six senses for revenue. Accordingly, the word *sales* is chosen as a suggestion to resolve the synonym conflict. This strategy is formalized as follows:

$$Suggestion_S = \{w \mid \underset{w \in syn(w')}{min} |Sense_D(w,p)| \},$$

where $syn(w')$ denotes the set of all words that have the same meaning as the original word w'.

After having defined the techniques to automatically detect and refactor homonym and synonym ambiguities, we now evaluate them with regard to real process models, which is the main concern of the next section.

6.5 Evaluation

The evaluation challenges the presented techniques against real-world process models in order to demonstrate their capabilities. More specifically, three process model collections from different industries and a varying degree of standardization have been used. The overall goal of the evaluation is to learn

Table 6.4: Potential Ambiguity Conflicts within the Test Collections

Characteristic	**SRM**	**TC**	**AIC**
No. of Models	604	803	1,091
No. of Activity Labels	2,432	12,088	8,339
No. of Unique Actions	321	728	1,567
No. of Unique Business Objects	891	3,955	3,268
No. of Potential Homonym Conflicts	748	3000	2441
No. of Potential Synonym Conflicts	174	648	812

whether the techniques can effectively detect homonyms and synonyms and successfully resolve them by suggesting more specific words. To this end, the evaluation investigates different evaluation scenarios. Section 6.5.1 introduces the employed process model collections of the evaluation. Section 6.5.2 discusses the evaluation of the detection algorithms. Finally, Section 6.5.3 elaborates on the evaluation of the resolution algorithms.

6.5.1 Model Repository Demographics

In order to achieve a high external validity, the techniques have been applied to different model collections from industry that vary with regard to specific criteria, such as collection size, modeling notation, degree of standardization, and domain. For the evaluation of the ambiguity detection and resolution techniques, we decided to focus on the activity labels since they a) represent a central construct for many process modeling languages and b) are particularly prone to ambiguity conflicts [285, 238]. Table 6.4 summarizes the main characteristics of these repositories. For further information of these repositories, please refer to Table 5.2. The selected process model collections show notable differences between the number of models and the total number of unique actions and business objects to specify the business process. Apparently, the SRM collection is the smallest collection and employs a controlled set of vocabulary words to define process models. This fact indicates an advanced standardization effort which might be reflected by the number of detected ambiguity conflicts. Opposite to that, the AIC collection is the largest collection using a wide set of words raising the expectation that the number of conflicts will be highest. The TC collection represents an average between these two extremes. Therefore, we consider these collections to be appropriate and representative to evaluate the proposed approaches.

6.5.2 Evaluation of Ambiguity Detection

In this section, we discuss the evaluation of the proposed detection techniques. First, Section 6.5.2.1 explains our evaluation strategy of the presented techniques. Then, Sections 6.5.2.2 and 6.5.2.3 discuss the results of the homonym detection and synonym detection techniques. Furthermore, they provide examples of frequently detected homonyms and synonyms in the model repositories.

6.5.3.1 Evaluation Setup

We asses the performance of the detection algorithms by comparing the performance of a basic approach and an advanced approach. The basic approach uses the computational lexicon WordNet to learn whether or not the respective word is ambiguous. The advanced approach has already been described in the previous sections. In order to quantify the performance of these approaches, we use precision, recall and the f-measure as metrics [21]. Here, the precision value describes the number of correctly detected ambiguity cases divided by the total number of ambiguity cases retrieved by the techniques. The recall reflects the number of correctly detected ambiguity cases divided by the total number of ambiguity cases in the repositories. The f-measure is the harmonic mean between precision and recall.

In order to be able to compute precision and recall for the collections introduced in Section 6.5.1, we require a benchmark with the human interpretations of the comprised terms. However, due to the high number of terms and process models in the test collections, a complete benchmark would require the manual judgment and annotation of 29,438 potential homonym term-model pairs and 80,609,259 potential synonym term-model pairs. It would be extremely difficult and time consuming to annotate all the homonyms and synonyms in the test collection.

To solve this problem, we draw a random sample from the test collections and apply statistical methods in order to make valid propositions for precision and recall. In particular, we apply the following procedure. We notice that each combination of a term and a process model is either ambiguous or not. Hence, repeatedly drawing instances from our test collections and assessing the ambiguity represents a binomial experiment $X \sim B(n,p)$ with a population size of n and the probability of finding an ambiguity p. Assuming a homonym probability of 0.01 and a synonym probability of 0.0001 for our overall test sample, we follow the recommendations by Berger [38] and Brown et al. [58] and apply the Jeffrey interval, which is most suitable for binomial references from large populations with a small probability p. Aiming for a significance level $\alpha = 0.05$ and a margin of error $\epsilon \leq 0.02$, we draw a random sample of 120 term-model pairs for homonyms and 125 term-model pairs for synonyms based on the sample size calculation formula provided by Piegorsch [325]. Additionally, we created two types of samples. The first sample includes term-model pairs from the algorithmic results, while the second one includes term-model pairs from the overall test set. With this strategy, we assess the performance of the algorithmic output and the overall test set for the synonym as well as the homonym detection technique.

Taking into account the literature of WSD evaluation (see for example [194, 362]), we asked six native English speakers to provide us with their

Table 6.5: Detection Results for Homonyms

		Basic Approach	**Advanced Approach**
Algorithmic Sample	No. True Positives	57	57
	No. False Positives	58	47
	No. False Negatives	1	1
	Precision	0.5	**0.55**
	Recall	0.98	**0.98**
	F-Measure	0.66	**0.7**
Complete Sample	No. True Positives	21	9
	No. False Positives	83	8
	No. False Negatives	5	17
	Precision	0.2	**0.53**
	Recall	**0.84**	0.35
	F-Measure	0.32	**0.42**

interpretation of the term-model pairs. We randomly assigned them to the test samples in such a way that each native speaker had to assess one homonym and one synonym sample. Overall, we got three judgments for each sample of term-model pairs. Each participant was provided with a target word and a process models in which this word was highlighted. For homonyms, we randomly selected four word senses and asked participants whether the target word could be used with the respective sense a) in general and b) in the context of the particular process model. For synonyms, we analogously selected alternative terms and asked users whether a replacement of the target term by an alternative term is meaningful a) in general and b) in the context of the particular process model. The participants were asked to provide feedback for each question on a 4-point-Likert-scale from *Strongly agree* to *Strongly disagree*. By using a 4-point-scale, we intentionally forced participants to make a final decision, which is necessary for the calculation of precision and recall. For the evaluation, we considered only the answers that relate to the particular process model and calculated the average of these which marked the term as ambiguous or not. Appendix B.1 and B.2 provide examples on these term-model pairs and the judgments required from the native speakers.

6.5.3.2 Homonym Detection Results

The results of the homonym detection approach are summarized in Table 6.5. The results show that the advanced approach works satisfactory in comparison to the basic approach. The advanced approach achieves a precision of 53% to 55%, while the basic approach is much more unstable and retrieves a considerable high number of false positives (precision between 20% and 50%). In contrast however, the recall scores of the basic approach ranges between

Table 6.6: Top 5 of Homonym Actions and Business Objects Ordered by Frequency

		Word	F	Word Sense
Actions	1	process	191	Perform operations to obtain required information Officially deliver
	2	create	175	Manufacturing a man-made product Causing something (create a furor)
	3	check	171	Be careful or certain to do something Hand over something to somebody
	4	review	111	Hold a review Appraise critically
	5	receive	65	To come into possession of To convert into sounds or pictures
Business Objects	1	incident	39	A single distinct event A disturbance
	2	request	38	The verbal act of requesting or asking A formal message postulating something
	3	case	27	Someone who is an object of investigation A portable container for carrying several objects
	4	notification	26	A request for payment Informing by words
	5	application	22	A computer program A verbal or written request for employment

84% and 98% in the samples. This observation relates directly to the low number of false negatives detected by the basic approach (1 in the precision sample and 5 in the recall sample) which implies that most of the ambiguous words have been detected and presented to the user. If we take the f-measure into account, we observe again a higher performance of the advanced approach. The f-measure amounts to 42% in the recall sample and 70% in the precision sample in contrast to 32% and 66%.

We further reflect upon the results by including the nature of the term-model samples into our consideration. In the algorithmic sample, we have chosen from those term-model pairs that have been retrieved by the algorithm. In this way, this sample included pairs with a fairly good chance of being ambiguous. In such a setting, each approach is equally capable of detecting a considerable number of ambiguities that are also relevant for the user. When looking at the details, we observe that the advanced approach is capable of reducing the number of false positives and keeping the relevant terms at the same time. In the complete sample, we randomly selected term-model

Table 6.7: Detection Results for Synonyms

		Basic Approach	**Advanced Approach**
Algorithmic Sample	No. True Positives	37	57
	No. False Positives	63	47
	No. False Negatives	5	1
	Precision	0.29	**0.56**
	Recall	**0.88**	0.58
	F-Measure	0.44	**0.57**
Complete Sample	No. True Positives	0	0
	No. False Positives	0	0
	No. False Negatives	2	2
	Precision	N/A	N/A
	Recall	0	0
	F-Measure	N/A	N/A

pairs from all three process model collections and also included terms that may remain undetected by the algorithm. In this sample, we observed a low precision and a high recall for the basic approach. In contrast, the results of the advanced approach are more balanced with a moderate precision. However, we observe a surprisingly low recall at the same time. The details revealed that our participants spotted specific word senses to be similar to each other and thus detected an ambiguous term. For example, the participants agreed that the term *link* in the activity *link pairs order to case* complies to the word senses *to be or to become joined or united* as well as *to make a logical or causal connection*, while the algorithm agrees only on the first word sense. Despite this, we still consider the algorithm to be superior to the basic approach and to retrieve more meaningful candidates to the users.

In addition to the quantitative analysis, we discuss qualitative results of the detection approach. Table 6.6 presents the top five homonym actions and business objects among all test collections. The most frequent homonym action is given by the verb *to process*. Our approach found out that this word is used with two particular word senses. First, the word is used as a general expression to perform a set of operations in order to create a required output or information. Second, it also identifies the activity of officially delivering a legal notice or summons. For the business objects, the technique detected the noun *incident* as the most frequent homonym. In this case, it is unclear if the incident refers to a single event or a disturbance, for example in IT applications. These words are good examples of homonyms that are frequently used and should rather be avoided.

6.5.3.3 Synonym Detection Results

The results of the synonym detection technique are summarized in Table 6.7. Similarly to the homonym detection approach, the results show that the advanced approach of this paper exceeds the capabilities of the basic approach. In terms of precision, the basic approach achieves 29% while the advanced approach reaches at least 56%. In terms of recall, the basic approach appears to outperform the advanced approach (88% compared to 58%). As discussed earlier, the high recall value is the result of the extensive number of retrieved instances that are produced by the basic approach. Taking the f-measure into account, we observe that the advanced approach is more balanced (57% in contrast to 44%).

We also discuss the results with regard to the samples of the user evaluation. In the algorithmic sample, the advanced approach dominates the capabilities of the basic approach. It does not only reduce the number of false positives by a significant amount, but it also keeps a large share of those synonym pairs that are relevant for the user. In the complete sample, the participants have detected two synonyms which have not been found by the basic or the advanced approach. Accordingly, it is not possible to calculate any values for precision or the f-measure. Interestingly, the users found the term pairs *(fill out, complete and submit)* as well *(document, register and communicate)*. In these cases, the participants interpreted the combination of two distinct verbs as being part of one single verb (troponymy), which was then considered to be a synonym. Since neither of these two approaches can detect the combination of verbs correctly, they do not detect a synonym in these cases. Overall, we consider the advanced approach to be superior to the basic one to retrieve the relevant cases for the user.

In addition to the quantitative perspective, it is again interesting to investigate qualitative examples. Table 6.8 gives an overview of the top five synonymous actions and business objects. It is interesting to note that the approach detected a group of three pairwise synonym words. The words *make*, *create*, and *produce* all refer to the activity of manufacturing products. For the business objects, the technique detects *customer*, *market* and *client* as synonyms for persons that pay for goods and services. Another example are the synonyms *purchase order* and *order* which refer to a commercial delivery document. Overall, these examples also illustrate the capability of our approach to detect synonyms that should be avoided.

6.5.3 Evaluation of Resolution

In this section, we discuss the evaluation of the resolution strategies. First, Section 6.5.3.1 elaborates on our evaluation setup. Then, Section 6.5.3.2 presents the results for the homonym resolution. Finally, Section 6.5.3.3 discusses the results of the synonym resolution.

Table 6.8: Top 5 of Synonym Actions and Business Objects Ordered by Frequency

		Word	**F**	**Word Sense**
Actions	1	check, control	173	Be careful or certain to do something
	2	create, produce	157	Manufacturing a man-made product
	3	post, send	142	Cause to be directed or transmitted to another place
	4	make, create	135	Manufacturing a man-made product
	5	survey, review	115	Hold a review
Business Object	1	customer, market	119	Someone who pays for goods or services
	2	customer, customer account	118	Someone who pays for goods or services
	3	purchase order, order	92	A document to request someone to supply something
	4	account, invoice	72	A statement of money owed for goods or services
	5	customer, client	36	Someone who pays for goods or services

6.5.4.1 Evaluation Setup

We assess the performance of the resolution strategies by comparing the degree of ambiguity before and after applying it to the test collections. Therefore, we define metrics that measure the basic characteristics of homonyms or synonyms respectively in order to operationalize the effects of ambiguity. Since a homonym represents a word referring to multiple senses, we measure the degree of homonymy by using the number of senses per word. Considering a set of process models $\mathcal{P}$ and differentiating between action and business object, the number of senses per word for actions (SpW_A) and for business objects (SpW_{BO}) is calculated as follows:

$$SpW_A = \sum_{P \in \mathcal{P}} \frac{|\bigcup_{a \in A_\lambda^P} \{s_i \mid s_i \in Senses_D(\alpha_A(\lambda(a)), v)\}|}{|\{\alpha(\lambda(a)) \mid a \in A_\lambda^P\}|}$$

$$SpW_{BO} = \sum_{P \in \mathcal{P}} \frac{|\bigcup_{a \in A_\lambda^P} \{s_i \mid s_i \in Senses_D(\beta_A(\lambda(a)), v)\}|}{|\{\beta(\lambda(a)) \mid a \in A_\lambda^P\}|}$$

For the whole repository the senses per word metric is calculated as follows:

$$SpW = SpW_A + SpW_{BO}$$

Table 6.9: Results of Homonym Resolution

		SRM	**TC**	**AIC**
SpW^P	Before Resolution	8.35	10.58	8.98
	After Resolution	1.71	1.73	2.94
	Reduction Effect	387.11%	512.48%	205.69%
SpW^P_A	Before Resolution	6.02	6.95	6.66
	After Resolution	1.83	1.47	1.54
	Reduction Effect	228.92%	371.92%	332.45%
SpW^P_{BO}	Before Resolution	8.14	11.06	9.46
	After Resolution	1.66	1.76	3.23
	Reduction Effect	389.11%	528.11%	193.22%

For the degree of synonymy, we calculate the number of words for each word sense. Given a process model repository $\mathcal{P}$ and the distinction between actions and business objects, the degree of synonymy is defined as follows:

$$WpS_A = \sum_{P \in \mathcal{P}} \frac{|\{\alpha_A(a) \mid a \in A^{\mathrm{P}}_{\lambda}\}|}{|\bigcup_{a \in A^{\mathrm{P}}_{\lambda}} \{s_i \mid s_i \in Senses_D(\alpha_A(\lambda(a)), v)\}|}$$

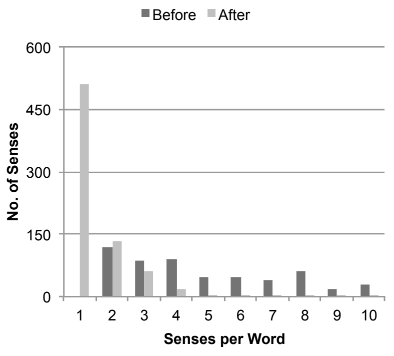
$$WpS_{BO} = \sum_{P \in \mathcal{P}} \frac{|\{\beta_A(a) \mid a \in A^{\mathrm{P}}_{\lambda}\}|}{|\bigcup_{a \in A^{\mathrm{P}}_{\lambda}} \{s_i \mid s_i \in Senses_D(\beta_A(\lambda(a)), v)\}|}$$

The number of words per sense for the whole process model repository is then calculated as follows:

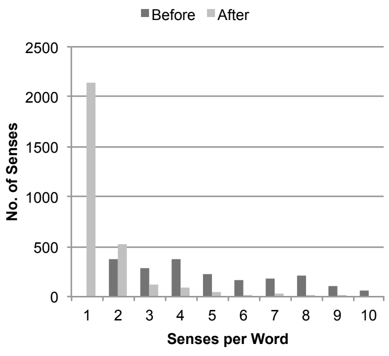
$$WpS^P = WpS_A + WpS_{BO}$$

6.5.4.2 Homonym Resolution Results

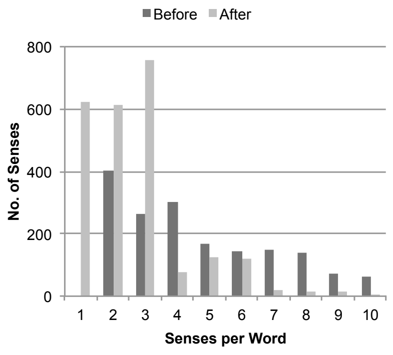
Table 6.9 summarizes the results for the introduced metrics. For the calculation, only those words have been considered that have at least two word senses. The table illustrates the significant effect of the homonym resolution technique for both, actions and business objects. We observe the biggest effect for the TC collection, in particular for the business objects. In the TC collection, the number of senses per action is reduced from 6.95 to 1.47 (Ambiguity Reduction Effect: 371.92%) and the number of senses per business object is reduced from 11.58 to 1.76 (Ambiguity Reduction Effect: 528.11%). Similar results can be observed for the SRM collection. The results also show that the AIC collection has the most problems with ambiguous terminology because after



(a) SRM



(b) TC



(c) AIC

Fig. 6.4: Senses per Word Distribution before and after Homonym Resolution

the application of the resolution the average degree of homonymy still amounts to 2.94. This is due to the extensive ambiguity caused by the business objects. According to the results, each business object may refer to three different word senses on average which confirms the initial assumption of the AIC collection to have the most issues with ambiguity.

Complementary, Figure 6.4 illustrates the homonym reduction effect by showing the senses per word distribution for all collections. The distribution figures demonstrate that the technique reduces the number of words with multiple senses to a notable extent. As a result of its application, almost all words with more than five senses are replaced. Consequently, the technique also significantly increases the number of words with a single sense. Hence, the degree of homonymy is considerably reduced in the collections. As an exception, we already mentioned the AIC collection. Although the number of words with one single sense has been increased, there is still a big number of

Table 6.10: Resolution Suggestions for the Top 5 Homonym Actions and Business Objects

		Homonym	**Word Sense**	**Suggestion$_{\mathbf{Hypo}}$**	**Suggestion$_{\mathbf{Hyper}}$**	**Suggestion$_{\mathbf{Lin}}$**
Actions	1	process	Perform operations	N/A	process (calculate)	process (process)
			Officially deliver	wash	process (deliver)	process (process)
	2	create	Manufacturing	overproduce	N/A	create (produce)
			Causing something	blast	N/A	create (create)
	3	check	Being certain to do sth	proofread	check (verify)	check (control)
			Handing over	N/A	check (consign)	check (check)
	4	review	Hold a review	N/A	review (inspect)	review (survey)
			Appraise critically	referee	review (evaluate)	review (critique)
	5	receive	Coming into possession	inherit	receive (get)	receive (receive)
			Converting	N/A	receive (convert)	receive (receive)
Business Objects	1	incident	A distinct event	cause celebre	incident (happening)	incident (incident)
			A disturbance	N/A	incident (disturbance)	incident (incident)
	2	request	Requesting or asking	call	request (speech act)	request (asking)
			A formal message	solicitation	request (message)	request (petition)
	3	case	Investigation subject	N/A	case (person)	case (subject)
			A portable container	gun case	case (container)	case (case)
	4	notification	A request for payment	N/A	notification (request)	notification (notice)
			Informing by words	warning	notification (informing)	notification (appraisal)
	5	application	A computer program	browser	application (program)	application (program)
			Request f. employment	credit application	application (request)	application (patent)

Table 6.11: Results of Synonym Resolution

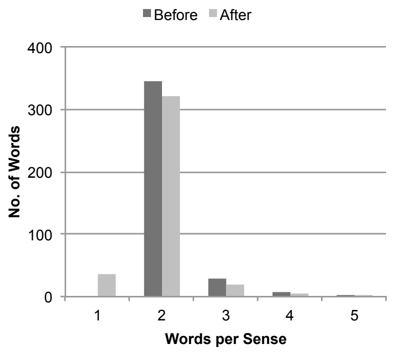
		SRM	**TC**	**AI**
WpS^P	Before Resolution	2.13	3.50	2.23
	After Resolution	1.98	3.24	2.06
	Reduction Effect	7.35%	7.61%	8.29%
WpS_A^P	Before Resolution	2.16	2.24	2.25
	After Resolution	1.82	2.03	2.01
	Reduction Effect	18.27%	10.54%	11.65%
WpS_{BO}^P	Before Resolution	2.12	2.19	2.22
	After Resolution	2.05	2.07	2.09
	Reduction Effect	3.54%	5.93%	6.36%

words having more than one senses. Note that not every word having multiple senses is necessarily ambiguous. Many words are associated with senses that are a closely related. Hence, the technique will never result in a sense distribution in which every word points to one single word sense.

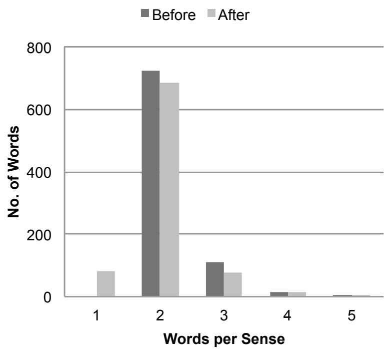
In addition to the quantitative results, we provide resolution examples for each of the Top 5 homonyms (see Table 6.10). For the sake of readability, we only list the first hyponym that is suggested. In general, the resolution strategies are capable of creating a decent number of suggestions to resolve the ambiguous words. However, we also note differences in the performance of these strategies. For the hyponym strategy, we note that it fails to retrieve hyponyms in eight cases due to the fact that there are simply no hyponyms existing. Moreover, we observe that the suggestions are quite specific for most of the cases and might miss the originally intended word senses. The hypernym strategy fails in only two cases. Moreover, if we compare the word sense and the suggestion, we more likely tend to accept the suggestion as it best describes the intended word sense. The Lin strategy is always capable to create a suggestion. However, we also notice that it is not capable of finding a suitable solution in nine cases since it proposes the same word as the original one. This typically occurs when each gloss word has a similarity score of 0. Nevertheless, we can consider the remaining Lin suggestions as a good complement when combining it with the other strategies.

6.5.4.3 Synonym Resolution Results

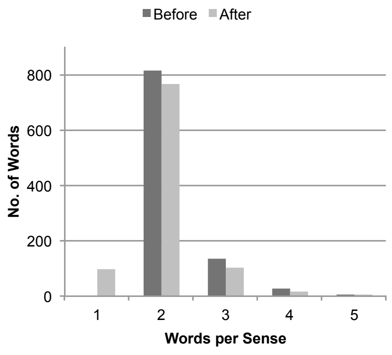
Table 6.11 summarizes the results for the synonym resolution metrics. Again, the calculation excluded those word senses that only have one corresponding word. From the data, we learn that the synonym resolution technique decreases the degree of synonymy for all three collections. The biggest effect can be observed for the AIC collection. In this collection, the WpS_A^P is reduced from 2.25 to 2.01 (Reduction Effect: 11.65%) and the WpS_{BO}^P is reduced from 2.22



(a) SRM



(b) TC



(c) AIC

Fig. 6.5: Word per Sense Distribution before and after Synonym Resolution

to 2.08 (Reduction Effect: 6.36%). While we yield comparable results for the TC collection, the effect for the SRM collection is significantly smaller in absolute numbers. In the SRM collection, the WpS_A^P metric deceases from 2.16 to 1.82 and the WpS_{BO}^P decreases from 2.12 to 2.05. These differences are the result of the lower ex-ante degree of synonymy within the SRM collection. The, in general, small absolute differences can be explained by the fact that many words do not have synonyms, i.e., their words per sense value is already one. Those words having synonyms typically have one or two. Hence, the effect of resolving a synonym results in a rather small difference of the metric. However, the numbers still show the positive effect of the synonymy resolution technique.

Figure 6.5 visualizes the results by showing the words per sense distribution. It confirms the explanation for the small absolute values. Only a few senses are represented by more than one word. Moreover, the figures also show a constant reduction of words referring to several word senses across all test collections.

Table 6.12: Top 5 of Synonymous Actions and Business Objects

		Word	**Word Sense**	**Suggestions**
Actions	1	check, control	Being certain to do sth	insure
	2	create, produce	Manufacturing	create
	3	post, send	Transmitting to another place	send
	4	make, create	Manufacturing	create
	5	survey, review	Hold a review	go over
Business Objects	1	customer, market	Someone who pays for goods	customer
	2	customer, customer account	Someone who pays for goods	customer
	3	purchase order, order	A supply request document	purchase order
	4	account, invoice	A statement of owed money	invoice
	5	customer, client	Someone who pays for goods	customer

Consequently, the synonym resolution technique increases the number of those senses to which only one word refers. As a result, it reduces the degree of synonymy in the collections. Again note that not every instance of synonymy is replaced since the respective words are not necessarily ambiguous.

We also discuss the qualitative results of the synonym resolution strategy. Similar to the homonym case, we create the resolution suggestions for the Top 5 synonymous actions and business objects as depicted in Table 6.12. The table shows on the one hand that the strategy is always capable of creating a suggestion based on the specificity of a word. On the other hand, we can distinguish two cases. First, the strategy can suggest a new alternative that comprises two words, such as the verb *go over* for the synonym actions *to survey* and *to review*. Second, it can select the most unambiguous alternative among two words, as for example for the business objects *invoice* and *account*. In general, we could not find counter-intuitive suggestions which lets us conclude that the specificity strategy fulfills its purpose.

6.6 Discussion

This section discusses results, limitations, and implications of the techniques and their evaluation. Section 6.6.1 summarizes the results of the evaluation. Section 6.6.2 reflects upon limitations of the ambiguity detection and resolution approach. Finally, Section 6.6.3 and 6.6.4 identify implications of the proposed approach for research and for practice.

6.6.1 Summary of Results

The evaluation of the presented algorithms reveals satisfactory results with regard to their efficacy. Table 6.13 provides a short summary of the main

Table 6.13: Summary of Results for Ambiguity Detection and Resolution

		Detection			Resolution
Homonyms	Min	P:0.53	R:0.35	F: 0.42	RE: 205.69%
	Avg.	P:0.54	R:0.67	F: 0.56	RE: 368.43%
	Max	P:0.55	R:0.98	F: 0.7	RE: 512.48%
Synonyms	Min	P:0	R:0	F: N/A	RE: 7.35%
	Avg.	P:0.28	R:0.29	F: N/A	RE: 7.75%
	Max	P:0.56	R:0.58	F: 0.57	RE: 8.29%

P:Precision, R: Recall, F: F-measure, RE: Resolution Effect

indicators for each technique. In case of homonyms, the detection approach achieves a stable f-measure of around 0.56 proving its goal of finding relevant homonyms. Regarding the resolution, the evaluation shows that the overall degree of homonymy could be reduced by the factor 3.5 on average, which resembles a significant reduction of homonym ambiguity in the test collections.

The results for the synonym detection are also stable in the sample taken from the algorithmic results (f-measure: 0.57). However, the approach was not capable to detect the synonyms in the sample from the complete test set. In this sample, the users spotted two synonyms, i.e. *(fill out, complete and submit)* as well as *(document, register and communicate)*. A possible explanation for this synonymy might be that the verbs *to complete* and *to submit* build a troponym of the verb *to fill out* such that the troponym shares a similar meaning with the combination of the two verbs. These instances are not detected by the approach since it does not consider troponym relations to form synonyms. Turning to the resolution results, the average resolution effect amounts to 7.75%. Since the synonym relation between two words is considerably rare, the effect of the resolution approach does not achieve large numbers. Nevertheless, the approach was capable to unify the terminology in the test collections and to reduce overlapping terminology to some extent.

6.6.2 Limitations

The findings from the evaluation are subject to some limitations. In particular, we discuss the internal validity, the conclusion validity, and the technology dependency.

Internal validity reflects the extent to which a causal conclusion based on a study is warranted. In case of the evaluation, the conclusion is based on the interpretations of the native speakers who perceive a label to be ambiguous or not. Their perception, however, might be influenced by instrumentation and maturation bias [441, p. 106-107]. *Instrumentation bias* may cause negative effects if the evaluation instrument is designed badly. In this particular case, the participants might put less effort in the evaluation task, which might negatively affect the outcome. We tried to mitigate this threat by carefully setting up the evaluation design and conducting several feedback loops with regard to the visual design and representation of the evaluation. *Maturation bias* may influence the perception and judgment of participants when time passes. For example, the participants might get tired or bored leading to less efforts for the evaluation. Due to the large number of scenarios which were presented to the participants, we implemented the evaluation in such a way that the participants had the possibility to save and quit at any time and resume later. In this way, we tried to minimize the maturation bias as far as possible.

Conclusion validity is concerned with issues that affect the ability to draw the correct conclusions from the outcome of the evaluation [441, p. 104-105]. The conclusion validity might be affected due to the a) limited number of collections, b) the focus on activity labels, and c) the sampling process. Regarding the *limited number of collections*, the three process model collections can hardly be seen as representative in a statistical sense. Therefore, we cannot completely guarantee that other model collections would yield different results. This risk has been mitigated by selecting process model collections that vary along different dimensions such as degree of standardization, domain, and size. The results of the evaluation suggest that the successful application of our techniques is not limited to a particular set or to a particular quality of process models. Regarding the *focus on activity labels*, the evaluation has been restricted to activity labels only, which raises the question if the inclusion of other labels, such as events or gateways, would also yield different results. The choice of restricting the evaluation to activity labels was motivated by the fact that the concept of an activity is central to any process modeling language [297] and that activity labels are particularly prone to ambiguity [285, 238]. Finally, the *sampling process* involves a statistical sampling of process models of the test collections for the purpose of evaluation. As a consequence, one might argue that the evaluation results are not reliable and that the true performance of the approach might be smaller than reported in the evaluation. In fact, the high number of potential ambiguity conflicts and the necessity of human annotators to evaluate the detected ambiguities (see Table 6.4) required a smaller test sample to calculate meaningful results for both techniques. The sheer number of potential ambiguity conflicts could not be handled comprehensively by humans and would have resulted in an extensively time-consuming task. We decided for a statistical sampling and selected suitable parameters for the determination of our sample size in order to overcome this restriction. Moreover, we created two different types of evaluation samples for each technique which assess different aspects of the techniques. Considering the restrictions on the evaluation, the evaluation strategy is appropriate for such a setting and reveals meaningful insights on the performance of the presented techniques.

The *technology dependency* refers to the fact that the presented techniques rely on the availability on language processing technologies of the respective language. On the one hand, the techniques require a computational lexicon enriched with the word senses as well as a suitable WSD algorithm. If these technologies are missing, there is hardly any chance to apply these techniques on process models. On the other hand, the field of natural language processing has brought forth a plethora of techniques that are also capable to deal with different languages. With regard to these techniques, it is to mention that the recent version of the BabelNet repository covers knowledge from more than 270 languages. Moreover, the integrated WSD algorithm, which makes use of the BabelNet lexicon, can perform sense disambiguation in different languages [304]. Therefore, we do not consider the technology as a limiting factor.

6.6.3 Implications for Research

One of the contributions of the presented ambiguity detection approach is a new direction to manage lexical ambiguity. This approach employs available technology from the field of natural language processing to formalize phenomena of lexical ambiguity, namely homonyms and synonyms. The combination of WSD technology with existing concepts of ambiguity formalization enables the adequate integration of contextual information and the meaningful distinction between truly and potentially ambiguous words. The usefulness of this integration has been proven in the evaluation leading to a high precision value. Although prior approaches also formalize such characteristics, e.g. by using linguistic patterns [97, 142] or metrics [125, 71, 429], they retrieve an extensive number of potential ambiguities that, however, might not be perceived as ambiguity by humans and thus evaluate to a small precision value. The presented approach balances both metrics as it is capable of finding those ambiguities that are truly harmful and show these to the users.

Regarding the resolution of ambiguous terminology, we argue that this task is best done by humans since there is currently no technique available that is capable to cope with the natural language understanding of humans and resolve ambiguities automatically [300]. As a consequence, it is often more beneficial to support humans with this task. Inline with this, another contribution of this technique are several strategies that use the available information from the process model and create a set of suggestions based on linguistic knowledge sources. Thus, the approach extents the capabilities of prior research which point to and explain specific ambiguities [71, 142] to the level of interactively selecting among several alternatives [158].

Finally, the presented techniques are not restricted to process models. For example, conceptual models are frequently used in requirements engineering to specify desired behavior and capabilities of the later system [66]. Similarly, these models have to be consistent and correct with regard to the textual fragments in order to avoid risks for the software development project [9]. Indeed, goal models [367, 333], feature diagrams [233], or use case diagrams [81] also use natural language fragments in the same fashion as process models. The proposed techniques thus provide means to support the software and requirements engineering process by automatically detecting ambiguous terminology in formal requirements documents and by automatically proposing more accurate terms. As a result, misconceptions of requirements are avoided and the likelihood of a successful completion of the software engineering initiative is increased.

6.6.4 Implications for Practice

The results of this paper have considerable implications for practice. Most importantly, the proposed techniques can be integrated into commercial modeling tools. In such a context, it can point modelers to ambiguous words and automatically generate more accurate alternatives among which the modeler may choose. In contrast to prior approaches, the approach provides advanced capabilities to distinguish between confirmed and rejected ambiguities and only notifies the user in case of a serious conflict with existing terminology. Overall, linguistic quality issues can be avoided right from the start.

For already existing model repositories, the presented techniques can help users in cleaning up terminology and develop a consistent domain vocabulary or glossary for a particular company. Given the size of model repositories in practice with thousands of models [369], the presented approach makes an important contribution to the effective and efficient management of terminology and its use in several process models. This is particularly useful when multiple modelers create models concurrently.

6.7 Summary

In this chapter, we have addressed the problem of automatically detecting and refactoring lexical ambiguity in process models. Based on the review of prior research, we have concluded that current techniques cannot be easily transferred to process models as they cannot deal with their specific characteristics. In particular, the shortness of the natural language text labels has imposed a major challenge. In order to adequately address this challenge, this chapter has introduced an approach to formalize lexical ambiguity with the help of semantic vectors initialized by performing WSD on the element labels of process models. Moreover, we have proposed necessary and extended conditions that allow us to automatically detect truly ambiguous homonyms and synonyms. Using the computational lexicon BabelNet, we have further implemented different resolution strategies, which automatically suggest alternative terms for replacement. The performance of the approach has been assessed with native English speakers. The results have shown that the technique detects a significant number of homonyms and synonyms within the test collections. Moreover, the introduced metrics *Sense per Word* and *Word per Sense* have further highlighted the positive effect of the ambiguity resolution technique. As a result, the overall ambiguity has been significantly reduced, which contributes to the consistency and quality of the process models and the process model repository.

7

Refactoring of Pragmatic Ambiguity

This chapter discusses the problem of pragmatic ambiguity that hinder the comprehension and sense-making of process models. So far, there are only limited research contributions that support users in managing this type of ambiguity. Therefore, this chapter proposes an approach to address pragmatic ambiguities in terms of detection and refactoring. Section 7.1 introduces to the problem of pragmatic ambiguities and gives an overview of related approaches. Section 7.2 explains and formalizes the main characteristics of pragmatic ambiguities. Then, Sections 7.3 and 7.4 present the techniques to refactoring pragmatic ambiguities. Sections 7.5 introduces the strategy of the user evaluations and discusses the usefulness of the recommendation-based approach by applying it to process model repositories from industry. Finally, Section 7.6 discusses the results and implications of the evaluation, before Section 7.7 summarizes the main points. The main results of this chapter have been published in [334].

7.1 Pragmatic Ambiguities in Process Models

The increasing adoption of BPM has stimulated the use of process models in different scenarios, such as providing knowledge for action [220] or analyzing and redesigning real-world processes [88]. For that purpose, process models are used by people from different organizational units with different knowledge and background. It is essential that they are comprehended and understood by these people in a consistent way, such that the interpretation leads to an equal implementation of the business process. However, if the process model does not provide sufficient information, the consistent comprehension and interpretation among several actors might be affected by pragmatic ambiguity. There is hardly any research available that addresses the issue of pragmatic ambiguity from a technical perspective. Section 7.1.1 introduces to the notion of pragmatic ambiguity and its consequences in order to understand this issue. Then, Section 7.1.2 discusses related approaches and their shortcomings.

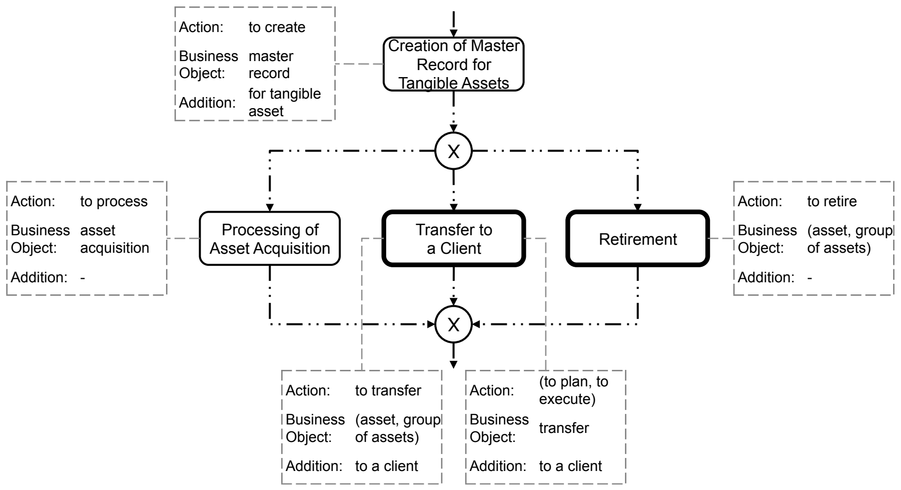

Action: to create
Business Object: master record
Addition: for tangible asset
Creation of Master Record for Tangible Assets
X
Action: to process
Business Object: asset acquisition
Addition: -
Processing of Asset Acquisition
Transfer to a Client
Retirement
Action: to retire
Business Object: (asset, group of assets)
Addition: -
X
Action: to transfer
Business Object: (asset, group of assets)
Addition: to a client
Action: (to plan, to execute)
Business Object: transfer
Addition: to a client

Fig. 7.1: Pragmatic Ambiguities in the Group Retirement Process; Adapted from [190]

7.1.1 The Issue of Pragmatic Ambiguities in Process Models

The studies of pragmatics is concerned with how people understand and communicate more than the literal meaning of words or sentences in a given situation or context. The sentence needs to have a deixis in order to correctly understand the meaning of a sentence within a context. As previously discussed, the deixis refers to a specific point to which the sentence has been issued. This point can describe a place, an object, or an action [43, p. 152]. If the deictic center cannot be determined clearly, it is hard to correctly understand the sentence and make sense out of it. This phenomenon is known as *pragmatic ambiguity*.

We face pragmatic ambiguity when a sentences misses the deictic center and when it allows for several meanings in the context in which the sentence is uttered. Similar to semantics, the context is given by the natural language text surrounding the sentence as well as the context beyond language, such as the situation or the background knowledge [40, 39, 198]. Depending on the flaws of sentence and context, we distinguish between deictic ambiguities and referential ambiguities. Deictic ambiguity emerges when the context is associated with several reference points of interpretation. Referential ambiguity occurs when an anaphor, e.g. it or they, refers to more than one element. In the following, we pay specific attention to deictic ambiguities.

These deictic ambiguities can nicely be illustrated by the example process model. Figure 7.1 depicts the *Group Retirement* process of the SAP reference model repository [190]. The process begins with the creation of a master record for the respective asset. Depending on the result, the process continues with either processing the related asset acquisition, the transfer to the client, or the retirement. If one of these activities has been conducted, we assume the process to finish.

Figure 7.1 also highlights the label components and two bold-edged activities that suffer from pragmatic ambiguity. These activities do not provide sufficient information to point to a clear deictic center, since specific label components are missing. The activity *Transfer to a client* informs the reader about a transfer that needs to be conducted to complete the process. However, it is unclear if a particular object, e.g. an asset or a group of several assets, needs to be transferred to the client or if a transfer itself needs to be planned, executed or put into action. Depending on the situation, it would be more consistent to either name the task *Transfer of Asset to a Client* or *Transfer Processing to a Client*, since the activities then specify a deictic center and resolve the pragmatic ambiguity. Similarly, the task *Retirement* does not specify which object needs to be retired. Based on the context, this might either be a single asset or a group. Hence, the process model refers to several reference points regarding its pragmatic interpretation.

The presence of pragmatic ambiguities does not only impact the model reader's ability to make sense of the model, but also has implications for the usage of process models for other scenarios, such as system design and analysis, process model compliance, or for process analysis and simulation. In the context of *system design and analysis* process models are helpful to document relevant parts of the organization and its systems [88] and to implement them in a process-oriented way [109]. However, if a process model and elements thereof are insufficiently specified with regard to the components, system developers have only limited information about the object and its properties and methods. For example, the task *Transfer to a client* from Figure 7.1 might either point to a technical transaction of digital object information or a physical transport of goods to a client. Depending on the situation, the technical implementation may involve completely different approaches and technologies and may not meet all requirements or cost expectations at the end of the project [46, 422]. In order to avoid this situation, additional feedback loops between analysts and developers are necessary, which will finally delay implementation and deployment [144, 24].

In the context of *process model compliance*, process models are used to prove that business processes and operations are in accordance with a given legislature [376, 140]. In such a setting, process models need to specify all necessary information such that the compliance may be confirmed in internal audits. However, if the process model has insufficiently specified model elements, the interpretation of auditors may deviate from the one of the company and raise additional questions about the processes of the company. For example, it might not be obvious to the auditor that the task *Transfer to a client* refers to a transfer of assets to a client. As a result, he might conclude that specific compliance requirements may not be fulfilled in the underlying business process. Ultimately, the process model might not be used to prove the implementation of compliance requirements and lead to a negative outcome of the audit process.

In a *process analysis and simulation* scenario, process model elements need to be enriched by additional information, such as activity costs, resources, roles, or time constraints. Similar to the aforementioned case, incomplete model elements may hinder the application of process analysis and simulation techniques as the additional information does not fit to the real operations. For example, considering the task *Transfer to a client* from Figure 7.1, there will be a notable difference between the execution times of retiring a single asset or an entire group of assets. If this inconsistency is not corrected, the results of analysis or simulation approaches do not reflect the current situation and may lead to wrong business decisions [24].

Against this background, it is necessary to appropriately manage pragmatic issues in process models. The next section will discuss prior research approaches to address these issues.

7.1.2 Managing Pragmatic Ambiguities in Process Models

We already discussed that pragmatic ambiguities arise if the process model can be interpreted from different angles and if these angles are valid. The different interpretations are the result of imprecise information pointing to several deictic centers. Relating to the cooperative principle [147, p. 45], there are two symptoms causing pragmatic ambiguity, i.e. process model elements that do not describe specific information at all (incompleteness) or that describe this information rather general (underspecificity). In order to manage the two cases, prior research has proposed several approaches which are summarized in Table 7.1. In general, we distinguish between approaches that focus on detection of underspecificity as well as on detection and refactoring of incompleteness.

Concerning the *detection of underspecificity*, Friedrich [135] quantifies the specificity of a text label in terms of depth within the WordNet hyponym tree. The deeper the word is located in the hyponym tree, the higher its specificity. The metric thus serves as an indicator to maintain a consistent level of detail in process models. Similarly, Leopold et al. [246] propose and evaluate a set of syntactic and semantic metrics that measure the granularity of single process models. The results suggest that these metrics are suitable to quantify the granularity of a process model within a process hierarchy or process architecture. Koschmider and Blanchard [208] develop a semi-automatic approach to detect non-uniformly specified process elements. The approach analyzes process element names with regard to the overall level of detail within the process model and highlights deviating elements. Finally, it is the user's responsibility to align these deficient elements. Gleich et al. [142] identify a set of lexical and syntactic language patterns that also point to pragmatic ambiguities. The authors operationalize these patterns by using regular expressions and POS tagging. For example, expressions, such as *many* or *few*, point to requirements that need further specification. Fabbrini et al. [119] define specific metrics that quantify characteristics of pragmatic ambiguity. Among others, they provide measures for underspecification and vagueness, which is reflected by

Table 7.1: Approaches for Managing Pragmatic Ambiguity

Author	Approach
Detection of Underspecificity	
Friedrich [135]	Metrics for the Specificity of Labels
Leopold et al. [246]	Label Granularity Measurement with Language Analysis
Koschmider and Blanchard [208]	Identification of Non-Uniformly Specified Elements
Gleich et al. [142]	Natural Language Patterns of Underspecification
Fabbrini et al. [119]	Measurement for Underspecification and Vagueness
Ferrari et al. [126]	Graph-based Similarity Calculation between Related Documents and Target Document
Ferrari et al. [127]	Graph-based Comparison between Interpretations of the Reader and the Document
Detection of Incompleteness	
Denger et al. [97]	Natural Language Templates for Describing Requirements
Kaiya and Saeki [185]	Linking Requirements Documents with Ontologies from the Application Domain
Siegmund et al. [398]	Verifying Requirements with Ontology Reasoning
Kösters et al. [214]	Two level verification approach for incompleteness in Use Case Models
Refactoring of Incompleteness	
Betz et al. [42]	Recommending Process Model Fragments with Similarity Metrics
Bobek et al. [45]	Recommending Model Fragments with Bayesian Inferences
Koschmider et al. [209, 210]	Recommending Model Fragments based on Frequency, Quality and Cost Criteria
Leopold et al. [243, 244]	Generating Abstract Model Element Names

the number of sentences containing words without modifier or the number of sentences containing vague expressions like *good* or *efficient*. Ferrari and Gnesi [126] as well as Ferrari et al. [127] employ a graph-based approach to detect pragmatic ambiguities in written documents. In a prior approach, the authors use related documents to detect pragmatic ambiguities in a target requirements document. To this end, the authors propose an algorithm that takes the concepts expressed in sentences, transforms them into a graph, and searches for corresponding concepts in the related documents. The paths resulting from the traversal of each graph are compared and, if their overall similarity score is lower than a given threshold, the requirements specification sentence is considered ambiguous. In a subsequent approach, the authors apply this strategy with the background knowledge of different readers [127].

With regard to the *detection of incompleteness*, Kösters et al. [214] introduce a two-level verification approach for UML use case models. The approach transforms these models into activity graphs, i.e. simplified process models, which are further enriched by attributes from class models. Based on that, the verification approach first investigates the formal parts of the use case model (e.g. pre- and post-conditions), before it checks for incomplete, inconsistent, or ambiguous specifications. Denger et al. [97] propose natural language templates in order to specify requirements. For example, if a requirement involves the reaction of the system to an event, the template demands the specification of a conjunction (If), an actor (the system), an action (receive), and an object (the signal). Kaiya and Saeki [185] use an ontology-based approach for the detection of incompleteness. The authors develop an ontology system that combines a domain thesaurus and inference rules. Hence, it is possible to analyze requirements specifications with respect to the semantics of the application domain and to identify incomplete statements in the requirements. Similarly, Siegmund et al. [398] also use ontology and reasoning technology to capture and validate requirements. They combine ontology consistency checking and rule-driven completeness tests. For example, one completeness rule demands that each functional requirement must specify at least one property, otherwise it is incomplete.

Concerning the *refactoring of incompleteness*, several researchers have proposed recommendation-based approaches to support modelers in closing gaps during process modeling. Betz et al. [42] use a combination of several similarity measures to find a suitable process model for recommendation. In particular, they utilize the string-edit distance metric by Levenshtein [250], a semantic similarity measure based on WordNet, and a similarity measure of the structural aspect of process models. Bobek et al. [45] train a Bayesian net based on existing configurable process models in order to recommend several alternatives on how to proceed. By conducting inferences on the Bayesian net with the process model, which has been created so far, the approach recommends the most probable elements or model fragments. Moreover, prior research also proposed modeling editors with recommendation features. Koschmider et al. [209, 210] developed a modeling editor with a recommender component for process model parts stored in a repository. Based on a user profile, the recommendation component looks for related process models and ranks them according to mandatory and optional criteria. The mandatory criteria include a modified version of the term and document frequency measure, the number of actual reuse cases for the respective model, and the number of change operations. Optional criteria include structural correctness, as well as the cost and quality of the process design. Leopold et al. [243, 244] propose a technique to suggest general names for process models. The technique implements several naming strategies for individual model fragments or process models which are build on linguistic features, such as dominating objects or conjunctions.

The overview of prior research revealed that the detection of underspecificity and incompleteness issues is sufficiently covered. Although incompleteness is a special case that is subsumed by underspecificity, available approaches help to detect such instances in both, text documents and process models. Afterwards, it is up to the users to resolve these inconsistencies to the best of their knowledge. Still, the resolution may turn out to be difficult because the process model itself only gives a limited amount of context information [249]. In this situation, the discussed recommendation approaches are of limited use. For example, the recommendation approaches of Betz et al. [42], Bobek et al. [45], and Koschmider et al. [209, 210] suggest suitable model fragments based on the process models stored in a repository. The recommended fragments may still contain incomplete elements, such that the problem is multiplied, if a suggested but incomplete fragment is reused by the modeler. The naming suggestion approach of Leopold et al. [243, 244] is tailored to create a general name for a process model or a sub process activity. It may not be suitable to create names for specific activities in a process model that already contain incomplete model elements.

Against this background, the aim of this chapter is to address incomplete process model elements and to fill these gaps with meaningful information that best matches the respective process model's area of concern.

7.2 Operationalization of Incomplete Model Elements

As already mentioned, the approach addresses those elements that have incomplete text labels. More specifically, we are interested in labels that do not comply with the minimal syntactical structure of naming conventions [395, 283, 261]. Typically, these rules are violated by not naming the essential components of the model element. In case of activities for example, we would be interested in those activities that do not contain at least one action and one business object. For that purpose, this phase employs the model parser introduced in Section 4.4.3 that annotates element labels with their components. Using the definitions 4.3, 4.4, and 4.5, the set of incomplete elements is defined as follows:

Definition 7.1. (**Incomplete Elements of a Process Model**). Let $\mathrm{P} = (A, E, G, F, R, P, L, \rho, \pi, \lambda, \gamma, \tau)$ be a process model as well as A_λ^{P} the set of labeled activities, E_λ^{P} the set of labeled events, G_λ^{P} the set of labeled gateways of P. The incomplete elements of P are given by the following subsets:

- $I_A = \{a \in A_\lambda^{\mathrm{P}} \mid \alpha_A(a) = \emptyset \ \vee \ \beta_A(a) = \emptyset\}$, as being the set of *incomplete activities.*
- $I_E = \{e \in E_\lambda^{\mathrm{P}} \mid \alpha_E(e) \cup \sigma_E(e) = \emptyset \ \vee \ \beta_E(e) = \emptyset\}$, as being the set of *incomplete events.*
- $I_G = \{g \in G_\lambda^{\mathrm{P}} \mid \alpha_G(g) \cup \sigma_G(g) = \emptyset \ \vee \ \beta_G(g) = \emptyset\}$, as being the set of *incomplete gateways.*

As an example, we consider the activity a' with the label $\lambda(a') =$ *Retirement* from Figure 7.1. This activity a' describes the action *to retire*

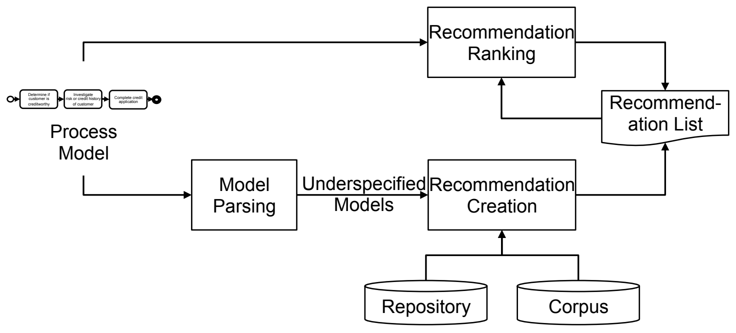


Fig. 7.2: Overview of the Recommendation Approach

($\alpha_A(a') =$ to retire) but no business object ($\beta_A(a') = \emptyset$). Given the previous definition, this activity is an element of the set of incomplete activities, i.e. $a' \in I_A$, because it names a valid action but an empty business object.

For the conceptual approach, we consider the incompleteness of model elements as a recommender problem involving the tasks of creating and ranking recommendations [161]. Figure 7.2 gives a general overview of the approach. The input of the approach is a set of process models that contain incomplete elements as defined previously. Depending on the components that are missing in the respective label, the approach creates a list of recommendations that are built upon four different context layers. Afterwards, the recommendation list is ranked according to the fitness of the recommendation with regard to the process model and given as output to the user. In the end, the user will receive an ordered list of recommendations among which he may choose a suitable alternative for the incomplete element labels.

The following sections will explain the different phases of the approach in more detail. Section 7.3 provides more details how recommendations may be created using different sources of knowledge, while Section 7.4 introduces a context-sensitive ranking algorithm to show only the relevant recommendations to the user.

7.3 Creation of Recommendations

This phase of the approach is concerned with the creation of recommendations for the incomplete model elements. The task of creating recommendations may be performed in a collaborative or a content-based way or in a combination thereof [161]. The collaborative creation of recommendations is based on direct user feedback and provides guidance on when and how a recommendation is given to the user. Content-based recommendation creation typically involves a filtering subsystem that is in charge of choosing the most useful items from a set. In the context of this thesis, we assume the situation in which an organization

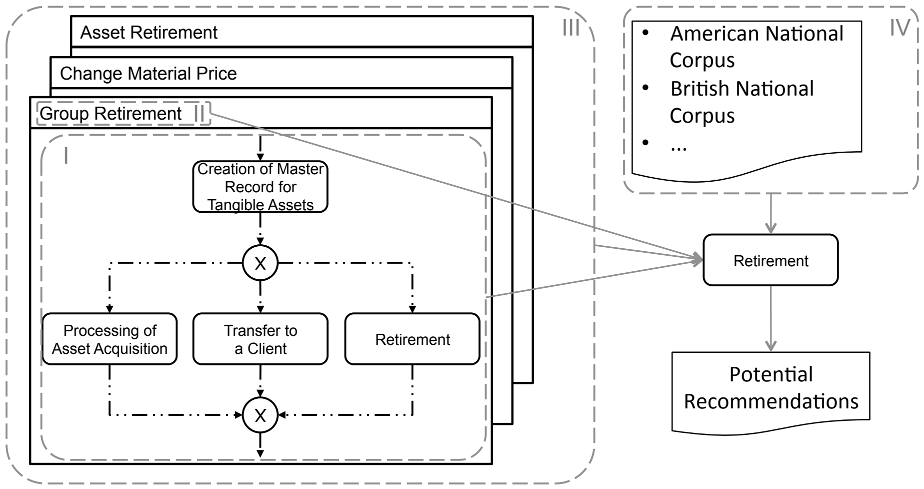

Asset Retirement
Change Material Price
Group Retirement
I
II
III
IV
Creation of Master Record for Tangible Assets
X
Processing of Asset Acquisition
Transfer to a Client
Retirement
X
• American National Corpus
• British National Corpus
• ...
Retirement
Potential Recommendations

Fig. 7.3: Recommendation Creation with 4 Layers of Context

has already created a repository with several hundreds or thousands of process models [369]. Accordingly, we can assume that the modeler will review an already existing process model with incomplete activities for which he requires a filtered list of best fitting recommendations. Based on this consideration, we develop a content-based recommendation creation with different context layers (see Figure 7.3).

The key feature of the recommendation creation strategy is the incorporation of several context layers. Specifically, we distinguish four layers of context: local, process model name, process model repository, and external. The list of potential recommendations is iteratively incremented by broadening the scope step-wise, from the most local to the most external (I to IV, in Figure 7.3). In the following, we discuss each recommendation creation strategy separately and explain how potential recommendations are created.

7.3.1 Local Context Strategy

The local context strategy considers all label components of each model element in a single process model to be a potential recommendation. The rationale is that the process model itself provides sufficient information to infer the component for the incomplete model element. Depending on the type of the incomplete element, this strategy aims to retrieve either a list of actions, business objects, or stati that are 'around' the respective model element. Considering the activity *Retirement* in the example of Figure 7.1, the local context strategy identifies *Asset Acquisition* and *Master Record* as potential business object recommendations. Formally, we describe the local context strategy S_L as follows:

Definition 7.2. (**Local Context Strategy**). Let P $= (A, E, G, F, R, P, L, \rho, \pi, \lambda, \gamma\, \tau)$ be a process model as well as A^{P}_{λ} the set of labeled activities, E^{P}_{λ} the set of labeled events, G^{P}_{λ} the set of labeled gateways of P. Further let $a' \in I_A$ be an incomplete activity, $e' \in I_E$ be an incomplete event, and $g' \in I_G$ be an incomplete gateway. The set of recommendations created from the local context are described as follows:

$$R_{Local}(a') = \begin{cases} \{\alpha_A(a) |\ a \in A^{\mathrm{P}}_{\lambda}\} & \text{, if } \alpha_A(a') = \emptyset \\ \{\beta_A(a) |\ a \in A^{\mathrm{P}}_{\lambda}\} & \text{, if } \beta_A(a') = \emptyset \end{cases}$$

$$R_{Local}(e') = \begin{cases} \{\alpha_E(e) |\ e \in E^{\mathrm{P}}_{\lambda}\} & \text{, if } \alpha_E(e') \cup \sigma(e') = \emptyset \\ \{\beta_E(e) |\ e \in E^{\mathrm{P}}_{\lambda}\} & \text{, if } \beta_E(e') = \emptyset \\ \{\sigma_E(e) |\ e \in E^{\mathrm{P}}_{\lambda}\} & \text{, if } \alpha_E(e') \cup \sigma(e') = \emptyset \end{cases}$$

$$R_L(g') = \begin{cases} \{\alpha_G(g) |\ g \in G^{\mathrm{P}}_{\lambda}\} & \text{, if } \alpha_G(g') \cup \sigma_G(g') = \emptyset \\ \{\beta_G(g) |\ g \in G^{\mathrm{P}}_{\lambda}\} & \text{, if } \beta_G(g') = \emptyset \\ \{\sigma_G(g) |\ g \in G^{\mathrm{P}}_{\lambda}\} & \text{, if } \alpha_G(g') \cup \sigma_G(g') = \emptyset \end{cases}$$

7.3.2 Process Model Name Strategy

As stated in [244], most naming concepts directly refer to the textual information captured in the process model name. The process model name context strategy considers the information contained in the model name to infer the missing object or action. However, since the process model itself may be regarded as a subprocess activity, this strategy is more suitable for incomplete actions. In this case, this strategy proposes to treat the model name as an activity label, extract its action and business object, and use this information to create a potential recommendation. However, this strategy relies on the assumption that the process model must specify a name. Taking the process model in Figure 7.1 as an example, this strategy takes the business object *group* as a potential recommendation for the incomplete activity *Retirement*. Interpreting the process model as being a subprocess activity a_P to which the functions $\alpha_A(a_P)$ and $\beta_A(a_P)$ extract an action or a business object respectively, we define the process model name strategy as follows:

Definition 7.3. (**Process Model Name Strategy**). Let P $= (A, E, G, F, R, P, L, \rho, \pi, \lambda, \gamma\, \tau)$ be a process model as well as A^{P}_{λ} the set of labeled activities. Further let $a' \in I_A$ be an incomplete activity. The set of recommendations created from the local context are described as follows:

$$R_{Name}(a') = \begin{cases} \{\alpha_A(a_P) |\ \alpha_A(a_P) \neq \emptyset\} & \text{, if } \alpha(a') = \emptyset \\ \{\beta_A(a_P) |\ \beta_A(a_P) \neq \emptyset\} & \text{, if } \beta(a') = \emptyset \end{cases}$$

7.3.3 Process Model Repository Strategy

The model repository strategy considers all process models of a repository to infer the missing component of an incomplete model element. The strategy also requires that the incomplete element and the complete element share a common component in order to avoid an extensive list of recommendations for incomplete activities. For example, in case an activity misses an action, we consider a complete activity to be a potential recommendation if both activities have the same business object. Otherwise, in case that the activity misses a business object, we consider a complete activity to be a potential recommendation if both activities have the same action. As an example, this strategy identifies the business object *leased asset* since this strategy identified the activity *Retirement of leased asset* to be complete and to share the action *to retire* with the incomplete label. The recommendations created from the process model collection are described as follows:

Definition 7.4. (**Process Model Repository Strategy**). Let P = (A, E, G, F, R, P, L, ρ, π, λ, γ τ) be a process model as well as A^{P}_{λ} the set of labeled activities, E^{P}_{λ} the set of labeled events, G^{P}_{λ} the set of labeled gateways of P. Further let $a' \in I_A$ be an incomplete activity, $e' \in I_E$ be an incomplete event, and $g' \in I_G$ be an incomplete gateway. The recommendations created from the process model repository are described as follows:

$$R_{Repository}(a') = \begin{cases} \{\alpha_A(a) |\ \beta_A(a) = \beta_A(a') \wedge a \in A^{\mathcal{P}}_{\lambda}\}, \text{ if } \alpha_A(a') = \emptyset \\ \\ \{\beta_A(a) |\ \alpha_A(a) = \alpha_A(a') \wedge a \in A^{\mathcal{P}}_{\lambda}\}, \text{ if } \beta_A(a') = \emptyset \end{cases}$$

$$R_{Repository}(e') = \begin{cases} \{\alpha_E(e) |\ \beta_E(e) = \beta_E(e') \wedge e \in E^{\mathcal{P}}_{\lambda}\}, \text{ if } \alpha_E(e') \cup \sigma_E(e') = \emptyset \\ \\ \{\beta_E(e) |\ (\alpha_E(e) = \alpha_E(e') \vee \sigma_E(e) = \sigma_E(e')) \\ \qquad\qquad \wedge\ e \in E^{\mathcal{P}}_{\lambda}\}, \text{ if } \beta_E(e') = \emptyset \\ \\ \{\sigma_E(e) |\ \beta_E(e) = \beta_E(e') \wedge e \in E^{\mathcal{P}}_{\lambda}\}, \text{ if } \alpha_E(e') \cup \sigma_E(e') = \emptyset \end{cases}$$

$$R_{Repository}(g') = \begin{cases} \{\alpha_G(g) |\ \beta_G(g) = \beta_G(g') \wedge g \in G^{\mathcal{P}}_{\lambda}\}, \text{ if } \alpha_G(g') \cup \sigma_G(g') = \emptyset \\ \\ \{\beta_G(g) |\ (\alpha_G(g) = \alpha_G(g') \vee \sigma_G(g) = \sigma_G(g')) \\ \qquad\qquad \wedge\ g \in G^{\mathcal{P}}_{\lambda}\}, \text{ if } \beta(g') = \emptyset \\ \\ \{\sigma_G(g) |\ \beta_G(g) = \beta_G(g') \wedge g \in G^{\mathcal{P}}_{\lambda}\}, \text{ if } \alpha_G(g') \cup \sigma_G(g') = \emptyset \end{cases}$$

7.3.4 External Corpus Strategy

If there are no meaningful recommendations retrieved by the three previous strategies, an alternative is to look for recommendations in an external source. Hence, this strategy considers the use of a general text corpus to search for terms which frequently co-occur with the missing element. For this purpose, one can employ sentence dependencies [68] (e.g. typed dependencies) to identify words that are frequently used with the target word. For this strategy, any available text corpus, such as the American National Corpus (ANC) or the British National Corpus (BNC) may be employed. In case of domain-specific process models, this strategy can also be adapted to use domain specific corpora or documents. As an example, we consider again the activity *Retirement* in Figure 7.1. According to the external corpora strategy, the objects *number* and *debt* are retrieved as potential recommendations for the missing business object. Considering Dep_C to be the set of all dependencies between an action and a business object that is retrieved from a corpus C, we formalize the external corpora strategy as follows:

Definition 7.5. (**External Corpus Strategy**). Let P = (A, E, G, F, R, P, L, ρ, π, λ, γ τ) be a process model as well as A^{P}_{λ} the set of labeled activities, E^{P}_{λ} the set of labeled events, G^{P}_{λ} the set of labeled gateways of P. Further let $a' \in I_A$ be an incomplete activity, $e' \in I_E$ be an incomplete event, and $g' \in I_G$ be an incomplete gateway. Based on the typed dependencies Dep_C retrieved from a corpus (see Definition 4.2), the recommendations created from the external corpus strategy are given as follows:

$$R_{External}(a') = \begin{cases} \{w |\ (\texttt{dobj}, w, \beta_A(a')) \in Dep_C\}\ , \text{if } \alpha(a') = \emptyset \\ \{w |\ (\texttt{dobj}, \alpha_A(a'), w) \in Dep_C\}\ , \text{if } \beta(a') = \emptyset \end{cases}$$

$$R_{External}(e') = \begin{cases} \{w |\ (\texttt{nsubj}, w, \beta_E(e)) \in D_C\}\ , \text{if } \alpha_E(e') \vee \sigma_E(e') = \emptyset \\ \{w |\ (\texttt{nsubj}, \alpha_E(e), w) \in D_C\}\ , \text{if } \beta_E(a') = \emptyset \end{cases}$$

$$R_{External}(g') = \begin{cases} \{w |\ (\texttt{nsubj}, w, \beta_G(g)) \in D_C\}\ , \text{if } \alpha(g') \vee \sigma_G(g') = \emptyset \\ \{w |\ (\texttt{nsubj}, \alpha_G(g), w) \in D_C\}\ , \text{if } \beta(g') = \emptyset \end{cases}$$

7.4 Context-Sensitive Ranking of Recommendations

As discussed before, the presentation of a large list of potential recommendations is not very helpful for users. Inline with content-based recommendation approaches, the goal is to present a list of most useful recommendations among which the user may choose according to their personal judgment. Therefore, we propose a ranking technique to sort the list of recommendations in a descending order. Inspired by the work of Bracewell et al. [51] who deals with ambiguous acronyms in biomedical texts, we use a semantic similarity calculation of a recommended item to a process model. The original method disambiguates acronyms by comparing the semantic similarity of an acronym expansion with a set of sense clusters, where each cluster describes a set of similar senses from the most frequent words extracted from a limited number of medical texts. The senses and the semantic similarity are computed using the lexical database WordNet [292].

As a single activity provides only limited context information, we create clusters with all activity labels from a process model. In order to measure the similarity of a recommendation for a missing component of an element label to its process model, the proposed method executes the following two tasks: the operationalization of the process model context and the cluster similarity calculation.

In order to operationalize the context of a process model, we make use of sense clusters that have been introduced in [363]. This strategy builds upon undirected weighted graphs where each word is a vertex and the weight of the edges represents the semantic similarity value between two vertexes. Afterwards, all maximal cliques are retrieved by using the Bron-Kerbosch algorithm [57]. The clique with the highest edge sum is stored as a cluster and all elements from the cluster are removed from the graph. This process is repeated until there are no more edges on the graph. Vertexes that are not assigned to any cluster remain unchanged in the graph. Also, a threshold limit can be used to prune low-score similarity values from the initial graph. As a result, we receive a set of sense clusters built upon semantically similar words from a process model.

The cluster similarity calculation involves the calculation of semantic similarity between a recommended item and the sense clusters and thus decides how good a recommendation fits to the process model. In this way, we retrieve all senses of a recommendation r and of a single sense cluster c and calculate the maximum pairwise-similarity score between them. The similarity measure for a recommended item $r \in R$ to a sense cluster $c \in C$ is the sum of similarity scores for a recommendation to each cluster sense. The following formula summarizes the calculation of semantic similarity between recommendation and clusters:

$$sim(r, C) = \sum_{r \in R} \arg\max_{c \in C}(sim(r, c))$$

Finally, the similarity between a recommendation r and a process model P is the sum of the similarity scores for a recommendation to each sense cluster of a process model C_P:

$$sim_P(r, P) = \sum_{C \in C_P} sim(r, C)$$

Algorithm 5 illustrates the steps to rank a list of recommendations R for a given process model P. The algorithm starts with the initialization of the result set that will contain the recommended items sorted by their relevance (Step 2). After all text labels of a process model have been extracted (Step 3), the algorithm determines the set of all sense clusters to operationalize the context of the process model (Step 4). Afterwards, the algorithm needs

Algorithm 5: Ranking Created Recommendations for a Process Model

```
1: rankRecommendations(RecommendationList $R$, ProcessModel P)
2: $rankedRecommendations \leftarrow \emptyset$
3: $elementLabels \leftarrow$extractTextLabels(P)
4: $senseClusters \leftarrow$retrieveSenseClusters($elementLabels$)
5: for all $r \in R$ do
6:    for all $C \in senseClusters$ do
7:       $sim_P \mathrel{+}= sim(r, C)$
8:    $rankedRecommendations \leftarrow (r, sim_P) \cup rankedRecommendations$
9: $sortDescendingBySimValue(rankedRecommendations)$
10: return $rankedRecommendations$
```

to decide how good the recommendation fits to the process model which is handled in a loop (Steps 5–8). For each combination of recommendation sense and sense cluster, the algorithm determines the similarity according to the given formula. Moreover, it accumulates these similarity scores to an overall similarity score sim_P that resembles the appropriateness of a recommendation to the process model (Step 7). Finally, the overall similarity score and the respective recommendation are added to the result set (Step 8). When each recommendation has been processed, the result set is sorted in descending order (Step 9). The algorithm terminates by returning a sorted result set that may drive the user decision by presenting the best ranked recommendations on the top positions.

7.5 Evaluation

The overall goal of the evaluation is to learn whether our techniques retrieve recommendations that are useful to the modeler in completing missing information in process models. First, we describe the test data (Section 7.6.1) and the setup of the evaluation (Section 7.6.3). Second, we present the results of our evaluation with regard to the creation of recommendations (Section 7.6.4) and their ranking (Section 7.6.2).

7.5.1 Model Repository Demographics

The recommendation-based approach has been applied to different model collections from industry that vary with regard to specific criteria, such as collection size, modeling notation, degree of standardization, and domain. For the evaluation of the created and ranked recommendations, we decided to restrict the evaluation on the activity labels since they a) represent a central construct for many process modeling languages and b) are particularly prone to ambiguity [285, 238]. Table 7.2 summarizes the main characteristics of the employed repositories. For a more detailed overview of these collections,

Table 7.2: Demographics of the Test Collections

Characteristic	**SRM**	**TC**	**AIC**
No. of Models	604	803	1,091
No. of Activities	2,432	12,088	8,339
No. of Incomplete Activities	311	500	741
No. of Affected Models	184 (30%)	290 (36%)	381 (34%)
Avg. No. Incomplete Activities per model	1.69	1.69	1.96

please refer to Table 7.2. The selected process model collections show a notable number of process models that are affected by incomplete activities and process models. Surprisingly, the SRM collection is the smallest and presumably most standardized collection and contains a notable number of process models with incomplete activities. In this data set, 184 models (30% of the models) are affected and contain on average 1.69 incomplete activities. Similar to that, the TC collection also has around 1.69 incomplete activities in its process models. However, the share of affected process models is slightly bigger than in the SRM collection. The AIC collection is the largest collection and assumed to have small degree of standardization. This collection shows a similar share of process models (34% of the models) that contain incomplete activities. However, the average number of affected activities is unlike higher amounting to almost 2 per model. Therefore, we consider these collections to be appropriate and representative to evaluate the proposed approaches. Nevertheless, it is to mention that the TC collection needed to be excluded from the evaluation due to non-disclosure reasons.

7.5.2 Evaluation Setup

To demonstrate the capabilities of our approach, we conduct an extensive user evaluation, which aims at assessing the usefulness of the recommendations. For this purpose, we outline our evaluation setup in terms of prototypical implementation, evaluation design, and evaluation metrics.

With regard to the *prototypical implementation*, we used Java 1.7 to implement our strategies and algorithms to create and rank the recommendations. Based on the list of incomplete model elements from our test collections, we created a set of potential recommendations by using the previously discussed strategies. As far as the external corpora strategy is concerned, we used the publicly open accessible version of the ANC corpus (OANC), which contains 15 million words of contemporary American English. Moreover, we also used the Stanford Tagger and Parser[1] to retrieve the typed dependencies, i.e. verb-direct object (dobj) and passive-sentence constructs (nsubj) to identify frequent co-occurrences of words. For the ranking algorithm and the employed similarity

[1] Please refer to http://nlp.stanford.edu/software/

metrics, we selected the Lin semantic similarity measure as it correlates best with human judgment [253].

Concerning the *evaluation design*, we require the decision of humans whether or not a recommendation is useful in the context of a given process model. However, since we face a considerably large number of incomplete labels combined with an enormous list of recommendations (1052 incomplete labels with approx. 28600 recommendations), we cannot let the user evaluate all recommendations and need to pursue another, more realizable evaluation strategy. In particular, we applied the statistical sampling strategy which has been explained in detail in Section 6.5.2. Accordingly, we utilize a statistical sampling among the recommendations and let the user evaluate a representative subset of recommendations. We notice that each recommendation is either useful or not. Hence, repeatedly drawing instances from our test data and assessing the usefulness represents a binomial experiment $X \sim B(n, p)$ with a population size of n and the probability of having a useful recommendation p. In order to determine this probability, we assessed the usefulness of ten recommendations, which then amounted to 75% in our overall test sample. Afterwards, we follow the recommendations by Piegorsch [325] and apply the Jeffrey interval, which suits for binomial references from large populations. Aiming for a significance level $\alpha = 0.05$ and a margin of error $\epsilon = 0.05$, we need to draw a random sample of 289 recommendations based on Piegorsch's sample size calculation formula. Taking the literature on recommender system evaluation into account [361, 161], it does not make sense to assess a single recommendation since the context of the recommendation is missing. Therefore, we randomly pick an incomplete activity and up to the ten highest ranked recommendations and repeat this procedure until we reached the necessary sample size. In the final evaluation design, we asked 5 users (2 modeling novices and 3 modeling experts) to provide us with their decision for the whole sample. Each participant was provided with an incomplete activity and the corresponding process model in which this activity was highlighted. The participants were then asked to provide feedback for each recommendation on a 4-point-Likert-scale from *very useless (-2)* to *very useful (+2)*. By using a 4-point-scale, we intentionally forced participants to make a final decision, which is necessary for our evaluation metrics. In Appendix C, the interested reader will find examples of the survey design.

As far as the *evaluation metrics* are concerned, we use acknowledged metrics from the recommender systems literature [161]. In particular, we employ the metrics precision and recall for our evaluation. In the context of recommendations, the precision value describes the number of relevant recommendations divided by the total number of relevant and non-relevant recommendations that have been created by the system. Recall is the number of relevant recommendations divided by the total number of retrieved and not retrieved relevant recommendations. As it is fairly easy to achieve a recall of 1 (by presenting all recommendations to the user), we further distinguish between three situations: i) the most highly ranked, ii) the five most highly

Table 7.3: Performance of Recommendation Creation

		S1	**S2**	**S3**	**S4**
Top 1	Useful Recommendations	15	0	3	5
	Useless Recommendations	6	0	1	1
Top 5	Useful Recommendations	15	0	23	20
	Useless Recommendations	16	3	24	13
Top 10	Useful Recommendations	58	0	58	33
	Useless Recommendations	46	4	51	36

S1: Local Context Strategy, S2: Model Name Strategy,
S3: Model Collection Strategy, S4: External Context Strategy

ranked (Top 5), and iii) all recommendations are shown to the user (Top 10). The assessment with different scenarios has several advantages. The Top 1 scenario provides insights from a full automation perspective. If the quality of the highest ranked recommendation was sufficiently good, there would be hardly any need to further involve the users in selecting a suitable alternative. The Top 5 scenario directly assesses the performance of the ranking algorithm. If the five highest recommendations achieve a high precision and recall, we have a good indicator that the approach only presents the relevant recommendations among which the user can choose. The Top 10 reports on the usefulness of the created recommendations in general. The higher the value of precision is, the better our recommendation creation strategies perform. Apparently, recall plays a minor role in this scenario since all recommendations (10 out of 10) are shown to the user leading always to a recall of 1.

7.5.3 Evaluation of Recommendation Creation

The performance results of the recommendation creation strategies are summarized in Table 7.3. The numbers show that the strategies are capable to create meaningful recommendations. However, we also observe notable differences between the strategies. In general, the local context and the model repository strategy perform best in producing useful recommendations. Also, the external context strategy provides a considerable amount of useful recommendations, although we observe a slightly larger number of useless ones. A possible explanation is that the external context strategy is not as closely related to the process models and thus is only capable to create general recommendations that cannot reflect model-specific terminology. To our surprise, the model name strategy did not create any useful recommendations. We assume that the model name resembles an abstract summary of the process which is not suitable to complement activities on a detailed level.

Moreover, we also take a closer look at the performance of the recommendation creation strategies with regard to the different scenarios. Beginning with the Top 1 scenario, we see that the local context strategy proposes a big number of useful recommendations and clearly outperforms the other strategies. We conclude that the local context strategy might be a good method to automatically fill the missing gaps in the label components. In the Top 5 and the Top 10 scenario, the picture is shifted in favor of the model repository and the external corpora strategy. Both strategies perform well in creating useful recommendations and also outperform the local context strategy in the Top 5 scenario. Then, in the Top 10 scenario, the recommendation from the local context and the model repository strategy are dominating again. On average, they produce more useful recommendations than the external context strategy. These results emphasize the advantage of using different context layers to infer the missing label component and to propose several alternatives among which the user may select. Finally, we may conclude that the local context and the model repository strategy are the favorite sources from which recommendations should be created.

7.5.4 Evaluation of Recommendation Ranking

The results of the ranking technique are shown in Table 7.4. In particular, we report on the numbers of precision and recall if the most highly ranked, the five and ten most highly ranked recommendations are considered by the users. In general, the numbers show that the performance of the recommendation creation and context-based ranking approach is promising in the Top 1 and Top 5 scenario and excellent for the Top 10.

The Top 1 scenario evaluates the approach from the perspective of automatically refactoring the incomplete activity. In this case, the approach would choose the highest ranked recommendation and replace the deficient activity label with the recommended one. In such a setting, the evaluation revealed a precision of 0.74 which implies that the first recommendation is considered to be useful in almost three out of four cases. With regard to automation, this may be regarded as a satisfactory result. However, the recall value of 0.18 also indicates that other recommendations might also be a suitable alternative for the deficient activity emphasizing that a semi-automated approach is more suitable to address incomplete activities.

In the Top 5 scenario, the precision reflects the capabilities of our techniques to present only the useful items to the user, while minimizing the number of useless recommendations at the same time. In this case, the number of 0.55 indicates that at least two out of five recommendations fit to the model context and are also regarded to be a suitable alternative to the incomplete activity. The recall emphasizes the coverage of useful items which our techniques can obtain by considering the entire pool of recommendations. It amounts to 0.58 in the Top 5 setting which shows that even if our techniques did not create any meaningful recommendations within the five highest ranked recommendations, more useful ones may be retrieved when looking at the following 5 recommendations.

Table 7.4: Recommendation Performance Results

	Top 1	**Top 5**	**Top 10**
No. Relevant & Retrieved	23	86	149
No. Non-relevant & Retrieved	8	69	137
No. Relevant & Not retrieved	126	63	0
Precision	0.74	0.55	0.52
Recall	0.18	0.58	1

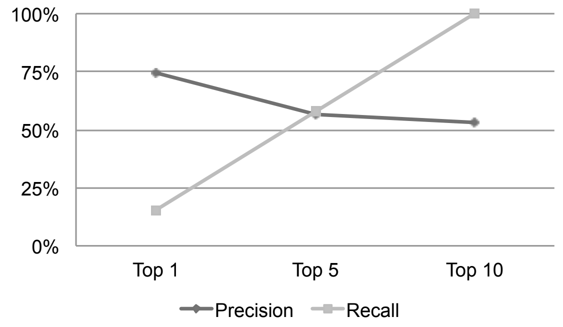


Fig. 7.4: Precision/Recall Curve for the Evaluation Scenarios

The results of the Top 10 scenario may also be interpreted from the perspective of usefulness in general. Since this scenario presents all the recommendations to the user at once, the precision value directly relates to the usefulness of the recommendations. The value of 0.52 indicates that the recommendation approach is capable of creating context-sensitive and useful recommendations. In particular, the users agreed that at least half of the recommendations are useful and appropriate for the given process models in our user experiment. In this scenario, the recall value is always 1, since the entire pool of recommendation is presented at the same time.

The ranking results also point to a trade-off conflict between the different scenarios. Figure 7.4 maps the scores of precision and recall to the different scenarios in order to illustrate this conflict. In general, this conflict is characterized by a strong increase in the recall scores and a constant slightly decrease of precision. If the users consider to automatically refactor the incomplete activities (Top 1), they take the risk of missing a more suitable recommendation, which is reflected by a high precision and a low recall value at the same time. Opposite to that, if they want to see the entire list of recommendations (Top 10), they can be sure that they will find a suitable recommendation, which is reflected by the recall of 1. The scores of the Top 5 scenario balance these extreme points. Here, the users are only presented a subset of all recommendations among which we can assume a suitable one for the incomplete activity.

Table 7.5: Top 5 Recommendations for the Group Retirement Process

Activity	**Top 5 Recommendations**	**Ranking**
Transfer to a Client	Transfer Time Specifications to a Client	25.76
	Transfer Balance Sheet Items to a Client	19.51
	Transfer Planned Sales Quantities to a Client	18.87
	Transfer Group to a Client	18.26
	Transfer Personnel Costs to a Client	16.24
Retirement	Retirement of Master Record	18.02
	Retirement of Leased Asset	16.42
	Retirement of Asset Acquisition	15.53
	Retirement of Group	14.81
	Retirement of Number	13.77

In addition to the quantitative results, we also discuss qualitative results of our recommendation techniques. Table 7.5 shows the five highest ranked recommendations for the two incomplete activities of Figure 7.1. In general, we can infer from the context of the process model that the two activities *Transfer to a Client* and *Retirement* are most likely applied to the business objects *Group* or *Asset*. Looking at the results of our techniques, we spot two recommendations that propose these two business objects. Moreover, the techniques also provide additional recommendations that make sense in the process model. In case of the first activity, we might also consider the objects *Balance Sheet Items* or *Planned Sales Quantities* to be valid for the missing business object. For the second activity, our approach further recommends the business objects *master record* and *asset acquisition*. Considering the context, we might still include the object asset acquisition as a potential candidate for the this activity. Overall, these recommendations are good examples to illustrate the capability of the approach to successfully provide context-sensitive recommendations to the user.

7.6 Discussion

This section discusses the results and implications of the recommendation approach based on the evaluation. To this end, Section 7.6.1 summarizes the main results of the evaluation with regard to the creation and ranking technique. Afterwards, Section 7.6.2 reflects upon limitations that need to be considered regarding the results. Finally, Sections 7.6.3 and 7.6.4 discuss implications of the proposed approach for research and for practice.

7.6.1 Summary of Results

The evaluation of the recommendation-based approach shows satisfactory results with regard to the usefulness of the suggestions and the quality of the

Table 7.6: Summary of Results for Pragmatic Ambiguity Refactoring

	Creation		Ranking	
Top 1	Total No. Useful Recommendations	23	Precision:	0.74
	Total No. Useless Recommendations	8	Recall:	0.18
Top 5	Total No. Useful Recommendations	86	Precision:	0.55
	Total No. Useless Recommendations	69	Recall:	0.58
Top 10	Total No. Useful Recommendations	149	Precision:	0.52
	Total No. Useless Recommendations	137	Recall:	1

ranking. Table 7.6 summarizes the main results of the evaluation for each scenario. In the Top 1 scenario, 23 out of 31 recommendations are considered to be useful by the users. We also see that most of these recommendations are created from the local context strategy. The Top 5 scenario reveals that 86 of 155 recommendations are meaningful to the users. Regarding the number of recommendations, the repository strategy and the external context strategy provide more meaningful recommendations to the user. Finally, the number of meaningful and useless recommendations are rather balanced in the Top 10 scenario. In general, the approach creates an equal number of good and bad recommendations. We observe that the local context and the model repository strategy perform equally well in creating good recommendations. We conclude that the local context and the model repository strategy are well suited to look for fitting recommendations. Moreover, it is not recommended to use the model name as a source of recommendations since the name appears to be too abstract in many cases.

The ranking of the recommendations is rather promising. The precision of the Top 1 scenario amounts to almost 75% strongly indicating the capabilities for automatic refactoring. In the Top 5 scenario, the evaluation shows that precision and recall are well balanced and confirm a satisfactory performance of the recommendation ranking approach since every second recommendation of the five highest ranked is helpful to the user. The ranking approach performs similar if we consider the Top 10 scenario.

7.6.2 Limitations

The findings of the evaluation and the approach are also subject to limitations, in particular regarding the validity of conclusion and the technology. *Conclusion validity* is concerned with issues that affect the ability to draw the correct conclusions from the outcome of the evaluation [441, p. 104-105]. The conclusion validity might mostly be affected due to the limited number of collections. For the evaluation, the two employed process model collections can hardly be seen as representative. Therefore, it cannot be excluded that the consideration of other process model repositories would lead to different results. In particular, the employed repositories, i.e. the SRM and the AIC repository, are quite opposite to each other with regard to their general characteristics. Hence, it appears beneficial to include a repository in the evaluation that mediates the characteristics of the SRM and AIC repository. Moreover, the results of the recommendation creation and ranking might also be threatened by the focus on activity labels and the statistical sampling process as well as by maturity and instrumentation, which has already been discussed in Section 6.6.2.

From the perspective of *technology*, there are performance issues that need to be considered before creating a new software system or embedding the techniques in an existing tool. The extensive use of semantic similarity calculations at word-level impedes the algorithm's performance due to the necessity of traversing a computational lexicon multiple times. However, since the approach does not yet focus on immediate recommendations but on already existing model repositories, where quality checks are typically conducted in terms of an initiative, we do not consider the performance as a major limitation yet. We also observe that the presented techniques heavily rely on the quality of the text labels present in the process model collection and the external corpora. The quality of recommendations might be affected if model elements contain grammatical errors. Moreover, the use of general corpora may cover the domain-specific language only poorly and might not be able to recommend domain-specific terms. In such cases, the use of a domain specific text corpus will yield better results. Finally, the proposed approach only uses basic natural language processing technology to create a list of recommendations. The performance of the approach may be enhanced by machine learning approaches improving the overall recommendation results. Nevertheless, the user evaluation shows that our techniques are already capable of creating a large share of useful recommendations despite the usage of imperfect input sources or basic techniques.

7.6.3 Implications for Research

The recommendation-based approach of this thesis provides a solution for a hardly addressed quality dimension of conceptual models. According to model quality frameworks (see e.g. [255, 219, 218]), the pragmatic quality of conceptual models involves the correspondence between the model and its interpretation in order to be comprehensible. However, the comprehensibility of models might be affected negatively if the complexity is increasing [284], the textual labels are ambiguous [285], or a consistent interpretation is not possible [126, 127]. The approach supports the maintenance of pragmatic quality as it ensures that the model is complemented by missing information from the modeling domain increasing the completeness of the model and ensuring its comprehensibility for several actors. Overall, the approach represents an important step towards automated quality assurance of conceptual models.

Furthermore, this thesis proposes a recommendation-based approach that uses several layers of context to solve the problem of incomplete element labels.

This aims for an improvement of the textual dimension of process models. Related recommendation approaches for process models put a strong emphasis on the generation of structural recommendations [204]. Among others, prior research has proposed recommendation techniques that employ similarity metrics [42, 168], a combination of business rules and structural constraints [167], the reuse of existing fragments with the help of the π-calculus and ontologies [266], a tagging-based approach [166], or Bayesian networks [45]. Moreover, prior research also proposed modeling editors that support the user and provide recommendations based on the existing process model that was modeled so far [209, 80, 210]. Although these approaches provide model fragments including a naming of model elements, they need to rely on existing knowledge bases such as the model repository. As a consequence, naming errors in the knowledge base also appear in the recommendations and are multiplied when users follow these recommendations. In contrast to that, the proposed approach addresses the naming problem at its root and contributes to the overall quality of the repository from which also other approaches benefit.

Finally, the recommendation approach also improves the understanding of the process model, which turns out to be beneficial in the aforementioned application scenarios, namely *process model compliance*, *system design and analysis*, and *process analysis and monitoring*. In the compliance scenario, the recommendation approach supports users to refactor audit-relevant process models that contain incomplete elements. Closing the gaps in such models facilitates the understanding of the auditors and helps the company to prove that their activities are in accordance to the given compliance rules. Also system design and analysis approaches benefit from the recommendations. In this case, the gaps in the process model are closed, such that the model itself may be used to correctly depict requirements of the later system. This decreases the necessity for additional feedback loops and supports the correct specification and implementation of the system. Revisiting the process model analysis scenarios, the recommendations further specify the incomplete activity and support analysts in retrieving the appropriate information for the analysis. In consequence, the analysis has a better chance to resemble the real business activities and to take the right measures depending on the results.

7.6.4 Implications for Practice

The proposed approach also has considerable implications for practice. Most importantly, the approach can be integrated into commercial modeling tools. Such modeling tools can support users and modelers by pointing to incomplete labels. The respective labels can be easily refactored after notifying the user about the respective issue. As a result, the approach may prevent element incompleteness during the process of modeling and avoid costly maintenance as early as possible.

The presented approach may also be used in scenarios with existing model repositories. It supports users in performing maintenance-related tasks that involve the inspection of process models with regard to specific errors and inconsistencies. In particular, the presented approach provides means to detect and refactor incomplete model elements. Following recommendation-based concepts, the approach supports users by creating a list of suitable alternatives to fill the gaps in process models. Since the detection and refactoring tasks are typically time-consuming and perceived as ineffective and boring [227, 17], the approach can help to reduce the required time significantly and increase the effectiveness of resolving incomplete instances.

7.7 Summary

This chapter has introduced a novel approach to refactor the problem of pragmatic ambiguities in process model elements, which occurs in every third process model of a model repository. Relying on existing linguistic concepts, the incompleteness of model elements has been identified to be a symptom of pragmatic ambiguity and to hinder the correct comprehension of the process model. In order to address the problem, a recommendation-based approach has been proposed that creates a list of recommendations and ranks this list according to their fitness to the process model. In particular, four different strategies have been presented that exploit the local process model, its name, the collection, and other external sources to come up with a set of initial recommendations. This chapter has also introduced a context-sensitive ranking to distinguish between useful and useless recommendations and to reduce the initial list. Both concepts have been implemented in a research prototype and evaluated in an extensive user experiment by sampling process models from real world repositories. The quantitative and qualitative evaluation results have demonstrated the capabilities of creating meaningful recommendations and stimulated further endeavors for practice and research. For instance, the proposed techniques complement existing techniques to ensure the quality assurance of process models with regard to the domain and enrich the set of model recommendation approaches.

8

Conclusion

The final chapter summarizes the thesis and its results. Section 8.1 gives a short summary of the main results, which are then discussed in a broader context in Section 8.2. In the end, Section 8.3 gives an outlook of future research endeavors.

8.1 Summary of Results

This thesis has proposed novel, linguistic-based approaches for the refactoring of process models. It has investigated existing concepts and technology from the linguistic branches of syntax, semantics, and pragmatics and transfers them to the context of process models. Due to the high degree of automation, the approach supports companies to maintain large process model repositories and to ensure the quality of its process models. In particular, the results of this thesis can be summarized as follows.

- **Framework of Process Model Analysis**: Prior research on process model analysis and refactoring put a strong focus on the correctness of the formal part of process models. As shown by current research, the textual part of process models equally contribute to their overall correctness and quality [236]. This thesis has elaborated a classification framework that brings together several different categories of refactoring in order to structure prior research on process model refactoring. The respective categories are the formal and textual content, the coverage of the refactoring, and the degree of automation. Besides its usefulness to organize prior research, the classification has also revealed a notable gap for the textual refactoring techniques and further motivated this thesis.
- **Conceptualization of Atomicity for Process Models**: Process modeling in practice typically involves causal modelers that are not sufficiently trained in process modeling [369, 419] and thus use a casual style of creating and naming model elements [285]. While specific naming styles are already

automatically checked and corrected [238], the techniques only provide part of the solution when modelers use complex natural language phrases to express logic that relates to the control flow or implicit elements. The imbrication of modeling language and natural language makes it impossible to draw reliable conclusions from formal analysis results. Therefore, we have motivated the conceptualization of the atomicity notion, which demands that process model elements shall not incorporate several actions or business objects in one element. Furthermore, we have assessed several process model repositories with regard to this property and identified nine patterns that violate the notion of atomicity.

- **Detection and Refactoring of Syntactic Ambiguity**: The assessment of the atomicity notion revealed that many process models already suffer from atomicity violation issues. Depending on the quality of the process models and the expertise of the model creators, 26% to 67% of process models contain elements that are not specified on an atomic level. In order to leverage the automatic refactoring of these models, we have formalized the notion of atomicity and the violation patterns with linguistic concepts and technology. The experimental evaluation of this approach has revealed that the techniques show good results for precision and recall, which indicates their capability to refactor a notable number of non-atomic process models from industry.
- **Detection and Refactoring of Semantic Ambiguity**: A fixed vocabulary represents an important asset in an organization to avoid misunderstandings and to ensure a precise communication of business goals and requirements. However, research also points out that the management of such a vocabulary is a challenging task [212]. We have proposed a technique for the automatic detection and refactoring of lexically ambiguous terms in process models in order to support modelers in maintaining a consistent and unambiguous terminology. In particular, we have operationalized synonym and homonym detection conditions and proposed a set of strategies to resolve these ambiguities. Both techniques have been evaluated by the help of real-world process models and native speakers and have shown promising results for the refactoring of semantic ambiguities.
- **Detection and Refactoring of Pragmatic Ambiguity**: A process model is used by modelers as a substitution of the business process to leverage its analysis and improvement. For this purpose, a model needs to reduce the complexity and only focus on the relevant parts. In many cases however, the process model does not provide sufficient details anymore to specify the relevant parts of the underlying business process. The process model has gaps which hinder the comprehension and its use as a knowledge resource. These gaps might point to several alternatives and confuse the reader actually applying the specified actions of the process model. For these reasons, we have proposed a recommendation-based approach that explicitly considers the context of the incomplete process model element. To this end, we have implemented several strategies to create

recommendations for incomplete elements. Moreover, we have developed a ranking approach that formalizes the context of the process model and orders the recommendation according to their context fitness. The approach has been evaluated by users who indicated that almost three out of five recommendations were useful to close the gaps in the process models.

8.2 Discussion

The results of this thesis also have implications for business process modeling in a broader context. With this regard, we discuss their importance of language consistency in conceptual models, as well as their implications for the quality of process models, business process modeling tools, and process modeling guidelines.

- *Importance of Language Consistency in Conceptual Models*: The number of various linguistic inconsistencies in the different model repositories from industry underline the importance of language consistency. We face around one third of process models from industry to be affected by language inconsistencies, such as semantic or pragmatic ambiguities. Such process models do not only hamper the understanding and sense making of model users, but also their reuse for various purposes [212]. Moreover, we also find other conceptual models that are affected by these inconsistencies, such as goal models [367, 333], use case models [81], or feature models [233]. The results of this thesis do not only support the revision of inconsistent models and model repositories, but also sustain the consistency of language and terminology over a longer period of time.
- *Quality of Process Models*: The quality of conceptual models has been extensively discussed by several authors (see e.g. Lindland et al. [255] and Krogstie et al. [218, 219]) who identified several dimensions equally contributing to the overall quality of process models. Building on the requirements of such quality frameworks, many techniques have been introduced that check and correct affected models in an automatic fashion. The research of this thesis makes an important contribution to existing techniques of model quality assurance by addressing inconsistencies of natural language from a syntactical, semantic, and pragmatic angle. Therefore, the research reported in this thesis represents an important step towards the automated quality assurance of process models.
- *Business Process Modeling Tools*: Today, many modeling tools, modeling environments, or model repositories provide comprehensive support to create business process models and checking various aspects of them. As a result, modeling errors can already be avoided during the modeling process. This interactive style of modeling gives modelers valuable feedback of their modeling skills and fosters reflection and improvement of the personal modeling style. The techniques of this thesis may also provide feedback

to the modelers by indicating a potentially ambiguous usage of a word or by notifying modelers about incompletely specified model elements. All in all, these techniques support modelers in their task and safe time and cost-intensive rework afterwards.

- *Teaching Process Model Labeling*: The number of linguistic inconsistencies also points to a weakness of how labeling of process model elements is taught by universities or companies. Despite the recommendations of various modeling guidelines [283, 395, 261, 34], the issue of bad labeling style is still not fully recognized by these organizations. Modeling trainers and teachers are not aware of the consequences of bad labeling and fail to communicate the importance of labeling for the overall model quality. Besides teaching the consequences of formal errors like deadlocks, modeling trainers should also pay attention to linguistic issues that are caused by ambiguous or vague labeling. With this regard, this thesis has shown various examples of bad labeling styles which can be communicated as *bad modeling practice*. This particularly holds for the atomicity violation patterns which nicely illustrate how specific natural language statements are translated into a correct model fragment that resembles the original intention of the statement.

8.3 Future Research and Concluding Remarks

There are several open questions that have not been addressed in this thesis. The main focus of this thesis was to develop novel techniques that employ linguistic concepts for the analysis and refactoring of process models. However, the application of such techniques comes along with several limitations that have already been discussed in the respective chapters and that point to future research directions.

First, it has to be mentioned that most of the presented techniques analyze process model activity labels and aim for a consistent labeling of these elements. The choice is motivated by the fact that activities represent a central construct for many process modeling languages and are particularly prone to ambiguity conflicts [285, 238]. Nevertheless, we are aware that process models also specify more elements that are worth analyzing, such as events, gateways, pools, and resources. Hence, one direction of future research will include the extension of the proposed techniques to completely cover all relevant elements of the process model.

Second, the proposed techniques have been developed for English process models so far and are not capable to analyze process model labels of a different language. The localization of the proposed techniques is, however, possible because non-English process models also follow regular structures [238] and because natural language processing techniques are also available for other languages (see e.g. [302, 155]). Therefore, the second direction of future research aims for an adaption and evaluation of the presented techniques to other languages.

Third, the proposed techniques have only been implemented in a prototypical environment in order to evaluate them with process models from industry. A professional implementation and deployment into an advanced modeling tool is still open. This direction of future research will involve the integration of existing model analysis techniques into a modeling environment such that users can check their process models from different perspectives. Moreover, such a tool would also point to the detected errors and explain how these errors could be corrected or resolved afterwards. The implementation of such a tool would also involve a close cooperation and evaluation with end-users in order to demonstrate the capabilities of the tool and to create a modeling tool that meets the user requirements.

Altogether, the techniques of this thesis complement existing model analysis techniques with regard to the natural language content of process models and stimulate further ideas to use available natural language processing techniques for process model analysis or other activities that are related to business process management.

References

1. van der Aalst, W.M.P.: Verification of workflow nets. In: Application and Theory of Petri Nets 1997, 18th International Conference, ICATPN '97, Toulouse, France, June 23-27, 1997, Proceedings. pp. 407–426 (1997)
2. van der Aalst, W.M.P.: The application of petri nets to workflow management. Journal of Circuits, Systems, and Computers 8(1), 21–66 (1998)
3. van der Aalst, W.M.P.: Formalization and verification of event-driven process chains. Information & Software Technology 41(10), 639–650 (1999)
4. van der Aalst, W.M.P.: Workflow verification: Finding control-flow errors using petri-net-based techniques. In: Business Process Management, Models, Techniques, and Empirical Studies. pp. 161–183 (2000)
5. van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M.: Business process management: A survey. In: Business Process Management, International Conference, BPM 2003, Eindhoven, The Netherlands, June 26-27, 2003, Proceedings. pp. 1–12 (2003)
6. van der Aalst, W.M.P., Kumar, A.: Xml–based schema definition for support of interorganizational workflow. Information Systems Research 14(1), 23–46 (2003)
7. van der Aalst, W.M.P., ter Hofstede, A.H.M.: Workflow patterns: On the expressive power of (petri-net-based) workflow languages. In: Proceedings of the 4th Workshop on the Practical Use of Coloured Petri Nets and CPN Tools. pp. 1–20 (2002)
8. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Trans. Knowl. Data Eng. 16(9), 1128–1142 (2004)
9. Abran, A., Bourque, P., Dupuis, R., Moore, J.W., Tripp, L.L.: Guide to the Software Engineering Body of Knowledge - SWEBOK. IEEE Press, 2004 version edn. (2004)
10. Adriaans, P.W.: Language Learning from a Categorial Perspective (1992)
11. Agirre, E., Edmonds, P.: *Word Sense Disambiguation: Algorithms and Applications*. Computational Linguistics 33(2), 1–28 (2007)
12. Agirre, E., de Lacalle, O.L.: UBC-ALM: Combining k-NN with SVD for WSD. In: Proceedings of the 4th International Workshop on Semantic Evaluations. pp. 342–345. Association for Computational Linguistics (2007)

13. Agirre, E., Stevenson, M.: Knowledge sources for wsd. Word Sense Disambiguation p. 217 (2007)
14. Ahrend, N., Leopold, H., Pittke, F.: Barriers and strategies of process knowledge sharing in public sector organizations. In: Multikonferenz Wirtschaftsinformatik (MKWI 2014) (2014)
15. Akmajian, A.: Linguistics: An introduction to language and communication. MIT press (2001)
16. Allen, J.F., Perrault, C.R.: Analyzing Intention in Utterances. Artificial Intelligence 15(3), 143–178 (1980)
17. Ambriola, V., Gervasi, V.: On the systematic analysis of natural language requirements with circe. Automated Software Engineering 13(1), 107–167 (2006)
18. Anda, B., Sjøberg, D.I.K.: Towards an inspection technique for use case models. In: Proceedings of the 14th international conference on Software engineering and knowledge engineering, SEKE 2002, Ischia, Italy, July 15-19, 2002. pp. 127–134 (2002)
19. Attaran, M.: Exploring the relationship between information technology and business process reengineering. Information & Management 41(5), 585–596 (2004)
20. Awad, A., Puhlmann, F.: Structural detection of deadlocks in business process models. In: Business Information Systems, 11th International Conference, BIS 2008, Innsbruck, Austria, May 5-7, 2008. Proceedings. pp. 239–250 (2008)
21. Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press / Addison-Wesley (1999)
22. Baker, C.F., Fillmore, C.J., Lowe, J.B.: The berkeley framenet project. In: 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, COLING-ACL '98, August 10-14, 1998, Université de Montréal, Montréal, Quebec, Canada. Proceedings of the Conference. pp. 86–90 (1998)
23. Baker, P.: Using Corpora in Discourse Analysis. Continuum Discourse Series, Bloomsbury (2006)
24. Barjis, J.: The importance of business process modeling in software systems design. Sci. Comput. Program. 71(1), 73–87 (2008)
25. Basel Committee on Banking Supervision: Basel iii: A global regulatory framework for more resilient banks and banking systems. Tech. rep., Bank of International Settlements (2010)
26. Basu, A., Blanning, R.W.: A formal approach to workflow analysis. Information Systems Research 11(1), 17–36 (2000)
27. Basu, A., Kumar, A.: Research commentary: Workflow management issues in e-business. Information Systems Research 13(1), 1–14 (2002)
28. Bateman, J.A.: Enabling technology for multilingual natural language generation: the KPML development environment. Natural Language Engineering 3(1), 15–55 (1997)
29. Batoulis, K., Eid-Sabbagh, R., Leopold, H., Weske, M., Mendling, J.: Automatic business process model translation with BPMT. In: Advanced Information Systems Engineering Workshops - CAiSE 2013 International Workshops, Valencia, Spain, June 17-21, 2013. Proceedings. pp. 217–228 (2013)
30. Bauer, L.: The linguistics student's handbook. Edinburgh University Press (2007)

31. Becker, J., Delfmann, P., Herwig, S., Lis, L., Stein, A.: Formalizing linguistic conventions for conceptual models. In: Conceptual Modeling - ER 2009, 28th International Conference on Conceptual Modeling, Gramado, Brazil, November 9-12, 2009. Proceedings. pp. 70–83 (2009)
32. Becker, J., Delfmann, P., Herwig, S., Lis, L., Stein, A.: Towards increased comparability of conceptual models - enforcing naming conventions through domain thesauri and linguistic grammars. In: 17th European Conference on Information Systems, ECIS 2009, Verona, Italy, 2009. pp. 2231–2242 (2009)
33. Becker, J., Rosemann, M., Schütte, R.: Grundsätze ordnungsmäßiger modellierung. Wirtschaftsinformatik 37(5), 435–445 (1995)
34. Becker, J., Rosemann, M., von Uthmann, C.: Guidelines of business process modeling. In: Business Process Management, Models, Techniques, and Empirical Studies. pp. 30–49 (2000)
35. Becker, J., Rosemann, M., von Uthmann, C.: Guidelines of business process modeling. In: Business Process Management, Models, Techniques, and Empirical Studies. pp. 30–49 (2000)
36. Ben-Ari, D., Berry, D.M., Rimon, M.: Translational ambiguity rephrased. In: Theoretical and Methodological Issues in Machine Translation of Natural Languages (1988). pp. 175–183 (1988)
37. Bentley, L.D., Whitten, J.L., Randolph, G.: Systems analysis and design for the global enterprise, vol. 417. McGraw-Hill/Irwin (2007)
38. Berger, J.O.: Statistical decision theory and Bayesian analysis. Springer (1985)
39. Berry, D.M., Kamsties, E.: Ambiguity in requirements specification. In: Perspectives on software requirements, pp. 7–44. Springer (2004)
40. Berry, D.M., Kamsties, E., Krieger, M.M.: From contract drafting to software specification: Linguistic sources of ambiguity - a handbook version 1.0 (2000)
41. Bertino, E., Ferrari, E., Atluri, V.: The specification and enforcement of authorization constraints in workflow management systems. ACM Trans. Inf. Syst. Secur. 2(1), 65–104 (1999)
42. Betz, S., Klink, S., Koschmider, A., Oberweis, A.: Automatic user support for business process modeling. In: Proceedings of the Workshop on Semantics for Business Process Management. pp. 1–12 (2006)
43. Bieswanger, M., Becker, A.: Introduction to English Linguistics. UTB für Wissenschaft: Uni-Taschenbücher, Francke (2010)
44. Blake, B.: Relational grammar. Routledge (2002)
45. Bobek, S., Baran, M., Kluza, K., Nalepa, G.J.: Application of bayesian networks to recommendations in business process modeling. In: Proceedings of the Workshop AI Meets Business Processes 2013 co-located with the 13th Conference of the Italian Association for Artificial Intelligence (AI*IA 2013), Turin, Italy, December 6, 2013. pp. 41–50 (2013)
46. Boehm, B.W., Papaccio, P.N.: Understanding and controlling software costs. IEEE Trans. Software Eng. 14(10), 1462–1477 (1988)
47. Bolloju, N., Leung, F.S.K.: Assisting novice analysts in developing quality conceptual models with UML. Commun. ACM 49(7), 108–112 (2006)
48. Booth, T.L.: Probabilistic representation of formal languages. In: Proceedings of the 10th Annual Symposium on Switching and Automata Theory. pp. 74–81. SWAT '69, IEEE Computer Society (1969)
49. Borah, P.P., Talukdar, G., Baruah, A.: Approaches for word sense disambiguation–a survey. International Journal of Recent Technology and Engineering 3(1) (2014)

50. Bouzeghoub, M., Gardarin, G., Métais, E.: Database design tools: An expert system approach. In: VLDB'85, Proceedings of 11th International Conference on Very Large Data Bases, August 21-23, 1985, Stockholm, Sweden. pp. 82–95 (1985)
51. Bracewell, D.B., Russell, S., Wu, A.S.: Identification, expansion, and disambiguation of acronyms in biomedical texts. In: Parallel and Distributed Processing and Applications - ISPA 2005 Workshops, ISPA 2005 International Workshops AEPP, ASTD, BIOS, GCIC, IADS, MASN, SGCA, and WISA, Nanjing, China, November 2-5, 2005, Proceedings. pp. 186–195 (2005)
52. Brants, S., Dipper, S., Eisenberg, P., Hansen, S., König, E., Lezius, W., Rohrer, C., Smith, G., Uszkoreit, H.: TIGER: Linguistic Interpretation of a German Corpus. Journal of Language and Computation 2(4), 597–620 (2004)
53. Brants, T.: Tnt: a statistical part-of-speech tagger. In: Proceedings of the sixth conference on Applied natural language processing. pp. 224–231. Association for Computational Linguistics (2000)
54. Breaux, T.D., Vail, M.W., Antón, A.I.: Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations. In: 14th IEEE International Conference on Requirements Engineering (RE 2006), 11-15 September 2006, Minneapolis/St.Paul, Minnesota, USA. pp. 46–55 (2006)
55. vom Brocke, J., Rosemann, M.: Handbook on Business Process Management 1: Introduction, Methods, and Information Systems. Springer Publishing Company, Incorporated, 1st edn. (2010)
56. vom Brocke, J., Simons, A., Niehaves, B., Riemer, K., Plattfaut, R., Cleven, A.: Reconstructing the giant: On the importance of rigour in documenting the literature search process. In: 17th European Conference on Information Systems, ECIS 2009, Verona, Italy, 2009. pp. 2206–2217 (2009)
57. Bron, C., Kerbosch, J.: Finding all cliques of an undirected graph (algorithm 457). Commun. ACM 16(9), 575–576 (1973)
58. Brown, L.D., Cai, T.T., DasGupta, A.: Interval estimation for a binomial proportion. Statistical Science pp. 101–117 (2001)
59. Brown, P.F., Cocke, J., Della Pietra, S.A., Della Pietra, V.J., Jelinek, F., Lafferty, J.D., Mercer, R.L., Roossin, P.S.: A Statistical Approach to Machine Translation. Computational Linguistics 16(2), 79–85 (1990)
60. Brown, P.F., Cocke, J., Pietra, S.A.D., Pietra, V.J.D., Jelinek, F., Lafferty, J.D., Mercer, R.L., Roossin, P.S.: A statistical approach to machine translation. Computational Linguistics 16(2), 79–85 (1990)
61. Brown, R., Recker, J., West, S.: Using virtual worlds for collaborative business process modeling. Business Proc. Manag. Journal 17(3), 546–564 (2011)
62. Bühler, K.: Theory of Language: The Representational Function of Language. Foundations of Semiotics, J. Benjamins Publishing Company (1990 (1934))
63. Buijs, J.C.A.M., La Rosa, M., Reijers, H.A., van Dongen, B.F., van der Aalst, W.M.P.: Improving business process models using observed behavior. In: Data-Driven Process Discovery and Analysis - Second IFIP WG 2.6, 2.12 International Symposium, SIMPDA 2012, Campione d'Italia, Italy, June 18-20, 2012, Revised Selected Papers. pp. 44–59 (2012)
64. Cabré, M., Sager, J.: Terminology: Theory, Methods, and Applications. Terminology and Lexicography Research and Practice, J. Benjamins Publishing Company (1999)
65. Cai, J.F., Lee, W.S., Teh, Y.W.: Nus-ml: Improving word sense disambiguation using topic features. In: Proceedings of the 4th International Workshop on

Semantic Evaluations. pp. 249–252. Association for Computational Linguistics (2007)
66. Cardoso, E.C.S., Almeida, J.P.A., Guizzardi, G.: Requirements engineering based on business process models: A case study. In: Workshops Proceedings of the 12th IEEE International Enterprise Distributed Object Computing Conference, EDOCw 2009, 1-4 September 2009, Auckland, New Zealand. pp. 320–327 (2009)
67. Ceccato, M., Kiyavitskaya, N., Zeni, N., Mich, L., Berry, D.M.: Ambiguity identification and measurement in natural language texts. Tech. rep., University of Trento (2004)
68. Cer, D.M., de Marneffe, M., Jurafsky, D., Manning, C.D.: Parsing to stanford dependencies: Trade-offs between speed and accuracy. In: Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta (2010)
69. Chan, Y.S., Ng, H.T., Zhong, Z.: NUS-PT: exploiting parallel texts for word sense disambiguation in the English all-words tasks. In: Proceedings of the 4th International Workshop on Semantic Evaluations. pp. 253–256. Association for Computational Linguistics (2007)
70. Chang, A.X., Manning, C.D.: Sutime: A library for recognizing and normalizing time expressions. In: Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012), Istanbul, Turkey, May 23-25, 2012. pp. 3735–3740 (2012)
71. Chantree, F., Nuseibeh, B., Roeck, A.N.D., Willis, A.: Identifying nocuous ambiguities in natural language requirements. In: 14th IEEE International Conference on Requirements Engineering (RE 2006), 11-15 September 2006, Minneapolis/St.Paul, Minnesota, USA. pp. 56–65 (2006)
72. Chapman, R.L., Roget, P.M., et al.: Roget's international Thesaurus. Harper & Row (1984)
73. Charmaz, K.: Grounded theory methods in social justice research. The Sage handbook of qualitative research 4, 359–380 (2011)
74. Charniak, E.: Statistical parsing with a context-free grammar and word statistics. In: Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island. pp. 598–603 (1997)
75. Chomsky, N.: Syntactic Structures. Mouton classic, Mouton De Gruyter (2002)
76. Christiansen, D.R., Carbone, M., Hildebrandt, T.T.: Formal semantics and implementation of BPMN 2.0 inclusive gateways. In: Web Services and Formal Methods - 7th International Workshop, WS-FM 2010, Hoboken, NJ, USA, September 16-17, 2010. Revised Selected Papers. pp. 146–160 (2010)
77. Claes, J., Vanderfeesten, I., Gailly, F., Grefen, P., Poels, G.: The structured process modeling theory (spmt) a cognitive view on why and how modelers benefit from structuring the process of process modeling. Information Systems Frontiers pp. 1–25 (2015)
78. Claes, J., Vanderfeesten, I.T.P., Pinggera, J., Reijers, H.A., Weber, B., Poels, G.: A visual analysis of the process of process modeling. Inf. Syst. E-Business Management 13(1), 147–190 (2015)
79. Claes, J., Vanderfeesten, I.T.P., Reijers, H.A., Pinggera, J., Weidlich, M., Zugal, S., Fahland, D., Weber, B., Mendling, J., Poels, G.: Tying process model quality to the modeling process: The impact of structuring, movement, and speed. In:

Business Process Management - 10th International Conference, BPM 2012, Tallinn, Estonia, September 3-6, 2012. Proceedings. pp. 33–48 (2012)

80. Clever, N., Holler, J., Shitkova, M., Becker, J.: Towards auto-suggested process modeling - prototypical development of an auto-suggest component for process modeling tools. In: Enterprise Modelling and Information Systems Architectures: Proceedings of the 5th International Workshop on Enterprise Modelling and Information Systems Architectures, EMISA 2013, St. Gallen, Switzerland, September 5-6, 2013. pp. 133–145 (2013)
81. Cockburn, A.: Writing effective use cases, vol. 1. Addison-Wesley Boston (2001)
82. Collins, M.: Head-driven statistical models for natural language parsing. Computational Linguistics 29(4), 589–637 (2003)
83. Consortium, B.: The british national corpus, version 3 (bnc xml edition). Tech. rep., Oxford University Computing Services (2007)
84. Corston-Oliver, S., Gamon, M., Ringger, E., Moore, R.: An overview of amalgam: A machine-learned generation module. In: Proceedings of the International Natural Language Generation Conference. pp. 33–40 (2002)
85. Cox, K., Phalp, K.: Replicating the crews use case authoring guidelines experiment. Empirical Software Engineering 5(3), 245–267 (2000)
86. D'Aprile, D., Giordano, L., Martelli, A., Pozzato, G.L., Rognone, D., Dupré, D.T.: Business process compliance verification: An annotation based approach with commitments. In: Information systems: Crossroads for organization, management, accounting and engineering, pp. 563–570. Springer (2012)
87. Davenport, T.: Process Innovation: Reengineering Work Through Information Technology. Harvard Business School Press (1993)
88. Davenport, T.H., Short, J.E.: The new industrial engineering: Information technology and business process redesign. Sloan Management Review 31(4), 11–27 (1990)
89. Davies, M.: The corpus of contemporary american english: 450 million words, 1990-present (2008)
90. De Marneffe, M.C., MacCartney, B., Manning, C.D., et al.: Generating typed dependency parses from phrase structure parses. In: Proceedings of LREC. vol. 6, pp. 449–454 (2006)
91. De Marneffe, M.C., Manning, C.D.: Stanford typed dependencies manual. Tech. rep., Technical report, Stanford University (2008)
92. De Marneffe, M.C., Manning, C.D.: The stanford typed dependencies representation. In: Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation. pp. 1–8. Association for Computational Linguistics (2008)
93. Deeptimahanti, D.K., Sanyal, R.: Semi-automatic generation of uml models from natural language requirements. In: Proceedings of the 4th India Software Engineering Conference. pp. 165–174. ACM (2011)
94. Dehnert, J., Rittgen, P.: Relaxed soundness of business processes. In: Advanced Information Systems Engineering, 13th International Conference, CAiSE 2001, Interlaken, Switzerland, June 4-8, 2001, Proceedings. pp. 157–170 (2001)
95. Deissenboeck, F., Pizka, M.: Concise and consistent naming. Software Quality Journal 14(3), 261–282 (2006)
96. Delfmann, P., Herwig, S., Lis, L., Stein, A.: Supporting distributed conceptual modelling through naming conventions - A tool-based linguistic approach. Enterprise Modelling and Information Systems Architectures 4(2), 3–19 (2009)

97. Denger, C., Berry, D.M., Kamsties, E.: Higher quality requirements specifications through natural language patterns. In: 2003 IEEE International Conference on Software - Science, Technology and Engineering (SwSTE 2003), 4-5 November 2003, Herzelia, Israel. pp. 80–90 (2003)
98. Desel, J., Erwin, T.: Modeling, simulation and analysis of business processes. In: Business Process Management, Models, Techniques, and Empirical Studies. pp. 129–141 (2000)
99. Desel, J., Esparza, J.: Free choice Petri nets, vol. Volume 40 of Cambridge tracts in theoretical computer science. Cambridge University Press, Cambridge, UK (1995)
100. Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., Makhoul, J.: Fast and robust neural network joint models for statistical machine translation. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. pp. 1370—1380. Association for Computational Linguistics (2014)
101. Dijkman, R.M., Dumas, M., van Dongen, B.F., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Inf. Syst. 36(2), 498–516 (2011)
102. Dijkman, R.M., Dumas, M., García-Bañuelos, L.: Graph matching algorithms for business process model similarity search. In: Business Process Management, 7th International Conference, BPM 2009, Ulm, Germany, September 8-10, 2009. Proceedings. pp. 48–63 (2009)
103. Dijkman, R.M., Dumas, M., García-Bañuelos, L., Käärik, R.: Aligning business process models. In: Proceedings of the 13th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2009, 1-4 September 2009, Auckland, New Zealand. pp. 45–53 (2009)
104. Dijkman, R.M., Dumas, M., Ouyang, C.: Formal semantics and analysis of bpmn process models using petri nets. Tech. rep., Queensland University of Technology (2007)
105. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Information & Software Technology 50(12), 1281–1294 (2008)
106. van Dongen, B.F., Dijkman, R.M., Mendling, J.: Measuring similarity between business process models. In: Advanced Information Systems Engineering, 20th International Conference, CAiSE 2008, Montpellier, France, June 16-20, 2008, Proceedings. pp. 450–464 (2008)
107. Dowding, J., Moore, R., Andry, F., Moran, D.: Interleaving syntax and semantics in an efficient bottom-up parser. In: Proceedings of the 32nd annual meeting on Association for Computational Linguistics. pp. 110–116. Association for Computational Linguistics (1994)
108. Dumas, M., Rosa, M., Mendling, J., Reijers, H.: Fundamentals of Business Process Management. Springer (2013)
109. Dumas, M., Van der Aalst, W.M., Ter Hofstede, A.H.: Process-aware information systems: bridging people and software through process technology. John Wiley & Sons (2005)
110. Dumas, M., García-Bañuelos, L., Dijkman, R.M.: Similarity search of business process models. IEEE Data Eng. Bull. 32(3), 23–28 (2009)
111. Effinger, P., Jogsch, N., Seiz, S.: On a study of layout aesthetics for business process models using BPMN. In: Business Process Modeling Notation - Second International Workshop, BPMN 2010, Potsdam, Germany, October 13-14, 2010. Proceedings. pp. 31–45 (2010)

112. Effinger, P., Siebenhaller, M.: Improving business process visualizations. Tech. rep., Universität Tübingen (2009)
113. Eid-Sabbagh, R., Ahrend, N.: Eine prozessplattform für die deutsche verwaltung. In: Informatik 2013, 43. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Informatik angepasst an Mensch, Organisation und Umwelt, 16.-20. September 2013, Koblenz. pp. 648–662 (2013)
114. Eid-Sabbagh, R., Kunze, M., Weske, M.: An open process model library. In: Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part II. pp. 26–38 (2011)
115. Elhadad, M., Balaban, M., Sturm, A.: Effective business process outsourcing: The prosero approach. IBIS 6, 8–31 (2007)
116. Ellis, C.A., Nutt, G.J.: Office information systems and computer science. ACM Comput. Surv. 12(1), 27–60 (1980)
117. Erman, L.D., Hayes-Roth, F., Lesser, V.R., Reddy, R.: The hearsay-ii speech-understanding system: Integrating knowledge to resolve uncertainty. ACM Comput. Surv. 12(2), 213–253 (1980)
118. Etzioni, O., Cafarella, M., Downey, D., Popescu, A.M., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Unsupervised named-entity extraction from the web: An experimental study. Artificial intelligence 165(1), 91–134 (2005)
119. Fabbrini, F., Fusani, M., Gnesi, S., Lami, G.: Quality evaluation of software requirement specifications. In: Proceedings of the Software and Internet Quality Week 2000. pp. 1–18 (2000)
120. Faber, P.B., Usón, R.M.: Constructing a lexicon of English verbs, vol. 23. Walter de Gruyter (1999)
121. Fabra, J., De Castro, V., Álvarez, P., Marcos, E.: Automatic execution of business process models: Exploiting the benefits of model-driven engineering approaches. Journal of Systems and Software 85(3), 607–625 (2012)
122. Fahland, D., Favre, C., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Analysis on Demand: Instantaneous Soundness Checking of Industrial Business Process Models. Data & Knowledge Engineering 70(5), 448–466 (2011)
123. Fahland, D., van der Aalst, W.M.P.: Model repair - aligning process models to reality. Inf. Syst. 47, 220–243 (2015)
124. Fahland, D., van der Aalst, W.M.: Repairing process models to reflect reality. In: Business Process Management - 10th International Conference, BPM 2012, Tallinn, Estland, 2012. Proceedings., pp. 229–245. Springer (2012)
125. Fantechi, A., Gnesi, S., Lami, G., Maccari, A.: Applications of linguistic techniques for use case analysis. Requir. Eng. 8(3), 161–170 (2003)
126. Ferrari, A., Gnesi, S.: Using collective intelligence to detect pragmatic ambiguities. In: 2012 20th IEEE International Requirements Engineering Conference (RE), Chicago, IL, USA, September 24-28, 2012. pp. 191–200 (2012)
127. Ferrari, A., Lipari, G., Gnesi, S., Spagnolo, G.O.: Pragmatic ambiguity detection in natural language requirements. In: IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering, AIRE 2014, 26 August, 2014, Karlskrona, Sweden. pp. 1–8 (2014)
128. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. pp. 363–370. Association for Computational Linguistics (2005)

129. Forcada, M.L., Ginestí-Rosell, M., Nordfalk, J., O'Regan, J., Ortiz-Rojas, S., Pérez-Ortiz, J.A., Sánchez-Martínez, F., Ramírez-Sánchez, G., Tyers, F.M.: Apertium: a free/open-source platform for rule-based machine translation. Machine Translation 25(2), 127–144 (2011)
130. Ford, H.: My Life and Work. Dubleday, Page and Company (1923)
131. Fowler, M.: Refactoring: improving the design of existing code. Pearson Education India (1999)
132. Frakes, W.B., Pole, T.P.: An empirical study of representation methods for reusable software components. IEEE Trans. Software Eng. 20(8), 617–630 (1994)
133. Francis, W.N., Kucera, H.: Brown corpus manual. Tech. rep., Department of Linguistics, Brown University, Providence, Rhode Island, US (1979)
134. Frederiks, P.J.M., van der Weide, T.P.: Information modeling: The process and the required competencies of its participants. Data Knowl. Eng. 58(1), 4–20 (2006)
135. Friedrich, F.: Measuring semantic label quality using wordnet. In: EPK. pp. 7–21 (2009)
136. Fromkin, V., Rodman, R., Hyams, N.: An Introduction to Language. Cengage Learning (2013)
137. Gadatsch, A.: Grundkurs Geschäftsprozess-Management: Methoden und Werkzeuge für die IT-Praxis: Eine Einführung für Studenten und Praktiker. Vieweg, 5 edn. (2005)
138. Gambini, M., La Rosa, M., Migliorini, S., ter Hofstede, A.H.M.: Automated error correction of business process models. In: Business Process Management - 9th International Conference, BPM 2011, Clermont-Ferrand, France, August 30 - September 2, 2011. Proceedings. pp. 148–165 (2011)
139. Gazdar, G.: Generalized phrase structure grammar. Harvard University Press (1985)
140. Ghose, A., Koliadis, G.: Auditing business process compliance. In: Service-Oriented Computing - ICSOC 2007, Fifth International Conference, Vienna, Austria, September 17-20, 2007, Proceedings. pp. 169–180 (2007)
141. Glas, R.: Das limas-korpus, ein textkorpus für die deutsche gegenwartssprache. Linguistische Berichte 40(1975), 63–66 (1975)
142. Gleich, B., Creighton, O., Kof, L.: Ambiguity detection: Towards a tool explaining ambiguity sources. In: Requirements Engineering: Foundation for Software Quality, 16th International Working Conference, REFSQ 2010, Essen, Germany, June 30 - July 2, 2010. Proceedings. pp. 218–232 (2010)
143. Gomez, F., Segami, C., Delaune, C.: A system for the semiautomatic generation of E-R models from natural language specifications. Data Knowl. Eng. 29(1), 57–81 (1999)
144. Gordijn, J., Akkermans, H., van Vliet, H.: Business modelling is not process modelling. In: Conceptual Modeling for E-Business and the Web, ER 2000 Workshops on Conceptual Modeling Approaches for E-Business and The World Wide Web and Conceptual Modeling, Salt Lake City, Utah, USA, October 9-12, 2000, Proceedings. pp. 40–51 (2000)
145. Greene, B., Rubin, G.: Automated grammatical tagging of english. Tech. rep., Department of Linguistics, Brown University (1971)
146. Gregor, S.: The nature of theory in information systems. MIS Quarterly 30, 611–642 (2006)
147. Grice, H.: Logic and conversation. In: Cole, P., Morgan, J. (eds.) Syntax and Semantics Volume 3: Speech Acts. Academic Press, New York (1975)

148. Grochla, E.: Grundlagen der organisatorischen Gestaltung. Poeschel Stuttgart (1982)
149. Gruhn, V., Laue, R.: Validierung syntaktischer und anderer epk-eigenschaften mit PROLOG. In: 5. Workshop der Gesellschaft für Informatik e.V. (GI) und Treffen ihres Arbeitskreises "Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (WI-EPK)", Wien, 30. November - 01. Dezember 2006. pp. 69–84 (2006)
150. Gruhn, V., Laue, R.: Checking properties of business process models with logic programming. In: Modelling, Simulation, Verification and Validation of Enterprise Information Systems, Proceedings of the 5th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, MSVVEIS-2007, In conjunction with ICEIS 2007, Funchal, Madeira, Portugal, June 2007. pp. 84–93 (2007)
151. Gruhn, V., Laue, R.: A heuristic method for detecting problems in business process models. Business Process Management Journal 16(5), 806–821 (2010)
152. Gruhn, V., Laue, R.: Detecting common errors in event-driven process chains by label analysis. Enterprise Modelling and Information Systems Architectures 6(1), 3–15 (2011)
153. Gschwind, T., Koehler, J., Wong, J., Favre, C., Kleinoeder, W., Maystrenko, A., Muhidini, K.: Ibm pattern-based process model accelerators for websphere business modeler. Tech. rep., IBM Research (2009)
154. Hammer, M., Champy, J.: Reengineering the Corporation: A Manifesto for Business Revolution. HarperBusiness (1993)
155. Hamp, B., Feldweg, H.: Germanet - a lexical-semantic net for german. In: Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications. pp. 9–15 (1997)
156. Harris, Z.S.: String Analysis of Sentence Structure. Mouton, The Hague (1962)
157. Havel, J., Steinhorst, M., Dietrich, H., Delfmann, P.: Supporting terminological standardization in conceptual models - a plugin for a meta-modelling tool. In: 22nt European Conference on Information Systems, ECIS 2014, Tel Aviv, Israel, June 9-11, 2014 (2014)
158. Hayne, S., Ram, S.: Multi-user view integration system (MUVIS): an expert system for view integration. In: Proceedings of the Sixth International Conference on Data Engineering, February 5-9, 1990, Los Angeles, California, USA. pp. 402–409 (1990)
159. Heitmeyer, C.L., Jeffords, R.D., Labaw, B.G.: Automated consistency checking of requirements specifications. ACM Trans. Softw. Eng. Methodol. 5(3), 231–261 (1996)
160. Henderson-Sellers, B.: On the Mathematics of Modelling, Metamodelling, Ontologies and Modelling Languages. Springer (2012)
161. Hernandez-del-Olmo, F., Gaudioso, E.: Evaluation of recommender systems: A new approach. Expert Syst. Appl. 35(3), 790–804 (2008)
162. Hevner, A., March, S., Park, J., Ram, S.: Design Science in Information Systems Research. MIS Quarterly 28(1), 75–105 (2004)
163. Hiai, F., Kosaki, H.: Meaning in Language: An Introduction to Semantics and Pragmatics. New York: Oxford University Press (2000)
164. Hinkelman, E.A., Allen, J.F.: Two Constraints on Speech Act Ambiguity. In: Proceedings of the 27th annual meeting on Association for Computational Linguistics. pp. 212–219. Association for Computational Linguistics (1989)

165. Hornby, A.S., Cowie, A.P., Gimson, A.C., Lewis, J.W.: Oxford Advanced Learner's Dictionary of Current English, vol. 1428. Cambridge Univ Press (1974)
166. Hornung, T., Koschmider, A., Lausen, G.: Recommendation based process modeling support: Method and user experience. In: Conceptual Modeling - ER 2008, 27th International Conference on Conceptual Modeling, Barcelona, Spain, October 20-24, 2008. Proceedings. pp. 265–278 (2008)
167. Hornung, T., Koschmider, A., Oberweis, A.: Rule-based autocompletion of business process models. In: CAiSE'07 Forum, Proceedings of the CAiSE'07 Forum at the 19th International Conference on Advanced Information Systems Engineering, Trondheim, Norway, 11-15 June 2007 (2007)
168. Hornung, T., Koschmider, A., Oberweis, A.: A recommender system for business process models. In: Workshop on Information Technologies & Systems (2009)
169. Hovy, E., Marcu, D.: Automated text summarization. The Oxford handbook of computational linguistics pp. 583–598 (2005)
170. Hutchins, W.J.: Machine Translation: Past, Present, Future. John Wiley & Sons, New York, NY, USA (1986)
171. Ide, N., Suderman, K.: The american national corpus first release. In: LREC. Citeseer (2004)
172. Ide, N., Véronis, J.: Introduction to the special issue on word sense disambiguation: The state of the art. Computational Linguistics 24(1), 1–40 (1998)
173. Ide, N., Véronis, J.: Introduction to the special issue on word sense disambiguation: The state of the art. Computational Linguistics 24(1), 1–40 (1998)
174. IEEE Computer Society: Ieee guide–adoption of the project management institute (pmi(r)) standard a guide to the project management body of knowledge (pmbok(r) guide)–fourth edition (2011)
175. Imai, M.: Kaizen: Der Schlüssel zum Erfolg der Japaner im Wettbewerb. Langen Müller/Herbig, München, 3rd edn. (1992)
176. Jaszczolt, K.: Semantics and pragmatics: Meaning in language and discourse. Pearson Education (2002)
177. Jayal, A., Shepperd, M.J.: The problem of labels in e-assessment of diagrams. ACM Journal of Educational Resources in Computing 8(4) (2009)
178. Jelinek, F.: Statistical Methods for Speech Recognition. The MIT Press (1998)
179. Jeston, J., Nelis, J.: Business process management. Routledge (2014)
180. Johnston, M., Busa, F.: Qualia structure and the compositional interpretation of compounds. In: Breadth and depth of semantic lexicons, pp. 167–187. Springer (1999)
181. Jones, T., Schulte, W.R., Cantara, M.: Magic quadrant for Intelligent Business Process Management Suites. Tech. rep., Gartner (2014)
182. Jordan, D., Evdemon, J., Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., et al.: Web services business process execution language version 2.0. OASIS standard 11, 10 (2007)
183. Jurafsky, D., Bates, R., Coccaro, N., Martin, R., Meteer, M., Ries, K., Shriberg, E., Stolcke, A., Taylor, P., Ess-Dykema, V., et al.: Automatic detection of discourse structure for speech recognition and understanding. In: Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on. pp. 88–95. IEEE (1997)
184. Jurafsky, D., Martin, J.H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. Prentice Hall, second edn. (2008)

185. Kaiya, H., Saeki, M.: Ontology based requirements analysis: Lightweight semantic processing approach. In: 2005 NASA / DoD Conference on Evolvable Hardware (EH 2005), 29 June - 1 July 2005, Washington, DC, USA. pp. 223–230 (2005)
186. Kamsties, E.: Understanding ambiguity in requirements engineering. In: Aurum, A., Wohlin, C. (eds.) Engineering and Managing Software Requirements, pp. 245–266. Springer (2005)
187. Kamsties, E., Berry, D.M., Paech, B.: Detecting ambiguities in requirements documents using inspections. In: Inspections in Software Engineering (2001). pp. 68–80 (2001)
188. Kamsties, E., Paech, B.: Taming ambiguity in natural language requirements. In: Proceedings of the Thirteenth International Conference on Software and Systems Engineering and Applications (2000)
189. Karagiannis, D., Kühn, H.: Metamodelling platforms. In: E-Commerce and Web Technologies, Third International Conference, EC-Web 2002, Aix-en-Provence, France, September 2-6, 2002, Proceedings. p. 182 (2002)
190. Keller, G., Teufel, T.: SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping. Addison-Wesley (1998)
191. Kemper, P., Bause, F.: An efficient polynomial-time algorithm to decide liveness and boundedness of free-choice nets. In: Application and Theory of Petri Nets 1992, 13th International Conference, Sheffield, UK, June 22-26, 1992, Proceedings. pp. 263–278 (1992)
192. Kern, R., Muhr, M., Granitzer, M.: KCDC: Word sense induction by using grammatical dependencies and sentence phrase structure. In: Proceedings of the 5th international workshop on semantic evaluation. pp. 351–354. Association for Computational Linguistics (2010)
193. Khan, R.N.: Business Process Management: a practical guide. Meghan-Kiffer Press (2004)
194. Kilgarriff, A.: Gold standard datasets for evaluating word sense disambiguation programs. Computer Speech & Language 12(4), 453–472 (1998)
195. Kilgarriff, A., Yallop, C.: What's in a thesaurus? In: Proceedings of the Second International Conference on Language Resources and Evaluation, LREC 2000, 31 May - June 2, 2000, Athens, Greece (2000)
196. Kim, S., Alani, H., Hall, W., Lewis, P., Millard, D., Shadbolt, N., Weal, M.: Artequakt: Generating tailored biographies with automatically annotated fragments from the web, semantic authoring. In: Annotation and Knowledge Markup Workshop in the 15th European Conference on Artificial Intelligence (2002)
197. Kingsbury, P., Palmer, M., Marcus, M.: Adding semantic annotation to the penn treebank. In: Proceedings of the Human Language Technology Conference. pp. 252–256. Citeseer (2002)
198. Kiyavitskaya, N., Zeni, N., Mich, L., Berry, D.M.: Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Requir. Eng. 13(3), 207–239 (2008)
199. Klein, D., Manning, C.D.: Fast exact inference with a factored model for natural language parsing. In: Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]. pp. 3–10 (2002)
200. Klein, D., Manning, C.D.: Accurate unlexicalized parsing. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 7-12 July 2003, Sapporo Convention Center, Sapporo, Japan. pp. 423–430 (2003)

201. Klein, S., Simmons, R.F.: A computational approach to grammatical coding of english words. J. ACM 10(3), 334–347 (1963)
202. Klinkmüller, C., Leopold, H., Weber, I., Mendling, J., Ludwig, A.: Listen to me: Improving process model matching through user feedback. In: Business Process Management - 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings. pp. 84–100 (2014)
203. Klinkmüller, C., Weber, I., Mendling, J., Leopold, H., Ludwig, A.: Increasing recall of process model matching by improved activity label matching. In: Business Process Management - 11th International Conference, BPM 2013, Beijing, China, August 26-30, 2013. Proceedings. pp. 211–218 (2013)
204. Kluza, K., Baran, M., Bobek, S., Nalepa, G.J.: Overview of recommendation techniques in business process modeling. In: Proceedings of 9th Workshop on Knowledge Engineering and Software Engineering (KESE9) co-located with the 36th German Conference on Artificial Intelligence (KI2013), Koblenz, Germany, September 17, 2013. (2013)
205. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al.: Moses: Open source toolkit for statistical machine translation. In: Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions. pp. 177–180. Association for Computational Linguistics (2007)
206. Koeling, R., McCarthy, D.: Sussx: Wsd using automatically acquired predominant senses. In: Proceedings of the 4th International Workshop on Semantic Evaluations. pp. 314–317. Association for Computational Linguistics (2007)
207. Kofax: Market analysis of multichannel capture, business process management, and smart process applications, 2013 through 2016 (2013)
208. Koschmider, A., Blanchard, E.: User assistance for business process model decomposition. In: Proceedings of the 1st IEEE International Conference on Research Challenges in Information Science. pp. 445–454 (2007)
209. Koschmider, A., Oberweis, A.: Designing business processes with a recommendation-based editor. In: Handbook on Business Process Management 1, pp. 299–312. Springer (2010)
210. Koschmider, A., Oberweis, A.: Recommendation-based business processes design. In: Handbook on Business Process Management 1, Introduction, Methods, and Information Systems, 2nd Ed., pp. 323–336. Springer (2015)
211. Koschmider, A., Reijers, H., Dijkman, R.: Empirical support for the usefulness of personalized process model views. In: Multikonferenz Wirtschaftsinformatik (2012)
212. Koschmider, A., Ullrich, M., Heine, A., Oberweis, A.: Revising the vocabulary of business process element labels. In: Advanced Information Systems Engineering - 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings. pp. 69–83 (2015)
213. Kosiol, E.: Organisation der Unternehmung. Gabler Verlag, Wiesbaden (1962)
214. Kösters, G., Six, H., Winter, M.: Coupling use cases and class models as a means for validation and verification of requirements specifications. Requir. Eng. 6(1), 3–17 (2001)
215. Krallmann, H., Bobrik, A., Levina, O.: Systemanalyse im Unternehmen: Prozessorientierte Methoden der Wirtschaftsinformatik. Oldenbourg Verlag (2013)
216. Krallmann, H., Frank, H., Gronau, N.: Systemanalyse im Unternehmen. Oldenbourg (1994)

217. Kroeger, P.R.: Analyzing grammar: An introduction. Cambridge University Press (2005)
218. Krogstie, J., Sindre, G., Jørgensen, H.: Process Models Representing Knowledge for Action: a Revised Quality Framework. European Journal of Information Systems 15(1), 91–102 (2006)
219. Krogstie, J., Lindland, O.I., Sindre, G.: Defining quality aspects for conceptual models. In: Information System Concepts: Towards a consolidation of views, Proceedings of the IFIP international working conference on information system concepts, Marburg, Germany, 28-30 March 1995. pp. 216–231 (1995)
220. Krogstie, J., Sindre, G., Jorgensen, H.: Process models representing knowledge for action: a revised quality framework. European Journal of Information Systems 15(1), 91–102 (2006)
221. Kumar, E.: Natural Language Processing. I.K. International Publishing House (2011)
222. Kunze, M., Weske, M.: Metric trees for efficient similarity search in large process model repositories. In: Business Process Management Workshops - BPM 2010 International Workshops and Education Track, Hoboken, NJ, USA, September 13-15, 2010, Revised Selected Papers. pp. 535–546 (2010)
223. La Rosa, M., ter Hofstede, A.H.M., Wohed, P., Reijers, H.A., Mendling, J., van der Aalst, W.M.P.: Managing process model complexity via concrete syntax modifications. IEEE Trans. Industrial Informatics 7(2), 255–265 (2011)
224. La Rosa, M., Lux, J., Seidel, S., Dumas, M., ter Hofstede, A.H.M.: Questionnaire-driven configuration of reference process models. In: Advanced Information Systems Engineering, 19th International Conference, CAiSE 2007, Trondheim, Norway, June 11-15, 2007, Proceedings. pp. 424–438 (2007)
225. La Rosa, M., Reijers, H.A., van der Aalst, W.M.P., Dijkman, R.M., Mendling, J., Dumas, M., García-Bañuelos, L.: APROMORE: an advanced process model repository. Expert Syst. Appl. 38(6), 7029–7040 (2011)
226. La Rosa, M., Wohed, P., Mendling, J., Ter Hofstede, A.H., Reijers, H., van der Aalst, W.M.P.: Managing Process Model Complexity via Abstract Syntax Modifications. IEEE Transactions on Industrial Informatics 7(4), 614–629 (2011)
227. Lami, G.: QuARS: A Tool for Analyzing Requirements. Tech. rep., DTIC Document (2005)
228. Lanz, A., Weber, B., Reichert, M.: Workflow time patterns for process-aware information systems. In: Enterprise, Business-Process and Information Systems Modeling - 11th International Workshop, BPMDS 2010, and 15th International Conference, EMMSAD 2010, held at CAiSE 2010, Hammamet, Tunisia, June 7-8, 2010. Proceedings. pp. 94–107 (2010)
229. Laue, R., Mendling, J.: The impact of structuredness on error probability of process models. In: Information Systems and e-Business Technologies, 2nd International United Information Systems Conference, UNISCON 2008, Klagenfurt, Austria, April 22-25, 2008, Proceedings. pp. 585–590 (2008)
230. Laue, R., Mendling, J.: Structuredness and its significance for correctness of process models. Inf. Syst. E-Business Management 8(3), 287–307 (2010)
231. Laue, R., Mendling, J.: Structuredness and its significance for correctness of process models. Inf. Syst. E-Business Management 8(3), 287–307 (2010)
232. Lavoie, B., Rainbow, O.: A fast and portable realizer for text generation systems. In: ANLP. pp. 265–268 (1997)

233. Lee, K., Kang, K.C., Lee, J.: Concepts and guidelines of feature modeling for product line software engineering. In: Software Reuse: Methods, Techniques, and Tools, 7th International Conference, ICSR-7, Austin, TX, USA, April 15-19, 2002, Proceedings. pp. 62–77 (2002)
234. Lehmann, T.: A framework for ontology based integration of structured it-systems. In: SWESE 2007 (2007)
235. Leopold, H., Mendling, J., Günther, O.: What we can learn from Quality Issues of BPMN Models from Industry. Software, IEEE PP(99), 1–1 (2015)
236. Leopold, H.: Natural Language in Business Process Models. Ph.D. thesis, Humboldt Universität zu Berlin (2013)
237. Leopold, H.: Natural Language in Business Process Models - Theoretical Foundations, Techniques, and Applications, Lecture Notes in Business Information Processing, vol. 168. Springer (2013)
238. Leopold, H., Eid-Sabbagh, R., Mendling, J., Azevedo, L.G., Baião, F.A.: Detection of naming convention violations in process models for different languages. Decision Support Systems 56, 310–325 (2013)
239. Leopold, H., Meilicke, C., Fellmann, M., Pittke, F., Stuckenschmidt, H., Mendling, J.: Towards the automated annotation of process models. In: Advanced Information Systems Engineering - 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings. pp. 401–416 (2015)
240. Leopold, H., Mendling, J.: Automatic derivation of service candidates from business process model repositories. In: Business Information Systems - 15th International Conference, BIS 2012, Vilnius, Lithuania, May 21-23, 2012. Proceedings. pp. 84–95 (2012)
241. Leopold, H., Mendling, J., Polyvyanyy, A.: Generating natural language texts from business process models. In: Advanced Information Systems Engineering - 24th International Conference, CAiSE 2012, Gdansk, Poland, June 25-29, 2012. Proceedings. pp. 64–79 (2012)
242. Leopold, H., Mendling, J., Polyvyanyy, A.: Supporting process model validation through natural language generation. IEEE Trans. Software Eng. 40(8), 818–840 (2014)
243. Leopold, H., Mendling, J., Reijers, H.A.: On the automatic labeling of process models. In: Advanced Information Systems Engineering - 23rd International Conference, CAiSE 2011, London, UK, June 20-24, 2011. Proceedings. pp. 512–520 (2011)
244. Leopold, H., Mendling, J., Reijers, H.A., La Rosa, M.: Simplifying process model abstraction: Techniques for generating model names. Inf. Syst. 39, 134–151 (2014)
245. Leopold, H., Niepert, M., Weidlich, M., Mendling, J., Dijkman, R.M., Stuckenschmidt, H.: Probabilistic optimization of semantic process model matching. In: Business Process Management - 10th International Conference, BPM 2012, Tallinn, Estonia, September 3-6, 2012. Proceedings. pp. 319–334 (2012)
246. Leopold, H., Pittke, F., Mendling, J.: Towards measuring process model granularity via natural language analysis. In: 4th International Workshop on Process Model Collections: Management and Reuse (PMC-MR 2013), Beijing, China (2013)
247. Leopold, H., Pittke, F., Mendling, J.: Automatic service derivation from business process model repositories via semantic technology. Journal of Systems and Software 108, 134–147 (2015)

248. Leopold, H., Smirnov, S., Mendling, J.: Refactoring of process model activity labels. In: Natural Language Processing and Information Systems, 15th International Conference on Applications of Natural Language to Information Systems, NLDB 2010, Cardiff, UK, June 23-25, 2010. Proceedings. pp. 268–276 (2010)
249. Leopold, H., Smirnov, S., Mendling, J.: On the refactoring of activity labels in business process models. Inf. Syst. 37(5), 443–459 (2012)
250. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. Cybernetics and Control Theory 10(8), 707–710 (1966)
251. Li, X., Roth, D., Tu, Y.: PhraseNet: Towards Context Sensitive Lexical Semantics. In: Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4. pp. 87–94. Association for Computational Linguistics (2003)
252. Lin, D.: Automatic retrieval and clustering of similar words. In: 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, COLING-ACL '98, August 10-14, 1998, Université de Montréal, Montréal, Quebec, Canada. Proceedings of the Conference. pp. 768–774 (1998)
253. Lin, D.: An information-theoretic definition of similarity. In: Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998. pp. 296–304 (1998)
254. Lin, X., Chan, L.M.: Personalized knowledge organization and access for the web. Library & information science research 21(2), 153–172 (1999)
255. Lindland, O.I., Sindre, G., Sølvberg, A.: Understanding quality in conceptual modeling. IEEE Software 11(2), 42–49 (1994)
256. Liu, H., Singh, P.: Conceptnet–a practical commonsense reasoning tool-kit. BT technology journal 22(4), 211–226 (2004)
257. Loisel, A., Duplessis, G.D., Chaignaud, N., Kotowicz, J.P., Pauchet, A.: A Conversational Agent for Information Retrieval based on a Study of Human Dialogues. In: International Conference on Agent and Artificial Intelligence. pp. 312–317 (2012)
258. Ludewig, J.: Models in software engineering - an introduction. Inform., Forsch. Entwickl. 18(3-4), 105–112 (2004)
259. Malesevic, A., Brdjanin, D., Maric, S.: Tool for automatic layout of business process model represented by UML activity diagram. In: Proceedings of Eurocon 2013, International Conference on Computer as a Tool, Zagreb, Croatia, July 1-4, 2013. pp. 537–542 (2013)
260. Mallery, J.C.: Thinking about foreign policy: Finding an appropriate role for artificially intelligent computers. In: MIT Political Science Department. Citeseer (1988)
261. Malone, T., Crowston, K., Herman, G. (eds.): Organizing Business Knowledge: The MIT Process Handbook. The MIT Press (2003)
262. Mani, I., Maybury, M.T.: Advances in automatic text summarization, vol. 293. MIT Press (1999)
263. March, S.T., Smith, G.F.: Design and natural science research on information technology. Decision Support Systems 15(4), 251–266 (1995)
264. Marcus, A., Maletic, J.I., Sergeyev, A.: Recovery of traceability links between software documentation and source code. International Journal of Software Engineering and Knowledge Engineering 15(5), 811–836 (2005)

265. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of english: The penn treebank. Computational Linguistics 19(2), 313–330 (1993)
266. Markovic, I., Pereira, A.C.: Towards a formal framework for reuse in business process modeling. In: Business Process Management Workshops, BPM 2007 International Workshops, BPI, BPD, CBP, ProHealth, RefMod, semantics4ws, Brisbane, Australia, September 24, 2007, Revised Selected Papers. pp. 484–495 (2007)
267. Martens, A.: On compatibility of web services. Petri Net Newsletter 65(12-20), 100 (2003)
268. Martens, A.: Wombat4ws–workflow modeling and business analysis toolkit for web services (2003)
269. Martens, A.: Analyzing web service based business processes. In: Fundamental Approaches to Software Engineering, 8th International Conference, FASE 2005, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005, Proceedings. pp. 19–33 (2005)
270. McCarthy, D., Koeling, R., Weeds, J., Carroll, J.: Unsupervised acquisition of predominant word senses. Computational Linguistics 33(4), 553–590 (2007)
271. McCarthy, D., Koeling, R., Weeds, J., Carroll, J.A.: Finding predominant word senses in untagged text. In: Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain. pp. 279–286 (2004)
272. Mellor, S.J.: MDA distilled: principles of model-driven architecture. Addison-Wesley Professional (2004)
273. Mendling, J.: Detection and prediction of errors in EPC business process models. Ph.D. thesis, Wirtschaftsuniversität Wien Vienna (2007)
274. Mendling, J.: Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness, Lecture Notes in Business Information Processing, vol. 6. Springer (2008)
275. Mendling, J.: Empirical studies in process model verification. T. Petri Nets and Other Models of Concurrency 2, 208–224 (2009)
276. Mendling, J., van der Aalst, W.M.P.: Formalization and verification of epcs with or-joins based on state and context. In: Advanced Information Systems Engineering, 19th International Conference, CAiSE 2007, Trondheim, Norway, June 11-15, 2007, Proceedings. pp. 439–453 (2007)
277. Mendling, J., van Dongen, B.F., van der Aalst, W.M.P.: Getting rid of the or-join in business process models. In: 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), 15-19 October 2007, Annapolis, Maryland, USA. pp. 3–14 (2007)
278. Mendling, J., Leopold, H., Pittke, F.: 25 challenges of semantic process modeling. International Journal of Information Systems and Software Engineering for Big Companies (IJISEBC) 1(1), 78–94. (2015)
279. Mendling, J., Neumann, G., van der Aalst, W.M.P.: Understanding the occurrence of errors in process models based on metrics. In: On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, OTM Confederated International Conferences CoopIS, DOA, ODBASE, GADA, and IS 2007, Vilamoura, Portugal, November 25-30, 2007, Proceedings, Part I. pp. 113–130 (2007)

280. Mendling, J., Nüttgens, M.: EPC modelling based on implicit arc types. In: Information Systems Technology and its Applications, International Conference ISTA'2003, June 19-21, 2003, Kharkiv, Ukraine, Proceedings. pp. 131–142 (2003)
281. Mendling, J., Nüttgens, M.: EPC syntax validation with XML schema languages. In: EPK 2003 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, Proceedings des GI-Workshops und Arbeitskreistreffens (Bamberg, Oktober 2003). pp. 19–30 (2003)
282. Mendling, J., Recker, J., Reijers, H.A.: On the usage of labels and icons in business process modeling. IJISMD 1(2), 40–58 (2010)
283. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7PMG). Information & Software Technology 52(2), 127–136 (2010)
284. Mendling, J., Reijers, H.A., Cardoso, J.: What makes process models understandable? In: Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings. pp. 48–63 (2007)
285. Mendling, J., Reijers, H.A., Recker, J.: Activity labeling in process modeling: Empirical insights and recommendations. Inf. Syst. 35(4), 467–482 (2010)
286. Mendling, J., Verbeek, H.M.W., van Dongen, B.F., van der Aalst, W.M.P., Neumann, G.: Detection and prediction of errors in epcs of the SAP reference model. Data Knowl. Eng. 64(1), 312–329 (2008)
287. Mens, T., Straeten, R.V.D., D'Hondt, M.: Detecting and resolving model inconsistencies using transformation dependency analysis. In: Model Driven Engineering Languages and Systems, 9th International Conference, MoDELS 2006, Genova, Italy, October 1-6, 2006, Proceedings. pp. 200–214 (2006)
288. Mens, T., Taentzer, G., Runge, O.: Analysing refactoring dependencies using graph transformation. Software and System Modeling 6(3), 269–285 (2007)
289. Meurers, D.: Natural language processing and language learning. The Encyclopedia of Applied Linguistics (2012)
290. Mili, A., Mili, R., Mittermeir, R.: A survey of software reuse libraries. Ann. Software Eng. 5, 349–414 (1998)
291. Miller, G., Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge, MA (1998)
292. Miller, G.A.: WordNet: a Lexical Database for English. Communications of the ACM 38(11), 39–41 (1995)
293. Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.J.: Introduction to wordnet: An on-line lexical database. International journal of lexicography 3(4), 235–244 (1990)
294. Moha, N., Guéhéneuc, Y.G., Duchien, L., Le Meur, A.F.: DECOR: A method for the specification and detection of code and design smells. IEEE Transactions on Software Engineering 36(1), 20–36 (2010)
295. Montes, A., Pacheco, H., Estrada, H., Pastor, O.: Conceptual model generation from requirements model: A natural language processing approach. In: Natural Language and Information Systems, 13th International Conference on Applications of Natural Language to Information Systems, NLDB 2008, London, UK, June 24-27, 2008, Proceedings. pp. 325–326 (2008)
296. Morris, C.W.: Foundations of the Theory of Signs. University of Chicago Press (1938)

297. zur Muehlen, M., Recker, J.: How much language is enough? theoretical and practical use of the business process modeling notation. In: Advanced Information Systems Engineering, 20th International Conference, CAiSE 2008, Montpellier, France, June 16-20, 2008, Proceedings. pp. 465–479 (2008)
298. zur Muehlen, M., Rosemann, M.: Multi-paradigm process management. In: CAiSE'04 Workshops in connection with The 16th Conference on Advanced Information Systems Engineering, Riga, Latvia, 7-11 June, 2004, Knowledge and Model Driven Information Systems Engineering for Networked Organisations, Proceedings, Vol. 2. pp. 169–175 (2004)
299. Murata, T.: Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE 77(4), 541–580 (1989)
300. Navigli, R.: Word sense disambiguation: A survey. ACM Comput. Surv. 41(2) (2009)
301. Navigli, R., Lapata, M.: An experimental study of graph connectivity for unsupervised word sense disambiguation. IEEE Trans. Pattern Anal. Mach. Intell. 32(4), 678–692 (2010)
302. Navigli, R., Ponzetto, S.P.: Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artificial Intelligence 193, 217–250 (2012)
303. Navigli, R., Ponzetto, S.P.: Joining forces pays off: Multilingual joint word sense disambiguation. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea. pp. 1399–1410 (2012)
304. Navigli, R., Ponzetto, S.P.: Multilingual WSD with just a few lines of code: the babelnet API. In: The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the System Demonstrations, July 10, 2012, Jeju Island, Korea. pp. 67–72 (2012)
305. Navigli, R., Velardi, P.: Structural semantic interconnections: A knowledge-based approach to word sense disambiguation. IEEE Trans. Pattern Anal. Mach. Intell. 27(7), 1075–1086 (2005)
306. Nenkova, A., McKeown, K.: A survey of text summarization techniques. In: Aggarwal, C.C., Zhai, C. (eds.) Mining Text Data, pp. 43–76. Springer US (2012)
307. Nichols, J.: Functional theories of grammar. Annual review of Anthropology pp. 97–117 (1984)
308. Niu, Z.Y., Ji, D.H., Tan, C.L.: I2r: Three systems for word sense discrimination, chinese word sense disambiguation, and english word sense disambiguation. In: Proceedings of the 4th International Workshop on Semantic Evaluations. pp. 177–182. Association for Computational Linguistics (2007)
309. Nordsieck, F.: Grundlagen der Organisationslehre. Poeschel (1934)
310. Novischi, A., Srikanth, M., Bennett, A.: LCC-WSD: System description for English coarse grained all words task at semeval 2007. In: Proceedings of the 4th International Workshop on Semantic Evaluations. pp. 223–226. Association for Computational Linguistics (2007)
311. Nüttgens, M., Rump, F.J.: Syntax und semantik ereignisgesteuerter prozessketten (EPK). In: Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen - Promise 2002, 9.-11. Oktober 2002, Potsdam. pp. 64–77 (2002)

312. Object Management Group: Business Process Model and Notation (BPMN). http://www.omg.org/spec/BPMN/2.0/ (2011)
313. Opdyke, W.F.: Refactoring: A program restructuring aid in designing object-oriented application frameworks. Ph.D. thesis, PhD thesis, University of Illinois at Urbana-Champaign (1992)
314. Op't Land, M., Proper, E., Waage, M., Cloo, J., Steghuis, C.: Enterprise Architecture. Springer (2008)
315. Ouyang, C., Dumas, M., Breutel, S., ter Hofstede, A.H.M.: Translating standard process models to BPEL. In: Advanced Information Systems Engineering, 18th International Conference, CAiSE 2006, Luxembourg, Luxembourg, June 5-9, 2006, Proceedings. pp. 417–432 (2006)
316. Ouyang, C., Dumas, M., ter Hofstede, A.H.M., van der Aalst, W.M.P.: From BPMN process models to BPEL web services. In: 2006 IEEE International Conference on Web Services (ICWS 2006), 18-22 September 2006, Chicago, Illinois, USA. pp. 285–292 (2006)
317. Ouyang, C., Dumas, M., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Pattern-based translation of bpmn process models to bpel web services. International Journal of Web Services Research (IJWSR) 5(1), 42–62 (2008)
318. Palmer, M., Gildea, D., Kingsbury, P.: The Proposition Bank: An Annotated Corpus of Semantic Roles. Computational Linguistics 31(1), 71–106 (2005)
319. Pantel, P., Lin, D.: Discovering word senses from text. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada. pp. 613–619 (2002)
320. Pedersen, T.: Duluth-WSI: SenseClusters applied to the sense induction task of SemEval-2. In: Proceedings of the 5th international workshop on semantic evaluation. pp. 363–366. Association for Computational Linguistics (2010)
321. Peffers, K., Tuunanen, T., Gengler, C.E., Rossi, M., Hui, W., Virtanen, V., Bragge, J.: The design science research process: a model for producing and presenting information systems research. In: Proceedings of the first international conference on design science research in information systems and technology (DESRIST 2006). pp. 83–106 (2006)
322. Peffers, K., Tuunanen, T., Rothenberger, M., Chatterjee, S.: A design science research methodology for information systems research. Journal of Management Information Systems 24(3), 45–77 (2007)
323. Pelleg, D., Moore, A.W.: X-means: Extending k-means with efficient estimation of the number of clusters. In: Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000. pp. 727–734 (2000)
324. Petri, C.A.: Kommunikation mit Automaten. Ph.D. thesis, Universität Hamburg (1962)
325. Piegorsch, W.W.: Sample sizes for improved binomial confidence intervals. Computational statistics & data analysis 46(2), 309–316 (2004)
326. Pinggera, J., Soffer, P., Fahland, D., Weidlich, M., Zugal, S., Weber, B., Reijers, H.A., Mendling, J.: Styles in business process modeling: an exploration and a model. Software and System Modeling 14(3), 1055–1080 (2015)
327. Pinggera, J., Soffer, P., Zugal, S., Weber, B., Weidlich, M., Fahland, D., Reijers, H.A., Mendling, J.: Modeling styles in business process modeling. In: Enterprise, Business-Process and Information Systems Modeling - 13th International Conference, BPMDS 2012, 17th International Conference, EMMSAD 2012, and

5th EuroSymposium, held at CAiSE 2012, Gdańsk, Poland, June 25-26, 2012. Proceedings. pp. 151–166 (2012)

328. Pinggera, J., Zugal, S., Weidlich, M., Fahland, D., Weber, B., Mendling, J., Reijers, H.A.: Tracing the process of process modeling with modeling phase diagrams. In: Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I. pp. 370–382 (2011)
329. Pittke, F., Leopold, H., Mendling, J.: Spotting terminology deficiencies in process model repositories. In: Enterprise, Business-Process and Information Systems Modeling - 14th International Conference, BPMDS 2013, 18th International Conference, EMMSAD 2013, Held at CAiSE 2013, Valencia, Spain, June 17-18, 2013. Proceedings. pp. 292–307 (2013)
330. Pittke, F., Leopold, H., Mendling, J.: When language meets language: Anti patterns resulting from mixing natural and modeling language. In: Business Process Management Workshops - BPM 2014 International Workshops, Eindhoven, The Netherlands, September 7-8, 2014, Revised Papers. pp. 118–129 (2014)
331. Pittke, F., Leopold, H., Mendling, J.: Automatic detection and resolution of lexical ambiguity in process models. IEEE Transactions on Software Engineering 41(6), 526–544 (2015)
332. Pittke, F., Leopold, H., Mendling, J., Tamm, G.: Enabling reuse of process models through the detection of similar process parts. In: Business Process Management Workshops - BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers. pp. 586–597 (2012)
333. Pittke, F., Nagel, B., Engels, G., Mendling, J.: Linguistic consistency of goal models. In: Enterprise, Business-Process and Information Systems Modeling - 15th International Conference, BPMDS 2014, 19th International Conference, EMMSAD 2014, Held at CAiSE 2014, Thessaloniki, Greece, June 16-17, 2014. Proceedings. pp. 393–407 (2014)
334. Pittke, F., Richetti, P.H.P., Mendling, J., Baião, F.A.: Context-sensitive textual recommendations for incomplete process model elements. In: Business Process Management - 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 - September 3, 2015, Proceedings. pp. 189–197 (2015)
335. Plachouras, V., Ounis, I., van Rijsbergen, C.J., Cacheda, F.: University of glasgow at the web track: Dynamic application of hyperlink analysis using the query scope. In: Proceedings of The Twelfth Text REtrieval Conference, TREC 2003, Gaithersburg, Maryland, USA, November 18-21, 2003. pp. 646–652 (2003)
336. Pollard, C., Sag, I.: Head-Driven Phrase Structure Grammar. Chicago University Press, Chicago, Illinois (1994)
337. Polyvyanyy, A., García-Bañuelos, L., Dumas, M.: Structuring acyclic process models. Inf. Syst. 37(6), 518 – 538 (2012)
338. Polyvyanyy, A., García-Bañuelos, L., Dumas, M.: Structuring acyclic process models. In: Business Process Management - 8th International Conference, BPM 2010, Hoboken, NJ, USA, September 13-16, 2010. Proceedings. pp. 276–293 (2010)
339. Polyvyanyy, A., Smirnov, S., Weske, M.: Process model abstraction: A slider approach. In: 12th International IEEE Enterprise Distributed Object Computing Conference, ECOC 2008, 15-19 September 2008, Munich, Germany. pp. 325–331 (2008)

340. Polyvyanyy, A., Vanhatalo, J., Völzer, H.: Simplified computation and generalization of the refined process structure tree. In: Web Services and Formal Methods - 7th International Workshop, WS-FM 2010, Hoboken, NJ, USA, September 16-17, 2010. Revised Selected Papers. pp. 25–41 (2010)
341. Ponzetto, S.P., Navigli, R.: Knowledge-rich word sense disambiguation rivaling supervised systems. In: ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden. pp. 1522–1531 (2010)
342. Porter, M.E.: Competitive advantage: Creating and sustaining superior performance. SimonandSchuster. com (2008)
343. Sampaio do Prado Leite, J.C., Freeman, P.A.: Requirements validation through viewpoint resolution. IEEE Transactions on Software Engineering 17(12), 1253–1269 (1991)
344. Pritchard, J., Armistead, C.: Business process management - lessons from european business. Business Proc. Manag. Journal 5(1), 10–35 (1999)
345. Procter, P.: Longman Dictionary of Contemporary English (1981)
346. Puhlmann, F.: Soundness verification of business processes specified in the pi-calculus. In: On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, OTM Confederated International Conferences CoopIS, DOA, ODBASE, GADA, and IS 2007, Vilamoura, Portugal, November 25-30, 2007, Proceedings, Part I. pp. 6–23 (2007)
347. Puhlmann, F., Weske, M.: Investigations on soundness regarding lazy activities. In: Business Process Management, 4th International Conference, BPM 2006, Vienna, Austria, September 5-7, 2006, Proceedings. pp. 145–160 (2006)
348. Pustejovsky, J.: The generative lexicon. Computational Linguistics 17(4), 409–441 (1991)
349. Pustejovsky, J.: The Generative Lexicon. MIT Press, Cambridge, MA. (1995)
350. Qiao, M., Akkiraju, R., Rembert, A.J.: Towards efficient business process clustering and retrieval: Combining language modeling and structure matching. In: Business Process Management - 9th International Conference, BPM 2011, Clermont-Ferrand, France, August 30 - September 2, 2011. Proceedings. pp. 199–214 (2011)
351. Rabiner, L., Juang, B.H.: Fundamentals of Speech Recognition. Prentice Hall (1993)
352. Ratnaparkhi, A.: A maximum entropy model for part-of-speech tagging. In: Proceedings of the conference on empirical methods in natural language processing. vol. 1, pp. 133–142. Philadelphia, USA (1996)
353. Raux, A., Eskenazi, M.: Optimizing Endpointing Thresholds using Dialogue Features in a Spoken Dialogue System. In: Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue. pp. 1–10. Association for Computational Linguistics (2008)
354. Ravichandran, D., Hovy, E.: Learning surface text patterns for a question answering system. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. pp. 41–47. Association for Computational Linguistics (2002)
355. Recker, J.: Towards an understanding of process model quality. methodological considerations. In: Proceedings of the Fourteenth European Conference on Information Systems, ECIS 2006, Göteborg, Sweden, 2006. pp. 434–445 (2006)
356. Recker, J.: Understanding Process Modelling Grammar Continuance. Ph.D. thesis, Queensland University of Technology Brisbane (2008)

357. Recker, J.: Opportunities and constraints: the current struggle with BPMN. Business Proc. Manag. Journal 16(1), 181–201 (2010)
358. Reijers, H.A., Freytag, T., Mendling, J., Eckleder, A.: Syntax highlighting in business process models. Decision Support Systems 51(3), 339–349 (2011)
359. Reijers, H.A., Mendling, J., Dijkman, R.M.: Human and automatic modularizations of process models to enhance their comprehension. Inf. Syst. 36(5), 881–897 (2011)
360. Reijers, H.A., Mendling, J., Recker, J.: Business process quality management. In: Handbook on Business Process Management 1, pp. 167–185. Springer (2015)
361. Resnick, P., Varian, H.R.: Recommender systems. Communications of the ACM 40(3), 56–58 (1997)
362. Resnik, P., Yarowsky, D.: Distinguishing systems and distinguishing senses: new evaluation methods for word sense disambiguation. Natural Language Engineering 5(2), 113–133 (1999)
363. Richetti, P.H.P., Baião, F.A., Santoro, F.M.: Declarative process mining: Reducing discovered models complexity by pre-processing event logs. In: Business Process Management - 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings. pp. 400–407 (2014)
364. van Rijsbergen, C.J.: Information Retrieval. Butterworths, London, 2 edn. (1979)
365. Rittgen, P.: Negotiating models. In: Advanced Information Systems Engineering, 19th International Conference, CAiSE 2007, Trondheim, Norway, June 11-15, 2007, Proceedings. pp. 561–573 (2007)
366. Robins, R.H.: Linguistics and Linguistic Evidence: The LAGB Silver Jubilee Lectures, 1984, vol. 25. Grevatt & Grevatt (1985)
367. Rolland, C., Souveyet, C., Achour, C.: Guiding goal modeling using scenarios. IEEE Transactions on Software Engineering 24(12), 1055–1071 (1998)
368. Rolland, C., Ben Achour, C.: Guiding the construction of textual use case specifications. Data & Knowledge Engineering 25(1), 125–160 (1998)
369. Rosemann, M.: Potential Pitfalls of Process Modeling: Part A. Business Process Management Journal 12(2), 249–254 (2006)
370. Rosemann, M.: Potential Pitfalls of Process Modeling: Part B. Business Process Management Journal 12(3), 377–384 (2006)
371. Rosemann, M., Schütte, R.: Grundsätze ordnungsmäßiger referenzmodellierung. Entwicklungsstand und Entwicklungsperspektiven der Referenzmodellierung 32, 16–33 (1997)
372. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Workflow exception patterns. In: Advanced Information Systems Engineering, 18th International Conference, CAiSE 2006, Luxembourg, Luxembourg, June 5-9, 2006, Proceedings. pp. 288–302 (2006)
373. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow resource patterns: Identification, representation and tool support. In: Advanced Information Systems Engineering, 17th International Conference, CAiSE 2005, Porto, Portugal, June 13-17, 2005, Proceedings. pp. 216–232 (2005)
374. Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow data patterns: Identification, representation and tool support. In: Conceptual Modeling - ER 2005, 24th International Conference on Conceptual Modeling, Klagenfurt, Austria, October 24-28, 2005, Proceedings. pp. 353–368 (2005)

375. Sadiq, S., Soffer, P., Völzer, H.: Business Process Management: 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014, Proceedings, vol. 8659. Springer (2014)
376. Sadiq, S.W., Governatori, G., Namiri, K.: Modeling control objectives for business process compliance. In: Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24-28, 2007, Proceedings. pp. 149–164 (2007)
377. Sadiq, W., Orlowska, M.E.: Applying graph reduction techniques for identifying structural conflicts in process models. In: Advanced Information Systems Engineering, 11th International Conference CAiSE'99, Heidelberg, Germany, June 14-18, 1999, Proceedings. pp. 195–209 (1999)
378. Sadiq, W., Orlowska, M.E.: Analyzing process models using graph reduction techniques. Inf. Syst. 25(2), 117–134 (2000)
379. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM 18(11), 613–620 (1975)
380. Sarbanes, P., Oxley, G.: Sarbanes-oxley act of 2002 (2002)
381. de Saussure, F.: Course on General Linguistics. Philosophical Library (Trans. Wade Baskin) (1959)
382. Scheer, A.: EDV-orientierte Betriebswirtschaftslehre. No. 236 in Heidelberger Taschenbücher, Springer, Berlin u.a. (1984)
383. Scheer, A.: ARIS - Modellierungsmethoden, Metamodelle, Anwendungen. Springer (1998)
384. Scheer, A.W.: ARIS - Vom Geschäftsprozess zum Anwendungssystem. Springer (1999)
385. Scheer, A.: ARIS: Business Process Modeling. Springer (2000)
386. Schmelzer, H.J., Sesselmann, W.: Geschäftsprozessmanagement in der Praxis. Hanser, München, Wien, 5th edn. (2006)
387. Schmid, H.: Probabilistic part-of-speech tagging using decision trees. In: Proceedings of the international conference on new methods in language processing. vol. 12, pp. 44–49. Citeseer (1994)
388. Schmitt, N.: An Introduction to Applied Linguistics. Routledge (2010)
389. Schuler, K.K.: Verbnet: a broad-coverage, comprehensive verb lexicon. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA, USA (2005)
390. Schütte, R., Rotthowe, T.: The guidelines of modeling - an approach to enhance the quality in information models. In: Conceptual Modeling - ER '98, 17th International Conference on Conceptual Modeling, Singapore, November 16-19, 1998, Proceedings. pp. 240–254 (1998)
391. Schütze, H.: Dimensions of meaning. In: Proceedings Supercomputing '92, Minneapolis, MN, USA, November 16-20, 1992. pp. 787–796 (1992)
392. Schütze, H.: Automatic word sense discrimination. Computational Linguistics 24(1), 97–123 (1998)
393. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. 34(1), 1–47 (2002)
394. Shanks, G., Tansley, E., Weber, R.: Using ontology to validate conceptual models. Communications of the ACM 46(10), 85–89 (2003)
395. Sharp, A., McDermott, P.: Workflow Modeling: Tools for Process Improvement and Application Development. Artech House Publishers (2001)
396. Shull, F., Travassos, G.H., Carver, J., Basili, V.R.: Evolving a set of techniques for oo inspections (1999)

397. Sidorova, N., Stahl, C., Trcka, N.: Soundness verification for conceptual workflow nets with data: Early detection of errors with the most precision possible. Inf. Syst. 36(7), 1026–1043 (2011)
398. Siegemund, K., Thomas, E.J., Zhao, Y., Pan, J., Assmann, U.: Towards ontology-driven requirements engineering. In: Workshop semantic web enabled software engineering at 10th international semantic web conference (ISWC), Bonn (2011)
399. Silver, B.: BPMN Method and Style, with BPMN Implementer's Guide. Cody-Cassidy Press, 2nd edn. (Jan 2011)
400. Simon, H.A.: The sciences of the artificial (3rd ed.). MIT Press, Cambridge, MA, USA (1996)
401. Sinur, J., Hill, J.B.: Magic quadrant for business process management suites. Gartner RAS Core research note pp. 1–24 (2010)
402. Smirnov, S.: Business process model abstraction: theory and practice. Universitätsverlag Potsdam (2010)
403. Smirnov, S., Dijkman, R.M., Mendling, J., Weske, M.: Meronymy-based aggregation of activities in business process models. In: Conceptual Modeling - ER 2010, 29th International Conference on Conceptual Modeling, Vancouver, BC, Canada, November 1-4, 2010. Proceedings. pp. 1–14 (2010)
404. Smirnov, S., Reijers, H.A., Weske, M.: A semantic approach for business process model abstraction. In: Advanced Information Systems Engineering - 23rd International Conference, CAiSE 2011, London, UK, June 20-24, 2011. Proceedings. pp. 497–511 (2011)
405. Smirnov, S., Reijers, H.A., Weske, M.: From fine-grained to abstract process models: A semantic approach. Inf. Syst. 37(8), 784–797 (2012)
406. Smith, A.: The Wealth of Nations: an Inquiry into the Nature and Causes of the Wealth of Nations. London (1776)
407. Smith, H., Fingar, P.: Business process management: the third wave, vol. 1. Meghan-Kiffer Press Tampa (2003)
408. Smith, R.F.: Business process management and the balanced scorecard: using processes as strategic drivers. John Wiley & Sons (2010)
409. Socher, R., Bauer, J., Manning, C.D., Ng, A.Y.: Parsing with compositional vector grammars. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers. pp. 455–465 (2013)
410. Stachowiak, H.: Allgemeine Modelltheorie. Springer-Verlag, Wien, New York (1973)
411. Stahl, H., Müller, J., Lang, M.K.: An efficient top-down parsing algorithm for understanding speech by using stochastic syntactic and semantic models. In: 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings, ICASSP '96, Atlanta, Georgia, USA, May 7-10, 1996. pp. 397–400 (1996)
412. Stevenson, A.: Oxford Dictionary of English. Oxford University Press (2010)
413. Sun, S.X., Zhao, J.L., Nunamaker, J.F., Sheng, O.R.L.: Formulating the data-flow perspective for business process management. Information Systems Research 17(4), 374–391 (2006)
414. Syal, P., Jindal, D.V.: An introduction to Linguistics: Language, grammar and semantics. PHI Learning Pvt. Ltd. (2007)
415. Taylor, F.: The principles of scientific management (1911)

416. Thomas, K.E., Sripada, S.: Atlas. txt: linking geo-referenced data to text for nlg. In: Proceedings of the Eleventh European Workshop on Natural Language Generation. pp. 163–166. Association for Computational Linguistics (2007)
417. Törner, F., Ivarsson, M., Pettersson, F., Öhman, P.: Defects in automotive use cases. In: 2006 International Symposium on Empirical Software Engineering (ISESE 2006), September 21-22, 2006, Rio de Janeiro, Brazil. pp. 115–123 (2006)
418. Toutanova, K., Manning, C.D.: Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In: Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13. pp. 63–70. Association for Computational Linguistics (2000)
419. Trkman, P.: The critical success factors of business process management. International Journal of Information Management 30(2), 125–134 (2010)
420. Tumay, K.: Business process simulation. In: Proceedings of the 27th conference on Winter simulation. pp. 55–60. IEEE Computer Society (1995)
421. Turing, A.M.: Computing machinery and intelligence. Mind pp. 433–460 (1950)
422. Van Lamsweerde, A.: Requirements engineering in the year 00: A research perspective. In: Proceedings of the 22nd international conference on Software engineering. pp. 5–19. ACM (2000)
423. Verbeek, H., Basten, T., van der Aalst, W.M.P.: Diagnosing workow processes using woflan. The Computer Journal 44(4), 246–279 (2001)
424. Vessey, I., Glass, R.L.: Strong vs. weak approaches to systems development. Commun. ACM 41(4), 99–102 (1998)
425. van der Vos, B., Gulla, J.A., van de Riet, R.: Verification of conceptual models based on linguistic knowledge. Data & Knowledge Engineering 21(2), 147 – 163 (1997)
426. Vossen, P.: A multilingual database with lexical semantic networks. Springer (1998)
427. Voutilainen, A.: A syntax-based part-of-speech analyser. In: EACL 1995, 7th Conference of the European Chapter of the Association for Computational Linguistics, March 27-31, 1995, University College Dublin, Belfield, Dublin, Ireland. pp. 157–164 (1995)
428. Voutilainen, A.: Hand-crafted rules. In: Syntactic wordclass tagging, pp. 217–246. Springer (1999)
429. Wang, Y., Gutiérrez, I.L.M., Winbladh, K., Fang, H.: Automatic detection of ambiguous terminology for software requirements. In: Natural Language Processing and Information Systems - 18th International Conference on Applications of Natural Language to Information Systems, NLDB 2013, Salford, UK, June 19-21, 2013. Proceedings. pp. 25–37 (2013)
430. Webb, N.: Cue-based Dialogue Act Classification. Ph.D. thesis, University of Sheffield (2010)
431. Weber, B., Reichert, M., Mendling, J., Reijers, H.A.: Refactoring large process model repositories. Computers in Industry 62(5), 467–486 (2011)
432. Weber, B., Zeitelhofer, S., Pinggera, J., Torres, V., Reichert, M.: How advanced change patterns impact the process of process modeling. In: Enterprise, Business-Process and Information Systems Modeling - 15th International Conference, BPMDS 2014, 19th International Conference, EMMSAD 2014, Held at CAiSE 2014, Thessaloniki, Greece, June 16-17, 2014. Proceedings. pp. 17–32 (2014)

433. Weber, I., Hoffmann, J., Mendling, J.: Semantic business process validation. In: Proceedings of the 3rd International Workshop on Semantic Business Process Management (SBPM'08), CEUR-WS Proceedings. vol. 472 (2008)
434. Weber, I., Hoffmann, J., Mendling, J.: Beyond soundness: on the verification of semantic business process models. Distributed and Parallel Databases 27(3), 271–343 (2010)
435. Webster, J., Watson, R.T.: Analyzing the past to prepare for the future: Writing a literature review. MIS Quarterly 26(2) (2002)
436. Weidlich, M., Dijkman, R.M., Mendling, J.: The icop framework: Identification of correspondences between process models. In: Advanced Information Systems Engineering, 22nd International Conference, CAiSE 2010, Hammamet, Tunisia, June 7-9, 2010. Proceedings. pp. 483–498 (2010)
437. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer, 2nd edn. (2012)
438. Williams, S.: Business process modeling improves administrative control. Automation pp. 44–50 (1967)
439. Wimmer, M., Kappel, G., Kusel, A., Retschitzegger, W., Schoenboeck, J., Schwinger, W.: From the heterogeneity jungle to systematic benchmarking. In: Models in Software Engineering - Workshops and Symposia at MODELS 2010, Oslo, Norway, October 2-8, 2010, Reports and Revised Selected Papers. pp. 150–164 (2010)
440. Wittgenstein, L.: Philosophical Investigations, The German Text with an English Translation. Basil Blackwell, Oxford, 4 edn. (2009)
441. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B.: Experimentation in Software Engineering. Springer (2012)
442. Wolf, K.: Generating petri net state spaces. In: Petri Nets and Other Models of Concurrency - ICATPN 2007, 28th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, ICATPN 2007, Siedlce, Poland, June 25-29, 2007, Proceedings. pp. 29–42 (2007)
443. Wollheim, R.: Art and its Objects. Cambridge University Press (1980)
444. Yan, Z., Dijkman, R.M., Grefen, P.W.P.J.: Fast business process similarity search. Distributed and Parallel Databases 30(2), 105–144 (2012)
445. Yan, Z., Dijkman, R.M., Grefen, P.W.P.J.: Fnet: An index for advanced business process querying. In: Business Process Management - 10th International Conference, BPM 2012, Tallinn, Estonia, September 3-6, 2012. Proceedings. pp. 246–261 (2012)
446. Yang, H., Roeck, A.N.D., Gervasi, V., Willis, A., Nuseibeh, B.: Analysing anaphoric ambiguity in natural language requirements. Requir. Eng. 16(3), 163–189 (2011)
447. Yang, H., Willis, A., Roeck, A.N.D., Nuseibeh, B.: Automatic detection of nocuous coordination ambiguities in natural language requirements. In: ASE 2010, 25th IEEE/ACM International Conference on Automated Software Engineering, Antwerp, Belgium, September 20-24, 2010. pp. 53–62 (2010)
448. Yngve, V.H.: Syntax and the problem of multiple meaning. Machine translation of languages. John Wiley & Sons, New York pp. 208–226 (1955)
449. Yu, H., Hatzivassiloglou, V.: Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In: Proceedings of the 2003 conference on Empirical methods in natural language processing. pp. 129–136. Association for Computational Linguistics (2003)

450. Zdravkovic, J., Kirikova, M., Johannesson, P. (eds.): Advanced Information Systems Engineering - 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings, Lecture Notes in Computer Science, vol. 9097. Springer (2015)
451. Zhong, Z., Ng, H.T.: Word sense disambiguation improves information retrieval. In: The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers. pp. 273–282 (2012)

A

Penn Treebank Tagset

Table A.1: Penn Treebank Part-of-Speech Tags, adapted from [184, p. 295]

Tag	**Description**	**Example**	**Tag**	**Description**	**Example**
CC	Coordin. Conjunction	and, but, or	RBS	Adverb, superlative	fastest
CD	Cardinal number	one, two	RP	Particle	up, off
DT	Determiner	a, the	SYM	Symbol	+,%, &
EX	Existential there	there	TO	to	to
FW	Foreign word	mea culpa	UH	Interjection	ah, oops
IN	Preposition/sub-conj	of, in, by	VB	Verb, base form	eat
JJ	Adjective	big	VBD	Verb, past tense	ate
JJR	Adj., comparative	bigger	VBG	Verb, gerund	eating
JJS	Adj., superlative	biggest	VBN	Verb, past participle	eaten
LS	List item marker	1, 2, One	VBP	Verb, non-3sg pres	eat
MD	Modal	can, should	VBZ	Verb, 3sg pres	eats
NN	Noun, sing. or mass	dog	WDT	Wh-determiner	which, that
NNS	Noun, plural	dogs	WP	Wh-pronoun	what, who
NNP	Proper noun, singular	IBM	WP$	Possessive wh-	whose
NNPS	Proper noun, plural	Carolinas	WRB	Wh-adverb	how, where
PDT	Predeterminer	all, both	(	Left parenthesis	(, [, {
POS	Possessive ending	s	)	Right parenthesis	),], }
PP	Personal pronoun	I, you, he	,	Comma	,
PP$	Possessive pronoun	your, ones	.	Sentence-final punc	(. ! ?)
RB	Adverb	quickly	:	Mid-sentence punc	(: ; ... -)
RBR	Adverb, comparative	faster			

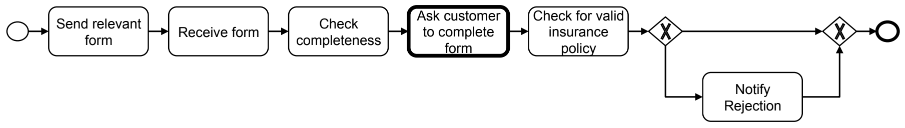
B

Example Scenarios from Ambiguity Detection Evaluation

B.1 Evaluation of Homonym Detection

Scenario 1

Consider the process model as depicted:



Consider specifically the activity **ask customer to complete form** and the general meaning that the word **ask** can refer to.

	Strongly disagree	Rather disagree	Rather agree	Strongly agree
Specifically in the context of this model, the word **ask** refers to **make a request or demand for something to somebody**.	○	○	○	○
In general, **ask** can refer to **make a request or demand for something to somebody**.	○	○	○	○

	Strongly disagree	Rather disagree	Rather agree	Strongly agree
Specifically in the context of this model, the word **ask** refers to **require as useful, just, or proper**.	○	○	○	○
In general, **ask** can refer to **require as useful, just, or proper**.	○	○	○	○

(to be continued)

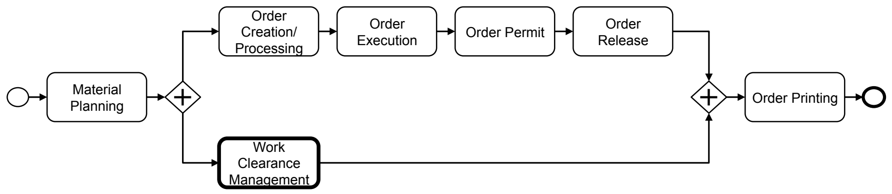
	Strongly disagree	Rather disagree	Rather agree	Strongly agree
Specifically in the context of this model, the word **ask** refers to **inquire about**.	○	○	○	○
In general, **ask** can refer to **inquire about**.	○	○	○	○

	Strongly disagree	Rather disagree	Rather agree	Strongly agree
Specifically in the context of this model, the word **ask** refers to **consider obligatory**.	○	○	○	○
In general, **ask** can refer to **consider obligatory**.	○	○	○	○

B.2 Evaluation of Synonym Detection

Scenario 1

Consider the process model as depicted:



Consider that the label **Work Clearance Management** was replaced by a computer program, changing it to the following new label.

	Strongly disagree	Rather disagree	Rather agree	Strongly agree
The new label **Assign Clearance Management** conveys the same meaning as the original label.	○	○	○	○
The new label **Assign Clearance Management** is grammatically correct.	○	○	○	○

(to be continued)

	Strongly disagree	Rather disagree	Rather agree	Strongly agree
The new label **Aggregate Clearance Management** conveys the same meaning as the original label.	○	○	○	○
The new label **Aggregate Clearance Management** is grammatically correct.	○	○	○	○

	Strongly disagree	Rather disagree	Rather agree	Strongly agree
The new label **Ship Clearance Management** conveys the same meaning as the original label.	○	○	○	○
The new label **Ship Clearance Management** is grammatically correct.	○	○	○	○

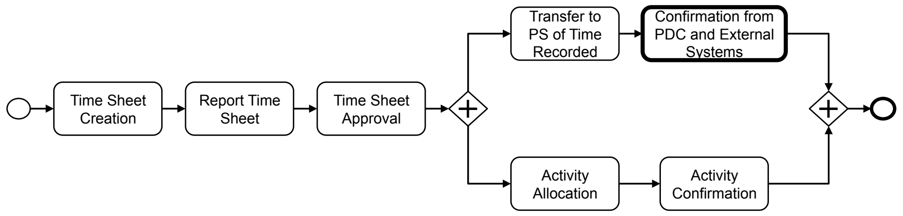
	Strongly disagree	Rather disagree	Rather agree	Strongly agree
The new label **Master Clearance Management** conveys the same meaning as the original label.	○	○	○	○
The new label **Master Clearance Management** is grammatically correct.	○	○	○	○

C

Example Scenario from Recommendation Evaluation

Scenario 6

Consider the process model as depicted:



Also consider the activity **Confirmation from PDC and External Systems** and think of possibilities to enrich this activity with an action or a business object.

	Very useless (-2)	Useless (-1)	Useful (+1)	Very useful (+2)
I regard the suggestion **Confirm Record from PDC and External Systems** as ...	○	○	○	○
I regard the suggestion **Confirm Production Order from PDC and External Systems** as ...	○	○	○	○
I regard the suggestion **Confirm Stock Placement from PDC and External Systems** as ...	○	○	○	○

(to be continued)

	Very useless (-2)	Useless (-1)	Useful (+1)	Very useful (+2)
I regard the suggestion **Confirm Time Sheet from PDC and External Systems** as ...	○	○	○	○
I regard the suggestion **Confirm Application Retraction from PDC and External Systems** as ...	○	○	○	○
I regard the suggestion **Confirm Removal f. Storage from PDC and External Systems** as ...	○	○	○	○
I regard the suggestion **Confirm Expression from PDC and External Systems** as ...	○	○	○	○
I regard the suggestion **Confirm Activity from PDC and External Systems** as ...	○	○	○	○
I regard the suggestion **Confirm Picking from PDC and External Systems** as ...	○	○	○	○
I regard the suggestion **Confirm Result from PDC and External Systems** as ...	○	○	○	○