Comparing and Aligning Process Representations

Han van der Aa

November 2017

SIKS Dissertation Series No. 2018-01
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Promotiecommissie:

prof.dr.	Frank van Harmelen	(Vrije Universiteit Amsterdam)
prof.dr.ir.	Wil van der Aalst	(Eindhoven University of Technology)
prof.dr.	Matthias Weidlich	(Humboldt University of Berlin, Germany)
prof.dr.	Barbara Weber	(Technical University of Denmark, Denmark)
prof.dr.	Manfred Reichert	(University of Ulm, Germany)

ISBN 978-94-028-0867-4

Published by ProefschriftMaken || www.proefschriftmaken.nl
Typeset in LaTeX by the author

VRIJE UNIVERSITEIT

Comparing and Aligning Process Representations

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. V. Subramaniam,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de Faculteit der Bètawetenschappen
op vrijdag 26 januari 2018 om 11.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Johannes Hendrikus van der Aa

geboren te Sint-Oedenrode

promotor:	prof.dr.ir. H.A. Reijers
copromotor:	dr. H. Leopold

Abstract

Processes within organizations can be highly complex chains of inter-related steps, involving numerous stakeholders and information systems. Due to this complexity, having access to the right information is vital to the proper execution and effective management of an organization's business processes. A major challenge in this regard is that information on a single process is often spread out over numerous models, documents, and systems. This phenomenon results from efforts to provide a variety of process stakeholders with the information that is relevant to them, in a suitable format. However, this disintegration of process information also has considerable disadvantages for organizations. In particular, it can lead to severe maintenance issues, reduced execution efficiency, and negative effects on the quality of process results. Against this background, this doctoral thesis focuses on the spread of process information in organizations and, in particular, on the mitigation of the negative aspects associated with this phenomenon. The main contributions of this thesis are five techniques that focus on the alignment and comparison of process information from different informational artifacts. Each of these techniques tackles a specific scenario involving multiple informational artifacts that contain process information in different representation formats. Among others, we present automated techniques for the detection of inconsistencies between process models and textual process descriptions, the alignment of process performance measurements to process models, conformance-checking in the context of uncertainty, and the matching of process models through the analysis of event-log information. We demonstrate the efficacy and usefulness of these techniques through quantitative evaluations involving data obtained from real-world settings. Altogether, the presented work provides important contributions for the analysis, comparison, and alignment of process information in various representation formats through the development of novel concepts and techniques. The contributions, furthermore, provide a means for organizations to improve the efficiency and quality of their processes.

Samenvatting

Bedrijfsprocessen kunnen heel ingewikkeld in elkaar zitten, omdat zij vaak bestaan uit complexe ketens van onderling samenhangende stappen, waarbij bovendien een groot aantal belanghebbenden betrokken is en ook nog allerlei IT systemen een rol spelen. Door deze complexiteit is het voor de uitvoering en het onderhoud van bedrijfsprocessen van cruciaal belang dat bij elke stap de juiste informatie over deze processen beschikbaar is. Een belangrijke complicatie in dit kader is dat informatie over een enkel proces vaak verspreid is over tal van modellen, documenten en systemen. Deze verspreiding ontstaat doordat organisaties trachten om de verschillende belanghebbenden binnen een proces te voorzien van alleen die informatie die van specifiek belang is voor hen. Deze opsplitsing van procesinformatie heeft voor een organisatie echter ook aanzienlijke nadelen. Het kan namelijk leiden tot ernstige onderhoudsproblemen, vermindering van efficiency en het kan een negatieve invloed hebben op de kwaliteit van de uiteindelijke resultaten. Om bij te dragen aan een oplossing voor deze negatieve gevolgen, richt dit proefschrift zich op het verminderen van de problemen die voortkomen uit de verspreiding van procesinformatie. De belangrijkste bijdragen in het proefschrift zijn vijf technieken die zich focussen op het verbinden en vergelijken van procesinformatie uit verschillende bronnen. Ieder van deze technieken biedt een oplossing voor een specifiek scenario waarbij meerdere informatiebronnen in verschillende vormen betrokken zijn. Zo presenteren we onder andere automatische technieken voor het opsporen van tegenstrijdigheden tussen procesmodellen en tekstuele documenten, voor het controleren van de naleving van dubbelzinnige procesomschrijvingen en voor het vergelijken van processen op basis van informatie uit IT systemen. We tonen het nut en de bruikbaarheid van deze technieken aan door middel van kwantitatieve evaluaties die zijn gebaseerd op gegevens van bestaande bedrijfsprocessen. Samengevat biedt dit proefschrift een aantal wetenschappelijke bijdragen voor het analyseren, vergelijken en verbinden van procesinformatie uit verschillende bronnen. Deze bijdragen worden gevormd door de introductie van nieuwe theoretische concepten en de ontwikkeling van technische oplossingen. Bovendien biedt dit werk een basis voor organisaties voor het verbeteren van hun efficiëntie en de kwaliteit van hun processen.

Contents

List of Figures

List of Tables

Acronyms

BPM Business Process Management
BPMN Business Process Model and Notation
CFG Context-Free Grammar
EPC Event-driven Process Chain
FLM First-Line Matcher
IDF Inverse Document Frequency
IG Information Gain
IS Information Systems
IT Information Technology
LSH Locality Sensitive Hashing
HMM Hidden Markov Model
MWBM Maximum Weighted Bipartite-graph Match
NLP Natural Language Processing
ORE Ontobuilder Research Environment
PCFG Probabilistic Context-Free Grammar
PPI Process Performance Indicator
SCOR Supply Chain Operations Reference
SLM Second-Line Matcher
7PMG Seven Process Modeling Guidelines
TTF Task-Technology Fit

1 Introduction

This chapter provides an introduction to this doctoral thesis. In Section 1.1, we motivate the need for the alignment and comparison of process information in organizations. Section 1.2 introduces the main contributions of our research. Section 1.3 describes the methodological background for the research presented in this thesis. Section 1.4 presents the publications that followed from this research. Finally, Section 1.5 provides an overview of the remaining chapters of the thesis.

1.1 Motivation

Business Process Management (BPM) is the art and science of analyzing how work is performed in an organization with the aim to ensure consistent outcomes and to take advantage of improvement opportunities [87]. The focal point of any BPM initiative is formed by one or more business processes. A *business process* refers to a set of related activities, which transform some input into an output that is valuable to a customer [121]. Such processes exist in every organization, from hospitals to financial institutions, and from family businesses to global enterprises.

Processes within organizations can be highly complex chains of inter-related steps, involving numerous stakeholders and information systems [17]. Due to this complexity, having access to the right information is vital to the proper execution and effective management of an organization's business processes [55]. Specifically, access to information on processes contributes to their efficient execution [139], their compliance to rules and regulations [31], and to their enhancement and redesign [134]. A major threat to this information need is that the information on a single process, also referred to as *process information*, is often spread out over numerous models, documents, and systems. We shall refer to this condition as the *fragmentation of process information*.

This fragmentation occurs because organizations typically use multiple *informational artifacts* to provide process information to stakeholders with different informational needs or preferences [55, 93]. However, the use of multiple informational artifacts also poses considerable challenges to organizations. First, the fragmentation

of process information can increase the effort required to access desired or necessary information. In particular, users may have to browse through numerous systems and documents to find the information they need [189, p.4]. Such cases can compromise the efficiency and effectiveness of process execution and decision making [55]. Second, fragmentation can result in the provision of incorrect process information to users. This problem can occur when different artifacts contradict each other, for instance when the artifacts have been developed independently [224]. The information captured in artifacts may also lose their validity over time if artifacts are not updated to reflect process changes [280]. When users execute a process based on invalid information, they may perform the process in an incorrect manner, resulting in *business process noncompliance* [30]. Such noncompliant acts can have severe consequences for organizations, including reduced productivity [35], a loss of control over processes [239], and even financial penalties imposed by authorities [173].

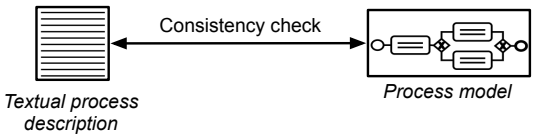
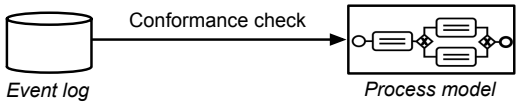
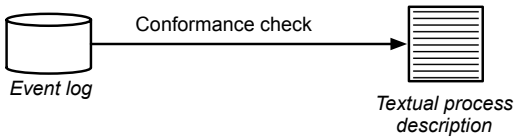
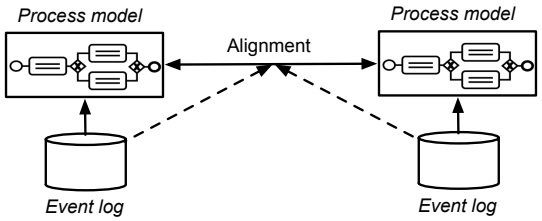
Despite the severity of these issues, there is only limited support for organizations to effectively deal with the negative effects of the fragmentation of process information. This lack of support exists despite the widespread recognition in research for the need to align process information from particular kinds of informational artifacts. In particular, a plethora of approaches have been developed that automatically align information between process models, so-called *process model matching techniques* (cf. [32, 58, 277]). Matching techniques also exist that address the need to align information between different *event logs* [178], or between event logs and process models [37, 39]. There is also support for scenarios beyond the alignment of information. For instance, several *querying* techniques exist that support users to search collections of process models. Other approaches exist that determine the *consistency* between process models [279] or that maintain consistency by *propagating changes* from one model to another [280].

A key problem is that these aforementioned approaches only focus on specific application scenarios. They typically address situations that involve highly-structured process information in the form of process models and event logs. Process information in less-structured formats, such as natural language documents, is largely ignored by existing work. Consequently, there is a considerable gap between the way in which organizations capture information on their processes and the support that exists for this situation. As a result of this research gap, organizations still struggle with consequences caused by the fragmentation of process information.

1.2 Contributions

This thesis focuses on the automated analysis of process information contained in various representation formats. As a foundation, we provide insights into the way in which organizations capture information on their processes and the problems that result from this. The main contributions of this thesis revolve around the definition of five techniques that focus on the comparison and alignment of process information in different informational artifacts. The techniques that we develop enable organizations to execute and maintain their process information more efficiently and to ensure that processes are executed according to their specification.

Table 1.1: Overview of the techniques presented in this thesis

Purpose	Illustration
1. Check consistency between a process model and a textual process description.	
2. Check conformance between an event log and a process model based on uncertain event-activity mappings.	
3. Check conformance between an event log and a textual process description. Considers ambiguity in the textual description.	
4. Transform unstructured natural language descriptions into measurable Process Performance Indicators (PPIs).	
5. Align process models based on information from associated event logs.	

As depicted in Table 1.1, each technique addresses a specific scenario involving multiple informational artifacts in different representation formats. With these techniques we focus on three main types of representation formats: process models, event logs, and natural language texts. The choice for these formats is justified by their role as *de facto* standards to capture a wide range of process information. Specifically, process models represent the most common means to capture *design-time* information on the flow of business processes in a structured manner [87, p.16]. Furthermore, event logs are the default way to capture *run-time* information on processes, as recorded during their actual execution [13, p.8]. Finally, natural language texts are widely-used as an all-purpose format to capture a broad variety of unstructured or semi-structured process information, such as process descriptions, work instructions, and guidelines [190], but also information on other process perspectives, such as Process Performance Indi-

cators (PPIs) [236].

The five techniques presented in this thesis provide a near-complete coverage of the combinations of the three considered representation formats. Specifically, we focus on scenarios involving alignments from model-to-model, log-to-log, model-to-text, log-to-model, and log-to-text[1]. Note that, given the lack of established application scenarios, we do not cover a scenario involving the comparison of process information from two natural language texts. Within the combinations covered by the techniques, our choice for the specific use cases is motivated by their practical relevance and by the gap that exists between prior research and the novelty of the conceptual developments required to address them. For example, the alignment between an event log and process model primarily focuses on *conformance checking* (cf. [15, 24, 282]), whereas in model-to-model scenarios the focus is most often on the establishment of alignments, so-called *matching* (cf. [79, 277, 284]).

The insights about the use of process information in organizations and the presented alignment techniques lead to several scientific contributions for the analysis, comparison, and alignment of process information, especially in the context of semi-structured representation formats. We identify one contribution that stems from the insights obtained about the fragmentation of process information, whereas five other contributions follow from the alignment techniques. Specifically, the six main contributions of this thesis are as follows:

1. *An overview of the causes and consequences of process-information fragmentation.* Organizations maintain a variety of process-information artifacts in order to provide different process stakeholders with the information they require. This fragmentation can have clear advantages, but also poses considerable threats to the efficient maintenance and execution of processes. As part of Chapter 2, we provide insights into the reasons why multiple artifacts are required for a single process and which effects this can have on organizations. These insights can be highly useful for organizations in their efforts to mitigate the negative consequences associated with the fragmentation of process information. Furthermore, they provide important guidance to researchers to develop means that support organizations.
2. *Inconsistency detection between process models and textual process descriptions.* Many organizations maintain textual process descriptions alongside graphical process models. Although this makes process information accessible to various stakeholders, there is a clear risk that model and text become misaligned when changes are not applied to both descriptions consistently. For organizations with hundreds of different processes, the effort required to manually identify and clear up such conflicts is considerable. We address this problem in Chapter 3 with a technique that automatically identifies inconsistencies between a process model and a corresponding textual description. The techniques builds on approaches for linguistic analysis that are tailored to extract relevant process information from textual descriptions. Furthermore, our technique uses specific metrics, so-called *predictors*, to quantify the likelihood that the two representation formats contain inconsistencies.

[1] Note that the fifth technique in the table covers both model-to-model and log-to-log alignment.

3. *Behavioral spaces as a means to capture behavioral uncertainty in processes.* The use of semi-structured and unstructured representation formats for process information, such as natural language texts, is widespread throughout organizations. Nevertheless, these formats are typically exempt from consideration by automated analysis techniques. A main reason for this is that these formats depend on the use of natural language to convey the behavior of a process. Because natural language is inherently ambiguous, it is often impossible to determine with certainty which process behavior is exactly described. Rather, there are numerous potential interpretations of the described process behavior. In order to be able to reason about properties such as conformance in the context of uncertain behavior, we introduce the concept of a *behavioral space* as a means to capture all potential interpretations of behavioral uncertainty. In this thesis, we leverage this concept in two ways. In Chapter 4, we use behavioral spaces to capture behavioral uncertainty caused by unclear relations between different informational artifacts. In Chapter 5, we use the concept to capture uncertainty caused by ambiguous textual process descriptions.
4. *Conformance checking in the context of behavioral uncertainty.* By capturing behavioral uncertainty in processes using behavioral spaces, automated analysis techniques can be used that are otherwise not applicable or may produce incorrect results. In this thesis we demonstrate the usefulness of behavioral spaces by applying them in the context of conformance checking. *Conformance checking* aims to determine if observed behavior, as recorded by IT systems, conforms to a particular process specification that denotes the allowed behavior. By using behavioral spaces for this task, we can obtain conformance-checking results in situations where traditional conformance-checking techniques yield untrustworthy results. The results obtained by our techniques are probabilistic, differentiating between *conforming*, *nonconforming*, and *potentially conforming* behavior. We apply our conformance-checking technique in Chapters 4 and 5.
5. *Transformation of natural language descriptions into measurable Process Performance Indicators (PPIs).* Monitoring process performance is an important means for organizations to identify opportunities to improve their operations. The definition of suitable PPIs represents a crucial task in this regard. Because PPIs need to be in line with strategic business objectives, the formulation of PPIs is a managerial concern. Managers typically start out to provide relevant indicators in the form of natural language PPI descriptions. Therefore, considerable time and effort have to be invested to transform these descriptions into PPI definitions that can be automatically monitored. To overcome these problems associated with manual performed transformations, Chapter 6 presents a technique that automates this task. The technique builds on a conceptualization of the way in which unstructured natural language descriptions convey the semantic components required to define PPIs.
6. *Process model matching based on event-log information.* Process model matching provides the basis for many process analysis techniques, such as inconsistency detection and process querying. So-called *matchers* aim to automatically identify correspondences between activities in two process models. The numerous techniques that have been developed for this purpose all share a focus on

process-level information. Therefore, they do not exploit process information contained in event logs associated with the models, which can be helpful for the identification of model correspondences. Chapter 7 shows the potential of using event-log information in process model matching. The matchers we introduce build on six conceptual notions that indicate similarity between event classes.

1.3 Methodological Background

The research presented in this thesis is conducted in the context of the Information Systems (IS) research discipline. IS research aims to further knowledge that aids in the productive application of Information Technology (IT) to organizations [129]. It is a strongly interdisciplinary field of study that applies theories from social sciences, economics, and computer science. IS research involves two complementary paradigms: behavioral science and design science (cf. [117, 122, 209]).

Behavioral science is concerned with the development and justification of theories that explain or predict phenomena surrounding the analysis, design, implementation, management, and use of information systems [122]. Such theories ultimately inform researchers and practitioners of the interactions among people, technology, and organizations that must be managed if an information system is to achieve its stated purpose: improving the effectiveness and efficiency of an organization. *Design science* is fundamentally a problem-solving paradigm that has its roots in engineering and the sciences of the artificial [252]. It seeks to create innovations that define the ideas, practices, technical capabilities, and products through which the analysis, design, implementation, management, and use of information systems can be effectively and efficiently accomplished [76, 261]. In particular, design science can be used to addresses problems that are characterized as *wicked*, which means that they do not have a definitive formulation [122]. Because of this wickedness, solutions to such design-science problems cannot be assessed by truth, but rather by utility. Therefore, Simon [252] advocates to accept satisficing solutions by designing and creating useful artifacts. In summary, it can be said that the goal of behavioral-science research is *truth* and the goal of design-science research is *utility* [122].

The research design of this thesis combines both paradigms, following the argument by Hevner et al. [122] that design and behavioral science are complementary: "*truth informs design and utility informs theory*." In this context, Hevner et al. suggest a set of seven guidelines for effective information systems research that are particularly applicable to works with a design-science focus. We use these guidelines as a basis to discuss in how far this thesis meets information systems research standards:

Guideline 1: Design as an artifact. The goal of design-science research in IS is the creation of purposeful IT artifacts that address important organizational problems. The artifact must be described effectively, enabling its implementation and application in an appropriate domain. An artifact can be a construct, model, method, or an instantiation [122].

In this thesis we address organizational problems caused by the spread of process information over various informational artifacts. Our contributions include novel conceptualizations of behavioral uncertainty and event-class similarity, which play key

roles in specific application scenarios. Furthermore, we provide five methods, also referred to as techniques, that allow organizations to more effectively deal with fragmented process information. For instance, one of the techniques allows organizations to automatically identify inconsistencies between textual process descriptions and process models. Thereby, the method supports organizations in their maintenance of process information, helping to ensure that all processes are executed correctly and efficiently. For each of these methods, we also present an instantiation (i.e., an implementation) in the form a Java prototype in order to demonstrate the method's applicability.

Guideline 2: Problem relevance. The relevance of design-science research is determined by addressing problems that are of value to a constituent community. For IS research, this community consists of practitioners who plan, manage, design, implement, operate, and evaluate information systems and the technologies that enable their development and implementation [122].

The general relevance of the research presented in this thesis stems from the widespread adoption of BPM in organizations (see e.g., [221, 254, 256]). Therefore, BPM research is of value to the community of BPM practitioners and the organizations in which BPM practices are applied. On a more specific level, the relevance of each of the problems that we address with our research was identified in existing literature or followed directly from the limitations of state-of-the-art solutions to effectively address the respective problems. For example, Chapter 4 presents a method for conformance checking in the context of uncertain mappings between event logs and process models. The relevance of this technique stems from the importance of conformance checking to ensure that processes are executed according to their specification [16, 21, 239], in combination with the inherent inability of mapping techniques to identify a single, correct mapping [40].

Guideline 3: Design evaluation. The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods. The business environment establishes the requirements upon which the evaluation of artifacts should be based. The evaluation of a designed IT artifact requires the definition of appropriate metrics and the gathering and analysis of appropriate data [122].

We evaluate all research contributions in this thesis based on data collections that were obtained from industrial and academic partners, i.e., so-called real-life data. For each contribution, we employ metrics that quantify the usefulness of the developed artifact for characteristics relevant to the specific problem addressed by the artifact. For example, to evaluate our technique for inconsistency detection between textual process descriptions and process models, we employ the well-known precision and recall metrics from the information-retrieval field [179, p.142]. In this context, *precision* quantifies the fraction of predictions made by the method that correspond to actual inconsistencies, whereas *recall* provides the fraction of actual inconsistencies that are successfully identified by the method.

Guideline 4: Research contribution. Design-science research must provide a novel, significant, and general contribution to the knowledge base. This requirement excludes so-called routine design, which is characterized by the application of existing knowledge and best-practices. Rather, scientific design requires the application of innovative methods to address unresolved problems or to improve existing solutions [122].

The research contributions of this thesis all meet this criteria by either address-

ing previously unresolved problems or by providing a considerable improvement over existing solutions. In particular, this thesis presents several techniques which are the first to address certain problems. For instance, we present the first approaches for the automated comparison of textual process descriptions to process models and the transformation of natural language PPI descriptions. Furthermore, we present conformance-checking methods that build on a novel conceptualization of process uncertainty that considerably improves their practical applicability over existing methods. Lastly, our technique for process model matching can obtain results that existing techniques are unable to achieve, because our method analyzes event-log information, which is ignored by others.

Guideline 5: Research rigor. Design-science research requires the application of rigorous methods in both the construction and evaluation of designed artifacts. In this context, rigor is derived from the effective use of the knowledge base, i.e., theoretical foundations and research methodologies. Success is predicated on the researcher's skilled selection of appropriate techniques to develop or construct an artifact and the selection of appropriate means to evaluate it [122].

In the creation of our design artifacts, we made effective use of research from various research streams, including process model analysis, Natural Language Processing (NLP), process model matching, and machine learning. For example, both of our conformance-checking techniques build on existing conformance-checking approaches used in process analysis [281], whereas our techniques that analyze textual process descriptions build on a state-of-the-art technique presented in [99]. Furthermore, we formalize the designed artifacts such that readers can successfully reconstruct their functionality.

Guideline 6: Design as a search process. Problem solving in design science can be defined as utilizing suitable means to reach desired ends while respecting laws imposed by the environment [252]. It is an inherently iterative process that consists of a generate/test cycle. Given the wickedness of most IS problems, it may not be possible to describe all means, ends, and laws relevant to a problem [269]. Hence, design-science research often aims for a *satisfactory solution* without the need to consider all possible solutions [252].

To varying degrees, all our research contributions deal with the analysis of natural language in process-information artifacts. Due to the variability of natural language, it is not possible define all aspects (e.g., laws) of the problems. This leads to a theoretically indefinite number of ways to address the problems on which our research focuses. Therefore, rather than undertaking the impossible task of searching through all possibilities, we aim to construct a satisfactory solution to each of the considered research problems. For example, due to variability of natural language, it is impossible to define an approach that can perfectly analyze every conceivable textual process description. As a result, our method for the detection of inconsistencies between textual descriptions and process models may not be able to identify every single inconsistency, but rather strives for the accurate identification of the vast majority of inconsistencies. We justify the satisfactory nature of our solutions by comparing the evaluation results of the solutions to state-of-the-art techniques or other relevant benchmarks.

Guideline 7: Communication of research. Design-science research must be presented both to technology-oriented as well as management-oriented audiences [122].

Technology-oriented audiences need sufficient detail to enable the implementation of described artifacts and their use within an appropriate organizational context. Management-oriented audiences need sufficient detail to determine if organizational resources should be committed to obtain and use the artifact within their specific organizational context. Zmud [294] furthermore suggests that the emphasis for management-oriented audiences should be on the importance of the problem, as well as the novelty and effectiveness of the solution.

Each of the contributions presented in this thesis has been published in peer-reviewed academic journals and/or international conferences (see Section 1.4). Furthermore, we provide prototypical open source implementations for each of the designed artifacts. Hence, the presented techniques are readily available to interested researchers and practitioners.

The discussion of the design-science research guidelines in relation to the work presented in this thesis illustrates that our research contributions fulfill internationally established research standards. Further, the discussion shows that the presented research makes a significant contribution to the body of knowledge of the IS discipline.

1.4 Publications

This doctoral thesis is a monograph that presents research on the comparison and alignment of process information in various representation formats. Parts of this research have also appeared in peer-reviewed, scientific outlets, which has currently led to the publication of 2 journal articles and 7 conference papers. In comparison to these publications, this doctoral thesis presents the conducted research in a more extensive and integrated manner. Furthermore, Chapter 5 contains considerable conceptual extensions and additional evaluations in comparison to the form in which the conformance-checking approach has currently been published. The following list provides an overview of the publications stemming from the research in this doctoral thesis:

The spread of process information in organizations:

- Van der Aa, H., Leopold, H., Mannhardt, F., Reijers, H.A.: On the fragmentation of process information: Challenges, solutions, and outlook. In: International Conference on Enterprise, Business-Process and Information Systems Modeling, pp. 3–18. Springer (2015)
- Van der Aa, H., Leopold, H., van de Weerd, I., Reijers, H.A.: Causes and consequences of fragmented process information: Insights from a case study. In: 23rd Americas Conference on Information Systems, AMCIS (2017)

Comparing process models to textual process descriptions:

- Van der Aa, H., Leopold, H., Reijers, H.A.: Detecting inconsistencies between process models and textual descriptions. In: International Conference on Business Process Management, pp. 90–105. Springer (2015)

- Van der Aa, H., Leopold, H., Reijers, H.A.: Comparing textual descriptions to process models: The automatic detection of inconsistencies. Information Systems 64, 447–460 (2017)

Reasoning in the presence of behavioral uncertainty:

- Van der Aa, H., Leopold, H., Reijers, H.A.: Dealing with behavioral ambiguity in textual process descriptions. In: International Conference on Business Process Management. pp. 271–288. Springer (2016)

- Van der Aa, H., Leopold, H., Reijers, H.A.: Checking process compliance on the basis of uncertain event-to-activity mappings. In: International Conference on Advanced Information Systems Engineering. pp. 79–93. Springer (2017)

Aligning and transforming natural language PPIs:

- Van der Aa, H., Del-Río-Ortega, A., Resinas, M., Leopold, H., Ruiz-Cortés, A., Mendling, J., Reijers, H.A.: Narrowing the business-IT gap in process performance measurement. In: International Conference on Advanced Information Systems Engineering. pp. 543–557. Springer (2016)

- Van der Aa, H., Leopold, H., del Rio-Ortega, A., Resinas, M., Reijers, H.A.: Transforming unstructured natural language descriptions into measurable process performance indicators using hidden markov models. Information Systems 71, 27–39 (2017)

Process model matching based on event-log information:

- Van der Aa, H., Gal, A., Leopold, H., Reijers, H.A., Sagi, T., Shraga, R.: Instance-based process matching using event-log information. In: International Conference on Advanced Information Systems Engineering. pp. 283–297. Springer (2017)

The author of this doctoral thesis has also contributed to several research projects beyond the scope of the monograph. These projects have currently led to the publication of 2 journal articles, 4 conference papers, and 1 workshop paper. These publications focus on, among others, the composition of workflow activities, business process noncompliance, process querying, evaluation of process model matching, and the prediction of deviations in logistics chains. The following list provides an overview of the publications:

Other publications:

- Van der Aa, H., Reijers, H.A., Vanderfeesten, I.: Composing workflow activities on the basis of data-flow structures. In: International Conference on Business Process Management, pp. 275–282. Springer (2013)

- Van der Aa, H., Leopold, H., Batoulis, K., Weske, M., Reijers, H.A.: Integrated process and decision modeling for data-driven processes. In: Business Process Management Workshops. pp. 405–417. Springer (2015)
- Van der Aa, H., Reijers, H.A., Vanderfeesten, I.: Designing like a pro: The automated composition of workflow activities. Computers in industry 75, 162–177 (2016)
- Andrade, E., van der Aa, H., Leopold, H., Alter, S., Reijers, H.A.: Factors leading to business process noncompliance and its positive and negative effects: Empirical insights from a case study. In: 22nd Americas Conference on Information Systems, AMCIS (2016)
- Di Ciccio, C., Van der Aa, H., Cabanillas, C., Mendling, J., Prescher, J.: Detecting flight trajectory anomalies and predicting diversions in freight transportation. Decision Support Systems 88, 1–17 (2016)
- Kuss, E., Leopold, H., Van der Aa, H., Stuckenschmidt, H., Reijers, H.A.: Probabilistic evaluation of process model matching techniques. In: Conceptual Modeling: 35th International Conference, ER 2016, Gifu, Japan, November 14-17, 2016, Proceedings 35. pp. 279–292. Springer (2016)
- Leopold, H., van der Aa, H., Pittke, F., Raffel, M., Mendling, J., Reijers, H.A.: Integrating textual and model-based process descriptions for comprehensive process search. In: International Conference on Enterprise, Business-Process and Information Systems Modeling. pp. 51–65. Springer (2016)

1.5 Thesis Outline

The remainder of this thesis is divided into the following seven chapters:

- *Chapter 2: Background.* This chapter provides background information essential to the alignment and comparison of process information contained in different informational artifacts. The chapter focuses on three relevant aspects: (i) process information in organizations, (ii) NLP tasks and techniques, and (iii) application scenarios and techniques of matching.
- *Chapter 3: Comparing Process Models to Textual Process Descriptions.* This chapter presents our technique for the automated detection of inconsistencies between process models and textual process descriptions. In particular, our technique identifies two types of inconsistencies: (i) process model activities that are not contained the accompanying textual description and (ii) conflicting orders between process model activities and process steps described in the text. A quantitative evaluation with 53 real-world model-text pairs demonstrates that our four-step approach accurately identifies these inconsistencies.
- *Chapter 4: Conformance Checking based on Uncertain Event-Activity Mappings.* This chapter presents a conformance-checking technique that can be used in the presence of uncertain event-to-activity mappings. Our technique provides

conformance-checking results without the need to select a single, possibly incorrect mapping to base conformance checks on. We achieve this by considering the entire spectrum of possible mappings generated by event-to-activity mapping techniques. As a result, our conformance-checking technique avoids the risk of drawing incorrect conclusions about conformance. A quantitative evaluation based on a large collection of real-world process models demonstrates that our technique can be used to obtain results in a vast number of cases where traditional conformance-checking techniques fail to do so.

- *Chapter 5: Dealing with Ambiguity in Textual Process Descriptions.* This chapter presents our method which checks the conformance of observed process behavior against ambiguous textual descriptions. The method builds behavioral spaces as a means to capture all possible interpretations of a textual process description in a systematic manner. By using behavioral spaces for conformance checking, we avoid the need to impose assumptions on the correct interpretations of ambiguous natural language texts. Therefore, conformance checks based on behavioral spaces avoid the risks associated with the assumption-based selection of interpretations. We use a quantitative evaluation with a set of 47 textual process descriptions to demonstrate the usefulness of behavioral spaces for conformance checking in the context of real-world textual descriptions.
- *Chapter 6: Transforming and Aligning Process Performance Indicators.* This chapter presents our approach for the automated transformation of natural language descriptions into measurable PPIs. The proposed approach first transforms an unstructured natural language description into a structured format. In this step, it extracts information on concepts relevant to the calculation of a PPI from the natural language description. Secondly, our approach aligns the process concepts contained in the description with the corresponding elements of a process model. To this end, it establishes links between a PPI description and the implementation of the process. As a result, the approach delivers structured and aligned PPI descriptions that can be directly used for automated monitoring of process performance. We use a quantitative evaluation with a set of 129 PPI descriptions obtained from practice to demonstrate that our approach works accurately.
- *Chapter 7: Process Model Matching using Event-Log Information.* This chapter presents techniques for process model matching based on event-log information. We present six different matchers to quantify process similarity based on event logs. A quantitative evaluation with real-world data shows that by just considering the information specific to event logs, our matchers can identify a considerable number of correspondences between event classes. By combining our instance-based matching techniques with traditional, model-based techniques users can strive to obtain matching results that cannot be obtained by using either of these techniques alone.
- *Chapter 8: Conclusion.* This chapter concludes the doctoral thesis. In this chapter, we summarize the main results presented throughout the thesis. Furthermore, we reflect on the implications of these results for research and practice, as well as on directions for future research that stem from the presented work.

2 Background

This chapter discusses background essential to the alignment of process information contained in different informational artifacts. The chapter consists of three parts, each of which focuses on an aspect relevant to any alignment task, as visualized in Figure 2.1. The input to an alignment task is formed by two informational artifacts related to a business process. Section 2.1 reflects on these input artifacts by discussing the role and presence of process information in organizations, whereas Section 2.2 provides formal definitions for the main types of informational artifacts contained in this thesis.

In the first step of an alignment task, relevant process information is extracted from the informational artifacts. A major part of the process information considered in this thesis is described in the form of natural language. Therefore, Section 2.3 discusses Natural Language Processing (NLP) techniques that form the foundation for this information-extraction step. Afterwards, the information extracted from the two artifacts must be related to each other in order to obtain the alignments that form the basis for various analysis techniques, such as the detection of inconsistencies between the artifacts and process querying. So-called *matching* techniques aim to automatically establish such alignments. Therefore, Section 2.4 presents an overview of the relevant aspects of matching.

2.1 Process Information in Organizations

This section gives an overview of the role and presence of process information in organizations. Section 2.1.1 introduces the discipline of BPM, which provides the context for the use of process information. Section 2.1.2 then defines the notion of process information and discusses its importance throughout the various phases of a BPM lifecycle. Afterwards, Section 2.1.3 describes the task of capturing process information in the form of informational artifacts and the associated causes of process-information fragmentation. Finally, Section 2.1.4 concludes this overview by considering the negative impact that the fragmentation of process information can have on organizations.

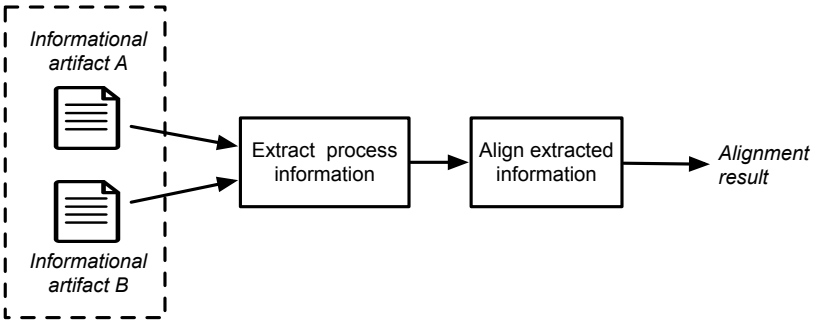


Figure 2.1: Overview of a generic alignment task

2.1.1 Business Process Management

Business Process Management (BPM) aims for the efficient coordination of business-related activities within and between organizations [186, p.5]. BPM represents one of the core concepts that enables organizations to flexibly react to constantly changing business environments [162, p.1]. The focal point and core concept of BPM, as well as of this thesis, is a *business process*. A multitude of definitions for this concept have been provided in literature. A seminal definition by Hammer and Champy [121, p.38] characterizes a business process as a collection of activities, transforming an input into an output that is valuable for a customer. This definition pinpoints two key concepts present in most definitions: a collection of *activities* (or process steps) and an *output* that provides value to a customer. Regarding the collection of activities in a business process, other definitions provide a more strict view. Davenport and Short describe this collection as a "*set of logically related-tasks*" [71], Dumas et al. [87, p.5] as a "*chain of events, activities, and decisions*," while Van der Aalst and Van Hee [18] emphasize that the order of activities is determined by pre-defined conditions. Hammer and Champy's definition also states that the output should provide value to a customer. This customer can either be external or internal to the organization [87, p.4]. Furthermore, Becker and Kugeler [46] stress that this goal is directed by the business objectives and environment of an organization. Without a loss of generality, we here adopt the definition by Dumas et al. because it emphasizes the core aspects of business processes that are relevant to this thesis.

Definition 2.1 (Business Process)**.** *A* business process *is a collection of inter-related events, activities and decision points that involve a number of actors and objects, and that collectively lead to an outcome that is of value to at least one customer [87, p.5].*

Despite the widespread use of the term Business Process Management, there is no agreement on its meaning. Rather, there are certain activities with respect to business processes that are commonly gathered under this name [229, p.11]. Notably: the design, analysis, modeling, implementation and control of business processes [18, 75, 247]. Since this thesis focuses on process information from various sources and for a broad variety of BPM purposes, we adopt the definition of BPM proposed by Van der Aalst et al. [17]:

Definition 2.2 (Business Process Management (BPM))**.** *BPM supports business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information [17].*

In the next section, we describe and define *process information* as information specifically related to BPM activities.

2.1.2 Process Information

To be able to reflect on the role of process information in BPM, we need to define what the term *process information* refers to. To do so, we first need to investigate the notion of *information* itself. In IT or IS contexts, information is typically defined by distinguishing it from *knowledge* and *data* [27]. There are different views on this distinction. What they share is the idea that the three form a hierarchy, such as depicted in Figure 2.2. For example, Vance [267] defines information as data that has been interpreted in the context of a meaningful framework, whereas knowledge is information that has been authenticated and thought to be true. Maglitta [177] characterizes data as raw numbers and facts, information as processed data, and knowledge as information that has been made *actionable*.

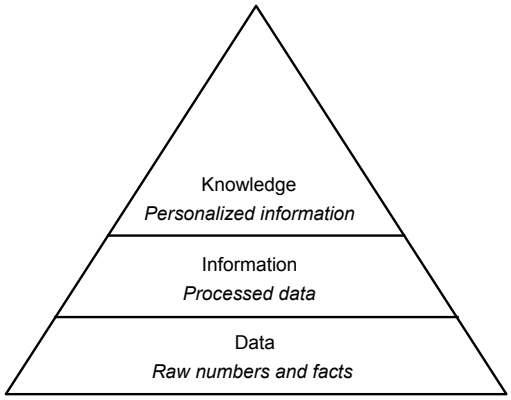


Figure 2.2: Hierarchy between data, information, and knowledge (adapted from [59])

The key characteristic that distinguishes knowledge from information throughout the definitions is that knowledge is information that exists in the mind of individuals [27]. It is *personalized* information, which may or not be unique or correct. As Fahey and Prusak [91] suggest, "*knowledge does not exist independently of the knower*." This implies that according to a strict definition, knowledge cannot exist in documents, systems, or other artifacts. Information is converted to knowledge once it is processed in the mind of individuals and knowledge becomes information once it is articulated and presented in the form of text, graphics, or other forms [27].

The typical characteristic used to distinguish data from information is that the latter has undergone some form of processing. However, as Alavi and Leidner [27] point

out, the presumption of a strict hierarchy between these two rarely survives a thorough evaluation. Whether something is data or information can depend on the particular need of a user. For instance, a timetable for a train can be regarded as raw data by some, but may very well provide *information* instead of data to a user who is interested in learning which train to take for a trip. Because of this fuzzy boundary, we adopt a broad view on what we consider process information in this thesis. Specifically, we consider anything that has a process-orientation, which has been captured in a document, system, or through other means, as information. This includes, for example, *event-log information* that has been automatically recorded and stored. However, this view on information excludes raw numbers without a known relation to a process or its activities. Instead, we regard those as data.

Building on this notion of information, we characterize *process information* as information that is relevant to BPM activities. Michelberger et al. [190] present a definition of process information that captures this:

Definition 2.3 (Process information)**.** *Process information refers to data that has been processed to support process users in the modeling, execution, monitoring, optimization, and design of processes, so that it has a meaning and value with respect to the process users' activities [190].*

There is no established taxonomy of informational artifacts that suits this or other definitions of process information. However, it is clear from literature that process information encompasses a broad variety of types and formats. Michelberger et al. [190] consider process information to include textual process descriptions, working guidelines, graphical process models, operational instructions, forms, checklists, lessons learned, and best practices. Other types of process information relevant to this thesis are (definitions of) PPIs [233] and event logs [13]. One way to understand why this broad range of process-information artifacts exists is to reflect on why and how process information is captured by organizations. This is considered in Section 2.1.3.

2.1.3 Capturing Process Information

Following Browning [55] and Figl and Recker [93], we can draw a parallel between the task of capturing process information, in any format, and the task of *modeling* a business process. This parallel is useful because key principles of modeling are also applicable to other means of capturing process information, such as textual descriptions and checklists. Before investigating the principles of modeling, it is important to recognize that these apply to the task of capturing information *manually*, as well as *automatically*. Automatically capturing process information occurs, for instance when recording event-log information or generating a process model from an event log. The key difference here is that when manually capturing information, decisions on what information to capture and how to do so are made while modeling (akin to *run-time* decisions). When automatically capturing information similar decisions have to be made. However, by contrast, the decisions are then made when selecting or designing the technique that performs the capturing (i.e., *design-time* decisions).

Different definitions for the concept of a model are proposed in literature. Stachowiak [258, p.131] defines a model as a simplifying mapping from reality that serves

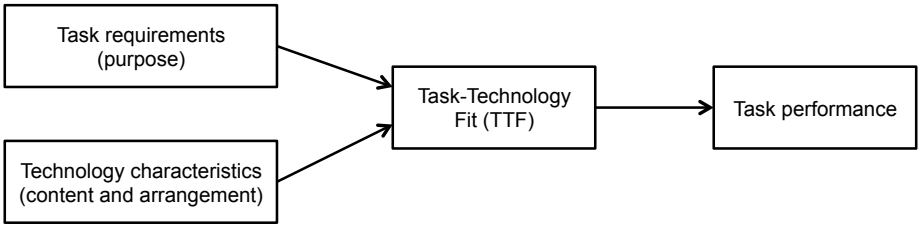


Figure 2.3: Main constructs in Task-Technology Fit (TTF) theory

a specific purpose. Steiger [259] defines it as an abstract representation of reality that is built, verified, analyzed, and manipulated to support a particular purpose, even if that purpose is merely to increase understanding of a situation. What these and other definitions share are two crucial, related characteristics: (i) a model is a *simplification* of reality and (ii) a model has a specific *purpose*. The combination of these characteristics results in the maxim by Box [50] that "*all models are wrong, but some are useful*." Models are "wrong", because they do not fully represent reality, but rather selectively abstract key information. Therefore, a model provides a simplification of reality tailored to the needs of a specific purpose.

To be able to reflect on the purpose of a model in detail, Browning [55] draws an insightful parallel between the usefulness of a model for a particular purpose and Task-Technology Fit (TTF) theory. TTF theory, introduced by Goodhue and Thomson [113], argues that a technology (i.e., a means to accomplish a task) will improve a user's performance if it matches the task's requirements well. In a BPM context, this means that a model of a process is more useful if it provides the right information in a suitable manner. Therefore, the informational content and representation format of an informational artifact must be in line with the artifact's intended purpose. This can also be seen in Figure 2.3, where the characteristics of a technology are defined according to its content and arrangement (i.e., its representation format).

Informational content

To accommodate for the information needs of various stakeholders involved in the BPM lifecycle, process-information artifacts often differ in terms of their *informational content*. These differences can be described according to two main dimensions: the provided level of detail and the perspective taken on a process.

First, informational artifacts vary in the *level of detail* that they provide to users. For instance, *process owners* and *process analysts* typically require an overview of an entire process [87, p.24]. Artifacts such as process models are well-suited for this purpose, because they can provide a high-level view on a process by describing it from start to end. For other stakeholders, most notably *process participants*, it is more important to understand how to perform certain tasks, than to understand the whole process. For these stakeholders, *work instructions* are more useful, because they provide detailed information on the execution of specific tasks [94].

Second, informational artifacts provide information regarding different *process per-*

spectives. A typical distinction can here be made between the technical and operational perspectives [280]. For example, a process model designed to support process implementation will emphasize technical details, such as the applications that should be called and the IT systems to be accessed. By contrast, a process model that takes an operational perspective will focus on the tasks that must be performed and, for instance, who should perform these. The difference between models that cover these perspectives is representative of the well-known *Business-IT-Gap* (cf. [149, 174]). As another example, *process owners* can be mainly concerned with aspects related to the *performance perspective* of processes [233], rather than operational or technical perspectives.

Representation Formats

A variety of representation formats can be used to capture process information in informational artifacts [287], including process models [72], natural language descriptions [212], spreadsheets [152], and checklists [230]. The representation format used to provide process information to users should be well-suited for its particular purpose. This suitability depends on two main factors. A format should convey its informational content in a useful manner and the intended users should be able to work with the utilized format well.

Representation formats emphasize different aspects of business processes. This means that the choice for a certain representation format depends on the intended focus of an informational artifact. For instance, *natural language text* can be very useful to provide process participants with detailed insights on how to perform complex tasks [39]. However, for a process participant who needs to be sure that all necessary steps are performed, a *checklist* is more useful. This latter format is more suitable because it emphasizes the information that is of primary importance for that purpose. Furthermore, process models have been found to be better suited to express complex execution logic of a process in a more comprehensive manner than natural language [186, p.23].

It is also important that users of an informational artifact are able to work well with an employed representation format. The ability of users to do so can depend on their familiarity and preferences with respect to different formats. Research by Figl and Recker [93] shows that people prefer different process representation formats depending on the application purpose and on the *cognitive style* of the user. For example, some participants were found to prefer textual descriptions over process models, whereas others preferred models over text for the same purpose. The influence of user preferences on the choice for model or text-based process representations is also recognized by Recker et al. [228] and Chakraborty et al. [60].

Fragmentation

The need for the availability of artifacts with different informational contents and representation formats means that often multiple artifacts are required to provide information on a single business process. Figure 2.4 depicts a classical example of this situation. The figure shows three informational artifacts that capture information on a loan application process: (1) a process model, (2) a natural language work instruc-

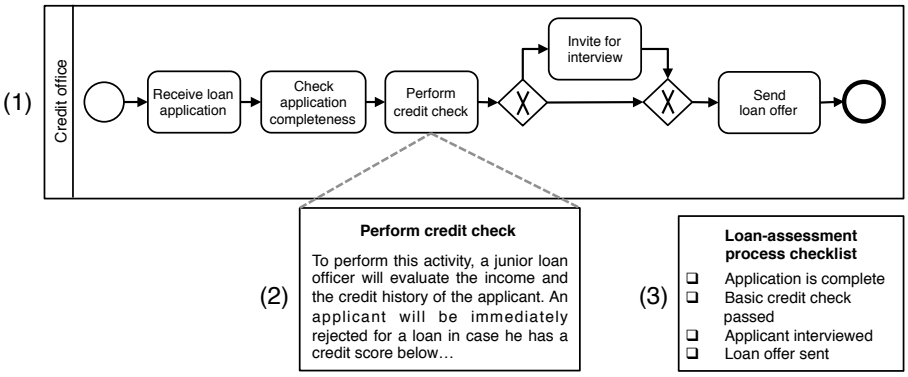


Figure 2.4: Loan application process with fragmented process information

tion, and (3) a checklist. This *fragmentation* has clear advantages because it allows an organization to provide users with the process information that is most relevant to them, in a suitable format. For instance, a process participant who performs a *credit check* in the loan application process can use the textual work instruction (2) to obtain all information relevant to the execution of this particular task. By contrast, a different participant can use the checklist (3) as a means to ensure that all necessary steps have been performed when assessing a loan application, before sending the *loan offer* to the applicant. However, despite these advantages of having various informational artifacts, the spread of process information can also pose considerable threats to the efficient execution and management of an organization's processes. We reflect on this in the next section.

2.1.4 The Problem of Fragmented Process Information

The fragmentation of process information over different informational artifacts can pose considerable problems to organizations. To understand the possible severity of these issues, it is important to grasp how fragmented process information can be in practice. For this purpose, we first illustrate this based on our observations during a case study described in [10]. In this case study, we investigated the existence and use of informational artifacts related to 23 processes of the procurement department of a major manufacturing company.

We found that process information was highly fragmented for some of the investigated processes. One of the most severe instances was observed for a *purchasing approval* process. We found more than 100 informational artifacts related to this single business process. Table 2.1 provides an overview of these artifacts, which we subdivided according to their information types and representation formats. Aside from the fragmentation in the form of this large number of informational artifacts, it is also important to note that these artifacts were spread out over various information systems.

[1]The organization used this format for materials in training settings.

[2]An informational artifact that contains both high-level and low-level process information, combining contents from textual process descriptions with work instructions.

Table 2.1: Informational artifacts for a purchasing approval process (from [10])

Information type	**Representation Format**	**Quantity**
Process model	EPC	1
Process description	Text document	5
	Slide set[1]	1
Work instruction	Text document	30+
	Spreadsheet	2
	Slide set	30+
	Video[1]	15
	E-learning module[1]	1
Hybrid documentation[2]	Text document	20+
	E-learning module	1

This latter problem occurs when process information is managed separately from the business processes themselves [190]. Process information is, for example, stored in shared drives, databases, enterprise portals, content management systems, and enterprise information systems [189, p.4].

Given a situation such as described above, the fragmentation of process information threatens the effective management and execution of an organization's business processes. We identify three main kinds of problems that occur due to fragmentation: (i) maintenance issues, (ii) efficiency issues, and (iii) business process noncompliance.

Maintenance Issues

Adapting business processes in order to respond to changing business needs is at the very heart of BPM [285]. Therefore, keeping process information in sync with continuous process improvement efforts is a highly important task [124]. This maintenance task is severely affected by the fragmentation of process information. Clearly, when more artifacts are affected by a process change, more effort is required to update all artifacts. However, the total effort can far exceed the time spent on actually updating information in the artifacts. If there is no traceability between process information contained in different artifacts, then considerable time must be invested to determine which information and which artifacts should be changed. In our case study, we found that the required efforts to maintain process information were very high. As one interviewee remarked: "*Sometimes a small change can mean tremendous effort*."

The amount of time and effort required to maintain process information represents a considerable problem in itself. However, it is crucial to recognize that organizations can only spend limited resources on maintenance, or any BPM activities in general [87, p.33]. Therefore, organizations may not be able to allocate sufficient resources to keep all process information in sync with reality. This comes with the highly problematic consequence that information can become outdated. In our case study, one of the interviewees described a recent clean-up effort in which 120 obsolete documents,

related to a handful of processes, were identified. When users execute processes based on such outdated information, this can lead to efficiency issues as well as business process noncompliance, as we discuss next.

Efficiency Issues

The fragmentation of process information can negatively affect the execution efficiency of processes. This can occur for two general reasons.

First, fragmentation can affect efficiency because it makes it hard for users to find the information that they need to perform their tasks in a process [55]. Because users may not have an overview of available process information [190], they may need to browse through numerous documents and systems before they find what they need. Such identification of information relevant to a specific work context has been recognized to be highly time-consuming and complex [238]. The time and effort required for this thus reduces the efficiency of the processes in which the user is involved. This consequence stands in sharp contrast to one of the major reasons why process information is fragmented. Namely, fragmentation occurs in order to provide users with the information that they need. However, this benefit is only achieved if users can actually *find* the right information in the first place.

Second, the execution efficiency of processes can be reduced because process participants do not have the right information that they need [35]. This can occur when the quality of process information is insufficient [191], such as when participants access outdated or otherwise incorrect processes. Furthermore, it can also happen that relevant process information has simply not been documented. For instance, in our case study, we observed cases where the authors of process descriptions chose not to describe certain steps that they perceived as trivial or irrelevant, but are in fact important to the users of the information. For instance, an interviewee—who oversees a process that provides reports (in the form of spreadsheets) related to strategic decisions—described the following: "*some process steps are not included [in the textual process descriptions], because they seemed natural to the designer. This leads to problems when working with the spreadsheets.*" Due to these issues, the interviewee spent the majority of her days cleaning up errors in these reports. In another case, a procurement manager acknowledged that he heavily relies on undocumented knowledge when performing his tasks. As a result, when he is absent, his replacement fails to perform certain tasks that should actually be performed on a daily basis.

These issues caused by the fragmentation of process information result in situations where considerable more time, effort, and, therefore, costs are required for the execution of an organization's processes than necessary.

Business Process Noncompliance

Business process noncompliance can be defined as any behavior that does not conform to the intended specification of a process [30]. Noncompliant actions can occur in various forms, such as the accidental omission of tasks, performing tasks incorrectly, or performing tasks without the proper authorization [263]. In the context of process-information fragmentation, noncompliant actions occur for the same reason as some

of the instances which lead to inefficient process execution: users execute processes based on incorrect or outdated process information [31]. In these cases, the use of incorrect process information negatively affects the outcome of the process. These effects of noncompliance can be severe, including a loss of control over business processes [239], reduced quality of process outcomes [31], or financial penalties imposed by authorities [173].

In our case study, we observed several instances of noncompliance caused by access to incorrect process information. In the above, we described a situation where a process owner spent considerable time correcting mistakes in spreadsheets-based reports, resulting from the incorrect execution of a reporting process. However, not all mistakes were caught in this way. As a result, management still received incorrect reports on strategic decisions made by the purchasing department. In another case, an employee used an outdated process specification while creating purchase orders. This ultimately led to extra costs incurred by the organization to clear up these problems with the supplier.

Problem Mitigation

The aforementioned problems demonstrate that the fragmentation of process information can have severe effects for organizations. From increased time spent on the execution and maintenance of processes, to noncompliance issues affecting the outcome of processes. There are two general directions that can help organizations to mitigate these problems: (i) reduce the extent of the fragmentation and (ii) maintain and use fragmented information more efficiently.

The extent of process-information fragmentation can be reduced by removing obsolete or redundant informational artifacts. Informational artifacts that do not have a distinct purpose from others should be discarded. In our case study, clean-up efforts revealed the existence of numerous outdated documents, as well as documents that were never actually used. Organizations can also take measures to prevent this situation by establishing guidelines for the documentation of process information. Such guidelines can, for instance, aim to standardize the content and structure of informational artifacts. General guidelines can, for instance, be found for process models in the form of the Seven Process Modeling Guidelines (7PMG) [187] and for use case descriptions [212]. A great additional benefit of standardization is that this makes process information easier to retrieve, which improves maintenance and execution efficiency. Nevertheless, it is important to realize that, even in spite of such organizational measures, the fragmentation of process information cannot be wholly avoided. Process stakeholders use informational artifacts for various purposes, which justifies the need for informational artifacts with different contents and representation formats. This calls for means to support this situation.

Technology can play an important role to support organizations to more efficiently maintain and use fragmented process information. Research has proposed concepts such as *data warehouses* as general solutions to provide users with information in various representation formats (see e.g., [62]). However, the use of data warehouses comes with great implementation efforts, requiring an overhaul of all existing process-information artifacts. For this reason, techniques that provide support to deal with the

current way in which processes are documented can be highly valuable. Existing techniques, among others, focus on the propagation of process changes [280], the detection of inconsistencies between process models [43, 123, 266], and the integration of information from different artifacts [157, 178]. A limitation is that this existing research focuses primarily on highly-structured process information in the form of process models and event logs. Process information in less-structured formats, such as natural language documents, is largely ignored by existing work. This leaves a considerable research gap, which we aim to address in this thesis.

2.2 Core Definitions

The main process-information artifacts that we consider in the remainder of this thesis are process models, event logs, textual process descriptions, and textual descriptions of PPIs. The latter two artifacts comprise unstructured natural language text, for which no formal definition can be provided. We formally define the former two artifacts, process models and event logs, in the remainder of this section.

2.2.1 Process Models

Process models can be created using a variety of modeling languages, such as Petri nets, Event-driven Process Chains (EPCs), and the Business Process Model and Notation (BPMN). The contributions of this thesis that take process models into account are independent of the specific notation used to define a process model. Therefore, we define process models using a generic definition adapted from Smirnov et al. [255], given in Definition 2.4.

Definition 2.4 (Process Model)**.** A process model is a tuple $M = (A, E, G, N, F, t)$, where:

- A is a finite set of activities,
- E is a finite set of events,
- G is a finite set of gateways,
- $N = A \cup E \cup G$ is a finite set of nodes,
- $F \subseteq N \times N$ is the flow relation, such that (N, F) is a connected graph,
- $t : G \rightarrow \{and, xor\}$ is a mapping that associates each gateway with a type.

Figure 2.5 depicts an exemplary process model conforming to this definition. Rounded rectangles denote activities from set A, such as "*Check document completeness*". Circles denote events from set E, such as "*Application received*". Furthermore, the arrows are used to depict the flow relation F that exist between nodes from the set $N = A \cup E \cup G$. Gateways from set G, which represent routing points in the flow F, are depicted by diamond shapes. The diamond shapes that contain a plus symbol indicate concurrent streams of action. This means that, in the provided example, the "*Check document completeness*" and "*Evaluate credit score*" activities can be executed at the same time or in an arbitrary order. Finally, diamond shapes containing a cross represent decision points. In Figure 2.5 either "*Decide on high-value loan*" or "*Decide on low-value loan*" can be executed for a case going through the process, but not both.

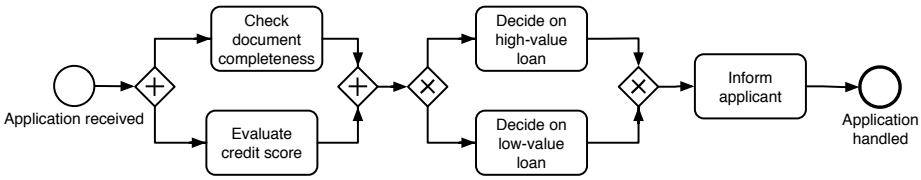


Figure 2.5: Exemplary process model of a loan application process

In order to precisely characterize the behavior that is allowed by a certain process model, we first define the predecessors and successors of nodes, based on definitions from Leopold [162, p.14].

Definition 2.5 (Predecessors and Successors)**.** *Let N be the set of nodes and $F \subseteq N \times N$ a binary relation over N representing the flow relation. For each node $n \in N$, we define the set of preceding nodes $\bullet n$ as $\{x \in N \mid (x, n) \in F\}$ and the set of succeeding nodes nodes $n\bullet$ as $\{x \in N \mid (n, x) \in F\}$.*

Based on the predecessors and successors of nodes, we can define the sets of *start* and *end* events of a process model. Given the set of events of a process model as E, we define the start events as $E_{\text{start}} = \{e \in E \mid \bullet e = \}$, i.e., the events without preceding nodes, and the end events as $E_{\text{end}} = \{e \in E \mid e\bullet = \}$, i.e., the events without succeeding nodes. For the process model in Figure 2.5, $E_{\text{start}} = \{$*Application received*$\}$ and $E_{\text{end}} = \{$*Application handled*$\}$.

Table 2.2: Activity identifiers for the process model depicted in Figure 2.5

ID	Activity
a	Check document completeness
b	Evaluate credit score
c	Decide on high-value loan
d	Decide on low-value loan
e	Inform applicant

Given the sets of starts and end events of a model, we can define the set of *allowed execution sequences* to capture all activity sequences that represent a proper execution sequence of a process model. Given an activity set A, an activity sequence $\sigma \in A^*$ represents a sequence of activity executions, also denoted as $\sigma = \langle a_1, \ldots, a_n \rangle$, where for each $1 \leq i \leq n$ it holds that $a_i \in A$. For a process model M, we denote the set of allowed execution sequences as $\Sigma_{\text{valid}}(M)$. An *allowed* execution sequence $\sigma \in \Sigma_{\text{valid}}(M)$ is a sequence of activities between an event in E_{start} and an event in E_{end} that follows the execution semantics of a process model. One way to formally determine these semantics is to convert a process model into a Petri net, which can be achieved through transformations described by Dijkman et al. [80]. Using the activity identifiers provided in Table 2.2, the execution semantics of the model result in the following set of valid

sequences for the model depicted in Figure 2.5: $\Sigma_{\text{valid}}(M) = \{\langle a, b, c, e\rangle, \langle b, a, c, e\rangle, \langle a, b, d, e\rangle, \langle b, a, d, e\rangle\}$.

2.2.2 Event Logs

Event logs are process-information artifacts that contain events recorded by systems throughout the execution of a business process. These events represent run-time information of how a process was actually executed. Table 2.3 shows a fragment of such an event log. The table shows events related to the loan application process depicted in Figure 2.5. This section presents the formal definitions of events and event logs that are of relevance to the remainder of this thesis. These definitions are based on the definitions provided by Van der Aalst [13, pp. 98–105].

Table 2.3: Excerpt of an exemplary event log

		Properties			
Case id	**Event id**	**Timestamp**	**Activity**	**Resource**	**...**
1	129301	03-08-2017:11.02	Check doc. completeness	Claudio	...
	129302	04-08-2017:15.24	Evaluate credit score	Adela	...
	129303	07-08-2017:12.34	Decide on high value loan	Ermeson	...
	129304	07-08-2017:13.30	Inform applicant	Jeroen	...
2	578401	09-10-2017:09.30	Check completeness	Franzi	...
	578402	10-10-2017:15.24	Evaluate score	Jennifer	...
	578403	11-10-2017:10:15.	Decide on low value loan	Giuseppe	...
	578404	11-10-2017:15.48	Inform applicant	Justine	...
...	...	...	...	...	...

Table 2.3 illustrates that each event is associated with various properties, also referred to as *attributes*. Definition 2.6 captures this.

Definition 2.6 (Event, Attribute)**.** *Let $\mathcal{E}$ be the event universe, i.e., the set of all possible event identifiers. Events may be characterized by various attributes, e.g., an event may have a timestamp, correspond to an activity, and be executed by a particular resource. Let AN be a set of attribute names. For any event $e \in \mathcal{E}$ and attribute name $n \in AN$: $\#_n(e)$ is the value of attribute n for event e. If event e does not have an attribute named n, then $\#_n(e) = \perp$.*

A number of standard attributes are generally assumed for events, although events are not required to have values for any of these [13, p.101]:

- $\#_{\text{activity}}(e)$ is the *activity* associated to event e;
- $\#_{\text{timestamp}}(e)$ is the *timestamp* of event e;
- $\#_{\text{resource}}(e)$ is the *resource* associated to event e.
- $\#_{\text{trans}}(e)$ is the *transaction type* associated to event e, such as *schedule*, *start*, and *complete*.

Table 2.3 illustrates the majority of these attributes. For instance, the event in the first row, with *Event id* 129301, corresponds to the "*Check doc. completeness*" *activity*, has the *timestamp 03-09-2017:11.02* and the *resource* executing the activity was *Claudio*. Although, for brevity, not depicted in the table, events can also have a *trans* attribute to capture the transaction type of an event. For example, the type *start* can be used to denote the moment at which the execution of an activity began, whereas *complete* can denote the moment at which the execution finished. These transactional statuses are particularly important when measuring process performance in the form of, for instance, processing times and waiting times.

Next, we define *event classifiers* as a means to group events together that share a common attribute. In many cases, events are grouped either based on their activity name, e.g., to create a set of events that all correspond to the "*Create credit score*" activity. However, events can also be classified based on (combinations of) other attributes, such the combination of the $\#_{\text{activity}}$ and $\#_{\text{trans}}$ attributes. Throughout this thesis, without loss of generality, we shall classify events based on the $\#_{\text{activity}}$ attribute, as captured in the following definition of *event classes*:

Definition 2.7 (Event Class)**.** *For any event $e \in \mathscr{E}$, $\underline{e} = \#_{\text{activity}}(e)$ is the event class of the event e.*

Aside from the standard attributes, events can be associated with activity-specific attributes. For instance, an event associated with the "*Evaluate credit score*" activity will, most likely, be associated with an attribute that indicates the *credit score* that results from the conducted evaluation. By contrast, events corresponding to different activities will be associated with other specific attributes.

An *event log* consists of cases and cases consist of events. The events for a case are represented in the form of a trace, i.e., a sequence of unique events. Moreover, cases, like events, can have attributes.

Definition 2.8 (Case)**.** *Let $\mathscr{C}$ be the case universe, i.e., the set of all possible case identifiers. Cases, like events, have attributes. For any case $c \in \mathscr{C}$ and attribute name $n \in AN$: $\#_n(c)$ is the value of attribute n for case c, where $\#_n(c) = \perp$ indicates that c has no attribute named n. Each case has a special attribute trace: $\#_{\text{trace}}(c) \in \mathscr{E}^*$. $\hat{c} = \#_{\text{trace}}(c)$ is a shorthand for referring to the trace of a case.*

Definition 2.9 (Trace)**.** *A trace is a finite sequence of events $\sigma \in \mathscr{E}^*$ such that each event appears only once, i.e., for $1 \leq i < j \leq |\sigma| : \sigma(i) \neq \sigma(j)$.*

Definition 2.10 (Event log)**.** *An event log is a set of cases $L \subseteq \mathscr{C}$ such that each event appears at most once in the entire log, i.e., for any $c_1, c_2 \in L$ such that $c_1 \neq c_2$: $\delta_{set}(\hat{c_1}) \cap \delta_{set}(\hat{c_2}) = \emptyset$.*

Finally, we shall use $\mathcal{E}(L)$ to refer to the set of event classes of an event log L, defined as $\mathcal{E}(L) = \{\#_{\text{activity}}(e) \mid c \in L \wedge e \in \hat{c}\}$.

2.3 Natural Language Processing

Natural Language Processing (NLP) is the scientific study of natural languages from a computational perspective [153]. *Natural* languages are the languages that people

speak, which have evolved over time and are hard to pin down with explicit rules [48, p.ix]. Research in NLP is highly interdisciplinary, involving concepts from the fields of computer science, linguistics, logic, and psychology [132]. NLP techniques are used in various application domains, including machine translation [143, 205], question answering [288, 290], text classification [130, 201], and information extraction [82, 184]. In this thesis, NLP is used for tasks that are related to this last domain: the extraction of information. Specifically, we use NLP techniques to extract process information from natural language texts. For instance, given the sentence "*Whenever a new order is received, an engineer first checks if the required parts are available*", we can build on NLP techniques to determine (i) which process steps are described, (ii) who performs these steps, and (iii) which ordering relations exist between them.

In this section, we first describe the three main NLP tasks that are relevant to this thesis. For the description of these tasks, our main goal is to clarify their usefulness, i.e., what can be achieved with them. Therefore, we put less emphasis on the technical aspects of approaches and implementations. Section 2.3.1 discusses *part-of-speech tagging*, which assigns a role to words in a sentence. Then, Section 2.3.2 considers parsing techniques, which identify the grammatical structure of sentences. Afterwards, Section 2.3.3 considers *reference resolution* in natural language texts. Following the discussion of the three main NLP tasks, Section 2.3.4 discusses the problem of *natural language ambiguity*, which plays an important role in this thesis. Finally, Section 2.3.5 concludes this part with a discussion of existing works that apply NLP techniques in a BPM context.

2.3.1 Part-of-Speech Tagging

Part-of-speech tagging, or simply *tagging*, is the task of assigning a part of speech, i.e., a tag indicating a word category, to each word in a natural language text [135]. This task provides important information for information-extraction purposes because its output provides insights into the role of words and their neighbors. For instance, when extracting process information, it is particularly important to understand if a multi-purpose term such as "*order*" corresponds to a *verb* or a *noun* in a particular sentence.

Part-of-speech taggers take as input a natural language text, most often a single sentence, and, as output, return a part-of-speech tag for each word in the text. As a basis for these taggers, various *tagsets* are available. These sets of part-of-speech tags are strongly language-dependent because tags reflect the word classes used by a language. For English, a seminal tagset is the one consisting of 87 tags used to annotate the *Brown corpus* [98]. Popular tagsets that evolved from the Brown tagset are the 61-tag *C5* set [106] and the 45-tag *Penn Treebank* set [181]. We here reflect in more depth on the latter, because it is used by the taggers that are part of the NLP tools employed in this thesis. Table 2.4 provides an overview of this set.

The following shows an example of the output provided by a tagger using the Penn Treebank tagset for the sentence "*An order can arrive at the department.*":

An/DT order/NN can/MD arrive/VB at/IN the/DT department/NN ./.

It is straightforward for human readers to interpret this sentence and provide the

Table 2.4: Overview of the Penn Treebank tagset (from [135, p.131])

Tag	Description	Example	Tag	Description	Example
CC	coord. conjunction	*and, or*	RB	adverb	*extremely*
CD	cardinal number	*one, two*	RBR	adverb, comparative	*never*
DT	determiner	*a, the*	RBS	adverb, superlative	*fastest*
EX	existential there	*there*	RP	particle	*up, off*
FW	foreign word	*noire*	SYM	symbol	*+, %*
IN	preposition or sub-conjunction	*of, in*	TO	"to"	*to*
JJ	adjective	*small*	UH	interjection	*oops, oh*
JJR	adject., comparative	*smaller*	VB	verb, base form	*fly*
JJS	adject., superlative	*smallest*	VBD	verb, past tense	*flew*
LS	list item marker	*1, one*	VBG	verb, gerund	*flying*
MD	modal	*can, could*	VBN	verb, past participle	*flown*
NN	noun, singular or mass	*dog*	VBP	verb, non-3sg pres	*fly*
NNS	noun, plural	*dogs*	VBZ	verb, 3sg pres	*flies*
NNP	proper noun, sing.	*London*	WDT	wh-determiner	*which, that*
NNPS	proper noun, plural	*Azores*	WP	wh-pronoun	*who, what*
PDT	predeterminer	*both, lot of*	WP$	possessive wh-	*whose*
POS	possessive ending	*'s*	WRB	wh-adverb	*where, how*
PRP	personal pronoun	*he, she*			

appropriate tags. However, even for such a simple sentence, automatically assigning a tag to each word is not trivial. This problem occurs because certain words in the sentence have multiple usages and meanings, which are associated with different parts-of-speech. This problem is referred to as *ambiguity* [135]. For example, "*order*" can refer to a noun (i.e., an order from a customer), but also to a verb (i.e., to order something). This means that "*order*" can have multiple associated tags: *NN* when it is used as a noun and *VB* or *VBP* for a verb. Similar problem occur for the term "*can*", which may be a noun, a verb, or a modal verb. Such tag ambiguity is highly common. A study by DeRose [77] found that over 40% of the words (or tokens) in the Brown corpus could be associated with multiple possible tags. The challenge of (automated) tagging is to resolve these ambiguities through *disambiguation*. Although each tagging algorithm addresses this challenge in its own manner, most approaches can be categorized as either rule-based or stochastic, or as a combination of the two [135, p.135].

Rule-based tagging algorithms use large collections of manually established rules to disambiguate among possible tags for words in a sentence. Rule-based approaches (cf. [115, 138, 141]) typically use a two-stage architecture. The first stage employs a dictionary to assign a list of potential parts-of-speech to each word, as illustrated in Table 2.5. The second stage uses the potential tags as input to a database of disambiguation rules. These rules eliminate possibilities until a single tag can be assigned to each word. For instance, a rule can be used to specify that an ambiguous word is a noun rather than a verb if it follows a determiner [135, p.135]. For example, "*order*"

Table 2.5: Possible tags for an exemplary sentence, with correct tags in bold

Word	Possible Tags	Word	Possible Tags
An	**DT**	at	**IN**
order	**NN**, VB, VBP	the	**DT**
can	NN, VB, VBP, **MD**	department	**NN**
arrive	**VB**, VBP		

is tagged with *NN* when it precedes a determiner such as "*an*", i.e., `an/DT order/NN`. The interested reader is referred to [138] for detailed insights into modern rule-based approaches.

Stochastic taggers disambiguate among potential tags by using an annotated training corpus to determine the probability of a given word having a given tag in a given context [135, p.135]. An important conceptual difference to rule-based taggers is that stochastic taggers *learn* methods for disambiguation, rather than depend on manually established rules. For example, given a tagged corpus, a stochastic parser can infer that the term "*order*" has a high (or possibly even 1.0) probability of receiving a *NN* tag when it appears directly after a determinant (*DT*) or an adjective (*JJ*). The parser will learn this, because it can count the number of times that such a sequence occurs. The earliest stochastic tagger was defined by Bahl and Mercer [36]. Several stochastic approaches followed up on this in the next decades (cf. [65, 69, 77]). A widely employed algorithm for stochastic tagging is the Hidden Markov Model (HMM) [196]. HMMs use *Bayesian inference* to determine the most probable sequence of tags for the words in a sentence.

Compared to rule-based taggers, HMM-based taggers have the great advantage that they assign tags by considering an entire sentence, rather than just looking at a word and its neighbors in isolation.

Both rule-based and stochastic tagging approaches have their own merits and downsides. A clear downside of rule-based taggers is that the construction and maintenance of a hand-crafted rule set is a highly time-consuming and labor-intensive task [136]. Furthermore, these approaches do not consider words in the context of an entire sentence, which can be necessary in certain case. Compared to rule-based taggers, HMM-based taggers have the great advantage that they assign tags by considering an entire sentence, rather than just looking at a word and its neighbors in isolation. However, a downside of stochastic taggers is that they require a large amount of training data to achieve high levels of accuracy. Furthermore, these approaches do not use any prior linguistic knowledge that might improve their accuracy. Recognizing these differences, several *hybrid* tagging approaches have been developed that combine aspects of rule-based and stochastic tagging. For example, works by Brill [53] and Brants [51] combine rule-based linguistic knowledge with stochastic methods that support context-dependent disambiguation.

In this thesis, we employ taggers, specifically HMMs, as part of our approach PPI-transformation approach described in Chapter 6.

2.3.2 Sentence Parsing

Natural language parsing, or simply *parsing*, is the task of assigning a grammatical structure to an input string. The difference to tagging is that parsing primarily focuses on the relations that exist among words, whereas tagging focuses on the role of individual words. A so-called *parser* takes as input a natural language text, most commonly a single sentence, and as output provides a structured representation of the grammatical relations for the words within this sentence. In this section we describe parsing according to two such representations: parse trees and dependency grammars.

Parse Trees

Parse trees are a commonly employed means to represent the grammatical structure of a sentence. The use of a tree to represent grammatical structure is based on the notion of *constituency*. Constituency states that groups of words can behave as a single unit or *phrase*, i.e., a constituent [135, p.385]. For instance, *noun phrases* can often be regarded as single units from a grammatical viewpoint. In these cases, it does not matter if a noun phrase consists of a single word, such as "*they*", or of a larger group of words, such as "*the parts required for a customized bike*"; in both cases, the grammatical relation of the noun phrase to surroundings words remains the same. For instance, for both noun phrases, a verb has to follow using the third person-plural form, e.g., "*they arrive*" and "*the parts required for a customized bike arrive*." A parse tree captures constituents in a hierarchical manner, such as visualized in Figure 2.6.

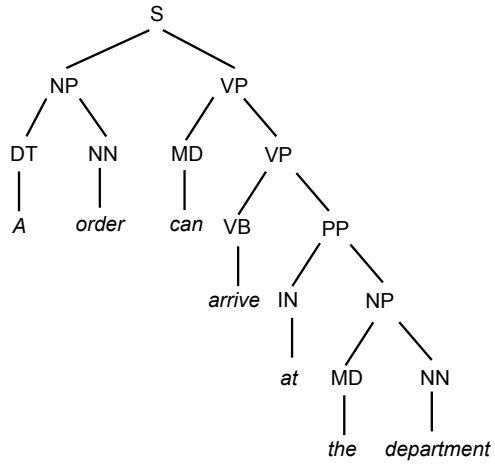


Figure 2.6: Example of a parse tree.

The way in which a parse tree can be structured is defined by so-called *production rules*. The most commonly used mathematical foundation for this purpose is the Context-Free Grammar (CFG), as originally formalized by Backus [34] and Chomsky [64]. A production rule in a CFG expresses the way in which the *symbols* of a language can be grouped and ordered together. For example, the following rule ex-

presses that a sentence can consist of a noun phrase (NP) followed by a verb phrase (VP):

$$S \rightarrow NP\ VP$$

In turn, the following rules specify that a noun phrase can be composed of either a pronoun (e.g., "*he*" or "*they*"), a proper-noun (e.g., a name), or of a determiner (*Det*) followed by a *Nominal*, where a *Nominal* refers to a constituency of one or more *Nouns*:

$$NP \rightarrow Det\ Nominal$$
$$NP \rightarrow Proper\text{-}Noun$$
$$NP \rightarrow Noun \mid NominalNoun$$

The goal of parsing is to automatically identify the most likely parse tree for a given sentence. This task can be viewed as a search through the space of possible parse trees to find the correct grammatical structure. Similar to part-of-speech tagging, it is important to disambiguate among the different meanings or different grammatical roles that words can have. In order to resolve these ambiguities, many parsers are based on *probabilistic* models of syntax, such as the Probabilistic Context-Free Grammar (PCFG) [127, p.237]. These models extend regular CFGs with information on the likelihood with which certain rules occur. As a basis for the parsing task, two sets of constraints guide the search process: one set contains the rules of the PCFG and the other is based on the input sentence. Both of these sets pose constraints on the final output of the parser. For example, constraints based on the input sentence define the leaves of the parse tree to be constructed. These two sets of constraints give rise to two search strategies: top-down and bottom-up search [88]. *Top-down* parsers search for a parse tree by trying to build from the root node S down to the leaves. These parsers thus start from the grammatical constraints of the CFG. By contrast, *bottom-up* parsers search for a parse tree by building up from the words of the input sentence [118, 183]. We refer the interested reader to e.g., [135, p.428] for a detailed explanation of both search strategies.

In this thesis, we employ parsing techniques in the approaches presented in Chapters 3 and 5. There, we use parse trees to extract phrases that correspond to relevant process concepts from textual process descriptions. For this task, we employ the PCFG parser included in the *Stanford Parser* [140].

Dependency Grammars

Dependency grammars provide an alternative means to represent the grammatical structure of sentences. These grammars capture the grammatical relationships among the words in a sentence using dependency relations. A *typed dependency relation* describes a relation of a certain type that exists between two words. For example, the relation *dobj(receive, goods)* denotes the *direct object* of a verb, i.e., "*goods*" is the object of the verb "*receive*". A commonly employed dependency grammar is defined by Marneffe et al. [74]; the dependencies in this grammar are referred to as the *Stanford Dependencies*. Table 2.6 presents some of the most common dependency relations in this grammar.

Table 2.6: Exemplary Stanford dependencies (from [74])

Dependency	**Description**	**Example**
nsubj	Nominal subject	*nsubj(arrive, order)*
dobj	Direct object	*dobj(receive, goods)*
det	Determiner	*det(the, department)*
amod	Adjectival modifier	*amod(order, new)*
nmod	Nominal modifier	*nmod(arrive, company)*
case	Case-marking	*case(arrive, at)*

To obtain the grammatical dependencies for a sentence, it is possible to exploit the close relation that exists between the dependencies and the structure of parse trees. Due to this relation, Stanford dependencies can be automatically derived from a parse tree [73, 289]. As such, developments in parsing algorithms can be utilized to generate grammatical dependencies with high accuracy. Alternatively, *dependency parsers* exist that directly compute grammatical dependencies from a sentence [170, 260]. While dependency parsers for English are less accurate than methods that derive dependency relations from parse trees, their application can be beneficial for other languages [73]. In particular, dependency parsers can handle languages with relatively free word ordering, such as *Czech*. In these cases, a phrase-structure grammar requires an abundance of rules to capture all word orders, whereas the use of grammatical dependencies avoids this problem [135, p.415].

In this thesis, we use grammatical dependencies in the approaches presented in Chapters 3 and 5. We use dependencies next to parse trees, because dependencies are more convenient for the extraction of word-pairs, such as verbs and their subjects. For this purpose, we use the Stanford Dependencies implementation provided by the Stanford Parser [140].

2.3.3 Reference Resolution

Reference resolution is the task of determining what entities are referred to by linguistic expressions [135, p.695]. We illustrate the importance of references in natural language texts by considering the following passage:

"*John von Neumann was one of the founding fathers of computing. The Hungarian-born mathematician invented the* merge sort *algorithm in 1945. He also made major contributions to the fields of mathematics, physics, economics, and statistics.*"

To properly interpret this passage, it is paramount to recognize that all three underlined phrases refer to same entity: *John von Neumann*. Reference resolution is the task that aims to achieve this. Here, we discuss two cases of this task: anaphora resolution and coreference resolution.

Anaphora resolution is the task of finding the *antecedent* of a single pronoun, e.g., *he* or *it*. For example, identifying that "*he*" refers to *John von Neumann*. Anaphora

resolution approaches take as input a natural language text, often comprising more than one sentence, and as output return the most likely antecedent for each pronoun in the text. A variety of approaches exist for this purpose, such as the seminal *Hobbs algorithm* [125], and more advanced techniques that followed up on this [107, 160]. Common among these approaches is that they check the agreement of certain features between the pronoun and potential antecedents. For instance, an important feature is the *numerical agreement* between pronoun and a candidate. A singular pronoun, e.g., "*he*", will not be used to refer to a plural entity such as "*managers*". Other typical features are the textual distance between pronoun and antecedent, and grammatical roles [135, p.701]. By analyzing a set of features, anaphora resolution approaches aim to identify the most likely candidate for an anaphoric reference.

Coreference resolution is the task of finding expressions in a text that refer to the same entity, i.e., finding expressions that corefer. This task encompasses the subtask of anaphora resolution. A set of coreferring expressions is called a *coreference chain*. For instance, the set {"*John von Neumann*", "*The Hungarian-born mathematician*", "*he*"} forms a single chain in the aforementioned passage. Coreference resolution presents a highly complex challenge, for which numerous techniques have been developed. These techniques employ machine learning methods, such as HMMs [257] and decision trees [185], but also exploit external knowledge sources, such as encyclopedic entries and dedicated corpora [217, 268]. Despite great advancements in the used techniques, the accuracy of resolution approaches still strongly depends on the context and on the availability of relevant information in corpora [220].

Reference resolution plays an important role in the approaches that focus on the analysis of textual process descriptions, Chapters 3 and 5. These texts typically contain numerous coreferences that must be resolved to properly understand the relations between described process steps.

2.3.4 Natural Language Ambiguity

Ambiguity is a pervasive problem in natural language comprehension [231]. As a result of this, the challenge that most NLP tasks face can be viewed as the need to resolve ambiguity, so-called *disambiguitation* [135, p.4]. In fact, disambiguation plays an important role in all of the previously considered NLP tasks: tagging, parsing, and reference resolution. In many cases, NLP techniques can successfully resolve ambiguity, especially when it occurs at the word-level. For example, given the sentence "*An order is received*," disambiguation techniques used in both taggers and parsers will correctly identify that "*order*" represents a noun rather than a verb in this sentence. Disambiguation can be highly challenging, especially in the context of coreference resolution. However, it is important to differentiate between ambiguity that can be theoretically resolved, i.e., given sufficiently advanced techniques, and *true* ambiguity that cannot be resolved without further contextual information, neither by automated approaches nor by humans.

To illustrate true ambiguity, consider the sentence "*The boy saw a man on the hill with a telescope*." This sentence can have numerous plausible interpretations. These interpretations vary, among others, on who is on the hill ("*the boy*" or "*a man*") and on the possessor or the location of the telescope ("*I*", "*a man*", or it is "*on the hill*").

For example, the boy can be using a telescope to see the man who is standing on the hill or the boy can be standing on the hill and see a man who has a telescope. Without further contextual knowledge, it is not possible to disambiguate among these different interpretations As a result, NLP techniques typically return what they consider to be the most likely interpretation, but there is no guarantee that this interpretation captures the correct meaning of the sentence.

Natural language ambiguity plays an important role in this thesis. As described in the previous section, the resolution of references, a key example of disambiguation, is necessary for the interpretation of textual process descriptions. Furthermore, in Chapter 5 we pay particular attention to the impact that true ambiguity has on the analysis of process descriptions from a compliance-checking perspective.

2.3.5 Applications in Business Process Management

A variety of works exist that apply NLP in the context of BPM. To provide an overview, we here subdivide the discussion into research that focuses on text *inside* process models and research that analyzes natural language *outside* of process models, captured in external text documents.

Approaches that analyze language inside process models focus on the labels of process model elements, mostly activity and event labels. Such techniques serve various purposes. A number of approaches aim to improve the *quality* of process models. These approaches analyze the quality and consistency of activity labels in a model, for example by detecting and/or correcting inconsistent use of terminology [148] or violations of labeling conventions [45,164,271]. Others aim to improve modeling quality by detecting common modeling errors [120] or ambiguously labeled activities [213, 214]. Some approaches *add information* to a process model by annotating model elements with semantic or ontological information [49, 97, 165]. Lastly, there are approaches that *distill information* by analyzing the labels of activities in a process model repository. Mostly, such techniques focus on the detection of *service candidates*, which correspond to process patterns that occur repeatedly throughout process models in a collection [100, 142, 168, 207].

Approaches that analyze process-related text documents primarily focus on the elicitation of process models from natural language texts. Several approaches have been developed for this purpose in recent years. These approaches elicit process models from different kinds of texts. Some focus on specific text types, such as the generation of models from use cases [253], group stories [111], or methodological descriptions [90], while others take general textual process descriptions as input [99, 108]. These techniques combine extensive use of NLP tools with specifically tailored analysis techniques, in order to extract process-related information from natural language texts. Despite the use of state-of-the-art NLP tools, the existing approaches have been found to produce inaccurate models, which require extensive manual revision [250].

The research presented in this thesis builds on several of the aforementioned approaches. Most prominently, in Chapter 5 we build on existing work on the generation of process models from natural language texts.

2.4 Matching

The automated techniques presented in this thesis focus on the comparison and integration of process information contained in different artifacts. These techniques share a reliance on the identification of *correspondences* between process concepts contained in the different artifacts. For example, correspondences between parts of a text and components of a process model are required in order to determine if a textual process description describes a process in the same way as a process model. In recent years, a plethora of so-called *matching* techniques have been developed that set out to automatically identify such inter-artifact relations [109]. Matching techniques are developed and applied in various fields, including schema matching and ontology alignment. *Schema matching* takes two database schemas as input and produces a mapping between elements of the schemas that correspond semantically to each other [224]. These mappings, also referred to as *alignments*, play a central role in applications, such as schema integration [44, 206], data warehousing [47], and semantic query processing [272]. *Ontology alignment* concerns the identification of correspondences between the elements of two ontologies [203]. Ontologies are abstract models that explicitly define concepts, their properties, and their inter-relations for a specific domain (cf. [119, 264]). Uses of ontology alignments include instance translation [68], ontology extension [85], and ontology merging [204]. The usefulness of matching has also been recognized in the context of BPM, most prominently in the form of *process model matching* techniques, which establish alignments between activities from different process models. These are, among others, used to detect differences between models [156], for the harmonization of process model variants [157], process querying [131], and to automatically propagate process changes [280]. Other techniques focus on the alignment of other process-information artifacts, such as techniques that align event classes from event logs to process model activities [37, 251].

In the remainder of this section, we elaborate on the matching task and matching techniques. We first illustrate the goal of matching in Section 2.4.1, by considering a process model matching scenario. Then, Section 2.4.2 provides definitions relevant to the matching task. In Section 2.4.3, we discuss measures to compute *textual similarity*, which represent a core component of nearly all matching techniques. Lastly, Section 2.4.4 presents a reflection on existing techniques for matching in a BPM context.

2.4.1 Matching Example

In this section, we discuss the goal of matching based on a process model matching scenario. Given two process models with their respective sets of activities A_1 and A_2, the goal of process model matching is to identify the activities (or sets of activities) from A_1 and A_2 that represent similar behavior. We illustrate this using the process models depicted in Figure 2.7.

This figure depicts two process models, both describing admission processes of universities, and their highlighted correspondences. Some of the correspondences in Figure 2.7 are evident. For example, both process models contain an activity with the label "*Send decision letter*," which shows that these activities represent similar process

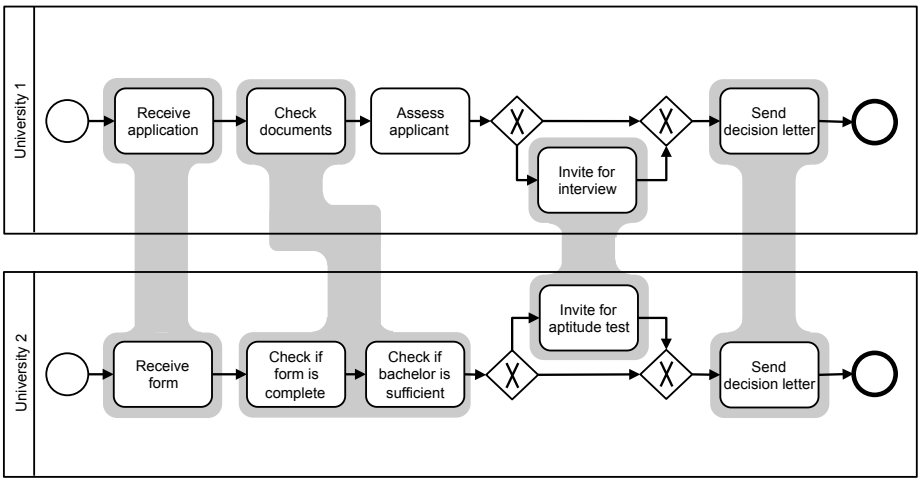


Figure 2.7: Two process models and their correspondences (adapted from [155])

steps. The correspondence between "*Receive application*" and "*Receive form*" is also clear, even though their activity labels are not completely identical. However, other correspondences are less straightforward. The activity "*Check documents*" in the model of *University 1* corresponds to two activities in the other model, "*Check if form is complete*" and "*Check if bachelor is sufficient*". This example illustrates that the cardinality of correspondences does not have to be one-to-one, but can also be one-to-many and even many-to-many. In this particular case, the one-to-many relation illustrates that the "*Check documents*" activity encompasses the "*Check if form is complete*" and "*Check if bachelor is sufficient*" activities from the second model.

Furthermore, upon closer inspection, it becomes clear that these correspondences are highly specific to the context in which they appear, i.e., in the context of university admission processes. For example, in other contexts, the correspondences between "*Check documents*", on the one hand, and "*Check if form is complete*" and "*Check if bachelor is sufficient*", on the other, are likely to not exist.

2.4.2 The Matching Task

The goal of matching is to automatically identify correspondences between concepts in two informational artifacts. We formally define this *matching task* based upon notions from [103]. Let C_1 and C_2 be two sets of concepts to be aligned, e.g., two sets of process model activities for process model matching scenarios. A matching task creates an $n \times n'$ *similarity matrix* $\mathcal{M}(C_1, C_2)$ over $C_1 \times C_2$. Each $M_{i,j}$ in the matrix represents a degree of similarity, usually a real number in $[0, 1]$, between the i-th concept in C_1 and the j-th concept in C_2. To perform the matching task, matching techniques typically perform several sequential steps, in which different classes of *matchers* are applied. An important distinction exists between so-called First-Line Matchers (FLMs) and Second-Line Matchers (SLMs) [104].

A FLM establishes a similarity matrix by analyzing the concepts contained in the artifacts C_1 and C_2. For any pair of concepts $c_1 \in C_1$ and $c_2 \in C_2$, an FLM produces a score [0,1] that quantifies the similarity between c_1 and c_2 by considering certain similarity characteristics. Most FLMs focus on the *labels* associated with c_1 and c_2, such as the *activity labels* in Figure 2.7. In the context of processes, other useful input for FLMs include behavioral and structural relations between process concepts. In contrast to FLMs, SLMs ignore any information contained in the informational artifacts. Instead, they establish a similarity matrix from an input of one or more other similarity matrices. Two commonly used categories of SLMs are ensemble matchers and decision makers. *Ensemble matchers* combine the values of multiple similarity matrices into one. For example, an ensemble matcher can be used to combine the results of an FLM that analyzes labels and another FLM that considers behavioral relations between process concepts. *Decision makers* are SLMs that take a non-binary similarity matrix (with values in the range [0,1]), as created by a FLM or an ensemble matcher, and convert it into a binary matrix (with values in $\{0, 1\}$). For example, a decision maker can be used to select only those correspondences with a similarity score above a certain threshold. The decision maker turns these scores into 1, whereas correspondences with a lower similarity score receive a 0. Decision makers are typically used to provide the final output of the matching task, i.e., to select the correspondences that are included in the alignment provide by the matching technique.

In the next section, we discuss measures used by FLMs to quantify similarity between text fragments, e.g., between activity labels. We specifically highlight these measures because they represent a crucial component of most matching techniques and because we employ them in the matching techniques presented in this thesis.

2.4.3 Similarity Measures

Textual contents are an important source of information to determine the similarity between concepts in a matching task. For example, for process model matching, similarity of activity labels is a vital component of most matching techniques. Labels that are syntactically similar (e.g., "*receive application*", "*receive online application*") or semantically similar (e.g., "*deliver products*", "*send order*") are good indicators that their corresponding activities describe similar process behavior. A plethora of *similarity measures* exist that can be used to compute the degree of similarity between two text fragments. Below, we first discuss certain *normalization* steps that can be applied on text fragments in order to improve the usefulness of such measures. Then, following Algergawy et al. [28] we separately consider the streams of syntactic and semantic similarity measures. Without a loss of generality, we illustrate the concepts and measures throughout this section with example fragments in the form of process model activity labels.

Normalization

In order to make text fragments more comparable, they are typically first normalized. This normalization phase can consist of several steps, including:

- *Tokenization*: A fragment is split into a set of tokens, each corresponding to a single word or term. Typically, tokenization can be performed by considering whitespace characters contained in the texts, e.g., "*deliver goods*" → {"*deliver*", "*goods*"}. However, in other cases different delimiters may be required for tokenization, such as punctuation or uppercase characters, e.g., "*deliverGoods*" → {"*deliver*", "*goods*"}.
- *Expansion*: Abbreviations and acronyms are expanded, e.g., "*PO*" → {"*purchase*", "*order*"}. Various approaches exist that address this task, for instance by extracting information on abbreviations from corpora of textual documents, see e.g., [200, 293].
- *Stemming*: Suffixes from words are removed in order to transform differing terms to the same form, e.g., *connected*, *connecting*, and *connection* are all stemmed to the term *connect*. The most commonly applied stemming algorithm is *Porter's algorithm* [219].
- *Lemmatization*: All terms are transformed into their grammatical base forms. Lemmatization can be performed using standard NLP tools, such as the Stanford CoreNLP [180]. An important advantage compared to stemming is that lemmatization can handle more complex transformations, such as irregular verbs. For instance, *sing*, *sang*, and *sung* are all mapped to the common lemma *sing*. A downside is that lemmatization is *taxonomy-based*; it can only be applied to terms that are included in a taxonomy or dictionary. By contrast, stemming is rule-based, which means that it can also be applied to unknown words.
- *Stop-word removal*: Common words that are of little value when considering textual similarity are removed from consideration [179]. Typical words to be removed are *closed class* terms, such as prepositions ("*the*", "*an*") and conjunctions ("*and*", "*or*").

As a result of the normalization phase, activity labels are split into sets of tokens from which information that can obscure the effective computation of similarity has been removed. For example, by combining normalization steps, given two activity labels "*Receive the purchase order*" and "*PO receipt*", we can obtain equal sets of tokens for each label: {"*receive*", "*purchase*", "*order*"}. Due to this normalization, any similarity measure will now recognize the similarity between these activities, whereas this might not have been the case for the non-normalized labels. Note that the discussed normalization steps abstract from certain details that might be relevant to the computation of similarity in particular use cases. For instance, tokenization abstracts from the order in which terms occur in a label by producing a set of tokens. While this abstraction is valuable in most use cases, such as the previous example, it is imaginable that term order is relevant in other alignment scenarios. Similarly, certain stop-words that are generally removed in normalization steps (such as "*and*" and"*or*") might be relevant to the computation of similarity in specific cases. Therefore, it is important to consider which normalization steps are suitable to the specific use case for which they are applied.

Given two sets of tokens ω_1 and ω_2 resulting from a normalization phase, the next step to compute label similarity is to determine the similarity between individual terms, i.e., $sim(t_1, t_2)$ for $t_1 \in \omega_1$ and $t_2 \in \omega_2$. For this purpose, both syntactic and semantic similarity measures can be used.

Syntactic Similarity Measures

Syntactic similarity measures compute the degree of similarity between two tokens by comparing their character sequences. A plethora of measures have been developed for this purpose, see e.g., Cohen et al. [66] and Navarro [202] for overviews. Two of the most commonly employed measures are the Levenshtein distance and the N-gram distance:

- *Levenshtein distance*: Given two tokens t_1 and t_2, the Levenshtein distance, also referred to as the *edit distance*, is given by the minimum number of single-character operations (i.e., insertion, deletion, and substitution of a character) required to transform t_1 into t_2 [291]. The following equation formalizes this:

$$sim_{edit}(t_1, t_2) = 1 - \frac{editDistance(t_1, t_2)}{\max(|t_1|, |t_2|)} \tag{2.1}$$

  In Equation 2.1, $editDistance(t_1, t_2)$ denotes the minimum cost of edit operations necessary for the transformation between t_1 and t_2. This computation can include varying costs for the different operations.

- *N-gram distance*: An *n-gram* is a slice of n characters of a longer string [70]. Typical values for n are 2, where the slices are referred to as *di-grams*, and 3, referred to as *tri-grams*. For example, the term "*order*" consists of the following bi-grams: {_o, or, rd, de, er, r_}. The n-gram distance between two tokens t_1 and t_2 quantifies the fraction of overlapping n-grams. The main benefit of n-gram-based similarity is that it is resistant to a wide variety of textual errors. This follows because the decomposition of the string means that any errors (e.g., typographical errors) affect only a limited number of the decomposed parts [57]. The n-gram distance between two tokens t_1 and t_2 is defined as:

$$sim_{n\text{-}gram}(t_1, t_2) = \frac{2 \times |n\text{-}gram(t_1) \cap n\text{-}gram(t_2)|}{|n\text{-}gram(t_1)| + |n\text{-}gram(t_2)|} \tag{2.2}$$

Syntactic similarity measures have been widely employed in various matching contexts. A clear advantage of syntactic measures such as the Levenshtein distance is that they can be applied without any external sources. This makes them language- and domain-independent [70]. Furthermore, they are able to recognize similarity in spite of typographical errors [57]. Table 2.7 illustrates this. The syntactic similarity (sim_{edit}) between *agreement* and *argeement* is high, despite the obvious error in the latter term. However, syntactic similarity measures also have their limitations. In particular, syntactic measures may assign high similar scores to terms with a highly dissimilar meaning. For example, the Levenshtein distance between "*contact*" and "*contract*" is minimal, resulting in a high similarity score. Such a similarity assessment can produce problematic results in process model matching, where the goal is to identify activities that refer to a *semantically* similar process step. Also, Table 2.7 shows that syntactic similarity measures cannot identify words with similar meanings, such as *contract* and *agreement*. Semantic similarity measures aim to overcome these issues.

Table 2.7: Comparison of syntactic and semantic similarity scores

t_1	t_2	sim_{edit}	sim_{lin}
agreement	*argeement*	0.88	n/a
contract	*contact*	0.88	0.10
contract	*agreement*	0.11	0.96

Semantic Similarity Measures

Semantic similarity measures consider word similarity based on the meaning of words, rather than their syntactic contents. For example, the words "*contract*" and "*agreement*" have a high semantic similarity, because they both describe an *exchange of promises*. Semantic similarity measures are based on NLP techniques and typically strongly rely on the use of external sources, such as text corpora or lexicons [28]. Following Agirre et al. [26], we distinguish two major streams of semantic measures: WordNet-based measures and distributional similarity measures.

WordNet-based measures, also referred to as taxonomy-based, consider similarity from the perspective of the semantic relations that can exist between words. For these measures, the most important relations to consider are synonymy, homonymy, hypernomy, and meronomy. Table 2.8 provides an overview of these. For example, the *synonymy* relation indicates that two words have the same meaning, whereas the *hyponomy* relation indicates that one word is more specific than another. Each semantic relation can be derived from a taxonomy of terms. The most commonly employed taxonomy for this purpose is based on WordNet. *WordNet* is a lexical database for the English language that organizes words into sets of synonyms, so-called *synsets* [194, 195]. From the relations that exist between these synsets, a taxonomy can be established that captures the semantic relations depicted in Table 2.8. The *Lin similarity* represents an implementation of a WordNet-based similarity measure:

Table 2.8: Overview of semantic word relations

Relation	**Description**	**Example**
Synonymy	Words have the same or similar meaning	bill & invoice
Homonymy	A word with multiple different meanings	*application* → *computer program* or *written request*
Hypernomy	Words have a hierarchical (*type-of*) relation between their meanings	*vehicle* & *car*
Meronymy	Words have a part-of relation between their meanings	*wheel* & *car*

- *Lin similarity*: Lin [169] proposes a theoretical notion of semantic similarity based on the *informational content* of the common ancestor of two terms in a

taxonomy. Lin similarity operationalizes this similarity theorem as follows:

$$sim_{Lin}(t_1, t_2) = \frac{2 \times \log P(anc(t_1, t_2))}{\log P(t_1) + \log P(t_2)} \tag{2.3}$$

In Equation 2.3, $anc(t_1, t_2)$ represents the synset that is the closest ancestor of t_1 and t_2 in the taxonomy and $\log P(t)$ denotes the occurrence probability of words in the synset containing token t. In this way, Lin similarity considers the meaning of terms, as well as their commonality.

Distributional similarity measures are based on the statistical analysis of co-occurrences of terms in large text collections. Specifically, given two terms t_1 and t_2, these measures consider how similar the words that surround t_1 and t_2 are to each other. The underlying intuition is that words with a similar meaning occur in similar contexts [193]. The great advantage of second order measures is that they do not rely on manually established lexical resources such as WordNet, which are unavailable for numerous domains and languages [145]. Due to this independence, second order similarity measures can be trained to deal with context-specific terms [128]. This is particularly relevant in the business settings where process model matching is mostly applied, because processes in these settings often use domain-specific terms.

Several methods provide distributional similarity measures, such as methods proposed by Landauer et al. [159], commonly known as *Latent Semantic Analysis*, Grefenstette [116], Dorow and Widdows [84], and Rapp [226]. Here, we describe a similarity measure proposed by Kolb [145, 146] in more detail:

- *DISCO*: The DISCO similarity measure distinguishes itself from other distributional similarity measures because it takes the distance between a word and its neighboring words into account. In particular, it computes the distribution based on weighted *word vectors*, which take into account the co-occurrence of words, as well as their relative position. Intuitively, this measure finds the words that share a maximum number of common co-occurrences. For brevity, we here omit the depiction of the extensive equations required for this computation.

In this thesis, we employ both Lin and Disco similarity to compute the semantic similarity between process concepts. Furthermore, we combine this with the use of the Levenshtein distance for terms that are not recognized by our semantic measures, e.g., *agreement—argeement*.

Token-Set Similarity

The previously considered syntactic and semantic similarity measures can be used to compute a similarity score between individual tokens. Several methods exist that use these token-similarity scores to compute a degree of similarity between two *sets* of tokens ω_1 and ω_2. One method is to take the average of the highest similarity score for each token in one set against all tokens in the other [176], as given by the following equation:

$$sim_{avg}(\omega_1, \omega_2) = \frac{\sum\limits_{t_1 \in \omega_1} [\max\limits_{t_2 \in \omega_2} sim(t_1, t_2)] + \sum\limits_{t_2 \in \omega_2} [\max\limits_{t_1 \in \omega_1} sim(t_1, t_2)]}{|\omega_1| + |\omega_2|} \tag{2.4}$$

When determining token-set similarity, it can also be useful to take the specificity of terms into account, aside from just the similarity of terms. *Term specificity* refers to the discriminatory power of terms in a given context. The underlying intuition is that terms that occur often in a particular context (e.g., in a particular business process) are less useful for the identification of similarity than relatively rare terms. To illustrate this, consider an activity labeled "*deliver order*" in an order handling process. The term *order* is most likely to occur in many activities in this process, which makes it hardly helpful when determining the similarity between activities. By contrast, the term *deliver* is likely more rare in this context, which make it a much better similarity indicator. A widely employed metric to compute term specificity is the Inverse Document Frequency (IDF), which assigns a low weight to common terms and a high weight to infrequently occurring terms. The IDF for a token t in a collection of process model activities A is given by Equation 2.5. In this equation, we use $l(a)$ to refer to the label of activity a.

$$idf(t, A) = \log \frac{|A|}{|a \in A \ : \ t \in l(a)|} \tag{2.5}$$

Mihalcea et al. [192] propose a measure that combines similarity scores with term specificity, given by Equation 2.6.

$$sim_{mih}(\omega_1, \omega_2) = \frac{\sum\limits_{t \in \omega_1} [\max\limits_{t' \in \omega_2} sim(t, t')] \times idf(t)}{2 \times \sum\limits_{t \in \omega_1} idf(t)} + \frac{\sum\limits_{t \in \omega_2} [\max\limits_{t' \in \omega_1} sim(t, t')] \times idf(t)}{2 \times \sum\limits_{t \in \omega_2} idf(t)} \tag{2.6}$$

The consideration of term specificity is highly important in process contexts, where certain terms with little discriminatory power are often repeated throughout the description of model of a process. Therefore, we employ the measure provided by Equation 2.6 for similarity computations for the computation of alignment throughout this thesis.

2.4.4 Matching in Business Process Management

Matching has received considerable attention in the context of BPM, resulting in the development of a plethora of matching techniques. The vast majority of these are process model matching techniques.

Nearly all process model matching techniques include one or more FLMs that focus on label similarity. A considerable number consider label similarity purely from a syntactic perspective, cf. [79, 172, 284]. These approaches typically use distance-based measures like the Levenshtein distance. Other matching techniques focus on the semantic similarity between activity labels, such as Lin similarity [167, 215, 249]. Although label similarity measures represent important components to most matching techniques, the dependencies that exist between process model activities also provide

useful insights for the matching task [133]. Therefore, *behavioral* or *structural* characteristics are often also taken into account by matchers. The underlying idea of such considerations is that activities which share structural properties are likely to be similar. For instance, if two activities both occur at the start of their respective process models, they are more likely to be similar than if one activity occurs at the start and one at the end of the process. Process model matching techniques such as [79, 133, 171] take such properties into account. Typically, these techniques combine structural similarity measures with measures for label similarity in order to improve the produced matching quality.

The importance of process model matching has been emphasized through the Process Model Matching Contests held in 2013 and 2015 [32, 58]. In these contests, various matching techniques were applied on the same data collections in order to compare their performance. The results show that the performance of matching techniques can be improved. For certain data collections, the best performing techniques manage to identify less than half of the actual correspondences correctly. Furthermore, it is interesting to observe that the performance of matchers varies greatly across different data collections. This indicates that the characteristics of process models have a significant effect on the performance of different matchers. Weidlich et al. [283] provide means to quantify such characteristics with the aim to predict the success of matching techniques on specific matching problems.

Throughout this thesis, we build on insights and techniques of various matching approaches. In Chapters 3 and 6 we use semantic similarity measures and other matching techniques to establish relations between process model elements and process information extracted from natural language. Chapter 4 builds on techniques that establish alignments between event classes of event logs and process model activities. Finally, Chapter 7 presents a new process model matching technique that establishes alignments by considering event-log information, rather than information contained in process models.

3

Comparing Process Models to Textual Process Descriptions

Organizational stakeholders involved in business processes have different preferences regarding the representation format used to visualize process information. As considered in Section 2.1.3, a user's cognitive style, their experience, and their purpose all influence the suitability of a particular format. As a result, some stakeholders and applications scenarios are better served by process information represented in the form of process models, whereas, in other cases, textual process descriptions are more suitable. For this reason, organizations have recognized the value of maintaining textual process descriptions alongside process models [166]. However, as shown in Section 2.1.4, the usage of two representation formats for the same process also comes with the risk of having to deal with inconsistencies between them. As a result of such inconsistencies, users may execute processes based on incorrect information. This can lead to inefficient process execution [35] as well as compliance issues [31]. Therefore, it is crucial that organizations keep their process descriptions in sync [124]. However, the effort required to identify and clear up conflicts for an entire process repository, potentially consisting of hundreds or even thousands of processes [237], is hardly manageable in a manual way. Against this background, this chapter introduces an approach that automatically detects inconsistencies between process models and textual process descriptions. In this way, the approach supports organizations by reducing the effort required to clear up any inconsistencies.

In the remainder of this chapter, Section 3.1 illustrates the problem and the challenges associated with the detection of inconsistencies between model and text. Section 3.2 presents our proposed inconsistency-detection approach. Specifically, our approach identifies two types of inconsistencies related to the control-flow of a process. First, it identifies process steps that are contained in the model but not in the textual description, i.e., *missing activities*. Second, our approach detects cases where a process model and a textual description describe process steps in a different, conflicting order. Section 3.3 demonstrates the usefulness of our approach through a quantitative evalua-

tion. The evaluation results show that our approach is indeed able to effectively identify inconsistencies in a collection of model-text pairs obtained from practice. Section 3.4 provides a reflection on the limitations of our approach and the evaluation. Section 3.5 considers streams of related research. Finally, we summarize the chapter in Section 3.6.

3.1 Problem Illustration

To illustrate the challenges that are associated with the detection of inconsistencies between textual and model-based process descriptions, consider the model-text pair shown in Figure 3.1. It includes a textual and a model-based description of a bicycle manufacturing process. The left-hand side provides a textual description of the process, which comprises eleven sentences. On the right-hand side, a corresponding model-based description can be seen, expressed using BPMN. The model contains nine activities, which are depicted using boxes with rounded edges. The diamond shapes that contain a *plus symbol* indicate concurrent streams of action; the diamond shapes containing a *cross* represent decision points, i.e., choices in the process. Lastly, the gray shades suggest correspondences between the sentences and the activities of the process model.

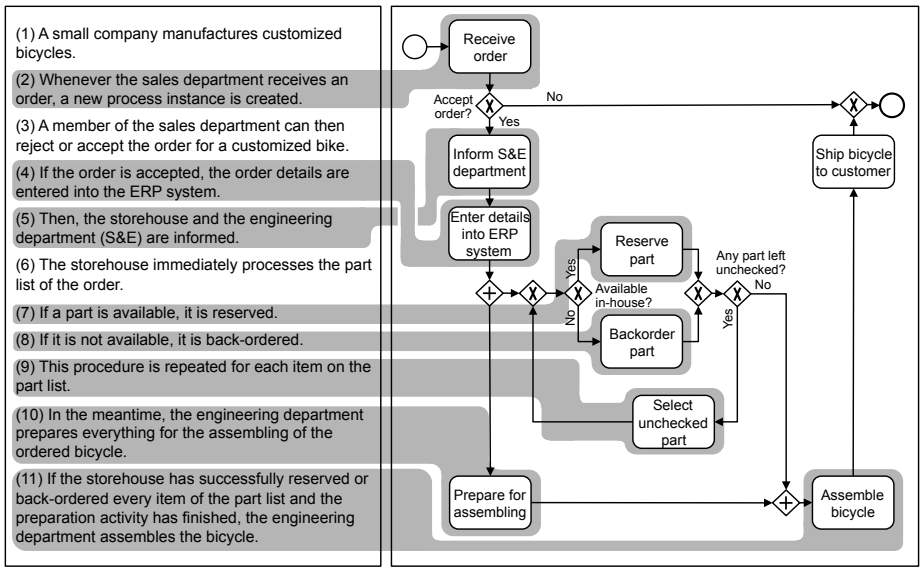


Figure 3.1: A textual and a model-based description of a bicycle manufacturing process

An in-depth look at the example reveals that many connections between the two artifacts are evident. For example, there is little doubt that sentence (7) describes the "*reserve part*" activity or that sentence (8) describes the "*back-order part*" activity. However, there are also clear inconsistencies between the two process representations. For instance, there is no sentence that relates to the "*ship bicycle to customer*" activity, i.e., that activity is missing from the textual description. Likewise, we can observe

that sentences (4) and (5) occur in a different order than their corresponding model activities. In other cases it is *less* straightforward to decide on the consistency—or lack thereof—between the representations. For example, sentence (9) indicates that a certain procedure of the process must be repeated for every part in the order. The textual description does not indicate that any action is associated with this repetition. By contrast, the process model does describe an explicit action for this repetition, in the form of the "*select unchecked part*" activity. Whether or not sentence (9) actually describes an action, and thus should be considered an inconsistency, seems to be open for debate. Nevertheless, it is important to recognize that such ambiguous cases, which are already difficult to resolve for human readers, pose even greater challenges to an automated technique.

This problem illustration shows that an appropriate inconsistency-detection technique must be able to (i) recognize structural as well behavioral process aspects in a natural language text and (ii) align this information with process model activities. In the next section, we describe an approach that achieves this by building on NLP and matching techniques.

3.2 Inconsistency-Detection Approach

Figure 3.2 depicts the main steps of our automated approach to detection inconsistencies between process models and textual descriptions. The approach takes a process model and an accompanying textual description as inputs. In the first step, the approach subjects the textual description to a linguistic analysis. This preprocessing step focuses on the extraction of core process information from the description's sentences. Next follows the creation of an alignment between process model activities and the preprocessed sentences. For this alignment task we combine semantic similarity scores with a consideration of the ordering relations that exist between the process steps in the model and text. In the third and final step, the approach predicts inconsistencies based on the established alignments. For this purpose, we introduce *predictors* that evaluate quality characteristics of the obtained correspondences. The final result of the approach is a set of predicted inconsistencies, both at a process level, as well as at a more fine-granular activity level. Sections 3.2.1 through 3.2.3 describe the individual steps of the approach in detail.

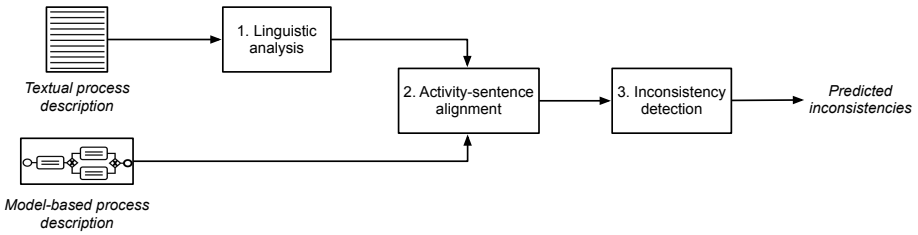


Figure 3.2: Overview of the proposed approach.

3.2.1 Linguistic Analysis

To be able to create an accurate activity-sentence alignment for a model-text pair, we first subject the textual process description to a linguistic analysis. The goal of this step is to preprocess the sentences in order to extract the most relevant process information contained in a sentence. As a result, we exclude details that may negatively affect the ability of our approach to accurately determine the similarity between a sentence and an activity. The linguistic-analysis step consists of two parts: (i) anaphora resolution and (ii) main clause extraction. We illustrate their application on sentence s_8 from the running example: s_8 ="*If it is not available, it is back-ordered*."

Anaphora Resolution

Anaphoric references, as introduced in Section 2.3.3, represent a considerable problem when analyzing textual process descriptions. These references are problematic because they obscure process information that is relevant to the establishment of alignments. As an example, consider the aforementioned sentence s_8 and the "*back-order part*" activity, referred to as a_5. The sentence and the activity both describe the act of *back-ordering* a *part*. A *part* thus represents the business object of the action. However, s_8 does not explicitly describe this. The sentence instead uses the pronoun "*it*" to refer to the term "*part*" contained in the previous sentence. The anaphoric reference thus obscures information that is important to determine the similarity between s_8 and a_5, namely that both actions are applied to the same business object. To overcome this problem, we apply an anaphora-resolution technique that resolves these backward references.

To resolve anaphoric references, our approach first identifies the objects contained in a sentence. We identify these objects by considering a number of Stanford Dependencies [74]. These dependencies, which were introduced in Section 2.3.2, denote grammatical relations between words [74]. Table 3.1 depicts the main dependencies used for the purpose of anaphora resolution. To identify the main term (i.e., the noun) of business objects in a sentence, the most important relations to consider are *direct objects* and *nominal subjects*. For instance, in sentence s_2, the relation *dobj(receives, order)* indicates that "*order*" is the object being "*received*". Nominal subjects similarly identify business objects in passive sentences, as seen in sentence s_7. There, *nsubj(reserved, part)* shows that the object "*part*" is "*reserved*". Since business objects can comprise multiple terms, e.g., "*the part list*", we also take *word specifiers* into consideration. Common specifiers include adjectival modifiers (e.g., "*small*", "*customized*"), possession modifiers (e.g., "*customer's*", "*his*"), and compounds (e.g., "*sales department*"). By considering these word specifiers, we can also extract complex business objects such as "*the customer's customized bicycle*."

Once the business objects of a sentence have been identified, we check if all objects in the sentence are anaphoric references. This is the case when all objects identified in the sentence are pronouns or determiners. For these sentences, we resolve the references by replacing the pronouns with the objects from the preceding sentence. For instance, in sentence s_8, we replace the two occurrences of "*it*" with the business object of sentence s_7: "*a part*". As a result, we resolve the anaphoric references and obtain

Table 3.1: Main Stanford Dependencies used for anaphora resolution

Purpose	**Description**	**Example**
Object identification	Direct object	*dobj(receive, goods)*
	Nominal subject	*nsubj(reserved, part)*
Word specifiers	Adjectival mod.	*amod(bicycle, customized)*
	Possessive mod.	*poss(bicycle, customer's)*
	Compound	*nn(department, sales)*

the following adapted sentence:
s'_8 = "*If a part is not available, a part is back-ordered*."

Main Clause Extraction

Sentences in a textual description describe actions that are performed in a process, the process flow, and also provide additional information. To accurately align them to process model activities, it is important to identify those parts of sentences that specifically relate to actions, while excluding unrelated information from consideration. In this context, the most problematic cases are sentences where a *conditional statement* contains terms similar to those used in activity labels. The dependent clause of sentence s_{11} provides such an example: "*If the storehouse has successfully reserved or back-ordered every item of the part list and the preparation activity has finished* [...]" This clause contains the same terms as the "*reserve part*" and "*back-order part*" activities. Still, it is clear that these activities are actually described elsewhere in the textual description. The conditional statement rather describes the requirement that these activities must have been previously completed. The inclusion of such terms in a conditional clause can indicate a strong similarity between sentence s_{11} and the "*reserve part*" and "*back-order part*" activities, which can reduce the quality of the generated alignment. Therefore, to mitigate the impact of conditional statements on the calculation of activity-sentence similarity, we extract the main clauses from such sentences.

In order to differentiate between conditional statements and main clauses, we use *parse trees* generated by the Stanford Parser [140]. Parse trees represent conditional statements as subordinate clauses, denoted as *SBAR*, starting with a specific term, such as e.g., "*if*", "*once*", or "*in case*". Figure 3.3 illustrates this for the sentence s_8. We omit these subordinate clauses from consideration by extracting the remainder of such sentences. For instance, for s_8 we exclude "*If it is not available*" and extract the main clause.

By combining anaphora resolution and main clause extraction, our linguistic analysis retains the following preprocessed sentence for the example: s'_8 = "*a part is back-ordered*." This preprocessed sentence now emphasizes the process information relevant to the determination of similarity because it includes the proper business object "*part*", rather than the pronoun "*it*". Furthermore, we omitted the precondition "*If it is not available*", which is not relevant when determining the similarity between s_8 and the

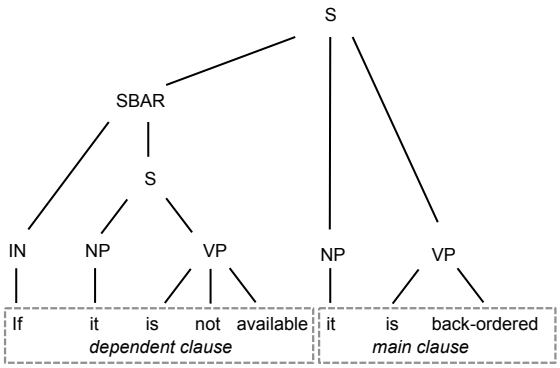


Figure 3.3: Simplified parse tree for sentence s_8.

contents of the activity labels in the process model. Therefore, the preprocessed sentences provide a suitable input for the alignment task, as described in the next section.

3.2.2 Activity-Sentence Alignment

This section describes how we obtain an optimal alignment between the activities of a process model and the sentences of a textual description. In this context, an alignment $\sim$ consists of a number of *pair-wise correspondences* between the activities from a set A and sentences from a set S. Each correspondence relates a single activity $a \in A$ to a sentence $s \in S$, denoted as $a \sim s$.

We here establish an alignment that maximizes the sum of the semantic similarity scores of its correspondences, while respecting certain ordering constraints imposed by the order of activities and sentences. Next, we describe the computation of semantic similarity and the ordering constraints in more detail.

Semantic Similarity

When establishing alignments, it is crucial to determine if a sentence s and an activity a refer to the same stream of action. To accurately assess this, we have to take the variability of natural language expressions from the sentences into account [22]. Therefore, we first *normalize* the textual contents of a sentence before determining their semantic similarity.

Normalization. We first perform a number of normalization steps in order to make an activity label and sentence more comparable. In particular, we perform tokenization, stop-word removal, and lemmatization, which are described in detail in Section 2.4.3. *Tokenization* splits a text or an activity label into a set of tokens, each corresponding to a single term, e.g., "*a part is back-ordered*" is split into the set {"*a*", "*part*", "*is*", "*back-ordered*"}. With *stop-word removal* we omit determiners, prepositions, and conjunctions from consideration, because these are of little value when determining similarity. As a result, we retain the token set: {"*part*", "*is*", "*back-ordered*"}. Fi-

nally, *lemmatization* transforms all remaining terms into their grammatical base forms. For instance, it transforms "*is*" into "*be*" and "*back-ordered*" into "*back-order*." In combination with our linguistic analysis step, our normalization step turns the original sentence "*If it is not available, it is back-ordered*", into the set of tokens ω_s = {"*part*", "*be*", "*back-order*"}.

Similarity computation. To quantify the similarity between the token sets that result from the normalization step, we use the similarity measure proposed by Mihalcea et al. [192]. This similarity measure combines *Lin similarity* and *IDF*, as previously introduced in Section 2.4.3. The use of Lin similarity allows us to identify semantically similar terms, such as "*part*" and "*component*." The IDF is used to assign a higher weight to terms that are relatively rare in the process descriptions, while it assigns a low weight to common terms. Consider, for instance, the term "*part*" in the running example. A brief look at the model and the text reveals that this term occurs in a large number of sentences and activities and, hence, has only little discriminative power. By contrast, the term "*receive*" only occurs in sentence s_2 and in activity a_1. Thus, this term is relatively likely to indicate that s_2 and a_1 describe the same process step, whereas this is not the case for the occurrence of "*part*" in, e.g., sentence s_6 and the activity "*back-order part*". Equation 3.1 provides the equation used to quantify the similarity between the token sets ω_a and ω_s, which, respectively, denote the set of tokens that stem from the normalization of an activity label and of a sentence.

$$sim(\omega_a, \omega_s) = \frac{\sum\limits_{t \in \omega_a} [\max\limits_{t' \in \omega_s} Lin(t, t')] \times idf(t)}{2 \times \sum\limits_{t \in \omega_a} idf(t)} + \frac{\sum\limits_{t \in \omega_s} [\max\limits_{t' \in \omega_a} Lin(t, t')] \times idf(t)}{2 \times \sum\limits_{t \in \omega_s} idf(t)} \tag{3.1}$$

Equation 3.1 provides a value between 0 and 1, where a higher value indicates that the (token sets of the) activity and sentence are more similar to each other.

Ordering Constraints

We impose constraints in our approach to ensure that the generated alignments respect the ordering relations included in a process model and a textual description. The consideration of these ordering constraints improves the ability of our approach to identify the correct correspondences between activities and sentences. Furthermore, they are vital to the identification of inconsistencies in the form of conflicting orders.

The order of process model activities can be extracted from the flow relation F of a process model. To represent the relations between activities, we use the *weak order relation* $\succ \subseteq A \times A$. Because the presence of loops in a process model can affect the restrictiveness of a weak order relation, we compute this relation over all valid execution sequences from Σ_{valid} that do not contain loops, i.e., those sequences that do not contain any repeated activities. Formally, we consider the following set of sequences: $\Sigma_{\text{nl}} = \{\sigma \in \Sigma_{\text{valid}} \mid 1 \leq i < j \leq |\sigma| : \sigma(i) \neq \sigma(j)\}$. Then, given two activities a_k, $a_l \in A$, a weak order relation $a_k \succ a_l$ defines that a_l cannot occur before a_k when executing the process, i.e., there is no trace in Σ_{nl} for which a_l occurs before a_k. This relation accounts for *choices* and *parallelism* that commonly occur in process models.

Extracting the inter-relations of actions in a textual process description is more complex. The inherent ambiguity of natural language makes it particularly difficult to extract the inter-relations of actions. An example of this is observed in sentence s_{10} of the running example. The phrase "*In the meantime, the engineering department prepares [...]*" can be interpreted in different ways. This results in multiple, potential points for the engineering department to start performing its tasks. Existing approaches for the analysis of textual process descriptions have been shown to provide suboptimal results when extracting such relations [29]. Therefore, we here take a conservative approach when imposing ordering restrictions. In particular, we recognize that textual descriptions generally describe actions in a chronological order [248]. For this reason, our rationale is to only allow for the generation of alignments that respect a chronological description. We capture this by introducing the strict order relation $\rightsquigarrow \subseteq S \times S$ to formally denote the order of the sentences in S. The expression $s_i \rightsquigarrow s_j$ states that sentence s_i precedes sentence s_j in the textual description.

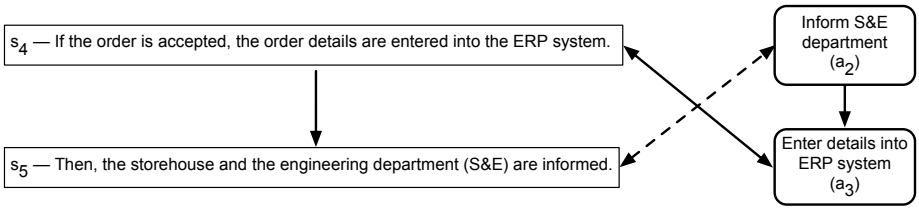


Figure 3.4: Correspondences disallowed by ordering constraints

Given the relations $\succ \subseteq A \times A$ and $\rightsquigarrow \subseteq S \times S$, we then define a constraint that avoids the creation of conflicting correspondences. Specifically, we require that two correspondences $a_k \sim s_i$ and $a_l \sim s_j$ cannot both be included in an alignment $\sim$ if $a_k \succ a_l$ and $s_j \rightsquigarrow s_i$. Intuitively, this restriction avoids the alignment of an activity a_k to a sentence s_i, if an action a_l that occurs *after* a_k is aligned to a sentence *before* s_i. Figure 3.4 depicts an example of this. Because of the ordering relations in this example, a correspondence $s_4 \sim a_3$ cannot be included in an alignment that also contains $s_5 \sim a_2$.

Finding the Optimal Alignment

Given the semantic similarity scores and the ordering constraints, our approach finds an optimal alignment $\hat{\sim} \subseteq S \times A$ which maximizes the sum of the similarity scores, while respecting the ordering constraints. Our approach establishes alignments in which every activity is aligned to exactly one sentence. This means that multiple process model activities can be aligned to the same sentence, but not the other way around.

Because of the choices and parallelism allowed by process models, finding the optimal alignment is not straightforward. Different combinations of activity execution orders must be considered as a possible solution in which the activities are contained in the textual process description. This can result in a huge number of possible alignments. Therefore, this optimization problem calls for an efficient solving approach. To find the optimal alignment $\hat{\sim}$, we adopt a *best-first search* algorithm similar to those used in machine translation problems [135]. Instead of aligning one language

to another, we here align the activities of A with sentences of S. Intuitively, the best-first search algorithm traverses a search space of partial hypotheses, which consists of activity-sentence alignments between A and S. The algorithm explores the search space by expanding the partial hypothesis with the highest possible score. Because the approach exempts unpromising hypotheses from further expansion, the explored search space is greatly reduced compared to a more naive approach. Since the algorithm merely affects computational efficiency—not the resulting optimal alignment—we abstract from further details and refer the interested reader to [135, 276] for a detailed description. We use the obtained optimal alignment $\hat{\sim}$ as input for the final step of the approach: the inconsistency-detection step.

3.2.3 Inconsistency Detection

This section describes the detection of inconsistencies through the analysis of the optimal alignments established in the previous step of our approach. For this analysis, we introduce so-called *predictors*. These predictors are metrics, designed to correlate with characteristics of the generated alignments that differ between consistent and inconsistent model-text pairs. In this section, we first introduce the general notion and desired properties of predictors for the detection of inconsistencies. Then, we describe two sets of proposed predictors: one with predictors that detect missing activities, the other to detect conflicting orders between model and text.

Detection using Predictors

Given an optimal alignment $\hat{\sim}$, we use predictors to quantify the probability that $\hat{\sim}$ contains *incorrect* correspondences. This notion of a predictor is inspired by similar ones used to analyze alignments in the context of schema and process model matching [103, 283]. The core premise underlying the predictors is that the similarity scores in the optimal alignments have different characteristics for consistent and inconsistent model-text pairs. In a consistent pair, every activity is aligned to a sentence with a high similarity score in the optimal alignment, while this is not the case for inconsistent model-text pairs.

Our proposed predictors adhere to desired structural properties of matching predictors: generalization and tunability [241]. *Generalization* refers to the applicability of predictors to tasks of different granularity levels. In the context of this work, we achieve this by defining predictors that can detect inconsistencies at two levels of granularity: the activity level and the process level. *Tunability* refers to the ability to tune predictors such that they put more emphasis on particular quality aspects of the results. The proposed predictors meet this requirement by allowing users to alternate between higher precision or higher recall.

Missing Activities

In a model-text pair where all the activities in A are also described by the set of sentences S, we expect that each activity $a \in A$ is aligned to a sentence $s \in S$ with a high similarity score $sim(a, s)$. However, if an activity a is not described in a textual

description, i.e., a is a *missing activity*, then a will not be aligned to a sentence with a high similarity score. To distinguish between *high* and *low* similarity scores, we can evaluate a similarity score according to three dimensions: (i) by its absolute value, (ii) by its value relative to the similarity scores of the activity with other sentences, i.e., a *horizontal comparison*, and (iii) by its value relative to the similarity scores of other activities, i.e., a *vertical comparison*. We introduce predictors to operationalize each of these perspectives. Each predictor presented here can be applied on an *activity level* to detect individual missing activities and on a *process level* to identify model-text pairs that contain one or more missing activities.

For our first predictor, we consider the absolute similarity score of the correspondence contained in an optimal alignment $\hat{\sim}$. A similarity score quantifies the likelihood that an activity a and sentence s describe the same part of a process. So, a higher value implies that activity a is less likely to be missing. By contrast, a low similarity score indicates that the sentence is not as similar to an activity. Therefore, such an activity is more likely to be missing from the textual description. To capture this property, we introduce a predictor *p-sim*. We apply this predictor at activity level and the process level as follows:

- *p-sim*(a): the likelihood that activity a represents a missing activity, given as the similarity score between activity a and the sentence s to which the activity is aligned in the optimal alignment, i.e., $sim(a, s)$ for $a \sim s \in \hat{\sim}$;
- *p-sim*($\hat{\sim}$): the likelihood that a model-text pair contains one or more missing activities, given by the lowest similarity score $sim(a, s)$ contained in the optimal alignment $\hat{\sim}$, i.e. $\min\{sim(a, s) \mid a \sim s \in \hat{\sim}\}$.

A second property that influences the confidence to be placed in a correspondence $a \sim s$ relates to the difference between $sim(a, s)$ and the similarity between a and other sentences in S. The similarity between an activity and a sentence is influenced by factors such as terminology and the amount of additional details that a sentence provides for a given action. Such factors can lead to considerable differences in the similarity scores between *correct* activity-sentence correspondences. To take these differences into account, we define predictors that consider a similarity score relative to other scores. The underlying notion is commonly applied in schema matching in the form of a *dominates* property (see e.g., [103, 273]).

The predictor we introduce for this purpose builds on the premise that an activity a is more likely to be described by a sentence s if $sim(a, s)$ is higher than the similarity score of a to other sentences in S. The left-hand similarity matrix in Table 3.2 illustrates this. Both activities a_2 and a_3 are aligned to sentences with a score of 0.5. For a_2 it is clear that its corresponding sentence s_2 is the most similar in the textual description. However, this clarity is missing for a_3 because two others sentences are just as similar to a_3 as its corresponding sentence s_3. This implies that s_3 is not more similar to a than these other sentences, which reduces the likelihood that the sentence actually describes the same process step as a. Thus, despite their equal similarity scores, a_3 is more likely to be a missing activity than a_2. To capture this, we define the predictor *diff-S*:

- *diff-S*(a): the difference between $sim(a, s)$ for $a \sim s \in \hat{\sim}$ and the average similarity score of a to sentences in S;

- *diff-S*($\hat{\sim}$): the lowest value *diff-S*(a) for any activity in $a \in A$, i.e., min{*diff-S*(a) | $a \in A$}.

Table 3.2: Examplary similarity matrices with correspondences in bold

	s_1	s_2	s_3	s_4
a_1	**0.6**	0.3	0.2	0.0
a_2	0.2	**0.5**	0.1	0.1
a_3	0.2	0.5	**0.5**	0.5

	s_1	s_2	s_3	s_4
a_4	**0.9**	0.3	0.2	0.0
a_5	0.2	0.4	**0.9**	0.1
a_6	0.2	0.5	**0.5**	0.5

Third, we compare the similarity score of a correspondence to the similarity scores of the other correspondences included in $\hat{\sim}$. Like *diff-S*, this property considers *sim*(a, s) relative to other scores in the similarity matrix in order to capture the differences that can exist among the similarity scores of correct correspondences. However, we here perform a vertical rather than a horizontal comparison. The premise underlying this property is that a correspondence $a \sim s$ is less likely to represent a correct correspondence if its similarity score is much lower than other similarity scores contained in the optimal alignment $\hat{\sim}$. This is illustrated in the two similarity matrices presented in Table 3.2. The correspondences in the left-hand matrix have an average similarity score of 0.53. For the right-hand matrix, this average score is 0.77. A score of 0.50 is thus close to the average score of the left-hand matrix. By contrast, this same score is much further below the average score of the right-hand matrix. Therefore, the correspondence $a_6 \sim s_3$ from the right-hand matrix is much less similar than the average similarity of the correspondences. This makes a_6 more likely to be inconsistent than a_3 from the left-hand matrix, despite their equal similarity scores. We operationalize this characteristic with the predictor *diff-A*:

- *diff-A*(a): the difference between *sim*(a, s) for $a \sim s \in \hat{\sim}$ and the average similarity scores of the correspondences in $\hat{\sim}$;
- *diff-A*($\hat{\sim}$): the lowest value *diff-A*(a) for any activity in A, i.e., min{*diff-A*(a) | $a \in A$}.

The predictors *p-sim*, *diff-S*, and *diff-A* each predict inconsistencies in the form of missing activities by quantifying a property that allows us to distinguish between high and low similarity scores.

Conflicting Orders

Inconsistencies in the form of conflicting orders occur when a process model and an accompanying textual description describe the steps of a process in different orders. For instance, the running example shows that the process model activity "*Inform S & E departments*" precedes "*Enter details into ERP system*," whereas the textual description describes these steps in the opposite order. The ordering constraints that we impose on alignments bar our approach from including correspondences between activities and sentences when their orders do not match. As a result, it cannot happen that both of these activities will be aligned to the sentences that actually describe them,

no matter how high the similarity scores are. Therefore, when conflicting orders exist between model and text, we can expect to observe large differences between the similarity scores contained in an optimal alignment and the similarity scores that would have been obtained without ordering constraints.

Table 3.3: Fragment of the similarity matrix for the running example

	s_2	s_3	s_4	s_5	s_6
a_2	0.1	**0.2**	0.2	0.7	0.2
a_3	0.1	0.1	**0.9**	0.2	0.1

To illustrate this, consider the similarity matrix in Table 3.3, where a_2 refers to "*Inform S & E departments*" and a_3 to "*Enter details into ERP system*". The similarity score $sim(a_2, s_5)$ is high (0.7), because both a_2 and s_4 describe the same process step. The similarity score for $sim(a_3, s_4)$ is also high (0.9), for the same reason. However, since a_2 precedes a_3 in the model ($a_2 \leq a_3$) and s_4 precedes s_5 in the textual description ($s_4 < s_5$), $a_2 \sim s_5$ and $a_3 \sim s_4$ cannot both be contained in $\hat{\sim}$ without violating the imposed constraints. Therefore, the approach can only align a_2 to s_3, despite its lower similarity score. As a result, the similarity score of the correspondence involving a_3 is much lower (0.2) than it would have been if ordering constraints had not been imposed (0.7). This difference of 0.5 in the similarity score for a single activity is an important indicator that there are ordering conflicts in the model-text pair.

We capture this difference between consistent and inconsistent model-text pairs in the *max-constrained* predictor. This predictor quantifies the maximum difference that exists between the aligned and potential score for a single activity in a model-text pair. In this way, it captures the largest similarity difference caused by the ordering restrictions. We operationalize this as follows:

- *max-constrained*($\hat{\sim}$): the maximal difference between the potential and aligned similarity scores for an activity $a \in A$.

The predictors defined for the detection of missing activities and conflicting orders each quantify conceptual notions that differ between the similarity scores of consistent versus inconsistent model-text pairs. In Section 3.3, we present a quantitative evaluation that demonstrates how well our approach is able to detect inconsistencies in practice.

3.3 Evaluation

This section presents a quantitative evaluation that demonstrates the ability of our approach to identify inconsistencies in model-text pairs. We made the prototypical implementation used for this evaluation publicly available.[1]

We manually annotated the inconsistencies in a collection of 53 model-text pairs obtained from practice. This annotation is referred to as the *gold standard* against

[1] Download from: www.hanvanderaa.com/downloads/modeltextcomparison

which we compare the results of our approach. Section 3.3.1 describes our test collection in detail. Next, we describe the setup of our evaluation in Section 3.3.2. Afterwards, we present the evaluation results in Section 3.3.3 and discuss the strengths and weaknesses of our approach in 3.3.4.

3.3.1 Test Collection

To evaluate our approach, we use a collection of 53 model-text pairs that originate from different sources, including academia, textbooks, industry, and public sector organizations. The majority of the model-text pairs was introduced in [99] to evaluate an approach for the generation of process models from textual process descriptions. We extended this collection with 7 processes obtained from a new industrial source. Table 3.4 presents the main characteristics of the model-text pairs contained in the test collection. The process models are heterogeneous with regard to several dimensions, such as size and complexity. Also, the corresponding textual descriptions vary in several aspects. For instance, they describe the processes from different perspectives (first and third person) and differ in terms of how explicitly and unambiguously they refer to the process model content. Finally, it is important to note that the ratio of sentences per activity (S/A) differs considerably throughout the collection. Some sources use a single sentence on average per activity, while other sources use up to 7 sentences. This increased ratio follows from the presence of more informational sentences and details on how certain process steps must be executed. Due to these varying characteristics, we believe that the collection is well-suited to achieve a high external validity of the results.

Table 3.4: Overview of the test collection

ID	Source	Type	MT	MT_m	A_m	MT_o	A	S	S/A
1	HU Berlin	Academic	4	1	1	0	9.0	10.3	1.1
2	TU Berlin	Academic	2	2	2	0	22.5	34.0	1.5
3	QUT	Academic	8	0	0	0	6.1	7.1	1.2
4	TU Eindhoven	Academic	1	1	2	0	18.0	40.0	2.2
5	VU Amsterdam	Industry	7	1	2	0	4.3	30.1	7.0
6	Vendor Tutorials	Industry	3	2	3	2	5.3	7.0	1.3
7	inubit AG	Industry	4	1	1	1	9.0	11.5	1.3
8	BPM Practitioners	Industry	1	0	0	0	4.0	7.0	1.8
9	BPMN Practice Handbook	Textbook	3	2	2	1	5.0	4.7	0.9
10	BPMN Guide	Textbook	6	5	7	0	7.0	7.0	2.2
11	Federal Network Agency	Public Sector	14	4	6	0	8.0	6.4	0.8
	Total	–	**53**	**19**	**26**	**4**	**7.6**	**12.0**	**1.6**

Legend: MT = Model-text pairs, MT_m = Model-text pairs with missing activities, A_m = Total missing activities, MT_o = Model-text pairs with conflicting orders, A = Activities per model (avg.), S = Sentences per text (avg.), S/A = Sentences per activity (avg.)

3.3.2 Setup

To conduct the evaluation, we implemented the presented inconsistency checking approach in the form of a Java prototype. The prototype uses the Stanford CoreNLP toolkit [180] for the tokenization of PPI descriptions and WS4J for the WordNet-based semantic similarity computation.[2]

To provide a basis for comparison of our automated approach, we manually identified inconsistencies in the test collection. We involved three researchers in the creation of this *gold standard*. Two researchers independently identified inconsistencies for each model-text pair. The inter-annotator agreement was high, having only 4 initial disagreements on whether or not an activity was missing. The cause for discussion involved implicitly described actions, such as seen for the "*select unchecked part*" activity in the bicycle manufacturing example. The differences were resolved in a discussion, involving the third researcher to settle ties. Out of the 406 activities contained in the process models, 26 are considered to be missing in the textual description. These activities occur in 19 different model-text pairs. Furthermore, 4 model-text pairs were found to contain conflicting orders.

In our evaluation, we compare the performance of three predictors for the detection of missing activities, *p-sim*, *diff-S*, and *diff-A*. We apply these predictors at the activity and process levels. For the detection of model-text pairs with conflicting orders, we evaluate the performance of the *max-constrained* predictor. For this part of the evaluation, we introduce a baseline against which we compare the performance of the predictor. For this baseline, we first create an alignment without imposing ordering constraints. Afterwards, we check if this alignment violates any ordering constraints. We refer to the heuristic that identifies these cases as *base-check*.

Aside from the choice for a certain predictor, the performance of the approach strongly depends on the quality of the generated alignments. Our approach uses several components to establish these alignments, most importantly the linguistic-analysis step and the consideration of ordering constraints. To demonstrate the added value of these individual components, we test the performance of our approach using different configurations to generate the alignments. In particular, we compare the following four configurations:

- **Baseline (BL):** As a baseline configuration, we aligne every activity a to the sentence s with the highest value for $sim(a, s)$;

- **Linguistic analysis (LA):** For this configuration, prior to the computation of similarity scores, we apply the linguistic analysis described in Section 3.2.1. We thus resolve anaphoric references and extract relevant clauses for the sentences in the textual description;

- **Ordering constraints (OC):** This configuration computes an alignment between activity set A and sentence set S that achieves a maximal similarity score, while respecting the ordering constraints described in Section 3.2.2;

[2] See: https://github.com/Sciss/ws4j

- **Linguistic analysis + ordering constraints (LA + OC):** This configuration applies the full alignment approach. We perform the linguistic analysis step *and* impose ordering constraints on the establishment of the optimal alignment.

We quantify the performance of our approach with standard information retrieval metrics. More specifically, we calculate precision, recall, and F_1-score by comparing the generated results against the manually created gold standard. *Precision* describes the fraction of predictions that are correct. *Recall* represents the fraction of all inconsistencies that are identified by our approach. We define these metrics in the context of this work as given by Equations 3.2 and 3.3.

$$pre = \frac{|E_I \cap E_\rho|}{|E_\rho|} \quad (3.2) \qquad rec = \frac{|E_I \cap E_\rho|}{|E_I|} \quad (3.3) \qquad F_1 = \frac{2 * pre * rec}{pre + rec} \quad (3.4)$$

Here, we use E_ρ to denote the entities, i.e., the set of model-text pairs or activities, that are predicted to be inconsistent with a prediction score in the range $[0, \rho]$. A higher value for ρ thus increases the number of entities that are included in E_ρ, i.e., that are predicted to be inconsistent. E_I denotes the set of model-text pairs or activities that contain a certain type of inconsistency. Finally, we also report the F_1-*score*, which provides the harmonic mean between precision and recall, as formalized by Equation 3.4.

3.3.3 Results

This section presents the results of the quantitative evaluation. We first asses the ability of the predictors to detect inconsistencies. Then, we consider how the individual components of the alignment approach contribute to the quality of the obtained results.

Predictor Performance

We computed precision and recall scores for increased values of the predictor score ρ for each of the predictors. The precision-recall graphs of Figures 3.5 and 3.6, respectively, depict the performance of the approach for the detection of missing activities at the activity level and the process level. Both graphs illustrate a trade-off between precision and recall. The reason for a trade-off to exist is that increasing the recall requires the consideration of lower confidence values. As a result, some activities are erroneously classified as missing, which negatively affects precision. The maximum results achieved by the predictors are presented in Table 3.5. This table also displays the predictor values that achieve the maximum F_1-score, which we here refer to as the optimal predictor values.

Missing Activities. The precision-recall graph of Figure 3.5 shows that our approach is able to detect missing activities with a good accuracy. The approach reaches a maximum F_1-score of 0.44 for the *p-sim* predictor. At this point, 17 missing activities have been detected (recall = 0.58) with a precision of 0.36. These results should be considered in light of the relatively low fraction of the total activities that are missing (0.07). The graph shows that beyond a recall of 0.15, *p-sim* consistently outperforms the other

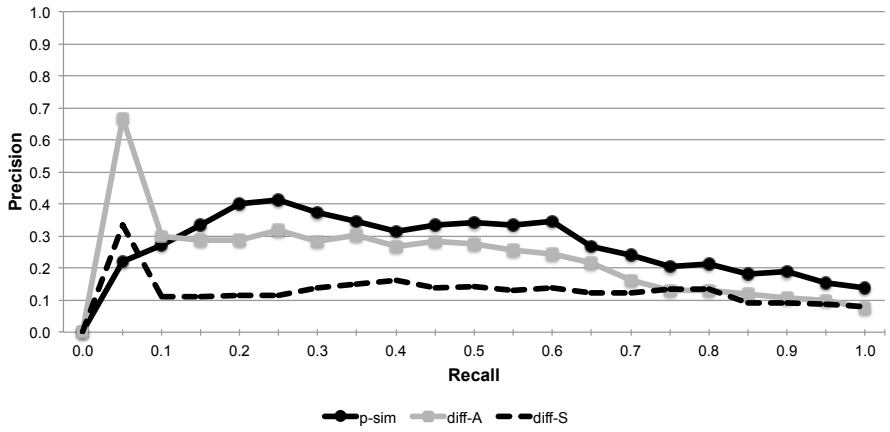


Figure 3.5: Precision-recall graph for the detection of missing activities (activity level)

predictors. The *diff-A* predictor reaches a maximum precision of 0.67 and a maximum F_1-score of 0.37. The predictor *diff-S* has the lowest performance, with a maximum precision of 0.33 and F_1-score of 0.21.

The precision-recall graph of Figure 3.6 shows that the approach performs even better at the process level, particularly using the *p-sim* predictor. This predictor correctly identifies more than 40% of the model-text pairs with missing activities with perfect precision. For increasing recall values, the approach maintains a high precision. The highest obtained F_1-score of 0.83 is reached when all incorrect model-text pairs have been detected (i.e., recall = 1.00), with a precision of 0.70. The *diff-A* and *diff-S* pre-

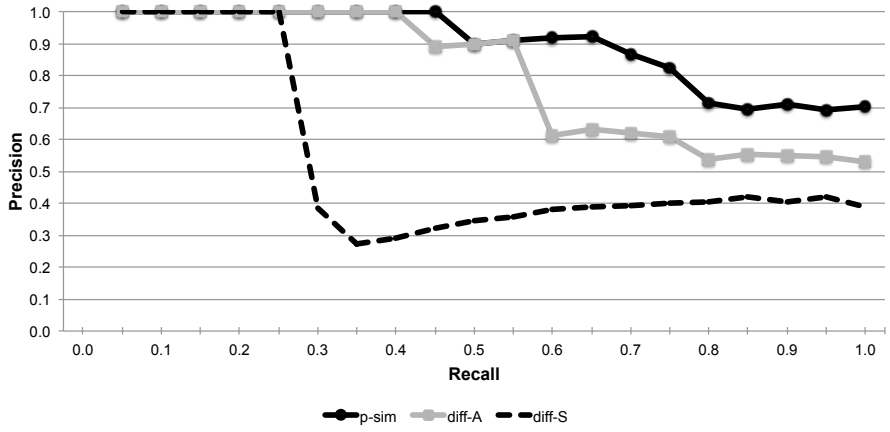


Figure 3.6: Precision-recall graph for the detection of model-text pairs with missing activities (process level)

Table 3.5: Predictor performance evaluation results

Predictor	Maximum precision	Maximum F_1-score	Optimal predictor value
p-sim(a)	0.39	0.44	0.74
diff-A(a)	0.67	0.37	1.18
diff-S(a)	0.33	0.21	0.92
p-sim($\hat{\sim}$)	1.00	0.83	0.74
diff-A($\hat{\sim}$)	1.00	0.69	0.52
diff-S($\hat{\sim}$)	1.00	0.58	0.47
base-check($\sim$)	0.16	0.28	–
max-constrained($\hat{\sim}$)	1.00	0.73	0.18

dictors display lower prediction accuracy, they reach maximal F_1-scores of 0.69 and 0.58, respectively.

Conflicting Orders. Lastly, we evaluated the usefulness of the *max-constrained* metric for the detection of model-text pairs with conflicting orders. We compare its performance to the baseline metric *base-check*. Note that we here omit the depiction of a precision-recall graph because there are only four cases with inconsistent orders in the test collection. The *max-constrained* predictor identifies these four cases with high accuracy. The predictor reaches a maximum precision of 1.00 and successfully identifies all inconsistent model-text pairs with a precision score of 0.57. This results in a maximum F_1-score of 0.73. Furthermore, we can observe that the *max-constrained* predictor greatly outperforms the baseline metric. The reason for this is that *base-check* yields a considerable amount of false positives, resulting in a precision of 0.16 and a maximum F_1-score of 0.28.

Alignment Configurations

The presented results demonstrate that our approach is able to detect inconsistencies with high accuracy by using the full configuration for the creation of alignments. Table 3.6 presents the performance of the approach for different configurations of the alignment approach. For the sake of brevity, we depict the highest F_1-score achieved by each predictor for a given configuration.[3]

The results clearly illustrate that each of the individual components of the alignment approach contribute positively to the prediction accuracy, since the scores of LA and OC are higher than the baseline scores reached by BL. The linguistic analysis improves the performance for all use cases and predictors. Notably, it increases the maximum performance for the detection of missing activities with 45%, from 0.22 to 0.32. The performance for the detection of model-text pairs with missing activities is similarly

[3]Note that the BL and LA configurations cannot detect inconsistent orders, since these configurations do not incorporate the ordering restrictions required to identify these.

Table 3.6: Highest F_1-measures for different configurations and predictors.

Predictor	**BL**	**LA**	**OC**	**LA + OC**
p-sim(a)	0.22	0.32	0.23	0.44
diff-A(a)	0.20	0.29	0.21	0.35
diff-S(a)	0.15	0.18	0.15	0.23
p-sim($\hat{\sim}$)	0.62	0.72	0.66	0.83
diff-A($\hat{\sim}$)	0.61	0.64	0.61	0.69
diff-S($\hat{\sim}$)	0.54	0.58	0.55	0.58
max-constrained($\hat{\sim}$)	–	–	0.50	0.73

increased from a maximum of 0.63 to 0.72. Though the inclusion of ordering constraints consistently improves the results in comparison to the baseline configuration, it does not yield gains as large as the linguistic analysis. For instance, the detection of model-text pairs with missing activities is improved from 0.63 to 0.66. However, the usefulness of the constraints becomes particularly apparent when they are used in combination with the linguistic analysis (LA +OC). Then, the results are improved from 0.32 to 0.44 and from 0.72 to 0.83 for the detection of missing activities at the activity level and process level, respectively.

3.3.4 Discussion

The evaluation results show that our approach successfully identifies inconsistencies between model-text pairs. Especially at the process level, the approach detects erroneous model-text pairs with a high prediction accuracy. The lower performance scores for inconsistency detection at the activity level can be attributed to two causes. First, it is inherently easier to detect that inconsistencies exist in a model-text pair than to identify exactly *where* these occur. Second, the difference occurs because the approach creates an alignment optimized for the entire process. Therefore, for inconsistent model-text pairs, it can happen that a missing activity also impacts the alignments of other activities in the process. As a result of such a shift, sometimes the wrong activities are predicted to be missing. This leads to lower precision at the activity level, while the prediction accuracy at the process level remains unaffected.

The results for the different predictors demonstrate that predictors considering the absolute similarity values (*p-sim*) outperform others at the activity and process levels. This implies that the absolute similarity value of an activity-sentence pair has the strongest correlation with inconsistencies from the considered factors. However, the performance of the other predictors indicates that the consideration of relative similarity also has its merits. This applies in particular to the *diff-A* metric, which compares an activity's similarity score to the other scores included in an optimal alignment. This predictor also achieves a high predictive accuracy.

The performance of the *max-constrained* predictor shows that the approach can accurately identify model-text pairs with conflicting orders between the process steps de-

scribed in the process model and those in the textual description. The *max-constrained* predictor was, furthermore, shown to greatly outperform the *base-check* baseline. The *base-check* predicts a large number of false positives, because it does not distinguish between small and large differences in similarity scores. The underlying cause is that, for various cases in the test collection, not all activities should be aligned to the sentence with the highest semantic similarity score. Instead, such activities are generally aligned to sentences with a slightly lower score (e.g., a score of 0.55 versus a score of 0.60). The *base-check* is not able to differentiate such cases from cases that truly contain conflicting orders. However, because true conflicting orders are characterized by much larger differences in similarity scores, the *max-constrained* predictor does successfully make this differentiation.

The obtained evaluation results have several implications for the application of the proposed approach in practice. The results reveal that both the linguistic analysis and ordering constraints consistently improve the accuracy with which inconsistencies are detected. This implies that the full configuration (BL + LA) should be used to generate alignments between model and text. The comparison of the different predictors furthermore demonstrates that the *p-sim* and *max-constrained* predictors achieve the best results on our heterogeneous data collection. As such, the evaluation shows that these predictors should be selected for application in practical settings. Finally, it is important to note the trade-off between activity-level and process-level predictors for the detection of missing activities. The former provide more fine-granular results, while the latter predictors achieve a higher predictive accuracy. Since clearing up inconsistencies will always involve manual effort, it is, therefore, worthwhile to consider the usage of process-level predictors to identify those model-text pairs in a collection that are most likely to contain inconsistencies.

3.4 Limitations

The evaluation demonstrates that our inconsistency-detection approach achieves promising results. However, these results need to be considered against the background of some limitations. In particular, we identify limitations related to the approach and limitations related to the evaluation.

The main limitation related to the approach is that the alignments we establish are not always fully correct. Three aspects of the approach play a role here. First, the employed NLP techniques, such as the Stanford Parser and the anaphora resolution technique, are not fully accurate. This means that the linguistic analysis step does not always yield the results that are necessary to properly compute the similarity between a sentence and an activity. Second, the employed semantic similarity metric is not always able to identify semantic relationships between words that are only semantically similar in a particular context. For instance, given the context of a loan application process, our approach is likely to not identify a relation between a "*check applicant's history*" activity and a sentence that describes this as "*determine if the applicant has had payment issues*." This problem occurs because the link between "*history*" and "*payment issues*" only exists in particular contexts. Third, our approach does not take discourse statements such as "*after*" or "*before*" into account. Instead, we use a con-

servative approach that only considers the order in the text, because existing techniques that analyze discourse statements have been shown to produce inaccuracies [250]. As a result of these limitations, our approach may identify wrong alignments. Therefore, the proposed approach remains a prediction approach that is intended as a means to support users. In this way, our approach greatly helps organizations to quickly detect inconsistencies between textual and model-based descriptions of their processes.

As for limitations related to the evaluation, we would like to point out that the presented quantitative results are bound to the specifics of the model-text pairs used in the evaluation. Still, we tried to compose a data set that is as heterogeneous as possible by collecting model-text pairs from a broad variety of external sources. This set included only four model-text pairs with conflicting orders. This should be taken into account when interpreting the results related to this type of inconsistency. Inconsistencies in the form of missing activities were better represented in the data collection, with 26 occurrences. Thus, for this type of inconsistency we are confident that our evaluation indeed shows a realistic picture of the performance of our approach in practice.

3.5 Related Work

The work presented in this chapter relates to two main streams of research: matching and transformations between conceptual models and natural language texts.

With respect to matching, Section 2.4 presents an extensive overview of this task and the numerous techniques that address it. In the context of the work presented in this chapter, it is important to recognize that none of those existing techniques focus on the alignment between process models and textual process descriptions. Therefore, existing approaches cannot be applied to detect inconsistencies between process models and textual process descriptions. Work by Sanchez et al. [245] forms an exception to this, although it should be noted that their approach represents a follow-up to ours.As an interesting extension, the approach from [245] also considers the resource and data perspectives when establishing alignments between model and text.

Research on the *transformation between conceptual models and natural language texts* is concerned with transforming a given representation into the other. Prior work has addressed both directions, model-to-text as well as text-to-model. Transformations of conceptual models into textual descriptions typically aim at making the information captured by the model available to a wider audience. Among others, such techniques have been defined for UML diagrams [188], object models [161], and process models [166]. Techniques transforming text into models usually aim at providing support for model creation. Text-to-model transformation techniques also cover a variety of conceptual models, including UML class diagrams [42] and entity-relationship models [110]. Several approaches specifically focus on the elicitation of process models from different types of text documents, such as group stories [111], use case specifications [253], and process descriptions [99].

Despite the empirically demonstrated usefulness of the discussed transformation techniques, they do not help to address the challenges associated with the detection of inconsistencies between process models and texts. Techniques for transforming models into texts focus on verbalizing the information from the model. Hence, they do not

provide any means to align the information from a given model-text pair. Techniques for transforming process descriptions into models do address the problem of inferring structural and behavioral process information from a natural language text. However, these approaches have been found to produce incomplete or inaccurate models that require extensive manual revision [250]. This makes them unsuitable to support the automatic detection of inconsistencies between textual and model-based process descriptions.

3.6 Summary

In this chapter, we presented an approach to automatically detect inconsistencies between textual and model-based process descriptions. Our approach first combines linguistic analysis, semantic similarity measures, and the consideration of ordering relations to obtain an alignment between the activities of a process model and the sentences of a textual process description. Afterwards, the approach analyzes the established alignments in order to detect process model activities that are missing from the associated textual process descriptions, as well as ordering conflicts that exists between the two representation formats. This analysis is performed by using a set of predictors that quantify particular characteristics that differ between the alignments of consistent and inconsistent model-text pairs. A quantitative evaluation with a collection of 53 real-world processes shows that our approach can indeed successfully identify inconsistencies in model-text pairs. We observed that our approach identifies all model-text pairs with conflicting orders with a precision of 0.57. This means that users only have to investigate 7 of the 53 model-text pairs in order to resolve all consistencies of this kind. For the 19 textual processes with missing activities, our approach achieves even higher levels of accuracy. The approach identifies all instances with a precision of 0.70. Since these results considerably exceed the baseline results against which we compared our approach, our evaluation demonstrates that our tailored NLP techniques as well as ordering restrictions positively contribute to the predictive accuracy of the approach. Therefore, our proposed inconsistency-detection approach can be used by organizations to quickly identify inconsistencies in their process repositories. As a result, our approach supports organizations to maintain and improve the quality of their process documentation in an efficient manner.

4

Conformance Checking based on Uncertain Event-Activity Mappings

Conformance-checking techniques enable organizations to automatically determinek whether business processes are executed according to their specifications. Particularly, they check if observed behavior, as recorded in an IT system and represented in the form of *event logs*, conforms to the allowed process behavior, typically captured in a *process model* [24]. The importance of conformance checking has been recognized in various contexts, such as legal compliance [239] and auditing [21]. Due to this importance, numerous conformance-checking techniques have been developed (cf. [15, 24, 199, 225]). A crucial requirement for all these techniques is that the events contained in an event log can be related to the activities of a process model [14]. Without knowing the relations between events and model activities, it is not possible to determine if the behavior within a trace conforms to the behavior specified by a process model. Despite this dependence on the existence of a, so-called, *event-to-activity mapping*, establishing these mappings is a highly complex task. In particular, mapping techniques face considerable challenges caused by, among others, cryptic event names, non-conforming behavior, and noise [37]. As a result, automated mapping techniques often cannot provide a certain solution to the mapping problem. Due to this uncertainty, the goal of mapping techniques becomes choosing the *best* mapping from a number of potential ones [175]. Hence, there is always the risk that the selected mapping is wrong, i.e., that the selected mapping does not correctly capture the relations between traces and a process model. In the context of conformance checking, selecting an incorrect mapping is particularly harmful. If the selected mapping is incorrect, the results obtained through conformance checking can become incorrect as well. Therefore, conformance-checking results based on a selected, potential mapping cannot be trusted. To overcome this issue, this chapter presents a conformance-checking technique that can be applied in spite of an uncertain mapping of events onto activities. Our technique assesses the conformance of a trace by considering the entire spectrum of potential mappings, rather than focusing on a single one. At the basis of our tech-

nique stands the novel concept of a *behavioral space* as a means to capture multiple interpretations of uncertain process behavior in a structured manner. By employing behavioral spaces, our conformance-checking technique avoids the risks associated with the selection of an incorrect mapping.

In the remainder of this chapter, Section 4.1 motivates the problem of conformance checking in the context of uncertain event-to-activity mappings. Section 4.2 describes our conformance-checking technique. Section 4.3 presents a quantitative evaluation of the usefulness of our technique. Section 4.4 discusses the limitations of the technique and its evaluation. Section 4.5 considers streams of related research. Finally, Section 4.6 summarizes the chapter.

4.1 Problem Illustration

In this section, we illustrate the problem of conformance checking in the context of mapping uncertainty. In this setting, the goal of conformance checking is to determine if behavior captured in event log *traces* is allowed by the behavior specified in the form of a *process model*. A trace captures an execution sequence of events. These events correspond to the *actual* behavior of a process, because they are extracted from information systems that record the execution of process steps. By contrast, process models are used in conformance-checking scenarios to specify the *allowed* behavior of a process. A crucial prerequisite for conformance checking is that the events in traces can be related to the activities of a process model. For example, given a trace $\sigma = \langle e_1, e_2, e_3, e_4, e_5 \rangle$ and the process model M depicted in Figure 4.1, the events in σ must be mapped to activities in model M. Otherwise, it is impossible to understand which activities have occurred in reality and, thus, whether or not σ conforms to M.

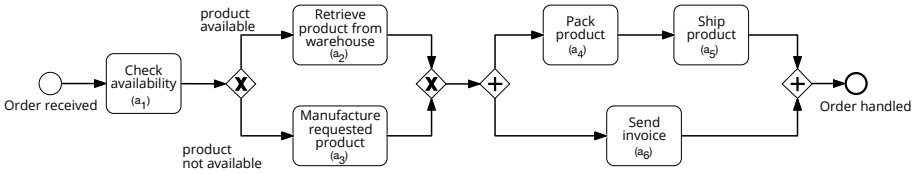


Figure 4.1: Process model for a simplified order handling process

Unfortunately, establishing a correct mapping between events and activities has proven to be a considerable challenge. Existing techniques addressing this task can at best indicate potential mappings and their likelihoods, instead of providing a definite solution [40, 41]. The reason why mapping techniques fail to provide definite solutions is that the information they can take into account when constructing mappings often does not suffice to identify relations with certainty. As an example, consider an event with the label "*product obtained*". By considering this label, it is not possible to determine with certainty whether this event corresponds to activity a_2 ("*retrieve product from warehouse*") or to activity a_3 ("*manufacture requested product*"). Both of these activities *obtain* a product, but in a different way. Even more problematic are the

commonly observed event labels with cryptic database field names such as *CDHDR* or *I_SM_E* [41]. In these cases, not even advanced linguistic analysis tools are able to identify reliable mappings.

The inability of techniques to reliably establish event-to-activity mappings leads to mapping uncertainty. As a result, mapping techniques generally construct a number of *potential* mappings without being able to determine with certainty which mapping is correct. Since existing conformance-checking techniques require a single event-to-activity mapping, mostly the mapping with the highest likelihood is selected as a basis for conformance checks. However, there is always the risk that this selected mapping is incorrect and that, consequently, conformance-checking results based on the selected mapping are incorrect as well.

The main issue of mapping uncertainty is that existing conformance-checking techniques can only be applied on a single, certain event-to-activity mapping. Therefore, in case of mapping uncertainty, these techniques require the selection of a single mapping from the set of potential mappings. However, this comes with the considerable risk that the selected mapping is incorrect and that, consequently, conformance-checking results based on the selected mapping are incorrect as well. To illustrate the risks of selecting a single, possibly incorrect, mapping, let $\sim_1$ and $\sim_2$ be two equally likely mappings between the trace σ and the process model M. Further assume that the mapping $\sim_1$ corresponds to the activity sequence $\pi_1(\sigma) = \langle a_1, a_2, a_4, a_5, a_6 \rangle$ and the mapping $\sim_2$ to $\pi_2(\sigma_2) = \langle a_1, a_2, a_3, a_5, a_6 \rangle$. The activity sequence σ_1 conforms to model M, whereas σ_2 does not, because it executes both activities a_2 and a_3, while it also skips execution of the mandatory activity a_4. Because the conformance of these two activity sequences differs for this scenario, the assessment of whether or not σ conforms to the process model M depends on the selection of a mapping relation. If $\sim_1$ is chosen, conformance-checking techniques will determine that σ conforms to M, whereas the opposite holds if $\sim_2$ is selected. Therefore, the conformance of σ fully depends on the ability to select a correct mapping in a situation where it is inherently uncertain what that mapping is.

The previous example illustrates that conformance-checking results based on the selection of a single, potentially incorrect mapping are not trustworthy. To provide a comprehensive solution to this problem, this chapter introduces a conformance-checking technique that takes the entire set of potential mappings into account. Therefore, we eliminate the need to select a single, possibly incorrect mapping.

4.2 Conformance-Checking Technique

This section describes the conceptual basis of our conformance-checking technique. It takes as input a trace, a process model, and an uncertain event-to-activity mapping. Note that the question of how to obtain an uncertain mapping, which consists of a number of potential event-to-activity mappings, is not the focus of this chapter. Potential mappings can be obtained using one or more mapping techniques, such as [38,40,251]. In the remainder, Section 4.2.1 first describes the notion of a behavioral space, which we use to capture the impact of mapping uncertainty on the process behavior described by trace σ. Then, Section 4.2.2 introduces the conformance-checking metrics that build on the obtained behavioral spaces.

4.2.1 Capturing Mapping Uncertainty using Behavioral Spaces

Mapping uncertainty results from multiple views on which behavior, in terms of process model activities, is described by a single trace. This uncertainty manifests itself through the existence of multiple possible event-to-activity mappings. A single event-to-activity mapping captures relations between events in a trace σ and the activities in a process model M, as defined in Definition 4.1.

Definition 4.1 (Event-to-Activity Mapping)**.** *Let $\sigma = \langle e_1, \ldots, e_n \rangle$ be a trace with a set of events E_σ and M a process model with an activity set A. An event-to-activity mapping is a surjective relation $\sim \subset E_\sigma \times A$. Elements of the relation are referred to as correspondences, where a correspondence $e \sim a \in (E_\sigma \times A)$ denotes a mapping relation between an event e and an activity a.*

In Definition 4.1, the relation $\sim$ is defined as *surjective*, because we assume that each event in a trace σ is always mapped to an activity. Furthermore, this implies that a 1:N relation can exist between events and activities. This cardinality captures the notion that events are typically more fine granular than activities [278]. As an illustration, consider a trace $\sigma = \langle e_1, e_2, e_3, e_4, e_5, e_6 \rangle$ and a mapping $\{e_1 \sim a_1, e_2 \sim a_2, e_3 \sim a_2, e_4 \sim a_4, e_5 \sim a_5, e_6 \sim a_6\}$. This mapping indicates that the trace σ, consisting of six events, corresponds to a sequence of five activities: $\langle a_1, a_2, a_4, a_5, a_6 \rangle$.

Mapping uncertainty leads to the existence of multiple potential event-to-activity mappings. Here, we capture this spectrum in the form of an *uncertain event-to-activity mapping* $\mathbb{EA}(\sigma, M)$, as defined in Definition 4.2.

Definition 4.2 (Uncertain Event-to-Activity Mapping)**.** *Let $\sigma = \langle e_1, \ldots, e_n \rangle$ be a trace and M a process model with an activity set A. An uncertain event-to-activity mapping is a tuple $\mathbb{EA}(\sigma, M) = (\mathbb{M}, \phi)$, with:*

- *$\mathbb{M}$: a set of event-to-activity mappings between σ and M;*
- *$\phi : \mathbb{M} \rightarrow [0, 1]$: a function that assigns a probability to each event-to-activity mapping $\mathcal{M}(\sigma, M) \in \mathbb{M}$, such that the sum of the probabilities for all mappings in $\mathbb{M}$ is equal to 1.*

In this definition, each mapping $\mathcal{M}(\sigma, M) \in \mathbb{M}$ represents a potential way to map the events in σ to the activities in A. The probability function ϕ assigns a probability p_i to each mapping $\mathcal{M}(\sigma, M) \in \mathbb{M}$. These probabilities generally follow from the confidence of an event-to-activity mapping technique. For instance, a technique based on semantic similarity scores, such as [40], can quantify the probability as the product of the similarity scores associated with each correspondence in the mapping. In this way, mappings with a higher semantic similarity receive a higher probability than the ones with a lower score. If no probabilities are available, the most straightforward solution is to assign an equal probability $p = 1 \,/\, |\mathbb{M}|$ to each mapping.

Given such an uncertain event-to-activity mapping $\mathbb{EA}(\sigma, M)$, we define the notion of a probabilistic behavioral space as a means to capture all process model behavior conveyed by trace σ according to the mapping $\mathbb{EA}(\sigma, M)$, i.e., the sequences of process model activities that follow from the different possible mappings. We shall refer to such a sequence of process model activities as a *trace translation* of trace σ, because it

represents a translation of the trace's events into process model activities. We denote a trace translation of σ with $\pi(\sigma)$.

Definition 4.3 (Trace Translation)**.** *Let $\sigma = \langle e_1, \ldots, e_n \rangle$ be a trace, M a process model with an activity set A, and $\mathcal{M}(\sigma, M)$ an event-to-activity mapping between σ and process model M. A trace translation $\pi(\sigma)$ represents a sequence of activities $\langle a_1, \ldots, a_m \rangle$ according to the mapping $\mathcal{M}(\sigma, M)$.*

Since an uncertain mapping $\mathbb{EA}(\sigma, M)$ consists of multiple event-to-activity mappings, a mapping $\mathbb{EA}(\sigma, M)$ results in different trace translations for σ. For instance in Section 4.1, we described an example where a single trace had two trace translations, $\pi_1(\sigma) = \langle a_1, a_2, a_4, a_5, a_6 \rangle$ and $\pi_2(\sigma_2) = \langle a_1, a_2, a_3, a_5, a_6 \rangle$, which resulted from two possible mappings. Together, the translations of a trace represent the spectrum of process behavior potentially conveyed by trace σ, i.e., the behavioral space of a trace. Since each mapping can be associated with a probability, we include a probabilistic component in our definition of a behavioral space, as captured in Definition 4.4.

Definition 4.4 (Probabilistic Behavioral Space)**.** *Let $\sigma = \langle e_1, \ldots, e_n \rangle$ be a trace, M a process model with an activity set A, and $\mathbb{EA}(\sigma, M)$ an uncertain event-to-activity mapping between σ and the activity set of process model M. We define a* probabilistic behavioral space *as a tuple $PBS_\sigma = (\Pi(\sigma), \phi)$, with:*

- $\Pi(\sigma)$*: the set of* trace translations *of trace σ over the activity set A as given by the event-to-activity mappings in $\mathbb{EA}(\sigma, M)$;*
- $\phi : \Pi(\sigma) \to [0, 1]$*: a function that assigns a probability to each trace translation in $\Pi(\sigma)$, such that the sum of the probabilities assigned to the trace translations is equal to 1.*

The set $\Pi(\sigma)$ comprises the set of potential trace translations of trace σ over the activity set A, where each translation $\sigma_i \in \Pi(\sigma)$ is based on a mapping $\mathcal{M}(\sigma, M)$ contained in $\mathbb{EA}(\sigma, M)$. This set, together with the probabilities provided by the function ϕ, provides the basis for the probabilistic conformance metric described next.

4.2.2 Using Behavioral Spaces for Conformance Checking

In this section, we demonstrate how to perform conformance checks by using behavioral spaces. To perform our conformance checks, we introduce a conformance metric that quantifies the conformance of a behavioral space PBS_σ to a process model M. The metric combines the conformance assessments for individual trace translations with probabilistic information. Specifically, it determines for each trace translation $\pi \in \Pi(\sigma)$ in a behavioral space whether it is conforming or not.

The goal of conformance checking is to determine if observed behavior in a trace σ is allowed by the behavioral specification of a process model M. Since uncertain event-to-activity mappings lead to multiple views on the process model behavior described by a trace (i.e., its trace translations), all these views need to be checked against the model M. In this section, we demonstrate how to perform a conformance check given a probabilistic behavioral space in order to obtain insightful conformance results and diagnostic information.

To perform conformance checks, we introduce a metric that quantifies the conformance of a behavioral space $PBS(\sigma)$ to a process model M. The metric combines conformance assessments for individual trace translations with probabilistic information. The metric determines for each trace translation $\pi(\sigma) \in \Pi(\sigma)$ in a behavioral space whether it conforms to M or not. In this way, the $ProbConf$ metric defined in Definition 4.5 represents the likelihood that σ conforms to model M.

Definition 4.5 (Behavioral Space Conformance)**.** *Let σ be a trace with a probabilistic behavioral space $PBS(\sigma) = (\Pi(\sigma), \phi)$ and M a process model with an activity set A and its set of valid execution sequences $\Sigma_{\text{valid}}(M)$. Then we define:*

- $\Pi_M(\sigma) = \Pi(\sigma) \cap \Sigma_{\text{valid}}(M)$ *as the set of trace translations in* $\Pi(\sigma)$ *conforming to process model M;*
- $ProbConf(\sigma, M) \in [0, 1]$ *: as the* behavioral space conformance *of trace σ to process model M, given as the sum of all probabilities $\phi(\pi(\sigma))$ for each $\pi(\sigma) \in \Pi_M(\sigma)$.*

Because of the probabilistic nature of the $ProbConf$ metric, the metric yields results that differ from the results obtained by traditional conformance-checking techniques. In traditional conformance-checking scenarios, i.e., without mapping uncertainty, a trace either conforms or does not conform to (a fragment of) a process model. By contrast, when using our technique, traces are either conforming, non-conforming, or *potentially conforming*. Potentially conforming traces are those traces for which some trace translations conform to a process model, whereas others do not. The conformance of these traces is associated with a certain probability $0 < p < 1$. Take, for instance, the process model from the running example and a trace $\sigma_1 = \langle e_1, e_2, e_3, e_4, e_5 \rangle$ with two trace translations $\sigma_1(\sigma_1)$ and $\sigma_2(\sigma_1)$:

$\pi_1(\sigma_1) = \langle a_1, a_2, a_4, a_5, a_6 \rangle$ with probability 0.7
$\pi_2(\sigma_1) = \langle a_1, a_2, a_3, a_5, a_6 \rangle$ with probability 0.3

While trace translation $\pi_1(\sigma_1)$ conforms to M, this does not apply for $\pi_2(\sigma_1)$. This latter translation skips the execution of the mandatory activity F. This leads to a conflict between the conformance of the different translations of σ_1 with respect to M. Therefore, the trace σ_1 is said to be potentially conforming to M with a probability of 0.7, i.e., $ProbConf(\sigma_1, M) = 0.7$. This shows that, even though we cannot make certain statements about the conformance of σ_1 to M, we do know that σ_1 is is more likely to conform than not. Furthermore, we also know the mapping conditions under which σ_1 is conforming or non-conforming. Namely, σ_1 conforms to M if the correspondence $e_3 \sim a_4$ holds, whereas the trace is non-conforming if $e_3 \sim a_3$ is true. This type of diagnostic information is very useful because it provides insights into which aspects of an uncertain mapping lead to uncertainty in the conformance-checking results for observed behavior.

It is important to note that the $ProbConf$ metric, despite its probabilistic nature and the presence of mapping uncertainty, can often still produce non-probabilistic (i.e., *deterministic*) conformance-checking results. To illustrate this, consider the traces σ_2 and σ_3 with the following associated trace translations:

$$\pi_1(\sigma_2) = \langle a_1, a_2, a_4, a_6, a_5 \rangle \quad \pi_1(\sigma_3) = \langle a_1, a_2, a_3, a_6, a_5 \rangle$$
$$\pi_2(\sigma_2) = \langle a_1, a_3, a_4, a_6, a_5 \rangle \quad \pi_2(\sigma_3) = \langle a_1, a_3, a_2, a_6, a_5 \rangle$$

For trace σ_2, mapping uncertainty has resulted in two trace translations that differ with respect to their second activity: a_2 for $\pi_1(\sigma_2)$ and a_3 for $\pi_2(\sigma_2)$. Despite this uncertainty, we can still with certainty state that σ_2 conforms to process model M. The reason is that both trace translations conform to M, since M allows for the execution of either activity a_2 or a_3. As a result, $ProbConf(\sigma_2, M) = 1.0$, thus yielding a deterministic result. In a similar fashion, we can be sure that trace σ_3 does not conform to M despite of its two different translations. Translations $\pi_1(\sigma_3)$ and $\pi_2(\sigma_3)$ execute activities a_2 and a_3 in two different orders. However, since model M only accepts the execution of one and not both of these activities, neither of the translations conform to M, leading to the deterministic outcome $ProbConf(\sigma_3, M) = 0.0$.

The previous example illustrates that our conformance-checking technique can be used to determine conformance with certainty in situations where traditional conformance-checking techniques would not be able to make trustworthy conformance assessments. In Section 4.3, we demonstrate the usefulness of this property in practical settings.

4.3 Evaluation

In this section, we present an evaluation that we conducted to demonstrate the capabilities of the proposed conformance-checking technique for uncertain event-to-activity mappings. The goal of this evaluation is to assess how the impact of mapping uncertainty on the conformance-checking task can be reduced by using our technique. To achieve this, we compare results obtained through our technique against results obtained by using a traditional conformance-checking technique. We apply these techniques on a collection of real-world process models and accompanying event logs. Specifically, we compare for how many traces in these event logs the two techniques can provide conformance-checking results with certainty.

In the remainder, Section 4.3.1 introduces the test collection used for the evaluation in detail. Section 4.3.2 describes the setup of our evaluation. Finally, Section 4.3.3 presents the evaluation results.

4.3.1 Test Collection

To perform the evaluation, we use a collection of real-world business process models from the *BIT process library*, first analyzed in an academic context by Fahland et al. [92]. The BIT process library consists of 886 process models from various industries, including the financial services and telecommunication domains. The same collection that has been used to test several event-to-activity mapping approaches [37,41], which motivates our choice for it. Hence, we believe that results obtained by using this collection present a realistic view on the applicability of the event-to-activity mapping approach against which we compare our conformance-checking technique. Furthermore, due to the size of the collection and its broad coverage of real-world process

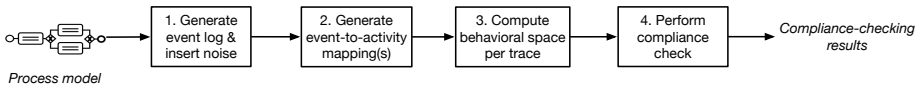


Figure 4.2: Overview of the evaluation setup

models, the collection seems well-suited to achieve a high external validity of the results.

From the test collection, we omitted any process model with soundness issues such as *deadlocks* or *livelocks*. Furthermore, we omitted a number of large models for which the event-to-activity mapping approach was not able to produce a results due to memory shortage. Note that the same filtering steps are also applied in [37]. As a result of the filtering, a collection of 598 process models remains available for usage in our evaluation.

4.3.2 Setup

Figure 4.2 depicts the three main steps of our evaluation setup. To perform these steps, we employ the *ProM6 framework*, which provides a vast amount of so-called *plug-ins* that implement process mining techniques.[1] In the first two steps of this evaluation, we build on existing plug-ins for event-to-activity mapping techniques, as described in [37]. For the third step, we implemented the generation of behavioral spaces and our proposed technique for conformance checking as a plug-in, which is available as part of the *BehavioralSpaces* package in ProM6.

In step 1 of the evaluation, we first generate an event log for each of the 598 process models in the test collection. In line with the evaluation in [37], we generate a log containing 1000 traces for each model. For process models that include loops, we generate traces with a maximum length of 1000 events. Since we are interested in conformance checking, we transform these fully conforming logs into logs containing non-conforming behavior. We achieve this by using a *noise-insertion* plug-in in ProM.[2] This plug-in randomly adds noise to a log (i.e., non-conforming behavior) by shuffling, duplicating, and removing events for a given percentage of traces. In this manner, we generate 11 different event logs, containing 0%, 10%, ..., 90%, 100% noise.

In step 2, we take a process model and an accompanying event log and use the mapping technique from [37] to establish an event-to-activity mapping. We have selected this particular technique because it returns all potential mappings in case of uncertainty. Furthermore, the technique is relatively robust in the context of non-conforming behavior. In case the approach computes a single mapping, i.e., there is no mapping uncertainty, we can conclude that for this process model and event log, traditional conformance-checking techniques suffice to determine the conformance of all traces in the log. If the mapping approach returns multiple possible mappings, i.e., there is mapping uncertainty, we continue with the third step of the evaluation.

In step 3, we first construct a behavioral space for a trace based on the uncertain

[1] See www.promtools.org for more information and to download the framework.
[2] Provided by the *Event2ActivityMatcher* package.

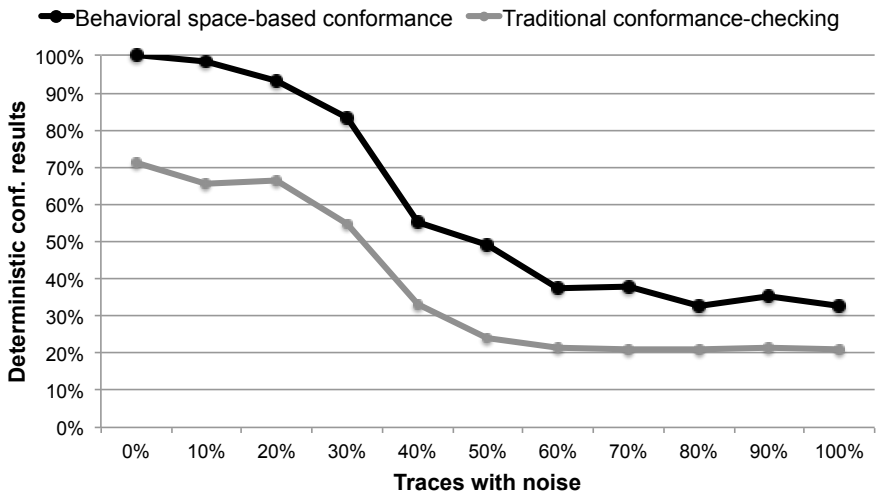


Figure 4.3: Evaluation results for deterministic conformance checking

event-to-activity mapping $\mathbb{EA}$ established in the previous step. We obtain a behavioral space by creating a trace translation for each of the potential event-to-activity mappings included in $\mathbb{EA}$, as described in Section 4.2.1. Afterwards, we perform a conformance check between the obtained behavioral space and the process model using the approach described in Section 4.2.2. If this method returns a $ProbConf$ value of 0.0 or 1.0 for the entire process model, we can conclude that our technique is able to provide a non-probabilistic conformance-checking result for the trace. This means that we know whether or not σ conforms to the process model with certainty. For other values, also the consideration of behavioral spaces does not suffice to be sure about the conformance of σ. Nevertheless, our technique still obtains probabilistic results and diagnostic information, whereas traditional conformance-checking techniques cannot provide any trustworthy results for these cases.

4.3.3 Results

Figure 4.3 presents the results of our evaluation experiments. The figure illustrates for which percentage of traces deterministic conformance-checking results are obtained by our and traditional techniques.

For noise level 0, where all traces in the event logs conform to the process models, we observe that the mapping approach establishes a single event-to-activity mapping for 71% of the models in the collection, which means that traditional techniques can provide (deterministic) results for 71% of the traces. Because these logs do not contain non-conforming behavior, the inability to establish mapping for certain models is caused by activities which are behaviorally identical to each other. Such cases can be seen for activities *C* and *D* of the running example. Because of these issues, traditional

conformance-checking techniques cannot assess the conformance of 29% of the traces. However, by using behavioral spaces, we can still determine the conformance of a trace when mapping uncertainty is caused by such behavioral equivalent activities. Hence, by using our proposed conformance-checking technique, we can provide deterministic conformance-checking results for all traces.

For increasing noise levels, the ability of the mapping approach to establish a single event-to-activity mapping diminishes. For logs with 10% and 20% noisy traces the approach can still establish a mapping for approximately 66% of the process models, as indicated by the 66% deterministic conformance results in Figure 4.3. However, this percentage sharply drops to 33% for event logs with 40% noise, followed by only 21% certainty for noise levels above 60%. As a result of these steep drops, the ability of traditional conformance-checking techniques to provide trustworthy conformance-checking results also sharply decreases. Although our behavioral space-based technique can also provide less deterministic results when the level of noise increases, this decrease is considerably less severe than for the benchmark. For instance, at 30% noise, our technique can still provide deterministic results for 83%, whereas traditional techniques can only provide such results for 55% of the cases. For the highest noise levels, our technique can still provide deterministic results for approximately 30% of the traces, which means that the technique outperforms the benchmark by close to 50%.

In summary, traditional conformance-checking techniques become less and less useful. For high noise levels, they can provide results for as little as 21% of the traces. Although the deterministic results obtainable through conformance checking with behavioral spaces is also affected by the increased levels of noise, the impact is much smaller. Therefore, we can conclude that in practical scenarios our conformance-checking technique is much wider applicable than traditional conformance-checking techniques. Furthermore, a crucial aspect in favor of our conformance-checking technique is that even in cases where also our technique cannot provide deterministic conformance-checking results, our technique still provides trustworthy conformance-checking information in the form of probabilistic results and diagnostic insights.

4.4 Limitations

The evaluation demonstrates that our conformance-checking technique can be used to obtain trustworthy results where traditional conformance-checking approaches fail to do so. However, these results need to be reflected against the background of some limitations. In particular, we identify limitations related to the utilized mapping technique, the conformance-checking technique, and limitations related to the evaluation.

Our proposed technique has to be considered in light of the limitation that the obtained conformance-checking results are dependent on the quality of the generated event-to-activity mappings. Most importantly, its results can be negatively affected if the correct mapping is not included in the set of potential mappings generated by any approach. Still, by applying our technique, we eliminate the need to select a mapping from the set of potential techniques. Hence, our technique significantly reduces the possibility of drawing incorrect conclusions.

A point of consideration regarding our conformance-checking technique is that we

base our conformance metric on the conformance of a full trace to a process model. This means that our technique does not distinguish between trace translations that are completely non-conforming and translations that are only slightly non-conforming. Such distinctions can be highly insightful [13, p.195], especially in order to investigate causes of non-compliance. We justify our choice for full trace conformance because the consideration of partial trace conformance would impede the interpretability of the $ProbConf$ metric. Such a metric would represent a product of probabilities and conformance levels, which are both non-integers in the range [0, 1]. Such values would lack clear meaning, given that a lower conformance score could be caused by various, orthogonal factors. Nevertheless, it is important to be aware that our metric is based on full trace conformance when interpreting the obtained results.

A limitation related to the evaluation of our technique is that our test collection consisted of partially generated data. While the process models used for the evaluation were obtained from a collection of real-world models, the event logs were automatically generated. This means that the obtained results may not fully reflect the situation in practice, where event logs related to the process models may have different characteristics. Nevertheless, we generated logs with varying levels of noisy behavior, which means that the utilized test collection contains behavior that can be observed in a variety of situations. Since our evaluation results show that our conformance-checking technique outperforms traditional techniques for all noise levels, we are confident that the improved performance also holds for real-world event logs.

4.5 Related Work

The work presented in this chapter primarily relates to two major research streams: event-activity mapping and conformance checking.

Our conformance-checking technique builds on techniques for the establishment of mappings between events and activities. Existing mapping techniques approach this goal in different ways, each with its own merits and limitations. An approach by Baier et al. [40] is the most extensive approach and the only one that focuses on the establishment of many-to-many relations between events and activities. The approach considers label similarity, work instructions associated with process models, and structural information in order to establish mappings. Despite the consideration of a variety of information, the approach still depends on human input in order to remove ambiguity, making it a semi-automated approach. Other approaches [37, 41] purely focus on behavioral similarity between events and activities. While these approaches only focus on one-to-one relations, they have been shown to be relatively robust to noise and non-conforming behavior. Work by Senderovich et al. [251] focuses on a specific situation for event to activity mapping. Specifically, the approach derives events based on information obtained from sensors in hospitals and aligns these logged events to process activities. Despite this specific context, the approach also deals with mapping uncertainty, which can lead to multiple potential event-to-activity mappings. Therefore, our conformance-checking technique is complimentary to all of these mapping approaches.

Process conformance-checking techniques are applied in various application sce-

narios, including process querying [33], legal conformance [239], and auditing [21]. A plethora of techniques exist for this purpose (cf. [15, 24, 199, 225]). In this chapter, we have used techniques that perform conformance checks based on behavioral profile relations, introduced in [279]. These techniques are computationally highly efficient, which makes them an ideal choice for conformance checking in the context of the potentially vast number of translations per trace. Other commonly used techniques perform conformance checks based on so-called *alignments*. These techniques, introduced in [15, 24], provide different diagnostic information than conformance checks based on behavioral profiles. Furthermore, the conformance checks can be considered to be more accurate in certain situations, because behavioral profile relations abstract from some details of process behavior. However, these techniques are computationally much more demanding than the highly efficient conformance checks based on behavioral profiles. For the purpose of efficiency, recent advances in decomposed conformance checking present a promising direction [199]. Since the interpretations in a behavioral space generally have considerable overlaps, such techniques can be useful in order to reduce the computation time required for conformance checking.

4.6 Summary

In this chapter, we introduced a conformance-checking technique that can be used in the presence of uncertain event-to-activity mappings. Our technique provides conformance-checking results without the need to select a single, possibly incorrect mapping to base conformance checks on. This is achieved by considering the entire spectrum of possible mappings generated by event-to-activity mapping techniques and capturing this spectrum in a behavioral space. Our probabilistic conformance-checking metric then provides insights into the fraction of conforming mappings, as well as useful diagnostic information. Therefore, our conformance-checking technique avoids the risk of drawing incorrect conformance conclusions. A quantitative evaluation based on a large collection of real-world process models demonstrated that our technique can be used to obtain results in a vast number of cases where traditional conformance-checking techniques fail to do so. In particular, our technique was able to provide deterministic results for up to 50% more cases than traditional techniques.

5

Dealing with Ambiguity in Textual Process Descriptions

The importance of automated conformance-checking has resulted in numerous conformance-checking techniques (cf. [15,33,63,279]), including the technique presented in Chapter 4. What these techniques have in common is that they rely on a structured specification of allowed behavior, mostly in the form of a *process model*. As a result, these techniques ignore the wealth of information that is contained in less structured forms of process documentation, such as textual process descriptions [99]. An important reason for this can be found in the inherent ambiguity of natural language. This ambiguity can lead to uncertainty about the exact specification of a process, which poses a considerable challenge to conformance-checking techniques. In prior work, text-to-process model generation techniques have circumvented this problem by introducing interpretation heuristics [99, 111, 253]. In this way, these techniques obtain a single process-oriented interpretation of the text, in spite of the presence of ambiguous sentences. This interpretation, however, contains assumptions on the correct interpretation of essentially undecidable ambiguity issues. So, there is always the risk that the derived interpretation conflicts with the proper way to execute the process. As a result, the focus on a single, assumed interpretation can lead to incorrect and, thus, untrustworthy conformance-checking results. To provide a rigorous solution for this problem, this chapter presents a conformance-checking technique that avoids the need to impose assumptions on the correct interpretation of ambiguous textual process descriptions. To achieve this, we again build on the concept of behavioral spaces, as introduced in Chapter 4, to capture the impact of behavioral ambiguity.

The remainder of this chapter is structured as follows. Section 5.1 illustrates the problems caused by ambiguity in textual process descriptions. Section 5.2 describes how the concept of a behavioral space can be used to capture ambiguity in textual process descriptions. The section, furthermore, presents a technique for the automated generation of behavioral spaces for this purpose. Section 5.3 illustrates the use of the generated behavioral spaces for conformance checking. Section 5.4 introduces a semi-

> After a claim is received, a claim officer reviews the request and records the claim information. The claim officer then validates the claim documents before writing a settlement recommendation. A senior officer then checks this recommendation. The senior officer can request further information from the claimant, or reject or accept the claim. In the former case, the previous steps must be repeated once the requested information arrives. If a claim is rejected, the claim is archived and the process finishes. If a claim is accepted, the claim officer calculates the payable amount. Afterwards, the claims officer records the settlement information and archives the claim. In the meantime, the financial department takes care of the payment.

Figure 5.1: Exemplary description of a claims handling process.

automated pruning technique that can be used to improve the quality of conformance-checking results. Section 5.5 demonstrates the usefulness of behavioral spaces and our proposed pruning technique through a quantitative evaluation using real-world data. Section 5.6 considers limitations of the proposed work and its evaluation. Section 5.7 discusses streams of related work. Finally, we summarize the chapter in Section 5.8.

5.1 Problem Illustration

In this section, we illustrate the problem associated with conformance checking of process behavior against textual process descriptions. The key challenge in this context is the ambiguity of natural language, as discussed in Section 2.3.4. Because conformance checking involves the comparison of observed versus allowed behavior, in this context we are particularly concerned with ambiguity related to the allowed process behavior described in a text. We shall refer to this as *behavioral ambiguity*. Behavioral ambiguity occurs when statements about the relations that exist between process steps can be interpreted in different ways. We illustrate the problem of behavioral ambiguity through the simplified description of a claims handling process, as presented in Figure 5.1. The description uses typical patterns to describe ordering relations, as observed in process descriptions obtained from practice and research [99].

At first glance, the description from Figure 5.1 may appear to be clear. However, on closer inspection, it turns out that the description does not provide conclusive answers to several questions regarding the proper execution of the described process. For instance:

Q1. Is it allowed that the claims officer records the claim information *before* reviewing the request?
Q2. Does it suffice for the claim officer to rewrite the settlement recommendation in case additional information has been requested?
Q3. Can the financial department start paying the claimant while the settlement information is still being recorded?

Based on the information provided in the textual description, these questions are not clearly decidable. This lack of decidability results from two forms of behavioral ambiguity: type ambiguity and scope ambiguity. *Type ambiguity* occurs when a textual description does not clearly specify the type of order relationship between two activities. For instance, the relation between the "*review request*" and "*record claim information*" activities in the first sentence is unclear. The term "*and*" simply does not allow us to determine whether these activities must be executed sequentially or whether they can be executed in an arbitrary order (Q1). *Scope ambiguity* occurs when statements in a textual description underspecify to which activity or activities they precisely refer. This type of ambiguity particularly relates to repetitions and parallelism. For instance, the statement "*the previous steps must be repeated*" does not clearly specify which activities must be performed again (Q2). Similarly, the expression "*in the meantime*" does not define when the financial department can start performing its activities (Q3).

As a result of such ambiguous statements, there are different views on how to properly carry out the described process. When deriving a single structured interpretation from a textual process description, as done by process model generation techniques (cf. [99, 111, 253]), there is always the risk that a derived interpretation conflicts with the proper way to execute the process. The focus on a single interpretation can, therefore, lead to wrong conclusions when reasoning about a business process. This can, for instance, result in a loss of efficiency by not allowing for parallel execution where possible (Q3). Furthermore, it can result in non-conformance with regulations, for example, by failing to impose necessary ordering restrictions (Q1) or by not repeating all of the required steps when dealing with the receipt of new claim information (Q2).

To avoid the problems associated with using an assumed interpretation, automated reasoning techniques should take into account all reasonable interpretations of a textual process description. Therefore, we next describe a technique that achieves this by employing the concept of behavioral spaces, as introduced in Chapter 4, in the context of behavioral ambiguity in textual process descriptions.

5.2 Capturing Ambiguity Using Behavioral Spaces

This section describes how process behavior from ambiguous textual process descriptions can be captured using behavioral spaces. For this purpose, we describe an approach that automatically generates a behavioral space from a textual process description. As shown in Figure 5.2, this approach consists of three main steps. First, a textual process description T is parsed in order to identify and analyze the set of behavioral statements $\mathcal{S}$. Second, the proposed approach generates behavioral interpretations for each statement in $\mathcal{S}$. Third and lastly, the different statement interpretations are combined into a collection of process interpretations that, together, comprise a behavioral space BS.

In the remainder of this section, we present the details on each of these three steps. Given the focus on behavioral ambiguity of this chapter, we mainly describe those aspects of the generation procedure that are specific to the consideration of ambiguity. Existing text-to-process model generation approaches, cf. [99], already address most

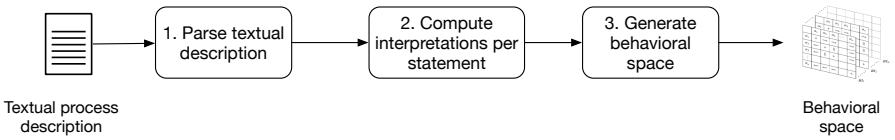


Figure 5.2: Steps involved to construct a behavioral space from a textual description

challenges related to the parsing of textual descriptions (step 1) and related to the extraction of behavioral relations for *unambiguous* behavioral statements (part of step 2). When illustrating the steps of our approach, we use the activity identifiers denoted in Table 5.1 to refer to activities described in the running example.

Table 5.1: Activities in the running example

ID	Activity	ID	Activity
a_1	Receive claim	a_8	Reject claim
a_2	Review request	a_9	Accept claim
a_3	Record claim information	a_{10}	Receive requested information
a_4	Validate documents	a_{11}	Calculate payable amount
a_5	Write settlement recommendation	a_{12}	Record settlement information
a_6	Check recommendation	a_{13}	Archive claim
a_7	Request further information	a_{14}	Arrange payment

5.2.1 Parsing Textual Process Descriptions

The first step in the approach is to parse a textual process description T. The goal of this parsing step is to identify a set of behavioral statements $\mathcal{S}$ and to extract behavioral information from these statements. Each behavioral statement in a text describes a single type of relation that holds between a number of activities. Therefore, a sentence in a textual description can contain more than a single behavioral statement. For example, the sentence "*After a claim is received, a claim officer reviews the request and records the claim information*", contains two separate behavioral statements. One statement describes the strict order relation between "*claim is received*" and "*reviews the request*". The other describes an ambiguous relation between "*reviews the request*" and "*records the claim information*".

For each behavioral statement in $\mathcal{S}$, the parsing step extracts the parts of the statement that refer to process concepts and their inter-relations. As such, the output of the parser consists of three *semantic components* that together comprise a behavioral relation of the form: <*source*, *relation type*, *target*>, for each behavioral statement. These components are a *source* reference, a *relation type* reference, and a *target* reference. We speak about *references*, rather than *process concepts* here, because textual descriptions often use different ways to refer to the same concept. For instance, "*previous activity*" provides a reference to some aforementioned activity; it is not a process concept itself.

To clarify the parsing results obtained in this manner, we present the parsing results for three statements in Table 5.2. We will use these examples in the remainder of this section to further illustrate the algorithm that generates statement interpretations.

Table 5.2: Exemplary outcomes of behavioral statement parsing

ID	Statement	Source ref.	Relation ref.	Target ref.
ς_1	After a claim is received, a claim officer reviews the request	*claim is received* (a_1)	*after*	*reviews the request* (a_2)
ς_2	a claim officer reviews the request and records the claim information	*reviews the request* (a_2)	*and*	*records the claim information* (a_3)
ς_3	In the meantime, the financial department takes care of the payment	ϵ (*not specified*)	*meantime*	*takes care of payment* (a_{14})

For this parsing step we can employ the parsers used by existing approaches for the generation of process models from texts, i.e., text-to-model generators, such as the ones described in Section 2.3.5. These approaches use a combination of standard NLP tools and *heuristic-based* techniques. NLP tools, such as the *Stanford Parser* [140], are used to identify the grammatical structure of sentences. This structure is important to extract the business object on which an action is being performed and to identify the actor who is performing the action. For example, the "*claims officer*" (actor) "*reviews*" (action) "*the request*" (business object). While the NLP tools mainly provide general techniques to analyze individual sentences and, thereby, extract activities, more tailored techniques are necessary to extract the ordering relations that exist between activities. For this purpose, model generation approaches employ heuristic-based techniques that recognize typical patterns used to express ordering relations. Such patterns include sequences like "After *activity i*, do *activity j*" or choices like "If *condition*, then *activity i*, else *activity j*". Since these techniques are extensively described in related work, we do not elaborate on them. The interested reader may consult the work by Friedrich et al. [99] for a detailed description of a state-of-the-art parsing technique.

5.2.2 Computing Statement Interpretations

Behavioral ambiguity in a textual process description occurs when behavioral statements can be interpreted in different manners. We refer to these statements as *ambiguous behavioral statements*. These statements result in multiple, conflicting activity relations. For instance, the statement "*a claim officer reviews the request and records the claim information*" results in two interpretations. It is unclear whether this statement implies a strict order or an interleaving order between the two described activities.

In the remainder, we use the term *statement interpretation* to refer to the set of behavioral relations that are associated with a single interpretation of a behavioral

statement. Unambiguous behavioral statements result in a single statement interpretation, whereas ambiguous statements lead to multiple interpretations. Definition 5.1 captures this notion. In this definition, we employ the *behavioral profile relations* from [279] to capture relations between activities. This choice is based on two main criteria. First, behavioral profile relations allow for an intuitive representation of the concepts we introduce in the remainder of this chapter. Second, the relations can be used for computationally efficient conformance checks. This feature is important because the computational complexity of conformance checks increases with the number of interpretations of a textual process description.

The behavioral profile relations capture the ordering restrictions that are in effect between pairs of activities. Four different behavioral profile relations can exist for an activity pair (a_i, a_j). The *strict order* relation $a_i \rightsquigarrow a_j$ is used to express that activity a_i cannot be executed after the execution of activity a_j. The *reverse strict order relation* $a_i \rightsquigarrow^{-1} a_j$ indicates the opposite restriction, namely that a_i cannot be executed after the execution of a_i.[1] The *exclusiveness* relation $a_i + a_j$ denotes that either activity a_i or activity a_j can be executed in a single process instance. Finally, the *interleaving order* relation $a_i \parallel a_j$ states that a_i and a_j can be executed in an arbitrary order. Using the activity identifiers specified in Table 5.1, this results in the relation $a_1 \rightsquigarrow a_2$ as a relation that holds between "*receive claim*" and "*review request*".

Definition 5.1 (Statement Interpretations)**.** *Let ς be a behavioral statement in the set of behavioral statements $\mathcal{S}$ of a textual process description T, A_T the set of activities described in T, and $\mathcal{R} = \{\rightsquigarrow, \rightsquigarrow^{-1}, +, \parallel\}$ the set of behavioral profile relations. We define Γ_ς as a set consisting of one or more statement interpretations for the statement ς. Each* statement interpretation *$\gamma \in \Gamma_\varsigma$ captures the behavioral relations that follow a possible interpretation of ς, which is defined as a partial function $\gamma : A_T \times A_T \nrightarrow \mathcal{R}$ that assigns a behavioral profile relation from $\mathcal{R}$ to a pair of activities from A_T, if any.*

In its second step, our approach constructs interpretations for each behavioral statement in $\mathcal{S}$. Algorithm 1 formalizes this part. The algorithm takes as input a parsed behavioral statement $\varsigma \in \mathcal{S}$, in which the semantic components have been identified. Then, it generates one or more statement interpretations, depending on the presence and the type of ambiguity in ς. Given a statement $\varsigma \in \mathcal{S}$, we distinguish three cases for this: (i) ς is unambiguous, (ii) ς contains type ambiguity, or (iii) ς contains scope ambiguity. For unambiguous statements, we obtain a single statement interpretation per behavioral statement. For the two types of ambiguous statements, we obtain multiple interpretations to reflect the various ways in which the statements can be interpreted. The following sections describe each of the three cases in detail.

Unambiguous Behavioral Statements

Generating a process interpretation for an unambiguous statement is relatively straightforward: all references in the statement can be resolved in a single and unambiguous manner. For these cases, we can generate a single statement interpretation by simply using the text-to-model generation approach to resolve the references and to construct

[1] Note that the reverse strict order relation $a_i \rightsquigarrow^{-1} a_j$ can only exist if and only if $a_j \rightsquigarrow a_i$.

Algorithm 1 Computing interpretations for a behavioral statement

```
1:  function computeStatementInterpretations(ParsedStatement ps)
2:      Set interpretations = new Set();
3:      if ps.isUnambiguous() then                          ▷ Statement is not ambiguous
4:          Set sourceActivities = ps.getSourceActivities();
5:          Set targetActivities = ps.getTargetActivities();
6:          RelationType r = ps.getRelationType();
7:          interpretations.add(createInterpretation(sourceActivities, r, targetActivities);
8:      if ps.hasTypeAmbiguity() then                       ▷ Relation type is ambiguous
9:          Set sourceActivities = ps.getSourceActivities();
10:         Set targetActivities = ps.getTargetActivities();
11:         for RelationType r ∈ RelationTypes.get(ps.getRelationReference()) do
12:             interpretations.add(createInterpretation(sourceActivities, r, targetActivities);
13:     if ps.hasScopeAmbiguity() then                      ▷ Scope of source reference is ambiguous
14:         Set targetActivities = ps.getTargetActivities();
15:         RelationType r = ps.getRelationType();
16:         Set sourceActivities1 = getPrecedingActivitiesWithSameResource(ps);
17:         interpretations.add(createInterpretation(sourceActivities1, r, targetActivities);
18:         Set sourceActivities2 = getPrecedingActivitiesWithSameObject(ps);
19:         interpretations.add(createInterpretation(sourceActivities2, r, targetActivities);
20:         Set sourceActivities3 = getActivitiesFromPrecedingControlFlowBlock(ps);
21:         interpretations.add(createInterpretation(sourceActivities3, r, targetActivities);
22: return interpretations;
```

a behavioral relation. Algorithm 1 describes this creation of an interpretation for unambiguous statements in lines 3–7. Since there is no ambiguity for any of the references, the employed text-to-model generation approach directly provides the set of source activities, target activities, and the relation type for the parsed statement. For example, in statement ς_1, the generation approach used to parse the statement will correctly identify that the "*claim is received*" activity a_1 should precede the "*reviews the request*" activity a_2, yielding the relation $a_1 \rightsquigarrow a_2$.

Statements with Type Ambiguity

A behavioral statement with type ambiguity describes a relation among a specific set of activities, but does not clearly define the type of relationship. Such statements can be identified because they have an ambiguous way to refer to the relation type, i.e. the relation reference is ambiguous. In line 8, Algorithm 1 checks if statement ς has type ambiguity by determining if the relation reference (*relationRef*) is a known ambiguous type indicator. In the context of this work, we focus on two ambiguous type indicators, namely the terms *and* and *or*. It is important to distinguish between cases where *relationRef* = "*and*" and cases where *relationRef* = "*and, then*" or similar. In the latter cases, the relation type is not ambiguous, since a sequential relation is clearly specified. Although these statements with unclear relation types are ambiguous, we can generate a set of statement interpretations that accurately capture the different possible interpretations.

If a statement indeed suffers from type ambiguity, the algorithm continues by resolving the unambiguous *source* and *target* activity references (lines 9–10). Afterwards, the algorithm generates a single statement interpretation for each applicable relation type (lines 11–12). For example, for statement ς_2 from Table 5.2, we generate one statement interpretation that contains a sequential relation and one that contains an interleaving order relation between activities a_2 and a_3.

Statements with Scope Ambiguity

Dealing with behavioral statements with scope ambiguity represents the most complex case of the three. These statements describe the existence of a relation, but do not specify between which activities this relationship holds. In practice, such ambiguity occurs with respect to the set of *source* activities to which a behavioral statement refers. For example, the statement ς_3 "*In the meantime, the financial department takes care of the payment*", does not specify to what activities "*meantime*" refers. Consequently, the source reference is empty (ϵ), as indicated in Table 5.2. Statements with scope ambiguity typically occur in the context of *parallelism* and *loops* because such behavioral patterns can be associated with textual references to groups of activities.

To automatically identify cases with scope ambiguity, we analyzed how existing techniques for the generation of process models from textual descriptions handle parallelism and loops. These techniques generally use heuristics to identify and analyze behavioral statements in a text. They build on predefined sets of indicators that pinpoint the different types of relations, e.g., "*while*" and "*in the meantime*" for parallel or interleaving order relations and "*is repeated*" for backward loops. Specifically, we analyzed the set of indicators used by the state-of-the-art technique from [99]. In this analysis, we isolated a subset of the parallelism indicators employed by the technique that typically result in statements with scope ambiguity. These indicators are presented in Table 5.3. What these indicators have in common is that their usage does not allow for the specification of a scope of the statement, i.e., these indicators cannot be associated with a source reference at all. Consider, for example, the sentence "*In the meantime, the financial department takes care of the payment*". It is syntactically not possible to specify to which set of activities a statement with *in the meantime* refers. By contrast, if we replace this construct with a non-ambiguous indicator, from the second set in Table 5.3, this problem can be avoided. For example, the indicator "*while*" can be used to specify the scope of the statement, e.g., "*While the claim is being archived*." By observing the usage of such ambiguous indicators, we can identify statements with scope ambiguity (line 13 of Algorithm 1) and generate interpretations for them (lines 14–21).

Table 5.3: Parallel indicators used in [99] and their classification

Class	**Contents**
Unambiguous	*while*, *as well as*, *in parallel to*
Ambiguous	*meanwhile*, *concurrently*, *meantime*, *in the meantime*

Although statements with scope ambiguity are highly problematic since they refer to an unknown set of activities, we can still identify a set of possible meanings for them. In particular, we can utilize the knowledge that statements such as "*the previous steps*" and "*in the meantime*" refer to distinct parts of a process. This means that the set of activities to which these statements refer *cannot* be any arbitrary combination of activities. The activities in the set must rather have a certain *commonality*, such as a set of activities that are all executed by the same person.

For this reason, we generate interpretations for statements with scope ambiguity based on sets of activities that have a certain common attribute. In particular, given a textual process description, we can identify sets of subsequently described activities that are (i) performed by the same resource, (ii) performed on or with the same (business) object, or (iii) are part of the same control-flow construct (e.g., a choice in the process). Based on this, we obtain different interpretations for the statement ς_3. In this statement, "*in the meantime*" can refer to different moments when the financial department can start taking care of the payment. This results in three possible statement interpretations, each with a different commonality:

1. *Common resource:* While the *claims officer*, the last described resource, is performing its tasks, after a senior claims officer has accepted the claim, i.e., in parallel to $\{a_{11}, a_{12}, a_{13}\}$;
2. *Common object:* While activities are being performed on the last mentioned business object (the *claim*), i.e., in parallel to $\{a_{13}\}$;
3. *Last control-flow construct:* While the last mentioned activity before the statement is being executed, i.e., in parallel to $\{a_{13}\}$.

These three possibilities result in three sets of relations that can follow from the same behavioral statement. Lines 14–21 describe the generation of a statement interpretation for each of the three sets of activities.

5.2.3 Generating a Behavioral Space

Using the interpretations of individual statements as a basis, we can construct views on the full process behavior described in a textual description. We refer to a specific view as a *process interpretation*. A process interpretation for a text T follows from the selection of a single *statement interpretation* for each statement ς in the set of behavioral statements $\mathcal{S}$. We define a process interpretation as given in Definition 5.2.

Definition 5.2 (Process Interpretation)**.** *Let T be a textual process description, A_T the set of activities described in T, $\mathcal{S}$ the set of behavioral statements in T with Γ_s the set of statement interpretations of a statement $\varsigma \in \mathcal{S}$, and $\mathcal{R} = \{\rightsquigarrow, \rightsquigarrow^{-1}, +, \|\}$ the set of behavioral profile relations. We then define a* process interpretation *as a tuple $P = (\mathcal{I}, BP)$, with:*

- *$\mathcal{I}$: a complete set of interpretations consisting of a single statement interpretation $\gamma \in \Gamma_\varsigma$ for each statement $\varsigma \in \mathcal{S}$, formally the following constraint holds: $\forall \varsigma \in \mathcal{S} : |\mathcal{I} \cap \Gamma_\varsigma| = 1$;*
- *$BP : A_T \times A_T \rightarrow \mathcal{R}$ a function that assigns a behavioral profile relation from $\mathcal{R}$ to each pair of activities from A_T.*

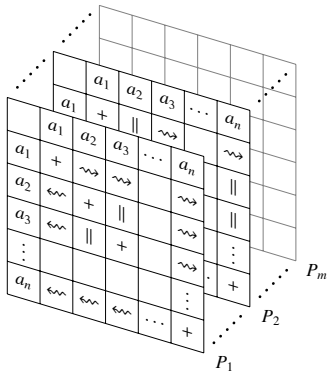


Figure 5.3: A behavioral space as a collection of m process interpretations

Recognize that in Definition 5.2 any activity relation that follows from a function $\gamma \in \mathcal{I}$ is part of BP, but that the reverse does not necessarily hold. BP defines a complete behavioral profile based on the interpretations included in $\mathcal{I}$, which includes relations that follow from the transitivity of the strict order and interleaving order relations [255]. For instance let $\gamma_1 \in \mathcal{I}$ be a statement interpretation that establishes the relation $a \rightsquigarrow b$ and let $\gamma_2 \in \mathcal{I}$ be a statement interpretation that establishes $b \rightsquigarrow c$. Then, BP will include the relation $a \rightsquigarrow c$ that follows due to transitivity[2], even though this relation is not part of any statement interpretation in $\mathcal{I}$. Despite the overlap between them, we include $\mathcal{I}$ alongside BP in Definition 5.2. This is done in order to preserve traceability between the obtained process behavior described by BP and the statement interpretation in $\mathcal{I}$ that provided the foundation for BP. As such, we can use this traceability to provide more useful diagnostic results when performing conformance checks.

A textual process description without ambiguity has exactly one process interpretation. For a textual process description T with behavioral ambiguity, the set of process interpretations in a behavioral space follows naturally as the set of possible combinations of statement interpretations for the ambiguous statements. For example, a text with two ambiguous behavioral statements ς_i and ς_j with, respectively, two and three different statement interpretations, will yield a set of $2 \times 3 = 6$ process interpretations. The concept of a behavioral space captures this spectrum of possible interpretations for a single process. Figure 5.3 visualizes this by depicting a three-dimensional view on the behavioral relations that exist between the activities in an ambiguous textual description. Definition 5.3 provides the formal definition of a behavioral space.

Definition 5.3 (Behavioral Space)**.** *Let T be a textual process description and $\mathcal{S}$ the set of behavioral statements in T. We define a* behavioral space *BS as a set of process interpretations of the textual process description T over the behavioral statements in*

[2]Note that this only applies if no interpretation $\gamma \in \mathcal{I}$ denotes a (different) relation between activities a and c.

$\mathcal{S}$.

Algorithm 2 presents an algorithm to generate a behavioral space based on the statement interpretations obtained in the previous step of our approach. A behavioral space should contain all possible combinations of statement interpretations for the behavioral statements. Lines 3–14 in Algorithm 2 describe the creation of these combinations. The underlying idea is that all existing text interpretations, starting from an empty list (line 3), are incrementally extended with a single statement interpretation (lines 8–11). This ensures that each possible combination of statement interpretations is included in the list. For example, as considered in the previous section, the claims handling process contains three ambiguous statements with, respectively, two, three, and two possible interpretations. This results in a total number of 12 ($2 \times 3 \times 2$) possible combinations of the statement interpretations and, thus, of 12 interpretations in BS. After all combinations have been generated, we compute a full behavioral profile for each of the interpretations and add them to the behavioral space (lines 15–17).

Algorithm 2 Generate a behavioral space from a textual process description

```
1:  function constructBehavioralSpace(Text text)
2:      BehavioralSpace behavioralSpace = new BehavioralSpace();
3:      List processInterpretations = new List();
4:      processInterpretations.add(new ProcessInterpretation());
5:      List newProcessInterpretations = new List();
6:      for Statement s ∈ text.getStatements() do
7:          List interpretations = computeStatementInterpretations(s);
8:          for Interpretation i ∈ interpretations do
9:              for ProcessInterpretation pi ∈ processInterpretations do
10:                 Interpretation newPI = pi.copy();
11:                 newPI.add(i);
12:                 newProcessInterpretations.add(newPI)
13:         interpretations = newInterpretations.copy();
14:         newInterpretations.clear();
15:     for ProcessInterpretation pi ∈ interpretations do
16:         computeFullBehavioralProfile(pi)
17:         if pi.getBehavioralProfile().isConsistent() then
18:             behavioralSpace.add(pi);
19:     return behavioralSpace;
```

To compute the complete behavioral profile for a process interpretation (line 16), we exploit the transitivity of the strict order and interleaving order relations [255]. In this way, we can obtain relations beyond those pair-wise relations that we extracted from a textual description. For example, if a text specifies that activity a_i is followed by a_j and a_j is followed by a_k, i.e. $a_i \rightsquigarrow a_j$ and $a_j \rightsquigarrow a_k$, then a_i is also followed by a_k, i.e., $a_i \rightsquigarrow a_k$. In lines 17 of the algorithm, we employ a technique defined by Smirnov et al. [255] to check the internal consistency of the obtained behavioral profile. We only include mutually consistent process interpretations to the behavioral space (line 18). For example, we exclude obviously incorrect interpretations in which one statement interpretation yields the relation $a_i \rightsquigarrow a_j$ and another the relation $a_i + a_j$.

Once the process interpretations have been obtained in this manner, the construction of the behavioral space is complete. The generated behavioral space can be used to obtained trustworthy conformance-checking results, despite the presence of behavioral ambiguity. Section 5.3 describes this.

5.3 Conformance Checking using Behavioral Spaces

By capturing behavioral ambiguity in a structured manner, behavioral spaces allow us to reason about process conformance without the need to arbitrarily settle ambiguity. In this section, we demonstrate this by showing how to perform a conformance check of an execution trace versus a behavioral space generated from a textual process description. This conformance check is highly similar to the one presented in Chapter 4, which performs a conformance check in the context of uncertain event-to-activity mappings. The key difference between traditional conformance checking and conformance checking using behavioral spaces lies in the potential outcomes of a check. In traditional conformance checking, a trace is either *conforming* or it is *non-conforming* with a business process. Due to the behavioral ambiguity captured in behavioral spaces, a trace can be conforming or non-conforming, but also *potentially conforming* with a behavioral space. The latter outcome occurs for traces that conform to one or more process interpretations in a behavioral space, but not with all of them.

To determine the level of conformance between a trace and a behavioral space, we consider the number of process interpretations of a textual process description to which a trace conforms. In particular, we quantify the *support* of a behavioral space BS for a trace σ as the ratio between the number of interpretations to which σ conforms and the total number of interpretations in BS:

$$\text{supp}(\sigma, BS) = \frac{|\{P \in BS \mid conf(\sigma, P)\}|}{|BS|} \tag{5.1}$$

In Equation 5.1, we use $conf(\sigma, P)$ to denote that a trace σ conforms to process interpretation P. Given that a process interpretation is defined based on behavioral profile relations, this conformance check involves the comparison of the behavioral profile relations of trace σ to the behavioral profile relations of P. This is achieved based on the conformance checking method described in [282]. The support metric $\text{supp}(\sigma, BS)$ quantifies the fraction of interpretations in BS that allow for a trace σ to occur. A support value of 1.0 indicates that a trace is without any doubt conforming with the behavioral space, i.e., independent of the chosen interpretation. A support of 0.0 shows that there is no interpretation under which a trace conforms to the behavioral space. Therefore, it can be said with certainty that the trace is non-conforming with BS. Finally, any trace σ with a support value $0.0 < \text{supp}(\sigma, BS) < 1.0$ is potentially conforming with BS. This implies that there are certain interpretations of the textual description to which the trace conforms. To illustrate the usefulness of the support metric, consider the following three partial execution traces of the running example:

- $\sigma_1 = \langle a_1, a_2, a_3, a_4, a_5 \rangle$
- $\sigma_2 = \langle a_1, a_3, a_2, a_4, a_5 \rangle$
- $\sigma_3 = \langle a_{11}, a_{14}, a_{12}, a_{13} \rangle$

The difference between the traces σ_1 and σ_2 is that, in σ_1, activity a_2 occurs before a_3, whereas these are executed in reverse order in σ_2, i.e., $a_2 \rightsquigarrow a_3 \in BP_{\sigma_1}$ and $a_3 \rightsquigarrow a_2 \in BP_{\sigma_2}$. Furthermore, recall that the behavioral relation between these two activities is given by the ambiguous behavioral statement ς_2. Depending on the interpretation of ς_2, there either exists a strict order or an interleaving order relation between a_2 and a_3, i.e., $R(a_2, a_3) = \{\rightsquigarrow, \|\}$. The relation $a_2 \rightsquigarrow_{\sigma_1} a_3$ from σ_1 is subsumed by both possible interpretations included in the behavioral space, since sub($\rightsquigarrow, \rightsquigarrow$) and sub($\rightsquigarrow, \|$) are both satisfied. Therefore, σ_1 is conforming with all interpretations in BS and, thus, has a support value of 1.0. For trace σ_2 we observe a different situation. While $a_3 \rightsquigarrow_{\sigma_2} a_2$ in trace σ_2 is subsumed by relation $a_2 \| a_3$, this relation is not subsumed by $a_2 \rightsquigarrow a_3$. Therefore, σ_2 does not conform to half of the process interpretations in BS. This results in supp(σ_2, BS) = 0.5.

Aside from providing information on the process interpretations with which a trace conforms, behavioral spaces allow us to obtain further diagnostic information from this conformance check. In particular, we can gain insights into the conditions under which a trace is conforming with a process description. For example, we can learn under which interpretations of the statement ς_3, "*In the meantime, the financial department takes care of the payment*", trace σ_3 is conforming. In σ_3, the financial department pays the settlement amount (a_{14}) before the claims officer records the settlement information (a_{12}). This conforms to one of the two interpretations of statement ς_3 and, therefore, results in a support value of 0.5. Furthermore, we know that this trace is conforming, if and only if "*in the meantime*" means "*while the claims officer is performing its tasks*" and not "*while the claims officer is archiving the claim*". Such diagnostic information can be useful when interpreting the support values for a trace or when aiming to resolve the ambiguity contained in a textual description.

5.4 Pruning Behavioral Spaces based on Information Gain

This section demonstrates how the behavioral space of a textual process description can be reduced to provide more accurate conformance-checking results. We refer to this act as *pruning* a behavioral space. In particular, we present a technique that supports users in resolving behavioral ambiguity in an efficient manner. The technique identifies those ambiguous behavioral statements in a textual description that have the biggest impact on the ambiguity in a process. To achieve this, we define an *information gain* metric. This metric quantifies the level of uncertainty in conformance-checking results caused by an ambiguous behavioral statement. It serves a similar purpose as the information gain metric used in the context of decision trees to quantify reductions in *information entropy* (cf. [222, 240]). Before introducing the information gain metric, we first consider how conformance-checking uncertainty can be reduced by resolving ambiguous statements and, thereby, pruning a behavioral space.

5.4.1 Conformance-Checking Uncertainty

In Section 5.3, we demonstrated that behavioral spaces allow for reasoning about conformance without the need to resolve behavioral ambiguity. This is achieved by introducing the notion of *potential conformance*, which captures situations where a trace is conforming to some process interpretations, but non-conforming to others. Such a classification provides valuable information, especially in the context of the diagnostic information that can be associated with it, i.e., for which statement interpretations a trace is conforming or non-conforming. Still, such cases represent a form of unclarity in the conformance-checking results, because it is not known if a trace is actually conforming or not. We will refer to this state as *conformance-checking uncertainty*.

The accuracy of conformance-checking results can be improved by reducing the level of conformance-checking uncertainty. This can be achieved by resolving the cause of ambiguity in ambiguous statements. For instance, by replacing an ambiguous type indicator such as "*and*" with either "*and, then*" or with "*and, meanwhile*". In these cases, there are less potentially conforming traces and more traces for which it can be stated with certainty that they are conforming or not. Behavioral spaces represent a powerful tool to support users in this endeavor. First, behavioral spaces support the improvement of conformance checking accuracy by providing insights into the *causes* (i.e., the ambiguous statements) and the *effects* (i.e., the different interpretations) of behavioral ambiguity. For instance, the behavioral space shows us that statement s_2 from the running example is ambiguous. Therefore, it is clear that if a user decides to resolve this ambiguity by selecting the correct interpretation of s_2, we reduce the overall conformance-checking uncertainty. Second, behavioral spaces can support users even further by letting them focus their resolution efforts on the ambiguous statements that are the greatest causes of conformance-checking uncertainty. We achieve this with the information-gain metric that we introduce next.

5.4.2 Information Gain

We introduce Information Gain (IG) as a metric that describes how much conformance-checking uncertainty can be resolved by selecting a single interpretation for an ambiguous statement. A proper quantification for this gain is to consider for how many traces the interpretations of a single ambiguous statement disagree about their conformance. By resolving the ambiguity in statements of which the interpretations disagree about the largest number of traces, a maximum of conformance-checking uncertainty can be removed. We define IG for a set of statement interpretations Γ_ς and a set of traces (or log) L in Equation 5.2.

$$IG(\Gamma, L) = |\bigcup_{\gamma \in \Gamma} C_L(\gamma) - \bigcap_{\gamma \in \Gamma} C_L(\gamma)| \tag{5.2}$$

In this equation, we use $C_L(\gamma)$ to refer to the set of traces from L that are conforming to the behavioral relations of γ. $IG(\Gamma, L)$ specifies the size of the set of traces that are conforming to at least one interpretation, but also non-conforming to at least one interpretation. This is computed by taking all traces that are allowed according to at

least one interpretation in Γ, i.e., the union of all sets $C_L(\gamma)$ for $\gamma \in \Gamma$, minus those traces that are allowed by all interpretations in Γ, i.e., the intersection of these sets.

The metric IG can be applied in two different ways, depending on the availability of an event log. If an event log is not available, a log L_G can be generated that contains all traces that are potentially conforming to the behavioral space BS. In this case, $IG(\Gamma, L_G)$ can be used to identify those phrases that lead to the biggest potential reduction in ambiguity. However, if an event log L_R related to the process is already available, the information gain metric can be used to compute the information gain in the context of truly observed behavior. In this case, $IG(\Gamma, L_R)$ represents the gain in ambiguity specific to the event log L_R.

To illustrate the usage of IG, consider the statements ς_2 and ς_3 used throughout this chapter (introduced in Table 5.2), on the one hand, and a log L with the following (partial) execution traces, on the other:

- $\sigma_1 = \langle a_2, a_3, a_{12}, a_{13}, a_{14} \rangle$
- $\sigma_2 = \langle a_2, a_3, a_{14}, a_{12}, a_{13} \rangle$
- $\sigma_3 = \langle a_2, a_3, a_{14}, a_{12}, a_{13} \rangle$
- $\sigma_4 = \langle a_3, a_2, a_{14}, a_{13}, a_{12} \rangle$

Recall that statement ς_2 has two interpretations, with the following sets of behavioral relations: $\{a_2 \rightsquigarrow a_3\}$ and $\{a_2 \parallel a_3\}$. It can be easily observed that these statements disagree about any trace in which a_3 occurs before a_2, i.e. of which the behavioral profile contains $a_3 \rightsquigarrow_t a_2$. Trace σ_4 is the only trace in L for which this is the case. Therefore, $IG(\Gamma_{s_2}, L) = 1$. To compute IG for statement ς_3, it suffices to consider the most restrictive and most flexible interpretations of the statement. The most *restrictive* interpretation states that a_{14} can only be executed in parallel, i.e., possibly before, activity a_{13}. By contrast, the most *flexible* interpretation of ς_3 specifies that a_{14} is in an interleaving order with a_{11}, a_{12}, and a_{13}. This means that from log L, only trace σ_1 is conforming to the former interpretation, whereas all four traces are conforming to the latter interpretation of ς_3. Therefore, three traces in L are in dispute by the interpretations in Γ_{s_3}, i.e. $IG(\Gamma_{s_3}, L) = 3$. From this, it can be concluded that the resolution of ambiguity in statement ς_3 has a greater impact on the ambiguity in log L than the resolution of ς_2.

By computing IG for all ambiguous statements in a behavioral space and resolving the statements with the highest information gain, users can efficiently reduce the level of conformance-checking uncertainty. As such, the behavioral space will be pruned by removing process interpretations that are not conforming with the resolved ambiguity. This greatly enhances the efficient usage of the notion of behavioral spaces for conformance checking.

5.5 Evaluation

In this section, we evaluate the usefulness of behavioral spaces for conformance checking in the context of ambiguous textual process descriptions. For this purpose, we conduct a two-stage evaluation. First, we assess the impact that the consideration of behavioral spaces has on conformance-checking results. In particular, we compare the

conformance-checking results obtained by using behavioral spaces to two alternative ways of dealing with behavioral ambiguity. Second, we demonstrate the effectiveness of the proposed IG metric for the reduction of uncertainty in conformance-checking results.

In the remainder, Section 5.5.1 first introduces the test collection used in both parts of the evaluation. Then, Sections 5.5.2 and 5.5.3 respectively describe the evaluation of the conformance-checking results and of the IG metric.

5.5.1 Test Collection

To perform our evaluation, we reuse the collection of textual process descriptions from the text-to-model generation approach by Friedrich et al. [99]. The collection contains 47 process descriptions from various industrial and scholarly sources. These textual descriptions were also used in the evaluation described in Chapter 3. Table 5.4 gives an overview of the characteristics of the test collection.

Table 5.4: Overview of the test collection

ID	Source	Type	T	S	W
1	HU Berlin	Academic	4	10.0	18.1
2	TU Berlin	Academic	2	34.0	21.2
3	QUT	Academic	8	6.1	18.3
4	TU Eindhoven	Academic	1	40.0	18.5
5	Vendor Tutorials	Industry	4	9.0	18.2
6	inubit AG	Industry	4	11.5	18.4
7	BPM Practitioners	Industry	1	7.0	9.7
8	BPMN Practice Handbook	Textbook	3	4.7	17.0
9	BPMN Guide	Textbook	6	7.0	20.8
10	Federal Network Agency	Public Sector	14	6.4	20.0
	Total		**47**	**9.2**	**17.2**

Legend: T = Number of textual process descriptions, S = Sentences per text (avg.), W = Words per sentence (avg.)

The data from Table 5.4 illustrate that the included process descriptions differ greatly in size. The average number of sentences ranges from 4.7 to 34.0. The longest process description contains a total of 40 sentences. Furthermore, the descriptions differ in the average length of the sentences. While the BPM Practitioners source contains process descriptions with rather short sentences (9.7 words), the process descriptions from the TU Berlin source contain relatively long sentences (21.2 words). Lastly, the process descriptions differ in terms of how explicitly and unambiguously they describe the process behavior. Among others, this results from the variety of authors that created the textual descriptions. Hence, we believe that the collection is well-suited for achieving a reasonably high external validity of the results.

5.5.2 Conformance Evaluation

To demonstrate the usefulness of behavioral spaces for conformance checking, we compare the conformance-checking results obtained by using behavioral spaces to two alternative ways of dealing with behavioral ambiguity. These two alternatives are: (i) imposing assumptions on the correct interpretation of behavioral statements, and (ii) ignoring ambiguous statements because they cannot be resolved. The goal of this part of the evaluation is to show that behavioral spaces provide a much more reasonable view on the process behavior allowed by a textual process description, when compared to the two alternatives. We first describe the details of the setup used for this part of the evaluation and then present and discuss the results.

Setup

To conduct the evaluation, we implemented a prototype to generate behavioral spaces from textual process descriptions. To achieve this, we build on the state-of-the-art text-to-process model generation approach by Friedrich et al. [99]. In particular, our Java prototype uses a library that is part of the RefMod-Miner[3], which implements the process model generation approach in a stand-alone tool. We use the library to automatically identify activities and extract behavioral profile relations that exist between activities.

We compare the conformance-checking results obtained by using behavioral spaces to two alternative ways of dealing with behavioral ambiguity. The first alternative reflects the possibility to deal with behavioral ambiguity by imposing assumptions on the correct interpretation of a text, i.e., by selecting a single interpretation for each ambiguous behavioral statement. Second, it is possible to deal with behavioral ambiguity by ignoring all ambiguous statements and, thus, only focusing on the behavioral relations that can be extracted with certainty from a text. Given these alternatives to behavioral spaces, we generate three behavioral models (BMs) for each of the 47 textual process descriptions as follows:

1. **Fully interpreted behavioral profile ($\mathbf{BP^{full}}$):** This behavioral model reflects an approach that imposes assumptions on the correct interpretation of ambiguous statements. To obtain BP^{full}, we generate a process model by using the text-to-model generation approach from [99] and, subsequently, extracting a behavioral profile from this model;

2. **Minimally restricted behavioral profile ($\mathbf{BP^{min}}$):** This behavioral model reflects an approach in which ambiguous statements are fully ignored. The resulting behavioral profile only captures the behavioral relations that can be extracted with certainty from the textual process description. To obtain BP^{min}, we remove all behavioral profile relations from BP^{full} that were extracted from the analysis of ambiguous behavioral statements;

3. **Behavioral space (BS):** The behavioral space generated for the textual description in accordance with the interpretation generation method described in Section 5.2.

[3] http://refmod-miner.dfki.de

We conduct our evaluation by comparing the sizes of the sets of traces that are (potentially) conforming with the three behavioral models, in accordance with the definitions provided in Section 5.3.[4] Using $C(BM)$ to refer to the collection of traces that are conforming or potentially conforming to a behavioral model BM, we quantify the size differences using the following two metrics:

$$R_1 = \frac{|\, C(BS)\,|}{|\, C(BP^{min})\,|} \qquad (5.3) \qquad\qquad R_2 = \frac{|\, C(BP^{full}))\,|}{|\, C(BS)\,|} \qquad (5.4)$$

R_1 quantifies the ratio between the number of traces allowed by a behavioral space and a minimally restricted behavioral profile. This measure reflects how much behavior that certainly does not conform to σ is allowed when ambiguous statements are ignored. R_2 quantifies the ratio between the number of traces allowed by a behavioral space and those allowed by a fully interpreted behavioral profile. This measure reflects how much behavior that is possibly conforming to σ is marked as nonconforming by an approach that imposes assumptions on ambiguous statements.

Results

Table 5.5 summarizes the evaluation results. The first interesting thing to note is how common textual process descriptions with behavioral ambiguity are. In total, 32 of the 47 textual process descriptions (70%) contained one or more ambiguous phrases. The majority of these cases, 28 in total, included just phrases with type ambiguity. Four cases contain statements with scope ambiguity, 3 of which also contain behavioral statements with type ambiguity.

Table 5.5: Evaluation results

Collection	**T**	$\mathcal{S}_{\textbf{type}}$	$\mathcal{S}_{\textbf{scope}}$	**A**	**BS**	$\mathbf{R_1}$	$\mathbf{R_2}$
Only type ambiguity	28	64	0	19.6	11.0	100.0%	37.8%
With scope ambiguity	4	13	4	24.0	76.5	16.4%	0.5%
Total	32	77	4	20.2	19.1	89.5%	33.7%

Legend: T = number of textual process descriptions, $\mathcal{S}_{type}$ = statements with type ambiguity, $\mathcal{S}_{scope}$ = statements with scope ambiguity, A = extracted activities per process (avg.), BS = interpretations per behavioral space.

For processes with just type ambiguity in their descriptions, there is a clear difference between the behavior allowed by fully interpreted behavioral profiles $C(BP^{full})$ and the behavior allowed by behavioral spaces $C(BS)$. As indicated by metric R_2, the fully interpreted behavioral profiles allow for only 37.8% of the behavior allowed by

[4] For processes that contain loops, we only include traces with at most one repetition.

the behavioral space. For the remaining 62.2% of the traces, we *cannot* state with certainty that they do not conform to the process described in the text. This difference results from ordering restrictions that the text-to-model generation algorithm imposes on activities, even when these ordering restrictions may not exist. Behavioral spaces do not impose such restrictions and, thus, mark traces that exhibit such execution flexibility as potentially conforming. This consideration of the cases with type ambiguity already illustrates the impact of assumptions on conformance checking. Nevertheless, this impact is much more severe for textual process descriptions that also contain statements with scope ambiguity.

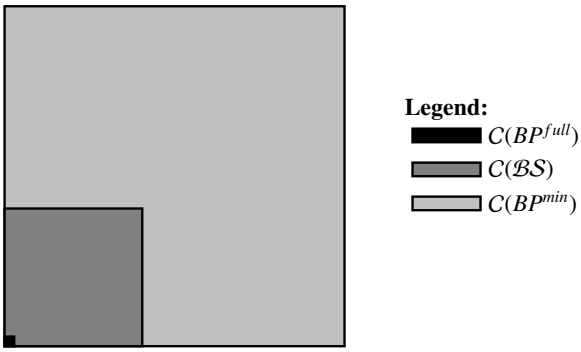


Figure 5.4: Visualization of three sets of conforming traces for cases with scope ambiguity

The behavioral models for the 4 cases with scope ambiguity show much larger differences among the behavior they allow. We visualize the relative sizes of the three sets of conforming traces in Figure 5.4. The light-gray area denotes the set of traces conforming with BP^{min}, i.e., the set of traces that remain when treating ambiguous statements as undecidable. The behavior allowed by the behavioral space, represented by the dark-gray area, is considerably smaller, as also indicated by the R_1 score of 16.4%. This number reveals that 83.6% of the traces in $C(BP^{min})$ are not conforming with any reasonable interpretation of the statements with scope ambiguity. Figure 5.4 also shows the considerable impact that the usage of single interpretations has on the number of conforming traces. The tiny size of the black area in the figure and the R_2 score of 0.5% both indicate that, for the cases with scope ambiguity, the fully interpreted behavior profiles allow for only a very small fraction of the behavior that is (potentially) conforming to a behavioral space. Again, the remaining 99.5% represent traces that do not necessarily conflict with behavior specified in a textual process description.

The evaluation results show the impact both of ignoring ambiguous statements and of imposing single interpretations on them. As visualized by Figure 5.4, behavioral spaces provide a balance between these loosely restricted and overly restricted behavioral models. In summary, behavioral spaces exclude a large number of nonsensical traces, which can be excluded by generating proper interpretations for ambiguous statements. Still, they allow for many more traces than the restricted models that are

obtained by imposing assumptions on the ambiguous statements in textual descriptions.

5.5.3 Pruning Evaluation

In the second part of the evaluation, we set out to demonstrate how effective the proposed pruning technique is at reducing uncertainty in conformance-checking results. Specifically, we assess how quickly conformance uncertainty can be reduced when we employ the *IG* metric, introduced in Section 5.4, to select ambiguous phrases. As a benchmark, we compare the results obtained in this manner to a random selection mechanism. We first describe the details of the setup used for this part of the evaluation and then present and discuss the results.

Setup

To evaluate the effectiveness of the *IG* metric, we make use of the behavioral spaces generated for the textual descriptions in the previous part of the evaluation. Specifically, we select the behavioral spaces for the 17 textual descriptions with more than one ambiguous behavioral statement. For these cases it is relevant to determine which ambiguous phrase should be resolved first. To compute values for the IG metric, we generate a log L_t for each textual description T that contains all traces that are potentially conforming to the behavioral space BS.

In this evaluation, we are interested in how much we can reduce conformance-checking uncertainty by using IG to select the ambiguous phrases we resolve first. We quantify this reduction by comparing the number of *potentially conforming* traces that remain after resolving the ambiguity in a statement to the original number of potentially conforming traces. Again, we use $C(BS)$ to denote the set of potentially conforming traces for a given behavioral space BS. Then, we compute the fraction of conformance uncertainty that remains after resolving k ambiguous statements as given by Equation 5.5. Here, BS_k represents the behavioral space that remains after resolving k ambiguous statements. BS_0 represents the behavioral space for which no ambiguity is resolved.

$$U(BS, k) = \frac{|C(BS_k)|}{|C(BS_0)|} \tag{5.5}$$

We compute the value $U(BS, k)$ for each $k \in 1, \ldots, n$, where n is the number of ambiguous statements in BS. As a benchmark, we compare these values against the uncertainty that remains after randomly selecting ambiguous statements to resolve. In particular, we compute the average value of $U(BS, k)$ for each of the $n!$ different orders in which ambiguous statements can potentially be selected. As such, this benchmark mimics the situation in which people blindly select ambiguous statements to resolve.

Results

Figure 5.5 visualizes the evaluation results. The curves represent the average reduction in uncertainty over the 17 relevant cases. The figure shows considerable differences

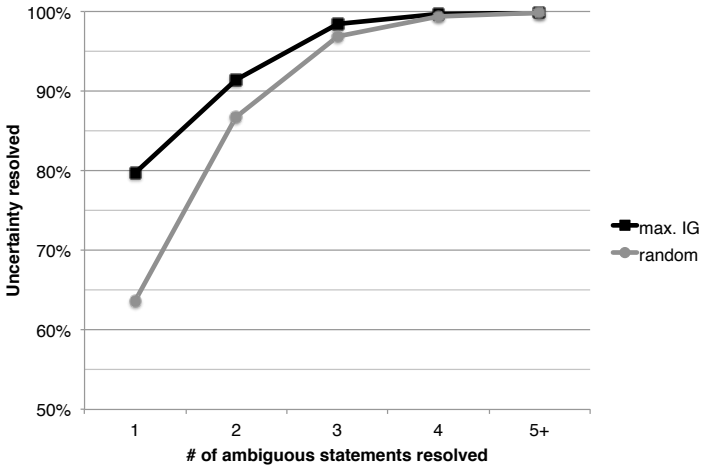


Figure 5.5: Comparison of uncertainty resolution using max. IG and random selection

between the reduction obtained by using the information gain metric versus the reductions obtained through random selection.

When interpreting these results, it is important to recognize that the minimum uncertainty reduction per statement observed in the test collection is 50.0%. This quantity represents the amount of uncertainty that is removed when resolving the most simple form of an ambiguous statement: a statement with type ambiguity between just two activities. Therefore, the reduction of 63.7% that is obtained by randomly resolving a single ambiguous statement appears is only 13.7% higher than the minimum improvement per statement. In comparison, usage of the IG metric results in a removal of 79.7% of the conformance-checking uncertainty in the first step. This reflects an improvement of 29.7 percentage points above the minimum improvement per statement. Therefore, the IG metric leads to an improvement that is more than twice as high as random selection.

The 79.7% reduction in uncertainty demonstrates that, in many textual descriptions, a single statement has a much bigger impact on the ambiguity than the others. These statements are typically the ones with scope ambiguity. This is the case because scope ambiguity results in a high number of potentially conforming traces, usually caused by interpretations with a considerable number of activities with interleaving order relations. This is, for instance, shown for statement ς_3 considered in the running example of this chapter. Similarly, statements with type ambiguity among more than two activities result in an increased number of interleaving order relations compared to statements with ambiguity between just two activities. Therefore, these statements result in a high number of potentially conforming traces. By resolving these ambiguous statements first, the information gain metric can be used to quickly resolve the vast majority of conformance-checking uncertainty.

5.6 Limitations

Our evaluation demonstrates the usefulness of behavioral spaces for reasoning about process conformance in the context of textual process descriptions. However, the evaluation results should be considered against the backdrop of some limitations. In particular, we are able to identify limitations related to the behavioral space generation approach, the conformance-checking technique, and related to the evaluation.

Limitations related to the *generation approach* concern two facets. First, it has to be considered that the natural language processing techniques on which we build our approach are not fully accurate. The model generation approach from [99] is heuristics-based, which means it does not cover all possible linguistic patterns that can be used to express behavioral relations. For instance, the approach cannot handle constructs corresponding to OR-gateways in process models, such as "*At least two of the following three activities should be performed*". Furthermore, the approach is not always able to properly recognize backwards loops in process descriptions. However, since this approach represents the state of the art, it does provide an accurate reflection of the quality of model generation approaches. Also, it is important to stress that our generation approach is largely independent of the underlying model generation approach. As long as this approach can provide information regarding source, relation type, and target references, behavioral spaces can be generated according to the algorithms detailed in Section 5.3. Second, it has to be taken into account that we generate interpretations for statements with scope ambiguity based on three commonalities. While these commonalities represent the plausible options in the context of process descriptions, it is possible that, in certain situations, other properties are necessary to capture the set of activities to which a statement refers. Still, the behavioral space will identify such statements as ambiguous and help users to analyze the impact of this ambiguity on conformance checks.

A point of consideration regarding our conformance-checking technique is that we perform conformance checks based on behavioral profiles. The expressive power of these relations has been shown to be less than that of process modeling notations such as Petri nets, which are used by certain other conformance-checking techniques [15]. Because of this lack of expressive power, behavioral profiles abstract from certain process behavior. As a result, conformance checking based on behavioral profiles is less restrictive than conformance-checking techniques based on Petri nets. [216] provides a detailed overview of the restrictions on expressiveness. A particular weakness in this regard relates to the handling of loops. When loops are present, behavioral profile-based conformance checking is considerably less restrictive than other techniques. However, as previously mentioned, these constructs are notoriously problematic when parsing textual process descriptions. Therefore, it is important to be aware of this limitation, but also to recognize its limited impact given the current state-of-the-art text-to-model generation approaches. Nevertheless, we have selected behavioral profiles because they enable highly efficient conformance checks, which is an important prerequisite given the numerous possible process interpretations that can exist in behavioral spaces. However, the primary reason is that the transitive property of behavioral profile relations allow us to combine the implications of different statement

interpretations in an intuitive and safe manner.

Finally, the obtained evaluation results must be considered in light of the specifics of the textual process descriptions included in the test collection. To mitigate the impact of these specifics on the results, we composed the test collection from several heterogeneous sources. As a result, the included descriptions contain a considerable degree of ambiguity. Furthermore, this data set has been previously used to evaluate the process model generation approach of Friedrich et al. [99]. Therefore, we are confident that our evaluation shows a realistic picture of the impact of behavioral ambiguity in practical settings.

5.7 Related Work

The work presented in this chapter primarily relates to three major research streams: conformance checking, the analysis of textual process descriptions and the representation of data uncertainty.

We previously provided an overview of conformance checking in Section 4.5, including a variety of its application scenarios and existing techniques. What these existing techniques have in common is that they all compare observed behavior against a structured process representation, such as a process model or business rules. The technique presented in this chapter differs in this regard, because it is designed to work with unstructured or semi-structured descriptions in natural language.

The relevance and widespread use of text documents as a source for process analysis has been emphasized in various contexts [99, 163, 166, 250]. The majority of works that consider the analysis of textual artifacts related to business processes focus on the automated derivation of process models from them. Respective techniques have been designed for textual process descriptions [99, 108], group stories [111], use case descriptions [253], and textual methodologies [270]. From these, the text-to-model generation technique from Friedrich et al. [99] is typically recognized as the state of the art [232]. Therefore, we used it as a basis for our own prototype and as a benchmark for our evaluation. Though none of these existing works mentions the problem of behavioral ambiguity explicitly, all techniques impose assumptions on the interpretation of ambiguous behavioral statements. This results in a single interpretation, i.e., a process model, for any given text. However, this comes with the great disadvantage that the behavior allowed by this representation is much stricter than the behavior specified in the textual description.

Similar to behavioral ambiguity in textual process descriptions, ambiguous or uncertain data is also present in other application contexts. In these cases, uncertainty can be caused by, among others, data randomness, incompleteness, and limitations of measuring equipment [210]. This has created a need for algorithms and applications for uncertain data management [25]. As a result, the modeling of uncertain data has been studied extensively (cf. [20, 126, 211, 246]). Our notion of a behavioral space builds on concepts related to those used in uncertain data models. For instance, similar to the behavioral interpretations captured in a behavioral space, the model presented by Das Sarma et al. [246] uses a set of *possible instances* to represent the spectrum of possible interpretations for an uncertain relation. Furthermore, the model described in [20]

uses conditions to capture dependencies between uncertain values. This notion has the same result as the sets of behavioral relations that we derive from uncertain behavioral statements and convert into different behavioral interpretations. Still, the technical aspects and application contexts of these uncertain data models, which mostly relate to querying and data integration [25], differ considerably from the process-oriented view of behavioral spaces.

5.8 Summary

In this chapter, we leveraged the concept of a behavioral space to deal with the ambiguity in textual process descriptions. A behavioral space captures all possible interpretations of a textual process description. By using behavioral spaces for conformance checking, we avoid the need to impose assumptions on the correct interpretations of ambiguous natural language texts. Therefore, conformance checks based on behavioral spaces provide trustworthy results: They avoid the risks associated with the selection of incorrect interpretations. Furthermore, we demonstrated the usefulness of behavioral spaces for the analysis of textual process descriptions. In particular, we used a quantitative evaluation with a set of 47 textual process descriptions to illustrate that a behavioral space strikes a reasonable balance between ignoring ambiguous statements and imposing fixed interpretations on them. Furthermore, we demonstrated the usefulness of a semi-automated pruning technique to quickly reduce the level of uncertainty remaining in conformance-checking results. In particular, by considering the information gain associated with the resolution of an ambiguous statement, the degree of conformance-checking ambiguity can be reduced twice as fast as through random selection.

6

Transforming and Aligning Process Performance Indicators

Monitoring Process Performance Indicators (PPIs) represents an important prerequisite for organizations in their strive for continuous process optimization [151]. Therefore, a key task for managers is to define suitable PPIs that are aligned with the strategic business objectives of the organization [233]. Typically, managers achieve this by describing relevant PPIs using natural language descriptions [235,286]. This has the great advantage that PPIs can be easily specified and understood by all stakeholders [158]. However, to be able to actually monitor PPIs, they must be defined in a way that supports their automated computation and relates them to the technical implementation of a business process in a Workflow Management or Enterprise Resource Planning System [96]. Although there are structured notations that achieve this (cf. [218, 233, 286]), they are not at all similar to the unstructured natural language descriptions used and preferred by managers.

Currently, the structured PPI definitions required to automatically monitor PPIs can only be obtained by manually transforming unstructured natural language descriptions [234]. Such a transformation requires considerable time and effort from a number of resources. To illustrate the causes of this, consider the situation we observed during our extensive research collaboration with the *Andalusian Health Service* [1, 235, 236]. There, the IT department received requests to measure PPIs for all organizational processes, as specified by other departments in natural language. The IT department had to manually establish SQL queries in order to compute the desired values for these PPIs. In many cases, the necessary interactions between business and IT led to miscommunication. For example, as a result of misinterpretations or incomplete specifications, the IT department often obtained incorrect PPI values. The time required to clear up these differences was considerable. In the general case, this necessary effort is even more problematic in light of the potentially vast size of process model repositories (possibly containing hundreds or even thousands of models [237]) and because processes and performance measures are subject to continuous change [280]. In sum, this makes the task of manually transforming PPIs into a structured notation hardly manageable in

practice.

To overcome this problem, this chapter presents an approach to automatically translate natural language PPI descriptions into PPIs that can be automatically monitored. In the remainder of this chapter, Section 6.1 illustrates the challenges associated with the transformation task. Section 6.2 defines templates that we use for the structured notation of PPIs and that represent the output format for our approach. Section 6.3 describes the transformation approach itself. Section 6.4 presents a quantitative evaluation that demonstrates the approach's usefulness. Section 6.5 reflects on the limitations of our transformation approach and the evaluation. Section 6.6 discuss streams of related research. Finally, Section 6.7 summarizes the chapter.

6.1 Problem Illustration

To illustrate the challenges associated with the transformation of natural language PPI descriptions into a structured notation, consider the process model shown in Figure 6.1. This figure depicts a simplified order handling process using BPMN. Table 6.1 provides six exemplary PPI descriptions related to this process. These descriptions use linguistic patterns that are similar to those that have been observed in practice [235].

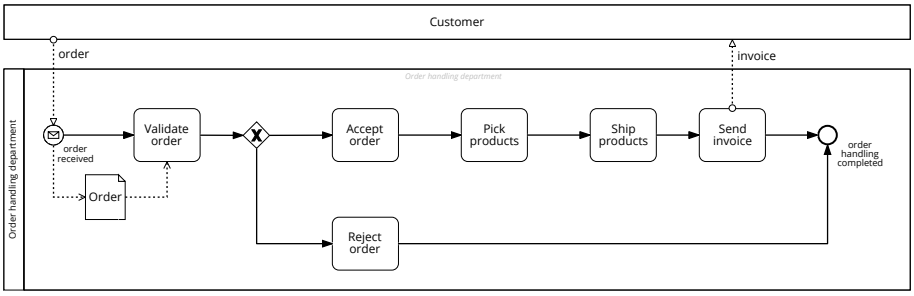


Figure 6.1: Process model for an order handling process

Table 6.1: PPIs for the order handling example

ID	Description
PPI1	Number of accepted orders.
PPI2	Average time between receipt and completion of an order.
PPI3	Average time to complete a received order.
PPI4	The percentage of rejected orders.
PPI5	The maximum time to transport an order.
PPI6	The total order amount per customer.

To actually compute values for these PPIs, the natural language descriptions must be transformed into a structured notation. In some cases, this transformation is relatively straightforward. For instance, it is clear that the description of *PPI1* refers to the

Table 6.2: Example of a structured notation for PPI2

Slot	**Value**
ID	*PPI2*
Description	*Average time between receipt and completion of an order*
Measure type	*time*
Aggregation	*average*
Start event	*Order received*
End event	*Order completed*

number of process instances for which the "*Accept order*" activity is executed. The transformation of *PPI2* is also relatively easy. For this PPI, we are interested in the average time between the "*order received*" and "*order handling completed*" events. Table 6.2 provides an example of how this PPI can be captured in a structured manner. Here we use *measure type* and *aggregation* to specify that the PPI computes the average time over process instances, whereas we use *start event* and *end event* to establish the link between the PPI and the events of the process model depicted in Figure 6.1. However, in other cases, the automated transformation of a PPI description into a structured definition is associated with considerable challenges. These challenges mainly relate to the flexible and inherently ambiguous nature of natural language, as preferred by human users, but much less suitable for automated interpretation [182, 250]. We identify three main challenges in this regard.

The first challenge to overcome is that natural language descriptions do not follow a specific structure or notation. These descriptions can express the same PPI using a broad variety of syntactic patterns [23]. *PPI2* and *PPI3*, for instance, both refer to the average time between the receipt and completion of an order. However, the two PPI descriptions use clearly distinct patterns to describe this measure. *PPI2* explicitly refers to the start and end points of the measure in chronological order. By contrast, *PPI3* describes these two points in a reverse order, i.e., the end point "*completed*" is described before the start point "*received*."

The second challenge to overcome is that natural language PPI descriptions can depend on implicit knowledge for their proper interpretation. Consider, for example, *PPI4*: "*The percentage of rejected orders*". This PPI refers to some fraction, where the numerator refers to the number of process instances in which the "*Reject order*" activity is executed. However, the PPI description does not specify the denominator for this fraction, i.e., it is not clear from the description by what number this numerator should be divided. Instead, the description depends on the implicit assumption that the denominator, most likely, refers to the total number of received orders.

Lastly, the third challenge that a transformation approach must address are problems caused by differences in terminology between PPI descriptions and process models. For example, the description of *PPI5* refers to the time it takes to "*transport orders*", whereas the process model refers to this action as "*product shipment*." Such differences occur in particular because PPIs and process models are generally defined by different organizational stakeholders, with their own perspectives on a process.

In summary, an automated transformation approach must be able to deal with (i) a variety of syntactic patterns, (ii) partially implicit PPI descriptions, and (iii) differences in terminology between description and model. The approach presented in this chapter addresses these challenges. Before describing this approach in detail, we introduce so-called templates, which represent the output required by an automated transformation approach.

6.2 Template-Based PPI Definitions

This chapter focuses on the transformation of natural language PPI descriptions into ones that follow a structured notation, which allows for their automated monitoring. This requires us to first define a structured notation into which we aim to transform the unstructured descriptions. We define this structured notation in the form of *PPI templates*. Each PPI template captures the different concepts of a PPI definition that are required for its automated computation. We refer to the placeholder of each concept in the template as a *slot*, to which a value, a so-called *slot filler*, must be assigned when defining a PPI. For the definition of these templates, we build on the PPINOT metamodel, introduced in [235]. We selected this metamodel because of its high degree of expressiveness and because it explicitly establishes links to process model elements. Due to these links, the PPINOT metamodel can be used to define structured PPIs whose values can be computed in an automated manner. Specifically, we base the templates for our approach on the templates and associated linguistic structures defined in [236]. In Section 6.2.1, we introduce the slots that correspond to the different semantic concepts that constitute the PPI templates. Section 6.2.2 describes the value domains associated with these slots.

6.2.1 Template Slots

We define four different PPI templates, each corresponding to a different type of measure that can underly a PPI: *numerical*, *temporal*, *data-based*, and *fractional* PPIs. These templates can capture the vast majority of PPIs that have been observed in empirical studies [234, 235]. Table 6.3 presents the four templates and provides an example for each of them.

Aside from an identifier and the natural language description, all of the templates capture four distinct semantic concepts, as illustrated by Figure 6.2: (i) measure type, (ii) aggregation function, (iii) group-by property, and (iv) one or more process concepts. The first three semantic concepts together characterize the measure of a PPI. The fourth concept links components of the PPI to the actual implementation of a business process. We now elaborate on these semantic concepts.

Measure type. The measure type classifies the nature of the measure underlying a PPI. In this work, we focus on the four most common measure types according to [236]: count, time, data, and fraction measures. A *count* PPI measures the number of times something happens. For instance, *PPI1* measures the number of orders that are accepted. *Time* measures consider the duration between two instants, i.e., the start or

Table 6.3: PPI templates and examples

Slot	**Value**	**Slot**	**Value**
ID	PPI1	ID	PPI2
Description	Number of accepted orders	Description	Average time between receipt and completion of an order
Measure type	count	Measure type	time
Aggregation	sum	Aggregation	average
Counted event	Accept order [completed]	Start event	Order received
Group-by	—	End event	Order completed
		Group-by	—
ID	PPI4	ID	PPI6
Description	The percentage of rejected orders	Description	The total order amount per customer.
Measure type	fraction	Measure type	data
Aggregation	sum	Aggregation	sum
Numerator	Reject order [completed]	Measured attr.	Order [amount]
Denominator	Order received	Group-by	customer
Group-by	—		

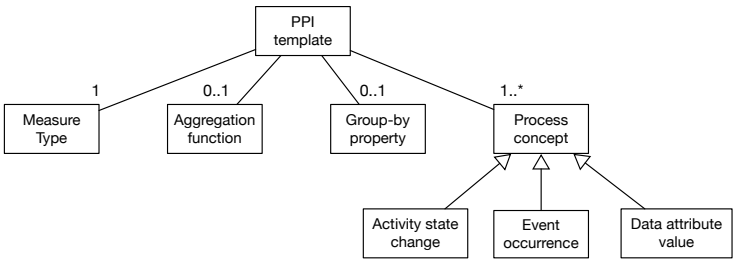


Figure 6.2: Semantic concepts in a PPI definition

completion of an activity or the execution of an event. *PPI2* and *PPI3* represent examples of such time measures. *Data* measures consider the attribute values of data objects, as seen for *PPI6*, which sums the amounts associated with "*order*" data objects. Finally, *fraction* measures divide the value of one measure by another. For example, *PPI4* divides the number of *rejected orders* by the number of *received orders*.

Aggregation function. Aggregation functions are used to aggregate the values of multiple process instances using a specific metric. For example, *PPI2* and *PPI3* consider the *average* time of individual order handling instances, whereas *PPI5* considers the *maximum* time it takes to transport an order. Note that the aggregation function is an optional slot in the templates; it is not necessary to specify it for every measure.

Group-by property. Group-by properties can be applied on measures with an aggregation function in order to compute a value aggregated over those process instances that have a certain property. These properties can be based on data attributes or on resources that are involved in the execution of a process instance. For example, *PPI6* requires a group-by property to determine the total order amounts *per customer*. A group-by property is an optional slot in a PPI template. If no such property is defined, the aggregated measure is simply applied on all process instances.

Process concepts. A process concept refers to a value or an occurrence during the execution of a business process that is relevant to the computation of a PPI. As shown in Figure 6.2, process concepts can refer to different things in the context of a PPI template. For instance, an activity is started, a process model event occurs, or the value of a data attribute changes. Each PPI template has one or more slots associated with process concepts. The exact number and the semantics of these slots differ per measure type. In particular, we define the following semantic roles for the different measure types:

- count measure: *counted event*;
- time measure: *start event*, *end event*;
- data measure: *measured attribute*;
- fraction measure: *numerator*, *denominator*.

Each of the semantic roles associated with a particular measure type should be filled with at least one process concept. For example, a time measure should always have a *start* and an *end* event. Furthermore, a slot may be filled with multiple events. For example, a count measure can be used to measure the total number of occurrences of multiple events, such as the number of accepted *and* the number of rejected orders.

6.2.2 Slot Domains

As illustrated in the previous section, PPI templates contain four different types of slots: measure types, aggregation functions, group-by properties, and process concepts. In a structured PPI definition, each of these slot types can only be filled with values from a closed class, i.e., from a specific *domain*. By filling slots with values from known

domains, we can ensure that values for the defined PPIs can actually be computed in an automated manner. Consider, for example, the textual description of *PPI2*. This description refers to the "*completion of an order*". Without incorporating domain knowledge, it is not clear to which process concept this statement exactly refers. Therefore, it is not possible to compute a value for this PPI based on this information. If we, instead, fill this slot with a value from the appropriate slot domain, i.e., by associating the slot with the "*Order handling completed*" process model event, we overcome this issue. Then, we know that the computation of a value for this PPI requires us to stop the measurement when this specific event occurs. Table 6.4 summarizes the domains for the four different slot types.

Table 6.4: Domains associated with template slots

Slot type	**Domain values**
Measure type	*count*, *time*, *data*, *fraction*
Aggregation function	*minimum*, *maximum*, *average*, *sum*
Group-by property	Data objects
Process concepts	Activities, events, & data objects

To increase the expressiveness of the PPI definitions that follow our templates, we base the slot domains on an extended version of the process model definition provided in Definition 2.4. In particular, we utilize the *abstract business process modeling language* designed to be used with the PPINOT framework, which provides the foundation for our PPI definitions. The metamodel for this abstract modeling language, defined in [233], introduces two main, additional concepts in process models: the notion of a *data object* and of a *runtime state*. A data object can be used to capture data that is relevant to a process and its execution, such as data attributes and resources that execute activities[1]. In general, the attributes that can be associated in this way are highly similar to event attributes, as given by Definition 2.6. Examples of data objects relevant to the running example are an *order*, the *order amount*, and an *invoice*.

Runtime states are used to denote values that are assigned or changed as the process instance is executed. For example, process model activities can be associated with states that correspond to the transaction types associated with events from event logs, defined in Section 2.2. In this way, runtime states can be used to differentiate between, for instance, the start and completion of activities for a process instance. Similarly, runtime states can denote the status of data objects. For example, in an order handling process, an *order* data object may have a status, such as *rejected* or *accepted* associated with it as a runtime state.

To operationalize these additional concepts, we extend the process model definition from Definition 2.4 into a definition that includes the data objects and runtime states necessary to capture additional details of a process:

[1]Note that this notion of data objects from [233] is broader than the concept of data objects in, for example, BPMN.

Definition 6.1 (Data-Aware Process Model)**.** A data-aware process model is a tuple $M_D = (A, E, G, N, F, t, O, u, S, v)$, where:

- A is a finite set of activities,
- E is a finite set of events,
- G is a finite set of gateways,
- $N = A \cup E \cup G$ is a finite set of nodes,
- $F \subseteq N \times N$ is the flow relation, such that (N, F) is a connected graph,
- $t : G \rightarrow \{and, xor\}$ is a function that maps each gateway with a type,
- O is a finite set of data objects,
- $u : O \rightarrow N$ is a partial function that maps data objects to nodes,
- S is a finite set of runtime states,
- $v : S \rightarrow N \cup O$ is a surjective function that maps runtime states to nodes and data objects.

Measure type & Aggregation The domains for measure type and aggregation function slots are independent of the process to which the PPI relates. A measure type slot always receives a value from one of the four measure types considered by our approach: D_{type} = {count, time, data, fraction}. Similarly, the domain for aggregation functions covers the most common functions identified in [236]: D_{aggr} = {minimum, maximum, average, sum}.

Group-by property The domain for group-by properties depends on the contents of the process model to which the PPI relates. The group-by property can be defined using any data object from the set O. This means that the group-by property can be used to group process instances based on data values or resources involved the execution of an instance. For instance, the group-by property of *PPI6* corresponds to the "*customer*" data object associated with a process instance.

Process concepts Process concept slots are filled with references to parts of a process implementation. Therefore, the domain associated with these slots differs per process model. This domain is based on a model's activities (and their transaction types), events, and data attributes. Given these elements, a process concept slot can be filled in three different ways, as previously indicated in Figure 6.2:

1. Process concepts in a PPI definition can refer to the state change of a process model activity. We consider state changes of activities rather than activities as a whole, because the calculation of PPIs often requires the consideration of an exact time instant. For example, in order to compute the time it takes to ship products, we are interested in the difference between the time instants of the *start* and *completion* of the activity execution. Specifically, a process concept here is aligned to the combination of a process model activity $a \in A$ and a runtime state mapped to a by the function v;
2. Process concept slots can correspond to the triggering of an event $e \in E$. Since such event occurrences are always instantaneous, this is not associated with a state change. As an example, consider PPI2. There, both the start and end times refer to process model events, i.e. to the "*order received*" and "*order handling completed*" events;

3. Process concept slots can be associated with data attribute values. For example, to compute PPI6, "*The total order amount per customer*", we have to consider the value of the data attribute *amount* of the *order* data object. Here, the slot of a process concept is aligned to a data object $o \in O$.

In the next section, we present our approach, which automatically fills the defined templates for natural language PPI descriptions.

6.3 Transformation Approach

To transform an unstructured natural language PPI description into a structured notation, we define the transformation task as a *template-filling* problem. In this section, we describe a transformation approach that addresses this problem. Given a natural language PPI description, our approach aims to obtain the information necessary to fill the slots of a PPI template. To achieve this, the transformation approach performs two subsequent steps, as depicted in Figure 6.3.

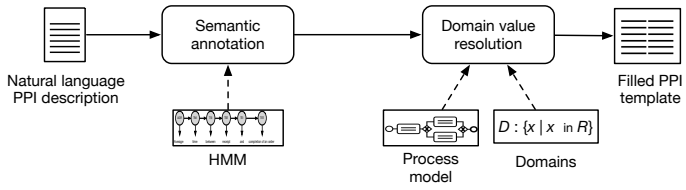


Figure 6.3: Overview of the proposed transformation approach

First, the approach parses a natural language description in order to identify the parts of the description that correspond to slots of the PPI template. For this parsing step we use HMMs. We refer to this parsing task as *semantic annotation*. Second, the approach determines the appropriate slot-fillers by matching the semantically annotated parts to values in the relevant domain, e.g., by matching concepts from the natural language description to activities in a process model. We refer to this task as *domain value resolution*. In the remainder of this section, Sections 6.3.1 and 6.3.2 respectively describe the semantic annotation and domain value resolution tasks in detail. Afterwards, Section 6.3.3 discusses the operationalization of our approach and its extensibility.

6.3.1 Semantic Annotation

The first step of our approach identifies those parts of a natural language PPI description that correspond to the semantic concepts contained in the PPI templates. For example, for *PPI5*: "*the maximum time to transport an order*", we aim to identify that "*maximum*" corresponds to the concept of an aggregation function, "*time*" corresponds to the measure type, and "*transport an order*" describes the events to be measured. For this semantic annotation task, we define a tag set Σ that can be used to annotate relevant semantic concepts in natural language PPI descriptions.

Semantic Tags

Table 6.5 presents the tag set we use for the semantic-annotation step. We establish this tag set Σ by defining two sorts of tags. First, Σ includes tags that correspond to each of the semantic concepts from the PPI templates. For example, we use *AGR* to refer to an aggregation function and FNE to denote a *numerator event* of a fraction PPI. Second, Σ contains tags used to annotate textual indicators that signal the transition of the description to a new semantic concept. For example, we use the tag TEI, i.e., a *time end indicator*, to show that a time end event will be described next. This can be seen for *PPI*2, "*Average time between receipt and completion of an order*" where the term "*and*" denotes a transition from the description of the start event ("*receipt*") to the end event ("*completion of an order*"). To illustrate the usage of the tag set, consider the following annotation of *PPI*2 from the order handling process:

```
average\AGR, time\TMI, between\TSI, receipt\TSE, and\TEI,
completion of an order\TEE.
```

From this annotation, we can observe that "*average*" corresponds to the aggregation function slot of a PPI template, "*receipt [of an order]*" to the start event and "*completion of an order*" to the end event.

Table 6.5: Tag set used for semantic annotation

Tag	Description	Examples
AGR	Aggregator function	average, sum, maximum
GBI	Group by indicator	per, for each
GBC	Group by concept	category, customer, department
CMI	Count measure indicator	number of, volume of, number of times
CE	Counted event	accepted orders, incidents
TMI	Time measure indicator	time, duration, throughput time
TSI	Time start indicator	from, between
TSE	Time start event	order received, product picking
TEI	Time end indicator	to, until
TEE	Time end event	order completion, product shipment
TBE	Time start and end event	service interruptions, product packaging
FMI	Fraction measure indicator	percentage of, ratio of, proportion for
FNE	Fraction numerator event	rejected orders
FDI	Fraction division indicator	divided by, relative to, as a percentage of
FDE	Fraction denominator event	received orders

In the tag set Σ, we purposefully do not define separate tags for data measures, because their semantic structure generally follows a similar syntactic pattern as other measures. Consider, for instance, PPI6: "*the total order amount per customer*." This description of a data measure follows an identical structure as a count measure would. However, in the context of this process model, "*amount*" refers to a data attribute of an

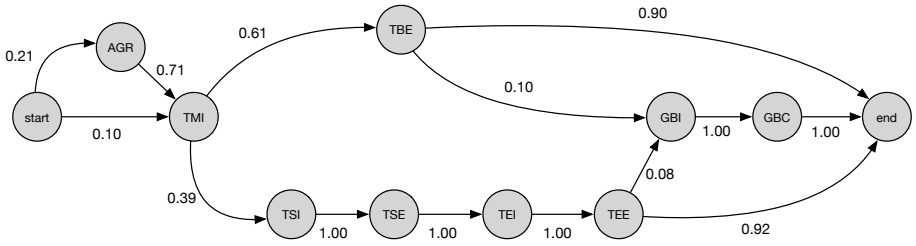


Figure 6.4: Fragment of a semantic prior $P(\Theta)$

order, rather than describing a regular count measure. Since such distinctions solely depend on the contents of the process model, i.e., on its data objects, we leave the differentiation between data and other measure types for the second step of our approach.

Semantic Annotation using HMMs

For the automated annotation of a PPI description, we use an HMM that assigns semantic tags to chunks of the description. Each word in a description is part of exactly one chunk and each chunk is assigned exactly one tag. We use $\pi =< \varphi_1, \ldots, \varphi_n >$ to denote the chunks of a partitioned description and $\Theta =< \theta_1, \ldots, \theta_n >$ to describe the sequence of tags assigned to the chunks, where $\theta_i \in \Theta$ is the tag that corresponds to the chunk $\varphi_i \in \pi$. Formally, given a word sequence W that constitutes a PPI description, the goal of the HMM is to then find the semantic representation of the meaning Θ that has the maximum *a posteriori* probability $P(\Theta \mid W)$ [275]. This probability is given by the following equation:

$$\hat{\Theta} = \arg\max_{\Theta} P(\Theta \mid W) = \arg\max_{M} P(W \mid \Theta) P(\Theta) \tag{6.1}$$

Equation 6.1 shows that, to compute $\hat{\Theta}$, the HMM combines two separate models: a semantic prior $P(\Theta)$ and a lexicalization model $P(W \mid \Theta)$. We now briefly describe how these models work in the context of our transformation approach. For a more detailed explanation of the technical aspects underlying HMMs, we refer the interested reader to [223].

Semantic prior. The semantic prior $P(\Theta)$ assigns a probability to an underlying semantic structure Θ. Intuitively, it models the probability that a PPI description follows a certain semantic structure, i.e., a sequence of tags from Σ. The model can be represented as a probabilistic finite automaton. The states of this automaton correspond to labels in the set Σ. The transition probabilities between states denote the probabilities that these states follow each other in a semantic structure. These probabilities can be learned by training an HMM on a collection of (partially) annotated PPI descriptions. We further reflect on the training of HMMs at the end of this section.

The semantic prior that results from a training phase can be used to determine the likelihood that PPI descriptions follow a certain semantic structure. As an example, consider the fragment of a semantic prior depicted in Figure 6.4. This figure shows the

semantic prior relevant to the annotation of time measures. From this semantic prior, we can observe that, despite the huge variability that natural language PPI descriptions can use to describe time measures, there is much less diversity in the semantic structure that they follow. For example, the prior shows that if a time measure (in this training set) has an *aggregation function*, then this is the first semantic concept that is described, whereas an optional *group-by property* occurs at the end of a description. Furthermore, the prior shows that a *time measure indicator* (TMI) is followed by a specification of an event that describes both the beginning and end of the measure (TBE) with a probability of 0.61, i.e. $TMI \rightarrow TSI$ has a probability of 0.61. In the other cases, TMI is followed by a *time start indicator* (TSI). By contrast, a TSI is always followed by a *time start event* (TSE) (probability of 1.00), because there are no other semantic structures in which a TSI occurs. This means that a TSI will never be directly followed by, for instance, a time end event or a group-by indicator.

Lexicalization Model. The lexicalization model $P(W \mid \Theta)$ quantifies the probability that a given word sequence W is used to model a certain semantic structure Θ. Intuitively, it quantifies the probability that W is used to a convey a meaning Θ. In particular, it models the transition probabilities between words given a certain context, i.e., a certain semantic concept. Probabilities in the lexicalization model have the form $P(word_n \mid word_{n-1}, context)$, which is the probability of taking a transition from one word to another given a particular context [196]. For example, the probability $P(time \mid throughput, TMI)$ denotes that the word *time* follows the word *throughput* in the context of a TMI (time measure indicator). As for the semantic prior, these probabilities are learned from a (partially) annotated training set. By training a lexicalization model, we can, for instance, learn that the term "*for*" is much more likely to be followed by the word "*each*" in the context of a *group by indicator* (GBI) than in the context of a *time start event* (TSE). Therefore, the HMM will assign a higher likelihood to the possibility that "*for each*" indicates a *group-by property* than that it is part of the description of a start event.

Probabilistic Annotation. Once the HMM has been trained, it can be used to assign tags to a previously unseen PPI description. To find the best tag sequence $\hat{M}$, the optimization problem denoted by Equation 6.1 considers the product of the probabilities from the semantic prior $P(\Theta)$ and lexicalization $P(W \mid \Theta)$ models. The combination of these two models follows intuitively. The probability that a word sequence W corresponds to a certain semantic structure Θ depends on both the likelihood of this semantic structure occurring ($P(\Theta)$) and the likelihood that W is associated with a certain semantic meaning ($P(W \mid \Theta)$). The task of finding the optimal sequence of labels $\hat{M}$ is also referred to as *decoding*. The most common decoding algorithm for HMMs is the *viterbi algorithm*, a dynamic programming algorithm [p.74] [135]. Because the algorithms only affect the computational efficiency of the optimization problem and not the final outcome, we abstract from its details and refer the reader to [95] for a detailed explanation. We previously showed the semantic annotation obtained for PPI2. For the remaining three PPI descriptions considered in Section 6.2, we obtain the following semantic annotations:

$PPI1 : \pi_1 \backslash M_1 =$ `number\CMI of accepted orders \CE`

$PPI4 : \pi_4 \backslash M_4$ = `the percentage\FMI of rejected orders \FNE`
$PPI6 : \pi_6 \backslash M_6$ = `the total\AGR order amount\CE per\GBI customer\GBC`

We take these semantic annotations as input for the next step of our approach: the domain value resolution step.

6.3.2 Domain Value Resolution

In the second step of our approach, we obtain a PPI definition by filling the slots of a PPI template with values from the appropriate domains. The semantic annotations obtained in the previous step tell us which chunks of text correspond to which slots in a PPI template. For example, from the annotation of *PPI*2, we know that the chunk φ_6 : "*completion of an order*" corresponds to the *end event* slot in a template. We illustrate the full connection between semantic annotation and template on the left-hand side of Table 6.6. However, this only presents an intermediate result of our approach. To be able to actually measure the value of this PPI definition, links have to be established to the events that occur during the execution of the process. For example, to compute a value for PPI2 we need to determine that the chunk φ_6 corresponds to the "*order handling completed*" process model event. Therefore, in this second step of our approach, we fill the template slots with values from the appropriate domains, as depicted in Table 6.6.

Table 6.6: Domain value resolution for PPI2

(a) Before resolution

Slot	**Value**
Measure type	*time*
Aggregation	*"average"*
Start event	*"receipt"*
End event	*"completion of an order"*
Group-by	—

(b) After resolution

Slot	**Value**
Measure type	*time*
Aggregation	*average*
Start event	*Order received*
End event	*Order handling completed*
Group-by	—

To perform this *domain value resolution* for aggregation functions and group-by properties, we consider the *semantic similarity* between a chunk φ and the respective domain values. For process concepts, we combine semantic similarity with *constraints*, which we impose based on the semantics of the various measure types. We next describe these two aspects in detail.

Semantic Similarity

We use semantic similarity to determine for a given chunk $\varphi \in \pi$ which value in its associated domain D_φ has the most similar meaning. We consider *semantic* rather than (just) *syntactic* similarity for this task because it allows us to deal with semantically related terms, such as synonyms. For example, by considering semantic similarity, we can learn that the chunk "*transport an order*" is most closely related to the "*product*

shipping" activity. The computation of semantic similarity is achieved in a manner that is largely equal to the one applied in the context of the alignment of process models to textual descriptions, as described in Section 3.2.2. More details on the employed normalization techniques and similarity measures are presented in Section 2.4.3.

Normalization. To determine the semantic similarity, we first apply a *normalization* step on a chunk φ and on the domain values D_φ. We perform three tasks in the normalization step. First, we split the textual contents of a chunk or a domain value into individual terms. Second, the function filters out stop words like "*the*", "*an*", and "*from*", which do not contribute to the similarity computation. Finally, the normalization step *lemmatizes* the remaining terms by transformaring all remaining terms into their grammatical base form. For instance "*is*" and "*been*" are both transformed into "*be*". To implement these steps we use the *Stanford Parser* and the associated toolkit [180]. As an illustration, the normalization step yields the following token set for the chunk φ ="*completion of an order*": ω_φ = {*"completion", "order"*}.

Semantic computation. Given two token sets ω_φ and ω_d, we quantify the semantic similarity using the similarity measure proposed by Mihalcea et al. [192], and described in detail in Section 2.4.3. This measure combines the semantic similarity of individual terms with word specificity scores. Equation 6.2 formalizes the measure for a chunk φ and a domain value d.

$$sim(\omega_\varphi, \omega_d) = \frac{\sum\limits_{t \in \omega_\varphi} [\max\limits_{t' \in \omega_d} sim_{so}(t, t')] \times idf(t)}{2 \times \sum\limits_{t \in \omega_\varphi} idf(t)} + \frac{\sum\limits_{t \in \omega_d} [\max\limits_{t' \in \omega_\varphi} sim_{so}(t, t')] \times idf(t)}{2 \times \sum\limits_{t \in \omega_d} idf(t)} \tag{6.2}$$

In this equation, $sim_{so}(t, t')$ captures the second-order similarity between two terms t and t' computed according to the approach by Kolb et al. [145].

The calculation of semantic similarity suffices for the aggregation function and group-by property slots, since the appropriate values for these slots solely depend on the textual similarity between chunk and domain value. By contrast, for the filling of process concept slots, we also have to consider constraints related to the semantics of a measure type.

Constraints on event alignment

When filling the slots of a PPI template, it is important that the end result makes sense from a semantic perspective. For instance, the start event of a time measure should always occur before its end event. Otherwise, the resulting metric will be semantically invalid. For this reason, we impose certain constraints on the inter-relations of the process model elements assigned to process concept slots. To achieve this, we formulate the slot-filling for event slots as an alignment problem. In the context of this work, an alignment $\sim$ consists of a number of pair-wise correspondences, each between a tagged chunk from a PPI description $\varphi \in \pi_e$ and an element of a process model $m \in M$. We denote such a correspondence as $\varphi \sim m$. We impose constraints on an alignment relation

$\sim$ through a constraint function Γ. In particular, we use Γ to impose three semantic constraints. The first constraint applies to time measures, the other two relate to fraction measures. Because the constraints are applicable to the inter-relations that exist between the correspondences assigned to different slots, we do not define constraints for count measures, which only have a single slot to be filled through the correspondences.

Time measures. For time measures, we impose an ordering constraint on the events and activities associated with the *start event* and *end event* slots. From a semantic viewpoint, a PPI in which the start event occurs after the end event does not make sense. Therefore, we ensure that the model element m_i aligned to the start event slot π_i occurs in the process model before the element m_j associated with the end event slot π_j. We achieve this by considering the flow relation F that exists between the nodes in a process model. Given a time-measure alignment $\sim = \{\pi_i \sim m_i, \pi_j \sim m_j\}$, we impose the constraint that there must be a path from m_i to m_j in the process model M_D, based on the flow relation F. Given the set of nodes N and two nodes $x, y \in N$, a *path* exists if there exists a sequence of nodes $n_1, \ldots, n_k \in N$ with $x = n_1$ and $y = n_k$ such that for all $i \in 1, \ldots, k-1$ holds: $(n_i, n_{i+1}) \in F$ [162, p.14]. In this way, the constraint captures that the aligned *start* element (m_i) occurs before the *end* element (m_j).

Non-trivial fractions. For fraction measures, we ensure that the obtained result is not a trivial measure, i.e., that the defined measure does not always yield 1.0. For this reason, we ensure that the numerator slot π_n and denominator slot π_d are not aligned to the same model element. Therefore, given an alignment $\sim = \{\pi_n \sim m_n, \pi_d \sim m_d\}$, it must hold that $m_n \neq m_d$.

Computable fractions. Lastly, we ensure that a fraction measure can actually be computed, instead of resulting in a divide-by-zero error. Therefore, we must ensure that the denominator slot is filled, even for those cases where a PPI description does not explicitly mention a denominator. This can be seen for PPI4, which simply refers to "*the percentage of rejected orders*," without specifying a denominator. In these cases, we align the slot π_d by default to a process concept that represents a more coarse-granular version of the process concept aligned to the numerator slot π_n. There are two possibilities for such default numerators. If π_n is aligned to a data object with a specific status, e.g., "*order [updated]*", we align π_d to the data object without the status specification, e.g., the "*order*" data object, if any. As such, we obtain a fraction measure that divides the number orders that have been *updated* by the *total* number of orders. For all other alignments of π_n, we align the denominator by default to the start event(s) of a process. In this way, the denominator corresponds to the total number of process instances. For example, PPI4 will then divide the number of *rejected orders* by the total number of *received orders*.

We combine these constraints with semantic similarity scores. Therefore, we set out to obtain an alignment that has the highest possible sum of semantic similarity scores for the correspondences in $\sim$, as long as $\sim$ abides to the alignment constraints Γ. The alignments obtained in this way, in combination with the domain value resolution

of the other slot types, provide the final result of our transformation approach: a filled-in PPI template.

6.3.3 Operationalization and Extensibility

The transformation approach described in this section can be regarded as an extensible framework for the transformation of natural language PPI descriptions into measurable indicators. In this chapter, we have so far described an instantiation of this framework that covers the most common types of PPIs according to insights obtained from practice [233, 235, 236]. This approach can thus be readily used for this set of PPIs. However, should users desire to extend the presented approach, for example, by incorporating different measure types, this can be achieved in a straightforward manner.

An extension requires the specification of new (or adapted) PPI templates and a retraining of the HMM. By designing new PPI templates in accordance to a structured PPI notation, such as the PPINOT metamodel [233], it can be ensured that also the values of these newly covered measure types can be computed in an automated manner. Once a new template has been defined, the HMM must be adapted to be able to generate PPI definitions in accordance to this template. To this end, tags must first be defined to cover newly introduced semantic concepts. As previously explained in Section 6.3.1, two tags must be defined for each new semantic concept: (i) a tag for the semantic concept itself and (ii) a tag for an indicator of the concept. Once the adapted tag set has been defined, several annotated PPI descriptions are required to retrain the HMM to incorporate the newly defined measure type.

The amount of new training data required depends on the (expected) variety of linguistic patterns that can be used to describe the measures. As the evaluation in the next section illustrates for fraction measures, sometimes only a handful of annotated PPI descriptions suffices. Two approaches are possible to train the HMM. In case an available training set $\mathcal{D}_T$ is fully annotated, i.e., all PPI descriptions in the training set are annotated, the models $P(\Theta)$ and $P(W \mid \Theta)$ can be learned by simply counting [275]. For instance, we can learn the probability that a TEE tag follows a TMI tag by taking the number of descriptions in $\mathcal{T}$ that contains the subsequence $< TMI, TEE >$ divided by the total number of descriptions in $\mathcal{T}$ containing $< TMI >$. In case $\mathcal{D}_T$ is only partially annotated, estimation algorithms can be used to compute the probability distributions. The most commonly used are *forward-backward* algorithms, such as the Baum-Welch method (see e.g., [223]). Due to the ability of these methods to work with partially annotated data, it is possible to add large amounts of additional training data, without the need to annotate them all.

Through similar adaptations, the approach can be extended in other ways. For example, new aggregation functions can be included or the HMM can be trained to work on specific application domains.

6.4 Evaluation

To demonstrate the capabilities of our transformation approach, we conducted a quantitative evaluation by comparing automatically generated, structured PPI definitions to

Table 6.7: Overview of the test collection

Source	P	PPIs	Count	Time	Data	Frac.	Aggr.	Group-by
Industry	10	47	25	20	1	7	12	8
SCOR	3	76	27	21	24	4	39	1
Total	13	129	52	41	25	11	51	9

a manually created *gold standard*. In this way, we assessed how well the automated approach approximates manual transformations of PPI descriptions. Therefore, it assesses the usefulness of our approach. To achieve this, we used a test collection consisting of 129 PPIs obtained from industry. Both the test collection and prototypical implementation used in this evaluation are publicly available.[2]

Section 6.4.1 provides detailed information on the used test collection. Section 6.4.2 describes the way in which we conducted the evaluation. Lastly, Section 6.4.3 presents and discusses the evaluation results.

6.4.1 Test Collection

To evaluate our approach, we use a collection of process models and accompanying natural language PPI descriptions from practice. Part of the test collection consists of an industrial data set stemming from prior research on the formalization of PPI definitions and service level agreements [234, 235]. In particular, this set consists of processes from three different organizations: (i) a healthcare institute, (ii) a university, and (iii) a telecommunications provider. This data collection was originally provided in Spanish, but translated to English in collaboration with the respective industrial partners. We augmented the industrial data set with a number of process models and PPIs from the Supply Chain Operations Reference (SCOR) reference framework. From this framework, we selected processes with a high number of associated performance indicators per process and a considerable complexity of the associated process model. The selected processes cover different aspects of the logistics domain: procurement, production, and delivery. The PPI descriptions obtained in this manner were not altered with respect to their contents. The set of descriptions varies greatly with regard to their language and structure.

We had to exclude nine PPI descriptions from the original data collection. These PPI descriptions could not be manually transformed into a structured notation based on the contents of the available process models, i.e., their computation required information that is not conveyed in the model. The resulting test collection consists of 12 process models with a total of 129 associated PPIs. Table 6.7 presents an overview of the further characteristics of the collection. This table shows the number of PPIs per type, and the number of PPIs that are associated with an aggregation function (aggr.) or group-by property.

[2]Download from: www.hanvanderaa.com/downloads/ppi-transformation

6.4.2 Setup

To conduct the evaluation, we implemented the presented transformation approach in the form of a Java prototype. The prototype uses the Stanford CoreNLP toolkit [180] for the tokenization of PPI descriptions and the semantic similarity implementation DISCO [145] to calculate second order similarity scores. We use this prototype to generate structured definitions for the PPI descriptions included in the test collection.

We compare the generated definitions to a manually created *gold standard*. For the creation of this gold standard, we partially built on information directly provided by the SCOR framework. This information included the alignment of the associated PPIs to process model elements. For the remaining alignments and PPIs, we used a similar procedure for the establishment of a gold standard as used in Chapter 3. Specifically, we let two researchers independently establish a gold standard by manually transforming the PPI descriptions. The creation of the gold standard yielded an inter-annotator agreement of 0.97. The five discrepancies between the two gold standards were resolved through a discussion.

To perform the comparison between the generated definitions and the gold standard, we computed the same metrics as used in the evaluation of Chapter 3. In particular, we computed *precision*, *recall*, and the F_1-score, as formalized by Equations 6.3, 6.4, and 6.5.

$$pre = \frac{|\mathcal{A} \cap \mathcal{G}|}{|\mathcal{A}|} \quad (6.3) \qquad rec = \frac{|\mathcal{A} \cap \mathcal{G}|}{|\mathcal{G}|} \quad (6.4) \qquad F_1 = \frac{2 * pre * rec}{pre + rec} \quad (6.5)$$

In these equations, $\mathcal{A}$ denotes the set of slots filled by our approach and $\mathcal{G}$ for the slots filled in the gold standard.[3] In the context of this evaluation, *precision* reflects the fraction of slots that our automated approach filled correctly according to the gold standard. *Recall* represents the fraction of slots filled in the gold standard that were also correctly filled by our approach.

To compute the evaluation results, we train our approach on a part of the PPI collection, referred to as the *training set*, and apply it on the remainder of the data collection, the *test set*. To avoid any bias in the result due to sampling, we perform a repeated *k-fold cross-validation*. In a k-fold cross-validation, a data set $\mathcal{D}$ is randomly split into k mutually exclusive subsets (i.e., folds), $\mathcal{D}_1, \ldots, \mathcal{D}_k$ of approximately equal size [144]. In each experiment run, our approach is then tested k times. Each time $t \in \{1, 2, \ldots, k\}$ we trained the HMM on $\mathcal{D} \setminus \mathcal{D}_t$ and tested it on $\mathcal{D}_t$. By repeating the cross-validation with different random splits of the data set, we can learn how much the results are affected by a particular partitioning of the data collection.

6.4.3 Results

In this section, we present the results of our evaluation experiments. We first provide an overview of the quantitative results and, afterwards, discuss the challenges that our

[3] Note that we exclude *null* values assigned to the optional aggregation function and group-by property slots from consideration.

approach faces in the context of a post-hoc analysis.

Overview

We used our prototype to conduct a k-fold cross validation with $k = 10$, which we repeated 30 times. We performed this cross validation for the *industry* and *SCOR* data collections separately, as well as for the combined collection. Table 6.8 summarizes these results for our approach and the baseline. It shows that our transformation approach performs well, obtaining an average F_1-score of 0.85. The approach achieves an average precision of 0.89, ranging between 0.78 for group-by properties and 0.93 for aggregation functions. The average recall obtained by the approach is 0.82, ranging from 0.72 for the alignment of process concepts and 0.96 for the identification of aggregation functions. From the low observed standard deviations (0.01 for the entire set of slots), we can learn that the performance of the approach is stable in the context of this data set. Furthermore, we observe that the alignment of process concepts differs considerably between the two data collections. For the other slot types, the average performance is comparable. Lastly, it is interesting to consider that our approach achieved a fully correct transformation for 67% of the PPIs in the data collection. Furthermore, the transformation approach returns at most one incorrect slot filler for a total of 86% of the PPIs.

Table 6.8: Evaluation results

Data source	Slot type	Prec.	Rec.	F_1
Industry	Measure type	0.90	0.90	0.90
	Process concepts	0.67	0.63	0.65
	Aggregators	0.92	1.00	0.97
	Group-by	0.74	0.74	0.74
	Total	0.78	0.76	0.77
SCOR	Measure type	0.94	0.94	0.94
	Process concepts	1.00	0.80	0.89
	Aggregators	1.00	1.00	1.00
	Group-by	–	–	–
	Total	0.98	0.88	0.93
Full collection	Measure type	0.92	0.92	0.92
	Process concepts	0.87	0.72	0.79
	Aggregators	0.93	0.96	0.94
	Group-by	0.78	0.78	0.78
	Total	0.89	0.82	0.85

Post-hoc Analysis

The quantitative evaluation shows that our transformation approach achieves a high result accuracy. A post-hoc analysis of these results reveals that the approach faces two main types of challenges. One challenge corresponds to the semantic annotation step and the other to the domain value resolution step of our approach.

The usage of HMMs for *semantic annotation* enables our approach to deal with a broad variety of linguistic patterns, even if a PPI description contains previously unseen terms or syntactic constructs. However, the parser can produce incorrect annotations if unseen terms play a prominent role in a PPI description. For instance, the collection of PPI descriptions from the SCOR framework contains a single PPI description that uses the term "*leadtime*" in a time measure. Because this is a unique occurrence in the used data collection, the HMM parser does not recognize this important logistic term. Therefore, it incorrectly annotates this PPI description as a count measure, which results in an incorrect measure type prediction. Furthermore, due to the semantic difference between count and time measures, it also results in an incorrect alignment of process concepts. A count measure refers to only a single process event, whereas a time measure requires both a start and an end event. Therefore, the misclassification will also lead to at least one event not being correctly aligned. Still, such cases are rare and their occurrences can be mitigated by extending the data collection used to train the HMM.

The *domain value resolution* step of our approach has to deal with the highly complex task of identifying the event that a chunk of text describes. The main challenge here is that certain correspondences depend on context-specific information for their identification. As an illustration, consider a PPI description from the industrial collection: "*the elapsed time between the technician arrival to headquarters and the closure of the intervention*". The start event of this description corresponds to the chunk "*the technician arrival to headquarters*", as correctly identified in the parsing step. However, identifying the correct process concept for this chunk is far from trivial. The process model accompanying this PPI does not contain any activity or event that describes an "*arrival*" or "*headquarters*". Instead, the event corresponds to the start of an activity labeled "*Perform field intervention*". To identify this correspondence correctly, background knowledge is required to establish the connection between the "*arrival of a technician*" and the start of a "*field intervention*". Our automated approach does not take such domain knowledge into account and, in some cases, identifies incorrect alignments between the PPI description and a process model. From the results described in Table 6.8, it becomes clear that such cases are mostly present in the industrial data collection.

Despite these challenges, the evaluation results demonstrate that the PPI definitions generated by our automated approach closely approximate PPI definitions manually created by experts. Therefore, we can conclude that our automated approach presents an efficient alternative for an otherwise highly tedious, manual task.

6.5 Limitations

The quantitative evaluation demonstrates that our approach achieves promising results in practical settings. However, these results need to be considered against the background of certain limitations. In particular, we identify limitations related to the approach itself and limitations related to its evaluation.

We identify two main limitations regarding the transformation approach First, it provides an automated alternative to a highly complex task. As a result, the PPI definitions that the approach generates are accurate, but not one hundred percent perfect. If desired, inaccuracies can be manually resolved by experts. This arguably requires less manual effort than it takes to manually define all PPIs from scratch. As such, our transformation approach allows users to trade-off between invested time and effort versus results quality. Second, it has to be considered that our transformation currently is not able to detect when it is dealing with a PPI description that cannot be transformed based on the available information. Such a situation can occur if the PPI describes a measure type that is currently not included in the set of templates or if the PPI description requires information that is not captured in a process model. The latter case was observed for the nine PPI descriptions that we had to exclude in the evaluation.

The presented quantitative evaluation results are bound to the specifics of the employed data collection. For instance, the included templates do not cover all imaginable PPIs, but rather the ones most observed in our empirical studies. As stated earlier, the impact of this limitation is diminished by the extensibility of our approach. Second, within each measure type, the data collection presents a sample of the natural language patterns that can be used to specify measures of this type. It is possible that in other organizations measures are described in different ways. These factors should be taken into account when interpreting the specified results. Still, we aimed to compose a data collection that is as heterogeneous as possible by obtaining data from various sources. Therefore, we are confident that our evaluation indeed shows a realistic picture of the performance of our approach in practice.

6.6 Related Work

The work presented in this chapter particularly relates to two main streams of research: (i) structured and measurable notations of PPIs and (ii) information extraction from natural language.

Performance measurement is an active research field in management science, which has gained interest in both academia and business [218]. Much work has been performed on the identification and classification of Key Performance Indicators in general settings [137] and those relevant for specific domains such as logistics, production, and supply chains (cf. [52, 61, 150, 265]). Within the context of *Process Performance Measurement*, great effort has been put on the formalization of PPI definitions. This has resulted in a number of notations and frameworks for the description and monitoring of PPIs [67, 112, 147, 198, 208, 218, 235, 243, 286]. The main differences among them are found in their expressiveness, i.e. the different types of PPIs that can be defined, and their features to directly support monitoring. Although the templates used in our

approach are inspired by the PPINOT metamodel, our approach is generally applicable. Specifically, the components of the templates can also be mapped to concepts in other frameworks, such as [198, 286].

As shown in Section 6.3, we approached the task of transforming natural language PPI descriptions as a so-called template-filling problem. Template filling, which is also referred to as slot filling [262] or semantic-based understanding [274], has been extensively studied and applied in a variety of contexts. A major application area for these techniques is *spoken language understanding*, where information is extracted from unstructured natural language text in the context of a dialog system [114]. The viterbi algorithm, which we employed for the extraction of information from the descriptions, also been applied in a broad variety of application scenarios. These include machine translation [54], part-of-speech tagging [154], and spoken language processing [197]. To achieve this, many approaches employ probabilistic models, such as (variations on) HMMs [196, 275].

6.7 Summary

This chapter presented an approach for the automated transformation and alignment of natural language PPI descriptions. Our approach takes as input an unstructured PPI description and produces a structured, template-based notation of a PPI. Because our template-based notation builds on the PPINOT metamodel, the values of the obtained PPI definition can be computed automatically. This makes the PPIs suitable for automated process performance monitoring. To achieve this, the approach builds on HMMs as linguistic parsers to identify the parts of a PPI description that correspond to slots in a PPI template. Then, we use semantic similarity measures and semantic constraints to fill the slots with the appropriate values belonging to particular domains. We evaluated the performance of our approach with a set of 129 real-world PPI descriptions and accompanying process models obtained from various industrial sources. The evaluation revealed that the structured PPI definitions generated by our approach are a good approximation of those created manually by experts. In particular, the approach achieves precision and recall values of, respectively, 0.89 and 0.82. Therefore, our approach represents a viable, automated alternative to an otherwise highly laborious and time-intensive, manual task. This enables organizations to more efficiently monitor the performance of their business processes and continuously adapt their monitoring activities to changing business needs.

7 Process Model Matching using Event-Log Information

Alignments between processes provide an important basis for a variety of application scenarios and techniques. These alignments are, among others, used for the detection of differences between models [156], the harmonization of process model variants [157], process querying [131], and the propagation of process changes [280]. The accuracy and, therefore, usefulness of such techniques is highly dependent on the correctness and completeness of the alignments that are established by process model matching techniques. However, despite the existence of a plethora of matching techniques, it has been shown that their results leave room for improvement [58]. A possible cause for this is that existing process model matching techniques focus exclusively on information related to the *specification* of processes, typically by just considering the information contained in process models themselves. Therefore, they ignore information that relates to the actual *execution* of the processes, as captured in event logs. These logs provide valuable information on data attributes, event durations, and other aspects specifically associated with the enactment of a process.

In this chapter, we present techniques that exploit event-log information for process model matching. Section 7.1 illustrates the usefulness of event-log information in a matching context. Then, Section 7.2 presents six event log-based process model matching techniques. The quantitative evaluation in Section 7.3 considers the performance of the individual matching techniques, as well as their performance in the form of a matching ensemble. We discuss limitations of our matching techniques and the evaluation in Section 7.4. Afterwards, we consider related work in Section 7.5 and summarize the chapter in Section 7.6.

7.1 Problem Illustration

To illustrate the usefulness of event-log information for process model matching, consider two process models, M_1 and M_2, which depict two (simplified) processes to han-

dle loan applications. Also, consider their respective sets of activities A_1 and A_2. Figure 7.1 illustrates these models, with M_1 at the top and M_2 at the bottom. The figure also highlights their correspondences, i.e., the activities that represent similar behavior.

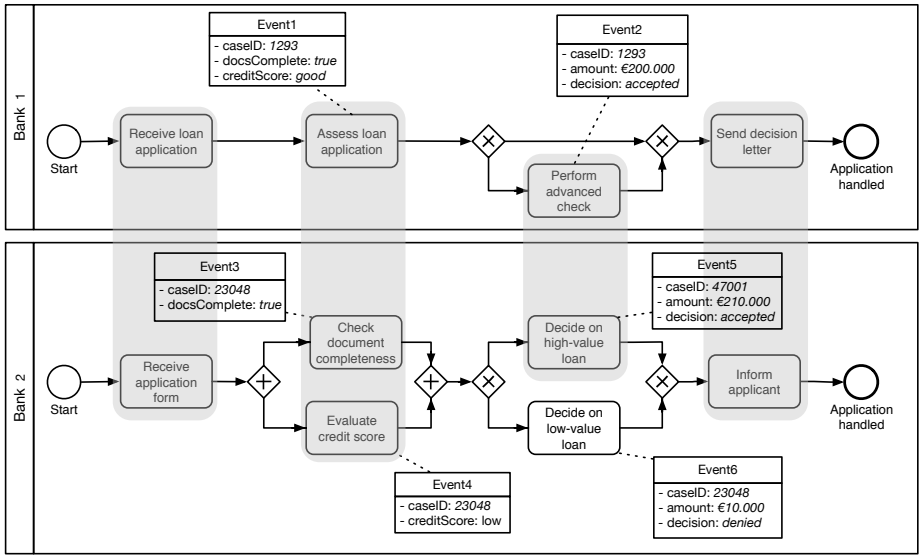


Figure 7.1: Two process models and their correspondences

In process model matching, we wish to automatically identify these correspondences between A_1 and A_2. By analyzing the labels of activities, some correspondences can be identified in a straightforward manner, such as the correspondence between "*receive loan application*" and "*receive application form*". However, the label-based identification of other correspondences is not as straightforward, if at all possible. Consider the "*assess loan application*" activity in M_1 and the activities "*check document completeness*" and "*evaluate credit score*" from M_2. For their correspondences, there is no obvious syntactic or semantic relation between the contents of their labels. This makes it difficult to recognize their similarity based on textual analysis. However, the events associated with these activities do provide valuable information about their similarity. *Event1* in M_1 includes attributes that describe the completeness of the filed documents and the credit score. The events *Event3* and *Event4* in M_2 are each also associated with one of these attributes. This similarity between event attributes provides a strong indication of an existing relation between the activities, which could not be derived without considering information beyond the contents of the process model.

In other cases, the names of attributes associated with events are useful but, by themselves, do not suffice to distinguish among potential correspondences. For example, the "*decide on low-value loan*" (a_i) and "*decide on high-value loan*" (a_j) activities from model M_2 both have events that contain an `amount` attribute. Therefore, the attribute names are not sufficient to determine which of these corresponds to the "*perform advanced check*" activity (a_k) in M_1. However, by analyzing the values associated with these attributes throughout an event log, this could be achieved. For instance, if events

corresponding to a_j and a_k are always associated with amounts above €200.000, while a_i always has a lower amount, the correspondence between a_j and a_k can be asserted.

In the next section, we describe how matching approaches take these and other notions of similarity between events into account.

7.2 Event Log-Based Matching

This section describes how information contained in event logs can be utilized for process model matching. In particular, we introduce six conceptual notions of similarity that each characterize a different aspect of similarity derivable from event-log information. Together, these notions provide a complete coverage of the prominent types of information contained in event logs: *ordering*, *frequencies*, *timestamps*, and *data attributes*. We consider one similarity notion for each of the first three types. Due to the versatility of the data perspective in event logs, we consider three different similarity notions related to the *data attributes* associated with events. To illustrate the operationalization of these similarity notions, we introduce a corresponding similarity *measure* for each of them. These measures can be used as FLMs, where the similarity scores obtained by the measures are used to populate a similarity matrix. Each measure produces a value in the range [0, 1], where a 1.0 score indicates perfect similarity between two event classes according to the particular similarity notion. The measures that we introduce can be applied without imposing any assumptions on the data. Furthermore, we also reflect on alternative measures that typically depend on certain assumptions or are computationally more complex.

The input for the matching techniques presented in this section are two process models and two corresponding event logs, i.e., two model-log pairs (M_1, L_1) and (M_2, L_2). We assume here that, given a process model M with activity set A and a corresponding event log L with the set of event classes $\mathcal{E}(L)$, each activity $a \in A$ corresponds to exactly one event class $E \in \mathcal{E}(L)$ and vice versa. Therefore, without loss of generality, we here refer to activities and event classes interchangeably. This means that we can, for example, refer to the similarity between the events of the event classes "*Asses loan application*" and "*Check document completeness*" from the running example.

Finally, it is important to note that the introduced matching techniques do not depend on the availability of process models. The matchers can also be used for scenarios in which the goal is to directly establish alignments between the event classes from different event logs. However, for clarity, we focus on process model matching in the remainder of this chapter, since this use case is more generally recognized.

7.2.1 Positional Similarity

Positional similarity focuses on the order in which events occur. The underlying idea of positional similarity is that if two event classes occur at similar stages in the execution of a process they are more likely to be similar. For example, the final activity in M_1, "*Send decision letter*", is more likely to be similar to "*Inform applicant*", which occurs at the end of M_2, than to "*Receive application form*", which occurs at the start of that process.

Similarity measure. We define a *relative position* measure RP that quantifies the average position at which events of a certain event class occur in traces. To account for varying trace sizes, we consider the position of an event relative to the length of a trace. Specifically, we use RP_e to denote the relative position of an event e in a trace, given as $\frac{j}{|\sigma|-1}$. Here we use $|\sigma|$ to denote the length of a trace and j denotes the index of e in the trace, where $j \in [0, n]$. This gives us, for example, for $\sigma = \langle a, b, c \rangle$, $RP_a = 0/2$, $RP_b = 1/2$ and $RP_c = 2/2$. Equation 7.1 provides the RP measure, in which we use RP_E to denote the average RP_e over all events $e \in \mathscr{E}$ with event class E.

$$RP(E_1, E_2) = 1 - |RP_{E_1} - RP_{E_2}| \tag{7.1}$$

Alternatives. Comparing the position of events in traces provides a basic measure for structural or behavioral similarity. Techniques for process model matching exist that use more advanced similarity measures, for example those based on *behavioral relations* [81]. Such measures can also be adapted to work on graph structures or behavioral relations derived from event-log information. Such derivation is done by techniques that automatically derive process models from event logs, i.e., so-called *process discovery* techniques [19].

7.2.2 Occurrence Similarity

The frequency with which events of a certain event class occur can provide useful information regarding its similarity to other event classes. For example, if two event classes E_1 and E_2 each occur only rarely in an event log, then E_1 and E_2 both correspond to some exceptional action, hinting at their potential similarity. In the running example, for instance, it can be expected that the majority of loan requests will be for comparably low amounts. This means that occurrences of the "*Perform advanced check*" and "*Decide on high-value loan*" events are relatively rare. Therefore, by comparing the frequencies with which these activities occur, the similarity between the activities can be identified. Furthermore, the consideration of frequencies can also be used to identify similarity between event classes that only occur once per trace and those that occur multiple times (i.e., rework activities).

Similarity measure. We define a measure $FREQ$ which compares the average number of occurrences of event classes per trace. We let $FREQ_E$ denote this average for an event class E, and use Equation 7.2 to formalize $FREQ$. Because it is possible that $FREQ_E > 1$, this measure is normalized to ensure a confidence score in $[0, 1]$.

$$FREQ(E_1, E_2) = 1 - |\frac{|FREQ_{E_1} - FREQ_{E_2}|}{\max(FREQ_{E_1}, FREQ_{E_2})}| \tag{7.2}$$

Alternatives. An alternative way to evaluate occurrence similarity is to consider the fraction of traces in which an event class occurs, rather than the average number of occurrences per trace. To illustrate the difference to the $FREQ$ metric, consider an event class E_3 that occurs in only 50% of the traces, but in those traces it occurs twice.

In this case, $FREQ_{E_3}$ equals 1.0 ($0.5 \cdot 2$), which would be the same for an event class E_4 that occurs once in every trace. Nevertheless, it is clear that the occurrences of E_3 and E_4 are very different. Furthermore, it is possible to use statistical tests for the comparison of frequencies, rather than compare averages. We reflect on these tests in the following.

7.2.3 Duration Similarity

The time it takes to execute activities can serve as an indicator that provides useful hints regarding their similarity. In our running example, it can be expected that activities that check loans with high amounts are extensive and, therefore, consume a significant amount of time. By contrast, the communication of the decision to the applicant can very well be automated, resulting in negligible activity durations. This means that, by considering the durations of activities, we can learn that the "*Perform advanced check*" activity is more similar to "*Decide on high-value loan*" than to "*Inform applicant*."

Similarity measure. A straightforward similarity measure for durations can be obtained by comparing the average durations of an event class in a log. The duration of a single event can be determined in two ways. In an ideal scenario, events are associated with transaction types (through the $\#_{\text{trans}}(e)$ attribute) that indicate whether an event denotes the start or end of an activity. In these cases, durations are determined by computing the difference between the timestamp of an event with the transaction type *start* and a corresponding event with the transaction type *complete*. If such transaction types are not available, an approximation can be obtained by taking the difference between the timestamp of an event and the timestamp of its preceeding event in the trace. Then, using DUR_E to denote the average duration of events of class E, Equation 7.3 provides a normalized measure that returns a score in $[0, 1]$.

$$DUR(E_1, E_2) = 1 - |\frac{|DUR_{E_1} - DUR_{E_2|}}{\max(DUR_{E_1}, DUR_{E_2})}| \tag{7.3}$$

Alternatives. Durations can vary significantly among occurrences of the same event class. Therefore, an alternative is to use statistical tests, e.g., the *t-test* or *Kolmogorov-Smirnov test* [227] to compare the statistical distribution of the durations for two event classes. By performing such tests, we do not only consider the averages of event durations, but also their variability. An important downside of such tests is that certain conditions have to be met in order to be able to apply them. For example, a *t-test* requires data to be normally distributed. Another consideration to take into account is the cost of computing the similarity. For instance, the Kolmogorov-Smirnov test is computationally intensive, which can negatively affect its applicability to matching problems.

7.2.4 Attribute name similarity

The names of attributes provide insights into the data values used or created by events. These attribute names can be useful similarity indicators to identify correspondences.

Their importance is demonstrated in the motivational scenario, where event classes that produce the same attributes (e.g., the `docsComplete` attribute) are recognized to be similar to each other.

Similarity measure. We define an attribute name similarity measure $ATTR$, which determines the level of overlap in attribute names among the attribute sets associated with two event classes. To quantify this overlap, we employ the well-known *cosine similarity* measure from the field of information retrieval [244]. The cosine similarity can be used to compute the similarity between two vectors. In the case of the $ATTR$ measure, these vectors represent the attributes associated with events of a certain class.

To illustrate this vectorization, consider the attributes associated with the "*Assess loan application*" (E) and "*Check document completeness*" (E') event classes from the running example. Using a 2D vector to represent the `docComplete` and `creditScore` attributes, we obtain the following vectors: $\mathbf{ATTR_E} = (1, 1)$ and $\mathbf{ATTR_{E'}} = (1, 0)$. Furthermore, we can add weights to these vectors by considering the commonality of attribute names in the context of the context. We achieve this by computing the IDF scores, introduced in Chapter 2, for each attribute name. Given two vectors $\mathbf{ATTR_{E_1}}$ and $\mathbf{ATTR_{E_2}}$, the cosine similarity between them is computed by Equation 7.4.

$$ATTR(E_1, E_2) = \frac{\mathbf{ATTR_{E_1}} \cdot \mathbf{ATTR_{E_2}}}{\| \mathbf{ATTR_{E_1}} \| \| \mathbf{ATTR_{E_2}} \|} \tag{7.4}$$

Alternatives. The ATTR measure only considers overlap in attributes with identical names. However, syntactic and semantic similarity measures can also be used to quantify the similarity between non-identical attribute names [102]. Section 2.4.3 presents an overview of such measures. Syntactic similarity measures, such as the Levenshtein distance [291], can be used to recognize that "*name*" and "*lastName*" describe similar attributes. Semantic similarity measures, such as WordNet-based similarity [56], can be used to recognize attribute names with similar meanings, such as "*amount*" and "*value*".

7.2.5 Attribute Value Similarity

The values of an attribute, associated with events of a given event class may provide insights into similarity beyond attribute name similarity. We can identify two general scenarios for this. First, an analysis of attribute values can be useful to determine similarity in the context of opaque or unrelated attribute names. For instance, it is difficult to relate two attributes `month` and `m` based on their labels. By contrast, if both attributes are associated with numeric values in the range 1–12 (or even month names), their similarity becomes more apparent. Second, attribute value similarity can be used to disambiguate event classes that use the same attributes. The motivational scenario provides an example of this. The event classes "*decide on high-value loan*" and "*decide on low-value loan*" in M_2 both consider an `amount` attribute. Events of the former class are associated with a higher range of values than events of the latter. Therefore, by considering the attribute values, we can identify that the former event

class is more likely to correspond to "*perform advanced check*" in M_1, which similarly occurs only for loan requests with a high amount.

Similarity measure. To quantify attribute value similarity for two *individual* attributes, we rely on techniques from the research area of schema matching [89] where *content-based* matching, (direct comparison of sets of attribute values) is combined with *constraint-based* matching. The latter aims to extract constraints from a set of values, such as upper and lower bounds for numerical values. For brevity, we refrain from presenting explicitly the equations used in this method. After considering the similarity of individual attributes, the similarity values obtained in this manner can be used as weights to calculate the cosine similarity between two attribute sets. Using VAL_E to refer to the value sets of the attributes of an event class E, we compute the VAL measure as given by Equation 7.5.

$$VAL(E_1, E_2) = \frac{\mathbf{VAL_{E_1}} \cdot \mathbf{VAL_{E_2}}}{\| \mathbf{VAL_{E_1}} \| \| \mathbf{VAL_{E_2}} \|} \tag{7.5}$$

Alternatives. Various alternative techniques exist to determine the similarity between two individual attributes, including identifying and comparing data types such as zip codes or geographical names, putting constraints on values, and identifying data patterns and distributions. The interested reader is referred to [224] for an overview of such techniques applied in a schema matching context.

7.2.6 Prerequisites Similarity

The input data used by an event can be an important indicator of event class similarity. Intuitively, this builds on the idea that the more similar the data that is used by event classes, the more similar their purpose. For example, in the motivational scenario, events of the classes "*send decision letter*" in M_1 and "*inform applicant*" in M_2 are the only ones to occur *after* an event has produced a value for the `decision` attribute.

A challenge here is that event logs typically do not have an explicit notion of *input data*. For instance, the the seminal definition by Van der Aalst [13] employed in this thesis (See Section 2.2.2) assigns attributes to events, but does not enforce a distinction between input and output attributes. Given this limitation, an event log can contain information on the input data in two general ways. First, input data elements of an event e might be part of the attribute set of e, as seen for the `amount` attribute of the "*perform advanced check*" event. In this case, similarity of inputs is already covered by the aforementioned attribute name and value similarity measures $ATTR$ and VAL. However, input data can also be derived from data attributes that were created *prior* to the execution of an event, which we operationalize next.

Similarity measure. We define a measure $PREQ$ that determines the similarity of prerequisites based on the attributes associated with prior events. Specifically, given an event e_i that occurs at position i in a trace t, we define $PREQ_{e_i}$ as the union of all attribute sets A_{e_j} for $0 < j < i$. $PREQ_E$ then denotes all attributes contained in a set $PREQ_e$ for all events $e \in \mathscr{E}$ with event class E. The similarity between two

prerequisites sets $PREQ_{E_1}$ and $PREQ_{E_2}$ is then computed in a similar manner as the $ATTR$ measure.

$$PREQ(E_1, E_2) = \frac{\mathbf{PREQ_{E_1}} \cdot \mathbf{PREQ_{E_2}}}{\| \mathbf{PREQ_{E_1}} \| \| \mathbf{PREQ_{E_2}} \|} \tag{7.6}$$

Alternatives. It is possible to consider the values of prerequisite attributes, rather than their names, as provided by the VAL measure, or by combining the two. Furthermore, alternative measures can consider two more factors in the similarity computation, namely frequency and proximity of prerequisite attributes. The *frequency* with which an attribute is created prior to the execution of an event $e \in E$ can be used to distinguish among mandatory and optional prerequisites. In the context of process matching, such a distinction was proposed by Sagi et al. [242]. The *proximity* between the creation of an attribute and an occurrence of an event can provide insights into their similarity. Intuitively, if an attribute is created by an event at index i in a trace, then this attribute is more likely to be a relevant prerequisite to its immediate sequel event e_{i+1} than it is to events that are further away. Frequency and proximity considerations can be integrated by adapting the weights of the elements of the vectors used by $PREQ$ accordingly.

7.2.7 Operationalization

The proposed similarity measures can be used as FLMs that quantify the similarity between the activities of two process models. In order to use them for a full-fledged matching task, the FLMs should be followed by second-line matching. In the evaluation presented in Section 7.3, we illustrate this by combining the similarity scores of all six FLMs using an *ensemble matcher*. Afterwards, we identify the most likely correspondence for each activity, by employing a *decision maker* to select the best combination of correspondences.

It is here important to stress that the proposed FLMs are intended to be complementary to traditional process model matching techniques, rather than to replace them. For instance, by using ensemble matchers, our event log-based matchers can be combined with matchers that consider the activity labels of the process models. In this manner, matching approaches can provide a full coverage of available process information in order to achieve high-quality matching results.

7.3 Evaluation

This section presents an empirical evaluation that demonstrates the usefulness of event-log information for process model matching. We evaluate the performance of the proposed log-based matchers as a standalone tool. Specifically, we compare the correspondences obtained by automatic matching based on our FLMs to a *gold standard* that contains the true correspondences between event classes. Our evaluation is based on real-world data, using a test collection of 105 event-log pairs. Section 7.3.1 provides further details on this test collection. Afterwards, Section 7.3.2 describes the way in which we conducted the evaluation. Finally, Section 7.3.3 presents and discusses the evaluation results.

7.3.1 Test Collection

To perform the evaluation, we use data from the *BPI Challenge 2015* [83], which consists of real-world event data related to the handling of construction permit applications by five Dutch municipalities. The event data describe similar processes, while their actual implementation differs considerably. To obtain a sufficiently large collection of event logs to match, we split the event data into event logs, each relating to a different subprocess (on average 17 subprocesses per municipality). We omit the event logs that contain less than five event classes in order to avoid trivial matching tasks. After this step, we have a total of 57 event logs. We create pairs of event logs that relate to the same subprocess from different municipalities. This results in a total of 105 event log pairs.

Table 7.1: Characteristics of the test collection

Measure	Traces	Event classes	Total corr.	True corr.	Log Overlap
Average	487.0	33.0	2,533.4	30.9	87.7%
Std.dev.	353.6	40.7	6,246.3	36.5	10.5%
Minimum	8	5	15	3	50.0%
Maximum	1409	172	26,832	156	100.0%

Table 7.1 provides an overview of the test collection. The table illustrates the great diversity between the subprocesses. This can, for example, be seen in the number of event classes per log, which ranges from 5 to 172. The column with *true correspondences* (True corr.) reflects the number of actual correspondences that exist between the event classes in a log pair. These true correspondences represent the *gold standard* for this evaluation. This gold standard could be directly established because corresponding event classes are associated with the same identifiers (i.e., `action_code`) across the event logs of the different municipalities. The last column in the table describes the overlap in terms of the event classes of a log pair, i.e., the fraction of event classes that appear in both logs. This measure indicates that, on average, 88% of the event classes in a log also appear in the gold standard. In the most extreme case, only 50% of the event classes from a log pair correspond to each other. Table 7.1 highlights the fact that even though the processes are similar across the five municipalities, considerable differences exist as well. The choice for this data collection is, furthermore, motivated by the lack of event logs associated with the collections typically used to evaluate matchers, i.e., the collections of the Process Model Matching Contests [32, 58].

Note that in order to provide objective evaluation results, we obscure all references to the names of event classes in this test collection. In particular, we hide the names and values of the following attributes: `concept:name`, `action_code`, `activityNameEN`, and `activityNameNL`.

7.3.2 Setup

To conduct the evaluation, we used the Ontobuilder Research Environment (ORE), an open source schema matching tool that enables researchers to run and evaluate matching experiments. We implemented the six FLMs in ORE and made their implementation publicly available as part of the tool.[1]

As described at the end of Section 7.2, establishing (exact) correspondences between the event class sets $\mathcal{E}(L_1), \mathcal{E}(L_2)$ of a log pair requires a similarity matrix $\mathcal{M}(\mathcal{E}(L_1), \mathcal{E}(L_2))$ and a decision maker. Here, we obtain the similarity matrices in two different manners, resulting in a two-part evaluation. In the first part, we use each of the six FLMs separately to construct $\mathcal{M}(\mathcal{E}(L_1), \mathcal{E}(L_2))$ based on a distinct similarity measure. This part of the evaluation provides insights into the performance of the *individual* FLMs and into the characteristics of the test collection. In the second part, we use an ensemble matcher that combines the scores of the six similarity matrices into a single matrix. By evaluating this *matching ensemble*, we obtain insights into the combined performance of the matchers and their complementary nature. We further reflect on the way in which the measures complement each other by computing correlations among the individual similarity scores.

After obtaining a similarity matrix $\mathcal{M}(\mathcal{E}(L_1), \mathcal{E}(L_2))$ we apply a decision maker on the matrix $\mathcal{M}(\mathcal{E}(L_1), \mathcal{E}(L_2))$ to obtain a set of exact correspondences, to which we will refer to as $\mathcal{C}(\mathcal{E}(L_1), \mathcal{E}(L_2))$. In particular, we apply the Maximum Weighted Bipartite-graph Match (MWBM) [105] to establish $\mathcal{M}(\mathcal{E}(L_1), \mathcal{E}(L_2))$. This decision maker is particularly well-suited in the context of the test collection, because it establishes 1:1 correspondences between event classes.

As in previous chapters, we again use the precision, recall, and F_1 measures to compare the automatically obtained set of correspondences C to the set $\mathcal{G}$ of *actual* correspondences included in the gold standard, as formalized by Equations 7.7, 7.8, and 7.9. *Precision* here reflects the fraction of the correspondences obtained by the matching techniques that is also included in the gold standard, whereas recall represents the fraction of the correspondences in the gold standard that is correctly identified by the matchers.

$$pre = \frac{C \cap \mathcal{G}}{C} \quad (7.7) \qquad rec = \frac{C \cap \mathcal{G}}{\mathcal{G}} \quad (7.8) \qquad F_1 = \frac{2 * pre * rec}{pre + rec} \quad (7.9)$$

Next, we presents the results of our quantitative evaluation.

7.3.3 Results

Table 7.2 presents an overview of the results obtained by using the individual FLMs and by using a matching ensemble based on all six FLMs. We will now elaborate on the results obtained through these two methods.

[1] `https://bitbucket.org/tomers77/ontobuilder-research-environment`

Table 7.2: Evaluation results

FLM	Precision	Recall	F_1-score
RP	0.24	0.25	0.25
FREQ	0.14	0.14	0.14
DUR	0.13	0.11	0.12
ATTR	0.05	0.04	0.04
VAL	**0.27**	**0.27**	**0.27**
PREQ	0.09	0.08	0.08
Ensemble	**0.38**	**0.38**	**0.38**

Matching results

The results presented in Table 7.2 show that the performance varies greatly across the various FLMs. The lowest performance results belong to *ATTR* and *PREQ* FLMs, which both consider similarity based on *attribute names*. These FLMs achieve F_1-scores of 0.04 and 0.08, respectively. A post-hoc analysis of the similarity matrices generated by these FLMs shows that, indeed, attribute names provide little discriminatory power in the context of this particular test collection. In fact, most event classes are associated with identical or nearly identical sets of attributes, which results in a similarity score of 1.00 for the vast majority of event class pairs. By contrast, *VAL* achieves the highest results with an F_1-score of 0.27. This shows that attribute *values* provide a substantially better indicator of similarity than attribute *names*. Furthermore, the performance of *RP* shows that the consideration of positional similarity also provides a relatively good indicator of similarity.

The last row of Table 7.2 presents the results obtained by an ensemble consisting of the six FLMs. For this ensemble, we applied a naïve weighting scheme, in which we computed the average score of the six similarity measures. The results demonstrate that the ensemble greatly outperforms individual FLMs, achieving an F_1-score of .38. A one-sided paired t-test reveals that this result is statistically significant ($p < 0.05$) when compared to the best performing individual FLM (*VAL*). The improved results of the ensemble illustrate that the six FLMs are complementary to each other and can enhance each other's performance.

Top-k results

The results shown so far indicate that the use of log data for process matching is a valid approach that can identify correspondences among activities by analyzing execution data. It is also clear that the use of log data alone does not suffice to establish a state-of-the-art matching tool. An F_1-score of about 0.4 indicates a far from random correlation between the decisions made by the ensemble and the true correspondences. Still, it requires the support of other techniques to strengthen its performance. Existing process model-based matchers represent good candidates, because they use valuable process model information (*e.g.,*, activity labels), which is purposefully not used by

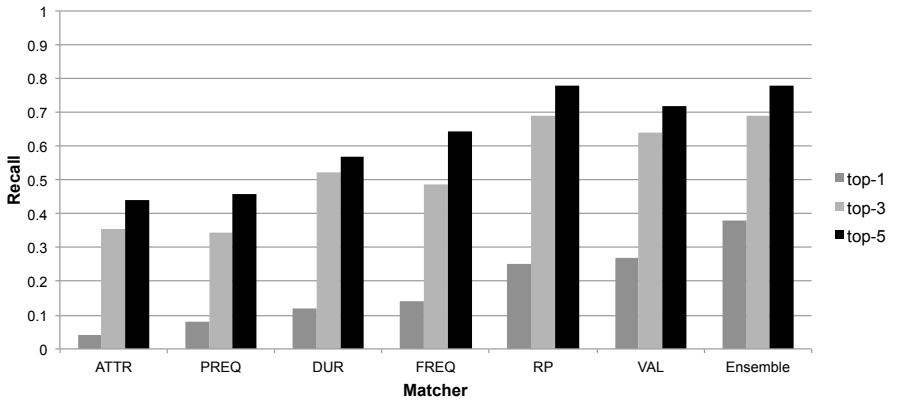


Figure 7.2: Recall scores for top-*k* results

our log-based matchers. Because numerous model-based matchers exist, each with their own strengths and weaknesses, we leave for future research the best way to tackle the combination of log-based and model-based matching techniques. Here, we investigate the obtained results in more depth and determine to what extent the log-based techniques lend themselves well to process matching.

Identified correspondences can be incorrect because often an event class has multiple correspondences with equal or near-equal similarity scores as the best candidates. The selection of a single, best correspondence then becomes an arbitrary selection among a handful of correspondences. This problem relates to the inherent issue of uncertainty in the matching task. Works on *matching monotonicity* [101] have found that this uncertainty prevents matchers from identifying a correct correspondence as the one with the highest similarity measure. However, these works argue that *good* matchers should contain the *correct correspondences* among the correspondences with the highest similarity scores, i.e. in the so-called top-*k* matches. If they succeed in this, a good matcher positions a true correspondence high enough for a human observer to confirm it after scanning only a few possible correspondences. To test this, we check for each event class whether its correct correspondence occurs within the top 3 or top 5 correspondences with the highest similarity scores.

Figure 7.2 presents results of the top-*k* analysis. For each matcher, we measure the recall of top-1, top-3, and top-5. As expected, the matching result improves significantly when the best correspondences are considered. This holds for all matchers, but with varying levels of success. The biggest gain is observed for the *RP* measure. There, the performance increases from a recall of 0.25 for top-1 to 0.69 and 0.78 for top-3 and top-5, respectively. As such, the *RP* measure performs nearly identically to the matching ensemble. The results indicate that all matchers show a monotonic behavior, though some more than others. It is interesting to see that while the top-1 performance of *RP* is worse than the one of *VAL*, a different picture is drawn for top-3 and top-5. There, *RP* surpasses *VAL* (in terms of Recall), performing just as well as the ensemble matcher. Finally, what is important to realize is that by considering

the top-3 and top-5 scores users need to just evaluate approximately 4% and 7% of the total possible correspondences. These small fractions already enable the respective identification of close to 70% and 80% of the true correspondences.

7.4 Limitations

Our evaluation demonstrates that the consideration of event-log information can be useful for the identification of correspondences between process models. However, these results need to be reflected against the background of some limitations. In particular, we identify limitations related to the proposed matchers and limitations related to the evaluation.

The main limitation of the proposed matchers is that they focus on characteristics that can provide useful insights into similarity, but cannot guarantee similarity or dissimilarity between event classes. For example, while the usefulness of positional similarity is shown through the good performance of the *RP* measure, it is well-imaginable that there are pairs of activities with a perfect *RP* similarity score, but which are actually not at all related to each other. In the same way, similarity of durations can be useful, but there can be many tasks in the same process with an execution duration of zero, i.e., automated tasks. By contrast, a certain process step can be very well automated in one process, whereas it is performed manually in another, resulting in complete dissimilarity according to the DUR metric. This characteristic of the event log-based matchers differs compared to model-based matching techniques that, for instance, analyze activity labels. If two activity labels are completely equal, then this nearly always does indicate a correct relation between the activities. Nevertheless, our matcher techniques are exactly intended to be used when such label-based matchers do not achieve the desired result. Furthermore, this limitation mainly implies that the FLMs perform best when used as part of a matching ensemble. In these cases, coincidental similarity according to one FLM will most likely be evened out by other FLMs included in the ensemble, as was also demonstrated in the evaluation of Section 7.3.

A limitation of the evaluation is that the presented results are affected by the characteristics of the utilized test collection. Therefore, the individual FLMs can perform differently when applied to other matching scenarios. For instance, the $ATTR$ matcher does not perform well on our collection because nearly all event classes are associated with the same set of attributes. However, this may well not be the case for other processes, which would result in a better performance of the $ATTR$ measure. Nevertheless, the test collection does represent a real-world collection with a great variety among its processes, as described in Section 7.3.2. Therefore, the evaluation still provides useful insights into the performance of the proposed matching techniques in practical settings. This especially applies to the performance of the matching ensembles. These ensembles analyze a variety of characteristics, which makes their results less prone to the specifics of a test collection.

7.5 Related Work

The work presented in this chapter relates to two main streams of research: process model matching and instance-based matching. Section 2.4.4 provides an in-depth overview of the former stream. The difference between our matching technique and the numerous existing process model matching techniques is that we take event-log information into account, rather than focus solely on information contained in the process models themselves. Since this information is derived from the consideration of process instances, our work also relates to existing *instance-based matching* techniques applied in other domains.

Instance-based matching has been previously explored in the context of schema matching and the related field of ontology alignment. Engmann and Maßmann [89] used two methods to enhance their COMA++ matcher. The first, a constraint-based matcher, identifies the field types using a list of patterns and numerical constraints attempted over the instance data per attribute. An approach similar to our own *VAL* measure. Their second method applies to text-based fields taken from the same domain in which the string-similarity of all instances is compared and averaged. A similar approach is suggested by Wang et. al. [273] which probe a Web query interfaces with keywords and then compare the vector-space similarity of the query result tokens. A similar approach is applied by Duan et. al. [86], using Locality Sensitive Hashing (LSH) techniques to compare instances over very large ontologies. Zaiß et. al. [292] use regular expressions to improve pattern identification of attribute domains. Our work differs from these works in that we make use of process-unique features to perform the matching task. For instance, the consideration of execution order, duration, and prerequisites are characteristics that are specific to the process perspective.

7.6 Summary

In this chapter we proposed instance-based process matching as a new element for the toolbox of matching process models. We introduced six FLMs that assess the similarity of two event classes from different event logs. Each FLM focuses on a different conceptual notion of similarity, resulting in a broad coverage of the process information stored in event logs. In particular, we presented matchers that quantify similarity according to: event positions, frequencies, durations, attribute names, attribute values, and prerequisites. We demonstrated the usefulness of these similarity metrics through a quantitative evaluation using real-world data. The evaluation showed that by just considering the information specific to event logs, the introduced matchers can identify a considerable number of correspondences between event classes. This performance was particularly achieved when the six FLMs were combined into a single matching ensemble. As an illustration, the results show that users would need to evaluate just 4% out of all possible correspondences in order to identify nearly 70% of the actual correspondences. By combining our instance-based matching techniques with traditional, model-based techniques users can strive to obtain matching results that cannot be obtained by using either of these techniques alone.

8 Conclusion

This chapter concludes this doctoral thesis. Section 8.1 provides a summary of the main results. Section 8.2 discusses the implications of the presented work for research and practice. Finally, Section 8.3 provides an outlook on directions for future research.

8.1 Summary of Results

In this thesis, we focused on the automated analysis of process information contained in various representation formats. As a basis for the development of these techniques, we considered why different representation formats are used in organizations and the problems that can result from this situation. The main contributions of this thesis are provided by five techniques that focus on the alignment and comparison of process information in different informational artifacts. We can summarize the main results presented in this thesis as follows:

- *An overview of the causes and consequences of process-information fragmentation.* Based on an analysis of existing literature and a case study, we identified various reasons for the existence of a variety of process-information artifacts in organizations, as presented in Chapter 2. Most importantly, different artifacts are used in order to provide stakeholders with the process information that is relevant to conduct their tasks, in suitable representation formats. We identified three main consequences of this spread of process information, i.e., of process-information fragmentation. First, fragmentation can lead to increased time and effort required to maintain process information. Because organizations often struggle with these required efforts, this can also lead process information becoming outdated. Second, the fragmentation of process information can negatively affect the execution efficiency of processes. Ultimately, this leads to more time, effort, and costs required for their execution. Third, when fragmentation causes users to struggle to find the process information needed to perform their

tasks, it can even lead to business process noncompliance. Such cases can lead to reduced quality of process outcomes and the incurrence of financial penalties.

- *Inconsistency detection between process models and textual process descriptions.* The use of textual process descriptions alongside process models makes process information more accessible to a variety of stakeholders. However, this comes with a clear risk that model and text become misaligned, for instance when process changes are not applied to both descriptions consistently. For organizations with hundreds of processes, the effort required to manually identify and clear up such conflicts is considerable. The approach presented in Chapter 3 addresses this problem by automatically detecting inconsistencies between a process model and a corresponding textual description. A quantitative evaluation using real-world data demonstrates that our approach achieves promising results. Particularly, our approach detected all model-text pairs in which a process model activity was not described in the text or where the order between process activities differed, with a high precision.
- *Behavioral spaces as a means to capture behavioral uncertainty in processes.* Uncertainty about conveyed process behavior plays an important role when analyzing semi-structured representation formats, such as textual process descriptions. Furthermore, uncertainty can result in the context of alignments between information from different artifacts. Given such uncertainty, it is often impossible to determine exactly which process behavior is conveyed by a process specification. Rather, there are numerous potential interpretations of the described process behavior. In order to still be able to reason about properties such as *conformance* in the context of uncertain behavior, this thesis introduced and applied the concept of a *behavioral space* as a means to capture all potential interpretation of behavioral uncertainty. We showed the need for and usefulness of behavioral spaces to capture uncertainty caused by alignments between process models and event logs in Chapter 4. Furthermore, Chapter 5 showed their application for the analysis of ambiguous textual process descriptions.
- *Conformance checking in the context of behavioral uncertainty.* By capturing behavioral uncertainty in processes using behavioral spaces, trustworthy conformance-checking results can be obtained in situations where existing conformance-checking techniques fail to do so. In this thesis, we showed the usefulness of behavioral spaces by using them in the context of conformance checking. The conformance-checking results obtained by our methods are probabilistic, differentiating between *conforming*, *nonconforming*, and *potentially conforming* behavior. We applied our method to two scenarios in order to illustrate the utility of behavioral spaces for conformance checking. Chapter 4 showed that behavioral spaces allow us to obtain conformance-checking results for numerous cases where traditional conformance-checking techniques cannot be applied. Similarly, Chapter 5 showed that the use of behavioral spaces provides a much better basis for conformance checking than other, naive ways of dealing with ambiguity in textual process descriptions.
- *Transformation of natural language descriptions into measurable Process Performance Indicators (PPIs).* The unstructured natural language used by managers to specify relevant PPIs differs considerably from the structured, formal

definitions required to actually monitor them. The time and effort required to transform the natural language descriptions into a suitable format impedes the effectiveness with which organizations can monitor their process performance. To overcome this problem, Chapter 6 presented an approach that automates the transformation task. A quantitative evaluation with a set of real-world PPI descriptions and accompanying process models revealed that our approach generates PPI definitions that are highly similar to those created manually. Therefore, our approach represents a viable, automated alternative to an otherwise highly laborious and time-intensive, manual task. This enables organizations to more efficiently monitor the performance of their business processes and continuously adapt their monitoring activities to changing business needs.

- *Process model matching based on event-log information.* Process model matching provides the basis for many process analysis techniques, such as inconsistency detection and process querying. Chapter 7 introduced a technique that uses event-log information in process model matching. Our matching technique builds on six conceptual notions of similarity between event classes. Furthermore, we provided various operationalizations that quantify these conceptual notions. We demonstrated the usefulness of these similarity metrics through a quantitative evaluation using real-world data. The evaluation showed that by just considering the information specific to event logs, the introduced matchers can identify a considerable number of correspondences between event classes. By combining our instance-based matching techniques with traditional, model-based techniques users can strive to obtain matching results that cannot be obtained by using either of these techniques alone. In this way, the accuracy and, therefore, usefulness of techniques that build on process model alignments is improved.

To achieve these results, we followed internationally established standards applicable to IS research, conducted according to the behavioral science and design science paradigms [122]. Most importantly, this involved addressing relevant problems and evaluating our proposed solutions using appropriate metrics and relevant, real-world data collections. In this way, our results represent a significant contribution to the body of knowledge of the IS research discipline.

8.2 Implications

In this section, we consider the implications of the work presented in this thesis. Section 8.2.1 reflects on the implications for practice and Section 8.2.2 on the implications for research.

8.2.1 Implications for Practice

The work presented in this thesis has several implications for organizations that strive to improve their organizational efficiency and process quality. In particular, we identify the following main implications for practice:

- *Insights for improved management of process information.* Our analysis of the causes and consequences of process-information fragmentation, presented in Chapter 2, shows the importance of the proper management of informational artifacts for organizations. In particular, organizations should strive to make the right process information available to stakeholders in suitable formats. However, organizations should simultaneously ensure that this fragmentation does not result in a situation in which process cannot be maintained or found. A main threat in this regard is the ad-hoc creation of process-information artifacts, which reduces the traceability that exists between process information in different artifacts. Therefore, organizations should aim to establish guidelines for the creation and maintenance of process information. In this way, process information becomes easy to retrieve and stays in sync with the correct way to execute the process it describes.
- *More efficient resolution of inconsistencies among process representations.* Although organizations often use multiple process representation formats in order to provide different stakeholders with information in a suitable format, this comes with the clear risk that the different representations do not convey the same information about a process. Our inconsistency-detection technique presented in Chapter 3 enables organizations to automatically detect such inconsistencies between process models and textual process descriptions. By using our approach, organizations can more efficiently detect and, therefore, resolve inconsistencies that exist in their process documentation. In this way, organizations can save considerable amounts of time and effort. This especially applies to organizations with large process repositories, possibly containing information on hundreds of processes.
- *Increased adaptability of process performance monitoring.* The monitoring of business processes represents an important prerequisite for organizations to improve their processes and, in this way, maintain a competitive advantage. This monitoring task is greatly affected by the continuous changes to which business processes and their environments are subject. As a result, organizations should continuously adapt the way in which they monitor their processes to these shifts. Our transformation approach presented in Chapter 6 supports organizations in this endeavor. The approach bridges the Business-IT gap that exists between the managers that specify the performance indicators to be monitored and the systems engineers involved in the actual monitoring of these indicators. In particular, it enables managers to specify automatically measurable PPIs without precarious and time-consuming communication with system engineers. In this way, our approach enables organizations to more easily adapt the monitoring of their business processes to emergent changes.
- *Improved detection and prevention of business process noncompliance.* Business process noncompliance can pose considerable threats to organizations. Noncompliant actions can lead to reduced efficiency, reduced process quality, and even to the incurrence of financial penalties. In this thesis, we presented two novel conformance-checking techniques that improve the ability of organizations to detect nonconforming behavior. Chapter 4 presented a technique that enables the detection of nonconformance in a more trustworthy manner than traditional

techniques. In particular, our technique avoids the need to impose, possibly incorrect, assumptions on the relations between observed events and a process model. Chapter 5 expands the applicability of conformance-checking techniques from merely structured process representations, to semi-structured representations, specifically in the form of textual process descriptions. In this way, we enabled conformance checking to take the wealth of process information that is stored in natural language documents into account. Ultimately, the presented techniques improve the detection of business process noncompliance, leading to increased organizational efficiency and process quality.

8.2.2 Implications for research

The work presented in this thesis has several implications for research in the field of business process management as well as IS research, in particular for the management and alignment of process information.

- *Comparison and alignment of process information in semi-structured formats.* Establishing alignments between process-information artifacts is an important prerequisite for the development of a variety of process analysis techniques, such as inconsistency detection, update propagation, and process querying. Most existing techniques focus on the creation of alignments between structured process representation formats, in the form of process models or process models and event logs. In this thesis, Chapters 3 and 6 presented techniques that align process information from other representation formats, specifically from textual process descriptions and natural language PPI descriptions. In this way, our research provides a basis for the development of analysis techniques that build on the alignments established by our techniques. Furthermore, our research also shows the potential for the development of further automated analysis techniques, specifically those that focus on process information contained in other semi-structured informational artifacts that have so far been largely ignored by research.
- *Enabled reasoning in the presence of behavioral uncertainty.* Uncertainty about process behavior plays a prominent role when analyzing unstructured or semi-structured process-information artifacts, as well as when reasoning based on automatically established alignments between artifacts. Our work presented in Chapters 4 and 5 has important implications in this regard. The introduction of the concept of behavioral spaces provides a structured manner to capture all possible interpretations of uncertain process behavior. By doing so, process analysis techniques can be applied without the need to select a single, possibly incorrect, interpretation of the process behavior. This demonstrates that the use of behavioral spaces opens up numerous possibilities for the development of process analysis techniques that were previously deemed unfruitful because of ambiguity or other causes of uncertainty.
- *Improved process model matching through the consideration of event-log information.* Alignments between process models provide an important basis for a variety of application scenarios and techniques. These alignments are, for in-

stance, used to detect differences between models, for the harmonization of process model variants, process querying, and for the automated propagation of process changes. The accuracy and, therefore, usefulness of such techniques is highly dependent on the correctness and completeness of the alignments that are established by process model matching techniques. In Chapter 7, we showed that the quality of these results can be improved through the consideration of event-log information. In particular, by considering the contents of event logs in the matching tasks, alignment results can be obtained that model-based matchers cannot achieve. Therefore, our research supports the usefulness of analysis techniques that build on process model alignments because it can lead to an improved alignment quality. Furthermore, our results show that the consideration of instance-level information can be worthwhile in future development of process model matching techniques.

8.3 Future Research

This doctoral thesis focused on the fragmentation of process information and, in particular, on technological developments that can mitigate its consequences. The work presented in this thesis opens up possibilities for further research that pursues the efficient use and maintenance of process information within organizations. In this pursuit, we can distinguish two general perspectives to mitigate the negative aspects of process-information fragmentation: the technical and the organizational perspective. The technical perspective focuses on mitigation by providing technical solutions to specific problems caused fragmentation. By contrast, the organizational perspective focuses on mitigation by reducing the extent of fragmentation itself.

8.3.1 Technical Perspective

This thesis primarily takes a *technical* perspective to mitigate the effects of process-information fragmentation. We presented conceptual and technical developments that focus on the more efficient use and maintenance of process information spread out over various artifacts. Different opportunities for future research stem from these developments.

A variety of scenarios in which organizations struggle with the impact of process-information fragmentation remain unaddressed by existing research. In particular, future research can focus on the development of techniques that (i) focus on more process perspectives, (ii) cover additional representation formats and (ii) that address additional use cases.

(i) Most of the techniques presented in this thesis focus on the comparison of process information from a control-flow perspective. While this is arguably the most relevant perspective in process analysis, it is important to recognize that the presented techniques could be extended to also consider, for example, the temporal, resource, and data perspectives. For instance, the consistency-checking technique from Chapter 3 could be extended to also detect inconsistencies with

respect to the resources that should execute activities according to the process descriptions;

(ii) Regarding the coverage of additional representation formats, we primarily see opportunities to develop alignment and comparison techniques that focus on process information contained in slide sets, checklists, and spreadsheets. The use of these representation formats has been recognized in literature and observed in our case study, as discussed in Section 2.1.4. However, these formats are currently not yet covered by existing alignment techniques;

(iii) Aside from broadening the spectrum of considered formats, techniques can also be developed that address use cases beyond the comparison and alignment of process information. For instance, our technique presented in Chapter 3 aims to detect inconsistencies between process model and textual descriptions, which can be used to detect changes between the process representations. Instead, by automatically propagating the effects of process changes from one representation to the other, process consistency can be ensured, rather than just detected. The conceptual contributions presented in this thesis can be particularly helpful in the pursuit of such new research directions. For instance, the approach used to extract process information from textual process descriptions and textual PPI descriptions can be utilized to support the extraction of process information from other semi-structured artifacts. Furthermore, the concept of a behavioral space can be applied to use cases in which the extraction or alignment of process information cannot provide deterministic results.

In this thesis, and in the above reflection, we have so far focused on application scenarios of alignment techniques in isolation. Typically, existing research focuses on the alignment or comparison of two informational artifacts. However, as discussed in Section 2.1.4, realistic settings can involve much larger numbers of informational artifacts related to a single process. It is for those processes that the problems caused by fragmentation can be me the most pressing. To better support such scenarios, research should strive for integrated solutions that enable the use and maintenance of all information related to a process and, even, entire process repositories, rather than consider pairs of informational artifacts in isolation. The main prerequisite for such support is that traceability should be established between process information from all artifacts. Note that the provision of holistic support through such techniques differs considerably from the use of data warehouses to accomplish similar goals. Data warehouses require a considerable overhaul of the way in which organizational members capture process information in the first place. By contrast, by using techniques such as presented and proposed in this thesis, organizational members can still document process information in the format that they prefer, whether this is a process model, a textual description, or any other representation format of choice.

Organizational Perspective

Although this thesis mainly focused on the technological perspective, the *organizational* perspective also plays an important role when dealing with process information in organizations. In particular, by taking this perspective into consideration, the negative aspects associated with process-information fragmentation can already be partially

avoided, reducing the need for their later mitigation. This thesis, in particular Section 2.1.4, already showed the crucial nature of taking such steps to reduce the effects of fragmentation. This can, for instance, be achieved by introducing guidelines that change the way in which organizations capture process information. However, further research is required in order to better understand how to achieve the desired effects. Currently, research that addresses the question of how to best capture process information has primarily focused on the suitability of representation formats for specific tasks (cf. [55, 93]. A main problem is that these scenarios and, therefore, also the usefulness of representation formats are typically considered in isolation, generally using experimental settings. There is limited reflection on how or why these representation formats co-exist in practical settings, let alone that these questions are investigated. Nevertheless, insights into this direction are necessary, because it is this co-existence that leads to problems when using and maintaining process information. A main direction to investigate in this regard is the trade-off that exists between the benefits of using more artifacts to provide stakeholders with the information they need versus the negative aspects associated with such further process-information fragmentation. Put differently, when does the creation of more process-information artifacts help stakeholders by providing them with the information they need and when does it reduce the ability of stakeholders to actually find this information? By answering such questions, important insights can be obtained that will enable organizations to manage their process information in a more considerate manner. Therefore, such insights would help organizations to further improve their efficiency and provide higher quality results.

Bibliography

[1] Van der Aa, H., Del-Río-Ortega, A., Resinas, M., Leopold, H., Ruiz-Cortés, A., Mendling, J., Reijers, H.A.: Narrowing the business-IT gap in process performance measurement. In: International Conference on Advanced Information Systems Engineering. pp. 543–557. Springer (2016)

[2] Van der Aa, H., Gal, A., Leopold, H., Reijers, H.A., Sagi, T., Shraga, R.: Instance-based process matching using event-log information. In: International Conference on Advanced Information Systems Engineering. pp. 283–297. Springer (2017)

[3] Van der Aa, H., Leopold, H., Batoulis, K., Weske, M., Reijers, H.A.: Integrated process and decision modeling for data-driven processes. In: Business Process Management Workshops. pp. 405–417. Springer (2015)

[4] Van der Aa, H., Leopold, H., Mannhardt, F., Reijers, H.A.: On the fragmentation of process information: Challenges, solutions, and outlook. In: International Conference on Enterprise, Business-Process and Information Systems Modeling, pp. 3–18. Springer (2015)

[5] Van der Aa, H., Leopold, H., Reijers, H.A.: Detecting inconsistencies between process models and textual descriptions. In: International Conference on Business Process Management, pp. 90–105. Springer (2015)

[6] Van der Aa, H., Leopold, H., Reijers, H.A.: Dealing with behavioral ambiguity in textual process descriptions. In: International Conference on Business Process Management. pp. 271–288. Springer (2016)

[7] Van der Aa, H., Leopold, H., Reijers, H.A.: Checking process compliance on the basis of uncertain event-to-activity mappings. In: International Conference on Advanced Information Systems Engineering. pp. 79–93. Springer (2017)

[8] Van der Aa, H., Leopold, H., Reijers, H.A.: Comparing textual descriptions to process models: The automatic detection of inconsistencies. Information Systems 64, 447–460 (2017)

[9] Van der Aa, H., Leopold, H., del Rio-Ortega, A., Resinas, M., Reijers, H.A.: Transforming unstructured natural language descriptions into measurable process performance indicators using hidden markov models. Information Systems 71, 27–39 (2017)
[10] Van der Aa, H., Leopold, H., van de Weerd, I., Reijers, H.A.: Causes and consequences of fragmented process information: Insights from a case study. In: 23rd Americas Conference on Information Systems, AMCIS (2017)
[11] Van der Aa, H., Reijers, H.A., Vanderfeesten, I.: Composing workflow activities on the basis of data-flow structures. In: International Conference on Business Process Management, pp. 275–282. Springer (2013)
[12] Van der Aa, H., Reijers, H.A., Vanderfeesten, I.: Designing like a pro: The automated composition of workflow activities. Computers in industry 75, 162–177 (2016)
[13] Van der Aalst, W.M.P.: Process Mining: Discovery, Conformance, and Enhancement of Business Processes. Springer (2011)
[14] Van der Aalst, W.M.P., Adriansyah, A., De Medeiros, A.K.A., Arcieri, F., Baier, T., Blickle, T., Bose, J.C., et al.: Process mining manifesto. In: International Conference on Business Process Management. pp. 169–194. Springer (2011)
[15] Van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2(2), 182–192 (2012)
[16] Van der Aalst, W.M.P., van Hee, K.M., van Werf, J.M., Verdonk, M.: Auditing 2.0: using process mining to support tomorrow's auditor. Computer 43(3), 90–93 (2010)
[17] Van der Aalst, W.M.P., Ter Hofstede, A.H., Weske, M.: Business process management: A survey. In: International conference on business process management. pp. 1–12. Springer (2003)
[18] Van der Aalst, W.M.P., Van Hee, K.M.: Workflow management: models, methods, and systems. MIT press (2004)
[19] Van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128–1142 (2004)
[20] Abiteboul, S., Kanellakis, P., Grahne, G.: On the representation and querying of sets of possible worlds. SIGMOD Rec. 16(3), 34–48 (Dec 1987)
[21] Accorsi, R., Stocker, T.: On the exploitation of process mining for security audits: the conformance checking case. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing. pp. 1709–1716. ACM (2012)
[22] Achananuparp, P., Hu, X., Shen, X.: The evaluation of sentence similarity measures. In: Data Warehousing and Knowledge Discovery, pp. 305–316. Springer (2008)
[23] Achour, C.B.: Guiding scenario authoring1. Information Modelling and Knowledge Bases X 51, 152 (1999)
[24] Adriansyah, A., van Dongen, B., Van der Aalst, W.M.P.: Conformance checking using cost-based fitness analysis. In: Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International. pp. 55–64. IEEE (2011)

[25] Aggarwal, C.C., Yu, P.S.: A survey of uncertain data algorithms and applications. IEEE Transactions on Knowledge and Data Engineering 21(5), 609–623 (2009)

[26] Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., Soroa, A.: A study on similarity and relatedness using distributional and wordnet-based approaches. In: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics. pp. 19–27. Association for Computational Linguistics (2009)

[27] Alavi, M., Leidner, D.E.: Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues. MIS quarterly pp. 107–136 (2001)

[28] Algergawy, A., Nayak, R., Saake, G.: Element similarity measures in xml schema matching. Information Sciences 180(24), 4975–4998 (2010)

[29] Allweyer, T.: BPMN 2.0: introduction to the standard for business process modeling. BoD–Books on Demand (2010)

[30] Alter, S.: Beneficial noncompliance and detrimental compliance: Expected paths to unintended consequences. In: Americas Conference on Information Systems (2015)

[31] Andrade, E., van der Aa, H., Leopold, H., Alter, S., Reijers, H.A.: Factors leading to business process noncompliance and its positive and negative effects: Empirical insights from a case study. In: 22nd Americas Conference on Information Systems, AMCIS (2016)

[32] Antunes, G., Bakhshandeh, M., Borbinha, J., Cardoso, J., Dadashnia, S., Di Francescomarino, C., Dragoni, M., Fettke, P., Gal, A., Ghidini, C., et al.: The process model matching contest 2015. GI-Edition/Proceedings: Lecture notes in informatics 248, 127–155 (2015)

[33] Awad, A., Decker, G., Weske, M.: Efficient compliance checking using bpmn-q and temporal logic. In: International Conference on Business Process Management. pp. 326–341. Springer (2008)

[34] Backus, J.W.: The syntax and semantics of the proposed international algebraic language of the zurich acm-gamm conference. Proceedings of the International Conference on Information Processing, 1959 (1959)

[35] Bagayogo, F., Beaudry, A., Lapointe, L.: Impacts of it acceptance and resistance behaviors: a novel framework. In: 34th International Conference on Information Systems (ICIS) (2013)

[36] Bahl, L.R., Mercer, R.L.: Part of speech assignment by a statistical decision algorithm. In: Proceedings IEEE International Symposium on Information Theory. pp. 88–89 (1976)

[37] Baier, T., Di Ciccio, C., Mendling, J., Weske, M.: Matching of events and activities-an approach using declarative modeling constraints. In: International Conference on Enterprise, Business-Process and Information Systems Modeling. pp. 119–134. Springer (2015)

[38] Baier, T., Di Ciccio, C., Mendling, J., Weske, M.: Matching events and activities by integrating behavioral aspects and label analysis. Software & Systems Modeling pp. 1–26 (2017)

[39] Baier, T., Mendling, J.: Bridging abstraction layers in process mining by automated matching of events and activities. In: International Conference on Business Process Management, pp. 17–32. Springer (2013)

[40] Baier, T., Mendling, J., Weske, M.: Bridging abstraction layers in process mining. Information Systems 46, 123–139 (2014)

[41] Baier, T., Rogge-Solti, A., Weske, M., Mendling, J.: Matching of events and activities-an approach based on constraint satisfaction. In: IFIP Working Conference on The Practice of Enterprise Modeling. pp. 58–72. Springer (2014)

[42] Bajwa, I.S., Choudhary, M.A.: From natural language software specifications to UML class models. In: Enterprise Information Systems, pp. 224–237. Springer (2012)

[43] Basten, T., Van der Aalst, W.M.P.: Inheritance of behavior. The Journal of Logic and Algebraic Programming 47(2), 47–145 (2001)

[44] Batini, C., Lenzerini, M., Navathe, S.B.: A comparative analysis of methodologies for database schema integration. ACM computing surveys (CSUR) 18(4), 323–364 (1986)

[45] Becker, J., Delfmann, P., Herwig, S., Lis, L., Stein, A.: Formalizing linguistic conventions for conceptual models. In: Conceptual Modeling - ER 2009, pp. 70–83. LNCS, Springer Berlin Heidelberg (2009)

[46] Becker, J., Kugeler, M., Rosemann, M.: Process management: a guide for the design of business processes. Springer Science & Business Media (2013)

[47] Bernstein, P.A., Rahm, E.: Data warehouse scenarios for model management. In: International Conference on Conceptual Modeling. pp. 1–15. Springer (2000)

[48] Bird, S., Klein, E., Loper, E.: Natural language processing with Python: analyzing text with the natural language toolkit. O'Reilly Media, Inc. (2009)

[49] Born, M., Dörr, F., Weber, I.: User-Friendly Semantic Annotation in Business Process Modeling. In: WISE 2007 Workshops, LNCS, vol. 4832, pp. 260–271. Springer (2007)

[50] Box, G.E.: Robustness in the strategy of scientific model building. Robustness in statistics 1, 201–236 (1979)

[51] 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)

[52] Brewer, P., Speh, T.: Using the balance scorecard to measure supply chain performance. Journal of Business Logistics 21, 75–93 (2000)

[53] Brill, E.: Some advances in transformation-based part of speech tagging. arXiv preprint cmp-lg/9406010 (1994)

[54] Brown, P.F., Pietra, V.J.D., Pietra, S.A.D., Mercer, R.L.: The mathematics of statistical machine translation: Parameter estimation. Computational linguistics 19(2), 263–311 (1993)

[55] Browning, T.R.: On the alignment of the purposes and views of process models in project management. Journal of Operations Management 28(4), 316–332 (2010)

[56] Budanitsky, A., Hirst, G.: Evaluating wordnet-based measures of lexical semantic relatedness. Computational Linguistics 32(1), 13–47 (2006)

[57] Cavnar, W.B., Trenkle, J.M., et al.: N-gram-based text categorization. Ann Arbor MI 48113(2), 161–175 (1994)
[58] Cayoglu, U., Dijkman, R.M., Dumas, M., Fettke, P., Garcıa-Banuelos, L., Hake, P., Klinkmüller, C., Leopold, H., Ludwig, A., Loos, P., et al.: The process model matching contest 2013. In: 4th International Workshop on Process Model Collections: Management and Reuse (PMC-MR'13) (2013)
[59] Chaffey, D., White, G.: Business information management: improving performance using information systems. Pearson Education (2010)
[60] Chakraborty, S., Sarker, S., Sarker, S.: An exploration into the process of requirements elicitation: A grounded approach. J. AIS 11(4) (2010)
[61] Chan, F.: Performance measurement in a supply chain. International Journal of Advanced Manufacturing Technology 21, 534–548 (2003)
[62] Chaudhuri, S., Dayal, U.: An overview of data warehousing and olap technology. ACM Sigmod record 26(1), 65–74 (1997)
[63] Chesani, F., Mello, P., Montali, M., Riguzzi, F., Sebastianis, M., Storari, S.: Checking compliance of execution traces to business rules. In: International Conference on Business Process Management. pp. 134–145. Springer (2008)
[64] Chomsky, N.: Three models for the description of language. IRE Transactions on information theory 2(3), 113–124 (1956)
[65] Church, K.W.: A stochastic parts program and noun phrase parser for unrestricted text. In: Proceedings of the second conference on Applied natural language processing. pp. 136–143. Association for Computational Linguistics (1988)
[66] Cohen, W., Ravikumar, P., Fienberg, S.: A comparison of string metrics for matching names and records. In: Kdd workshop on data cleaning and object consolidation. vol. 3, pp. 73–78 (2003)
[67] Costello, C., Molloy, O.: Building a process performance model for business activity monitoring. In: Information Systems Development, pp. 237–248. Springer US (2009)
[68] Crubézy, M., Musen, M.A.: Ontologies in support of problem solving. In: Handbook on ontologies, pp. 321–341. Springer (2004)
[69] Cutting, D., Kupiec, J., Pedersen, J., Sibun, P.: A practical part-of-speech tagger. In: Proceedings of the third conference on Applied natural language processing. pp. 133–140. Association for Computational Linguistics (1992)
[70] Damashek, M.: Gauging similarity with n-grams: Language-independent categorization of text. Science 267(5199), 843 (1995)
[71] Davenport, T.H., Short, J.E.: The new industrial engineering: information technology and business process redesign. Sloan Management Review pp. 11–27 (1990)
[72] Davies, I., Green, P., Rosemann, M., Indulska, M., Gallo, S.: How do practitioners use conceptual modeling in practice? Data & Knowledge Engineering 58(3), 358–380 (2006)
[73] 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)

[74] 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 (2008)

[75] Dellarocas, C., Klein, M.: A knowledge-based approach for designing robust business processes. In: Business Process Management, pp. 50–65. Springer (2000)

[76] Denning, P.J.: A new social contract for research. Communications of the ACM 40(2), 132–134 (1997)

[77] DeRose, S.J.: Grammatical category disambiguation by statistical optimization. Computational linguistics 14(1), 31–39 (1988)

[78] Di Ciccio, C., Van der Aa, H., Cabanillas, C., Mendling, J., Prescher, J.: Detecting flight trajectory anomalies and predicting diversions in freight transportation. Decision Support Systems 88, 1–17 (2016)

[79] Dijkman, R.M., Dumas, M., García-Bañuelos, L.: Graph matching algorithms for business process model similarity search. In: International Conference on Business Process Management, pp. 48–63. Springer (2009)

[80] Dijkman, R.M., Dumas, M., Ouyang, C.: Formal semantics and analysis of bpmn process models using petri nets. Queensland University of Technology, Tech. Rep (2007)

[81] Dijkman, R.M., Dumas, M., Van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Information Systems 36(2), 498–516 (2011)

[82] Doddington, G.R., Mitchell, A., Przybocki, M.A., Ramshaw, L.A., Strassel, S., Weischedel, R.M.: The automatic content extraction (ace) program-tasks, data, and evaluation. In: LREC. vol. 2, p. 1 (2004)

[83] van Dongen, B.: Bpi challenge 2015 (2015), https://doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1

[84] Dorow, B., Widdows, D.: Discovering corpus-specific word senses. In: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 2. pp. 79–82. Association for Computational Linguistics (2003)

[85] Dou, D., McDermott, D., Qi, P.: Ontology translation on the semantic web. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". pp. 952–969. Springer (2003)

[86] Duan, S., Fokoue, A., Hassanzadeh, O., Kementsietsidis, A., Srinivas, K., Ward, M.J.: Instance-based matching of large ontologies using locality-sensitive hashing. In: International Semantic Web Conference. pp. 49–64. Springer (2012)

[87] Dumas, M., Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer (2013)

[88] Earley, J.: An efficient context-free parsing algorithm. Communications of the ACM 13(2), 94–102 (1970)

[89] Engmann, D., Maßmann, S.: Instance matching with COMA++. In: BTW Workshops. pp. 28–37 (2007)

[90] Epure, E.V., Martín-Rodilla, P., Hug, C., Deneckère, R., Salinesi, C.: Automatic process model discovery from textual methodologies. In: Research Challenges

in Information Science (RCIS), 2015 IEEE 9th International Conference on. pp. 19–30. IEEE (2015)
[91] Fahey, L., Prusak, L.: The eleven deadliest sins of knowledge management. California management review 40(3), 265–276 (1998)
[92] 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)
[93] Figl, K., Recker, J.: Exploring cognitive style and task-specific preferences for process representations. Requirements Engineering 21(1), 63–85 (2016)
[94] Fisher, B.: Reengineering your business process. Journal of Systems Management 47(1), 46 (1996)
[95] Forney Jr, G.D.: The viterbi algorithm. Proceedings of the IEEE 61(3), 268–278 (1973)
[96] Franceschini, F., Galetto, M., Maisano, D.: Management by measurement: Designing key indicators and performance measurement systems. Springer (2007)
[97] Francescomarino, C., Tonella, P.: Supporting Ontology-Based Semantic Annotation of Business Processes with Automated Suggestions. In: International Conference on Enterprise, Business-Process and Information Systems Modeling, LNBIP, vol. 29, pp. 211–223. Springer (2009)
[98] Francis, W., Kucera, H.: Frequency analysis of english usage. Journal of English Linguistics 18 (1982)
[99] Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: International Conference on Advanced Information Systems Engineering. pp. 482–496. Springer (2011)
[100] Gacitua-Decar, V., Pahl, C.: Automatic business process pattern matching for enterprise services design. In: Services-II, 2009. SERVICES-2'09. World Conference on. pp. 111–118. IEEE (2009)
[101] Gal, A., Anaby-Tavor, A., Trombetta, A., Montesi, D.: A framework for modeling and evaluating automatic semantic reconciliation. VLDBJ 14(1), 50–67 (2005)
[102] Gal, A., Weidlich, M.: Model matching - processes and beyond. In: International Conference on Advanced Information Systems Engineering. pp. 525–526. Springer (2015)
[103] Gal, A.: Uncertain schema matching. Synthesis Lectures on Data Management 3(1), 1–97 (2011)
[104] Gal, A., Sagi, T.: Tuning the ensemble selection process of schema matchers. Information Systems 35(8), 845–859 (2010)
[105] Galil, Z., Micali, S., Gabow, H.: An o(ev\logv) algorithm for finding a maximal weighted matching in general graphs. SIAM Journal on Computing 15(1), 120–130 (1986)
[106] Garside, R., Leech, G.N., McEnery, T.: Corpus annotation: linguistic information from computer text corpora. Taylor & Francis (1997)
[107] Ge, N., Hale, J., Charniak, E.: A statistical approach to anaphora resolution. In: Proceedings of the sixth workshop on very large corpora. vol. 71, p. 76 (1998)
[108] Ghose, A., Koliadis, G., Chueng, A.: Process discovery from model and text artefacts. In: Services, 2007 IEEE Congress on. pp. 167–174. IEEE (2007)

[109] Giunchiglia, F., Shvaiko, P., Yatskevich, M.: Semantic matching. In: Encyclopedia of Database Systems, pp. 2561–2566. Springer (2009)

[110] Gomez, F., Segami, C., Delaune, C.: A system for the semiautomatic generation of ER models from natural language specifications. Data & Knowledge Engineering 29(1), 57–81 (1999)

[111] de Gonçalves, J.C., Santoro, F.M., Baiao, F.A.: Business process mining from group stories. In: Computer Supported Cooperative Work in Design, 2009. CSCWD 2009. 13th International Conference on. pp. 161–166. IEEE (2009)

[112] González, O., Casallas, R., Deridder, D.: MMC-BPM: A Domain-Specific Language for Business Processes Analysis. Business Information Systems 21, 157–168 (2009)

[113] Goodhue, D.L., Thompson, R.L.: Task-technology fit and individual performance. MIS quarterly pp. 213–236 (1995)

[114] Gorin, A.L., Riccardi, G., Wright, J.H.: How may I help you? Speech communication 23(1), 113–127 (1997)

[115] Greene, B.B., Rubin, G.M.: Automatic grammatical tagging of English. Department of Linguistics, Brown University (1971)

[116] Grefenstette, G.: Explorations in automatic thesaurus discovery, volume 278 of the springer international series in engineering and computer science (1994)

[117] Gregor, S.: The nature of theory in information systems. MIS quarterly pp. 611–642 (2006)

[118] Griffiths, T.V., Petrick, S.R.: On the relative efficiencies of context-free grammar. Communications of the ACM 8(5), 289–300 (1965)

[119] Gruber, T.R., et al.: A translation approach to portable ontology specifications. Knowledge acquisition 5(2), 199–220 (1993)

[120] 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)

[121] Hammer, M., Champy, J.: Reengineering the Corporation: Manifesto for Business Revolution, A. Zondervan (2009)

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

[123] Hidders, J., Dumas, M., Van der Aalst, W.M.P., ter Hofstede, A.H., Verelst, J.: When are two workflows the same? In: Proceedings of the 2005 Australasian symposium on Theory of computing-Volume 41. pp. 3–11. Australian Computer Society, Inc. (2005)

[124] Hill, J.B., Cantara, M., Kerremans, M., Plummer, D.: Magic quadrant for business process management suites, 2007. Gartner RAS Core Research Note G 152906 (2007)

[125] Hobbs, J.R.: Resolving pronoun references. Lingua 44(4), 311–338 (1978)

[126] Imieliński, T., Lipski Jr, W.: Incomplete information in relational databases. Journal of the ACM (JACM) 31(4), 761–791 (1984)

[127] Indurkhya, N., Damerau, F.J.: Handbook of natural language processing. Chapman and Hall/CRC (2010)

[128] Islam, A., Inkpen, D.: Second order co-occurrence pmi for determining the semantic similarity of words. In: Proceedings of the International Conference on Language Resources and Evaluation, Genoa, Italy. pp. 1033–1038 (2006)

[129] ISR: Editorial statement and policy. Information Systems Research 13(4), inside front cover (2002)

[130] Jackson, P., Moulinier, I.: Natural language processing for online applications: Text retrieval, extraction and categorization, vol. 5. John Benjamins Publishing (2007)

[131] Jin, T., Wang, J., La Rosa, M., Ter Hofstede, A., Wen, L.: Efficient querying of large process model repositories. Computers in Industry 64(1), 41–49 (2013)

[132] Joshi, A.K.: Natural language processing. Science 253(5025), 1242 (1991)

[133] Jung, J.Y., Bae, J.: Workflow clustering method based on process similarity. In: International Conference on Computational Science and Its Applications. pp. 379–389. Springer (2006)

[134] Jung, J., Choi, I., Song, M.: An integration architecture for knowledge management systems and business process management systems. Computers in industry 58(1), 21–34 (2007)

[135] Jurafsky, D., Martin, J.H.: Speech & language processing. Pearson Education India (2000)

[136] Jurish, B.: A hybrid approach to part-of-speech tagging. Final Report at Berlin-Brandenburgische Akademie der Wissenschaften, Berlin (2003)

[137] Kaplan, R.S., Norton, D.P.: The balanced scorecard: measures that drive performance. Harvard Business Review 83(7), 172 (2005)

[138] Karlsson, F., Voutilainen, A., Heikkilae, J., Anttila, A.: Constraint Grammar: a language-independent system for parsing unrestricted text, vol. 4. Walter de Gruyter (1995)

[139] Kettinger, W.J., Teng, J.T., Guha, S.: Business process change: a study of methodologies, techniques, and tools. MIS quarterly pp. 55–80 (1997)

[140] Klein, D., Manning, C.D.: Accurate unlexicalized parsing. In: Proceedings of the 41st Annual Meeting of the ACL-Volume 1. pp. 423–430. ACL (2003)

[141] Klein, S., Simmons, R.F.: A computational approach to grammatical coding of english words. Journal of the ACM (JACM) 10(3), 334–347 (1963)

[142] Knackstedt, R., Kuropka, D., Müller, O., Polyvyanyy, A.: An ontology-based service discovery approach for the provisioning of product-service bundles. In: European Conference on Information Systems. pp. 1965–1977 (2008)

[143] 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)

[144] Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Ijcai. vol. 14, pp. 1137–1145 (1995)

[145] Kolb, P.: Disco: A multilingual database of distributionally similar words. Proceedings of KONVENS-2008, Berlin (2008)

[146] Kolb, P.: Experiments on the difference between semantic similarity and relatedness. In: Proceedings of the ordic Conference on Computational Linguistics (ODALIDA). pp. 81–88 (2009)

[147] Korherr, B., List, B.: Extending the EPC and the BPMN with business process goals and performance measures. In: International Conference on Enterprise Information Systems. pp. 287–294 (2007)

[148] 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)

[149] Kovacic, A.: Business renovation: business rules (still) the missing link. Business process management journal 10(2), 158–170 (2004)

[150] Krauth, E., Moonen, H., Popova, V., Schut, M.C.: Performance measurement and control in logistics service providing. In: International Conference on Enterprise Information Systems. pp. 239–247 (2005)

[151] Kronz, A.: Managing of process key performance indicators as part of the aris methodology. In: Corporate performance management, pp. 31–44. Springer (2006)

[152] Krumnow, S., Decker, G.: A concept for spreadsheet-based process modeling. In: International Workshop on Business Process Modeling Notation. pp. 63–77. Springer (2010)

[153] Kumar, E.: Natural language processing. IK International Pvt Ltd (2011)

[154] Kupiec, J.: Robust part-of-speech tagging using a hidden markov model. Computer Speech & Language 6(3), 225–242 (1992)

[155] Kuss, E., Leopold, H., Van der Aa, H., Stuckenschmidt, H., Reijers, H.A.: Probabilistic evaluation of process model matching techniques. In: Conceptual Modeling: 35th International Conference, ER 2016, Gifu, Japan, November 14-17, 2016, Proceedings 35. pp. 279–292. Springer (2016)

[156] Küster, J.M., Koehler, J., Ryndina, K.: Improving business process models with reference models in business-driven development. In: International Conference on Business Process Management. pp. 35–44. Springer (2006)

[157] La Rosa, M., Dumas, M., Uba, R., Dijkman, R.M.: Business process model merging: An approach to business process consolidation. ACM Transactions on Software Engineering and Methodology (TOSEM) 22(2), 11 (2013)

[158] Lami, G., Gnesi, S., Fabbrini, F., Fusani, M., Trentanni, G.: An automatic tool for the analysis of natural language requirements. Tech. rep., Informe técnico, CNR Information Science and Technology Institute, Pisa, Italia, Setiembre (2004)

[159] Landauer, T.K., Foltz, P.W., Laham, D.: An introduction to latent semantic analysis. Discourse processes 25(2-3), 259–284 (1998)

[160] Lappin, S., Leass, H.J.: An algorithm for pronominal anaphora resolution. Computational linguistics 20(4), 535–561 (1994)

[161] Lavoie, B., Rambow, O., Reiter, E.: The modelexplainer. In: Eighth International Workshop on Natural Language Generation, Herstmonceux, Sussex (1996)

[162] Leopold, H.: Natural language in business process models. Springer (2013)

[163] Leopold, H., van der Aa, H., Pittke, F., Raffel, M., Mendling, J., Reijers, H.A.: Integrating textual and model-based process descriptions for comprehensive process search. In: International Conference on Enterprise, Business-Process and Information Systems Modeling. pp. 51–65. Springer (2016)
[164] Leopold, H., Eid-Sabbagh, R.H., 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)
[165] Leopold, H., Meilicke, C., Fellmann, M., Pittke, F., Stuckenschmidt, H., Mendling, J.: Towards the automated annotation of process models. In: International Conference on Advanced Information Systems Engineering. pp. 401–416. Springer (2015)
[166] Leopold, H., Mendling, J., Polyvyanyy, A.: Supporting process model validation through natural language generation. IEEE Transactions on Software Engineering 40(8), 818–840 (2014)
[167] Leopold, H., Niepert, M., Weidlich, M., Mendling, J., Dijkman, R.M., Stuckenschmidt, H.: Probabilistic optimization of semantic process model matching. In: International Conference on Business Process Management. pp. 319–334. Springer (2012)
[168] 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)
[169] Lin, D.: An information-theoretic definition of similarity. In: ICML. vol. 98, pp. 296–304 (1998)
[170] Lin, D.: Dependency-based evaluation of minipar. In: Treebanks, pp. 317–329. Springer (2003)
[171] Ling, J., Zhang, L., Feng, Q.: An improved structure-based approach to measure similarity of business process models. In: SEKE. pp. 377–380 (2014)
[172] Liu, K., Yan, Z., Wang, Y., Wen, L., Wang, J.: Efficient syntactic process difference detection using flexible feature matching. In: Asia-Pacific Conference on Business Process Management. pp. 103–116. Springer (2014)
[173] Lu, R., Sadiq, S., Governatori, G.: Compliance aware business process design. In: International Conference on Business Process Management. pp. 120–131. Springer (2007)
[174] Luftman, J., Papp, R., Brier, T.: Enablers and inhibitors of business-it alignment. Communications of the AIS 1(3es), 1 (1999)
[175] Madhavan, J., Bernstein, P.A., Domingos, P., Halevy, A.Y.: Representing and reasoning about mappings between domain models. AAAI/IAAI 2002, 80–86 (2002)
[176] Madhavan, J., Bernstein, P.A., Rahm, E.: Generic schema matching with cupid. In: vldb. vol. 1, pp. 49–58 (2001)
[177] Maglitta, J.: Know-how, inc. Computerworld 30(1), 1996 (1996)
[178] Mannhardt, F., de Leoni, M., Reijers, H.A.: Extending process logs with events from supplementary sources. In: 3rd Workshop on Data- & Artifact-centric BPM. pp. 235–247. Springer (2014)
[179] Manning, C.D., Raghavan, P., Schütze, H.: Introduction to information retrieval, vol. 1. Cambridge university press Cambridge (2008)

[180] Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. pp. 55–60 (2014)
[181] Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a large annotated corpus of english: The penn treebank. Computational linguistics 19(2), 313–330 (1993)
[182] Marshall, B., Chen, H., Madhusudan, T.: Matching knowledge elements in concept maps using a similarity flooding algorithm. Decision Support Systems 42(3), 1290–1306 (2006)
[183] Matsumoto, Y., Tanaka, H., Hirakawa, H., Miyoshi, H., Yasukawa, H.: Bup: a bottom-up parser embedded in prolog. New Generation Computing 1(2), 145–158 (1983)
[184] McCallum, A., Freitag, D., Pereira, F.C.: Maximum entropy markov models for information extraction and segmentation. In: Icml. vol. 17, pp. 591–598 (2000)
[185] McCarthy, J.F., Lehnert, W.G.: Using decision trees for coreference resolution. arXiv preprint cmp-lg/9505043 (1995)
[186] Mendling, J.: Metrics for process models: empirical foundations of verification, error prediction, and guidelines for correctness, vol. 6. Springer Science & Business Media (2008)
[187] Mendling, J., Reijers, H.A., Van der Aalst, W.M.P.: Seven process modeling guidelines (7pmg). Information and Software Technology 52(2), 127–136 (2010)
[188] Meziane, F., Athanasakis, N., Ananiadou, S.: Generating natural language specifications from UML class diagrams. Requirements Engineering 13(1), 1–18 (2008)
[189] Michelberger, B.: Process-oriented information logistics: aligning process information with business processes. Ph.D. thesis, Ulm University (2015)
[190] Michelberger, B., Mutschler, B., Reichert, M.: On handling process information: Results from case studies and a survey. In: International Conference on Business Process Management. pp. 333–344. Springer (2011)
[191] Michelberger, B., Mutschler, B., Reichert, M.: Towards process-oriented information logistics: why quality dimensions of process information matter. In: EMISA. pp. 107–120. Koellen-Verlag (2011)
[192] Mihalcea, R., Corley, C., Strapparava, C.: Corpus-based and knowledge-based measures of text semantic similarity. In: Proceedings of the 21st National Conference on Artificial Intelligence. vol. 6, pp. 775–780 (2006)
[193] Miller, G.: The organization of lexical memory: Are word associations sufficient. The pathology of memory pp. 223–237 (1969)
[194] Miller, G.A.: WordNet: a lexical database for english. Communications of the ACM 38(11), 39–41 (1995)
[195] 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)
[196] Miller, S., Bobrow, R., Ingria, R., Schwartz, R.: Hidden understanding models of natural language. In: Proceedings of the 32nd annual meeting on Associa-

tion for Computational Linguistics. pp. 25–32. Association for Computational Linguistics (1994)
[197] Mohri, M.: Finite-state transducers in language and speech processing. Computational linguistics 23(2), 269–311 (1997)
[198] Momm, C., Malec, R., Abeck, S.: Towards a model-driven development of monitored processes. In: Wirtschaftsinformatik (2). pp. 319–336 (2007)
[199] Munoz-Gama, J., Carmona, J., Van der Aalst, W.M.P.: Single-entry single-exit decomposed conformance checking. Information Systems 46, 102–122 (2014)
[200] Nadeau, D., Turney, P.D.: A supervised learning approach to acronym identification. In: Conference of the Canadian Society for Computational Studies of Intelligence. pp. 319–329. Springer (2005)
[201] Nasukawa, T., Yi, J.: Sentiment analysis: Capturing favorability using natural language processing. In: Proceedings of the 2nd international conference on Knowledge capture. pp. 70–77. ACM (2003)
[202] Navarro, G.: A guided tour to approximate string matching. ACM computing surveys (CSUR) 33(1), 31–88 (2001)
[203] Noy, N.F.: Semantic integration: a survey of ontology-based approaches. ACM Sigmod Record 33(4), 65–70 (2004)
[204] Noy, N.F., Musen, M.A.: The prompt suite: interactive tools for ontology merging and mapping. International Journal of Human-Computer Studies 59(6), 983–1024 (2003)
[205] Och, F.J.: Minimum error rate training in statistical machine translation. In: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1. pp. 160–167. Association for Computational Linguistics (2003)
[206] Parent, C., Spaccapietra, S.: Issues and approaches of database integration. Communications of the ACM 41(5es), 166–178 (1998)
[207] Paulraj, D., Swamynathan, S., Madhaiyan, M.: Process model-based atomic service discovery and composition of composite semantic web services using web ontology language for services (owl-s). Enterprise Information Systems 6(4), 445–471 (2012)
[208] Pedrinaci et al., C.: Sentinel: a semantic business process monitoring tool. In: International Workshop on Ontology-Supported Business Intelligence. pp. 26–30 (2008)
[209] Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S.: A design science research methodology for information systems research. Journal of management information systems 24(3), 45–77 (2007)
[210] Pei, J., Jiang, B., Lin, X., Yuan, Y.: Probabilistic skylines on uncertain data. In: Proceedings of the 33rd international conference on Very large data bases. pp. 15–26 (2007)
[211] Peng, L., Diao, Y.: Supporting data uncertainty in array databases. In: ACM SIGMOD International Conference on Management of Data. pp. 545–560. ACM (2015)
[212] Phalp, K.T., Vincent, J., Cox, K.: Improving the quality of use case descriptions: empirical assessment of writing guidelines. Software Quality Journal 15(4), 383–399 (2007)

[213] Pittke, F., Leopold, H., Mendling, J.: When language meets language: Anti patterns resulting from mixing natural and modeling language. In: Business Process Management Workshops. pp. 118–129. Springer (2014)
[214] 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)
[215] Pittke, F., Leopold, H., Mendling, J., Tamm, G.: Enabling reuse of process models through the detection of similar process parts. In: International Conference on Business Process Management. pp. 586–597. Springer (2012)
[216] Polyvyanyy, A., Armas-Cervantes, A., Dumas, M., García-Bañuelos, L.: On the expressive power of behavioral profiles. Formal Aspects of Computing 28(4), 597–613 (2016)
[217] Ponzetto, S.P., Strube, M.: Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In: Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics. pp. 192–199. Association for Computational Linguistics (2006)
[218] Popova, V., Sharpanskykh, A.: Modeling organizational performance indicators. Information Systems 35(4), 505–527 (2010)
[219] Porter, M.F.: An algorithm for suffix stripping. Program 14(3), 130–137 (1980)
[220] Pradhan, S.S., Ramshaw, L., Weischedel, R., MacBride, J., Micciulla, L.: Unrestricted coreference: Identifying entities and events in ontonotes. In: International Conference on Semantic Computing. pp. 446–453. IEEE (2007)
[221] Pritchard, J.P., Armistead, C.: Business process management–lessons from european business. Business Process Management Journal 5(1), 10–35 (1999)
[222] Quinlan, J.R.: Induction of decision trees. Machine learning 1(1), 81–106 (1986)
[223] Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
[224] Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. the VLDB Journal 10(4), 334–350 (2001)
[225] Ramezani, E., Fahland, D., Van der Aalst, W.M.P.: Where did i misbehave? diagnostic information in compliance checking. In: International conference on business process management. pp. 262–278. Springer (2012)
[226] Rapp, R.: A freely available automatically generated thesaurus of related words. In: LREC (2004)
[227] Razali, N.M., Wah, Y.B., et al.: Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. Journal of statistical modeling and analytics 2(1), 21–33 (2011)
[228] Recker, J., Safrudin, N., Rosemann, M.: How novices design business processes. Information Systems 37(6), 557–573 (2012)
[229] Reijers, H.A.: Design and control of workflow processes: business process management for the service industry. Springer-Verlag (2003)
[230] Reijers, H.A., Leopold, H., Recker, J.: Towards a science of checklists. In: Proceedings of the 50th Hawaii International Conference on System Sciences (2017)

[231] Resnik, P., et al.: Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. J. Artif. Intell. Res.(JAIR) 11, 95–130 (1999)
[232] Riefer, M., Ternis, S.F., Thaler, T.: Mining process models from natural language text: A state-of-the-art analysis. In: Multikonferenz Wirtschaftsinformatik (MKWI-16), March 9-11, Illmenau, Germany. Universität Illmenau (2016)
[233] del Río-Ortega, A., Cabanillas, C., Resinas, M., Ruiz-Cortés, A.: Ppinot tool suite: a performance management solution for process-oriented organisations. In: Service-Oriented Computing, pp. 675–678. Springer (2013)
[234] del Río-Ortega, A., Gutiérrez, A.M., Durán, A., Resinas, M., Ruiz-Cortés, A.: Modelling service level agreements for business process outsourcing services. In: International Conference on Advanced Information Systems Engineering. pp. 485–500. Springer (2015)
[235] del Rio-Ortega, A., Resinas, M., Cabanillas, C., Ruiz-Cortes, A.: On the definition and design-time analysis of process performance indicators. Information Systems 38(4), 470–490 (2013)
[236] del Río-Ortega, A., Resinas, M., Durán, A., Ruiz-Cortés, A.: Using templates and linguistic patterns to define process performance indicators. Enterprise Information Systems 10(2), 159–192 (2016)
[237] Rosemann, M.: Potential Pitfalls of Process Modeling: Part A. Business Process Management Journal 12(2), 249–254 (2006)
[238] Rowley, J.: The wisdom hierarchy: representations of the dikw hierarchy. Journal of information science 33(2), 163–180 (2007)
[239] Sadiq, S., Governatori, G., Namiri, K.: Modeling control objectives for business process compliance. In: International conference on business process management. pp. 149–164. Springer (2007)
[240] Safavian, S.R., Landgrebe, D.: A survey of decision tree classifier methodology. IEEE Trans. Systems, Man, & Cybernetics (1991)
[241] Sagi, T., Gal, A.: Schema matching prediction with applications to data source discovery and dynamic ensembling. The International Journal on Very Large Data Bases 22(5), 689–710 (2013)
[242] Sagi, T., Gal, A., Weidlich, M.: Measuring expected integration effort in service composition. In: Services Computing (SCC), 2014 IEEE International Conference on. pp. 645–652. IEEE (2014)
[243] Saldivar, J., Vairetti, C., Rodríguez, C., Daniel, F., Casati, F., Alarcón, R.: Analysis and improvement of business process models using spreadsheets. Information Systems 57, 1 – 19 (2016)
[244] Salton, G., McGill, M.J.: Introduction to modern information retrieval. McGraw-Hill, Inc. (1986)
[245] Sànchez-Ferreres, J., Carmona, J., Padró, L.: Aligning textual and graphical descriptions of processes through ILP techniques. In: International Conference on Advanced Information Systems Engineering (In press). Springer (2017)
[246] Sarma, A.D., Benjelloun, O., Halevy, A., Widom, J.: Working models for uncertain data. In: 22nd International Conference on Data Engineering. pp. 7–7. IEEE (2006)

[247] Schäl, T.: Workflow management for process organisations, volume 1096 of. Lecture Notes in Computer Science (1996)

[248] Schumacher, P., Minor, M., Schulte-Zurhausen, E.: Extracting and enriching workflows from text. In: Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on. pp. 285–292. IEEE (2013)

[249] Sebu, M.L., Ciocârlie, H.: Similarity of business process models in a modular design. In: Applied Computational Intelligence and Informatics (SACI), 2016 IEEE 11th International Symposium on. pp. 31–36. IEEE (2016)

[250] Selway, M., Grossmann, G., Mayer, W., Stumptner, M.: Formalising natural language specifications using a cognitive linguistic/configuration based approach. Information Systems 54, 191–208 (2015)

[251] Senderovich, A., Rogge-Solti, A., Gal, A., Mendling, J., Mandelbaum, A.: The road from sensor data to process instances via interaction mining. In: International Conference on Advanced Information Systems Engineering. pp. 257–273. Springer (2016)

[252] Simon, H.A.: The sciences of the artificial. MIT press (1996)

[253] Sinha, A., Paradkar, A.: Use cases to process specifications in Business Process Modeling Notation. In: IEEE International Conference on Web Services. pp. 473–480 (2010)

[254] Sinur, J., Schulte, W., Hill, J., Jones, T.: Magic quadrant for intelligent business process management suites. Gartner RAS Core Research Note G 224913, 27 (2012)

[255] Smirnov, S., Weidlich, M., Mendling, J.: Business process model abstraction based on behavioral profiles. In: Service-Oriented Computing, pp. 1–16. Springer (2010)

[256] Smith, H., Fingar, P.: Business process management: the third wave, vol. 1. Meghan-Kiffer Press Tampa (2003)

[257] Soon, W.M., Ng, H.T., Lim, D.C.Y.: A machine learning approach to coreference resolution of noun phrases. Computational linguistics 27(4), 521–544 (2001)

[258] Stachowiak, H.: Allgemeine modelltheorie. {Springer-Verlag, Wien} (1973)

[259] Steiger, D.M.: Enhancing user understanding in a decision support system: a theoretical basis and framework. Journal of Management Information Systems 15(2), 199–220 (1998)

[260] Temperley, D., Sleator, D., Lafferty, J.: Parsing english with a link grammar. In: Third International Workshop on Parsing Technologies (1993)

[261] Tsichritzis, D.: The dynamics of innovation. In: Beyond calculation, pp. 259–265. Springer (1997)

[262] Tur, G., Hakkani-Tür, D., Heck, L.: What is left to be understood in atis? In: Spoken Language Technology Workshop (SLT), 2010 IEEE. pp. 19–24. IEEE (2010)

[263] Unger, M., Leopold, H., Mendling, J.: How much flexibility is good for knowledge intensive business processes: A study of the effects of informal work practices. In: System Sciences (HICSS), 2015 48th Hawaii International Conference on. pp. 4990–4999. IEEE (2015)

[264] Uschold, M., Gruninger, M.: Ontologies and semantics for seamless connectivity. ACM SIGMod Record 33(4), 58–64 (2004)
[265] Vaidyanathan, G.: A framework for evaluating third-party logistics. Communications of the ACM 48(1), 89–94 (2005)
[266] Van Glabbeek, R., Goltz, U.: Refinement of actions and equivalence notions for concurrent systems. Acta Informatica 37(4-5), 229–327 (2001)
[267] Vance, D.: Information, knowledge and wisdom: The epistemic hierarchy and computer-based information systems. AMCIS 1997 Proceedings p. 124 (1997)
[268] Versley, Y.: Antecedent selection techniques for high-recall coreference resolution. In: EMNLP-CoNLL. pp. 496–505 (2007)
[269] Vessey, I., Glass, R.: Strong vs. weak approaches to systems development. Communications of the ACM 41(4), 99–102 (1998)
[270] Viorica Epure, E., Martin-Rodilla, P., Hug, C., Deneckere, R., Salinesi, C.: Automatic process model discovery from textual methodologies. In: Research Challenges in Information Science (RCIS), 2015 IEEE 9th International Conference on. pp. 19–30. IEEE (2015)
[271] 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)
[272] Wald, J.A., Sorenson, P.G.: Explaining ambiguity in a formal query language. ACM Transactions on Database Systems (TODS) 15(2), 125–161 (1990)
[273] Wang, J., Wen, J.R., Lochovsky, F., Ma, W.Y.: Instance-based schema matching for web databases by domain-specific query probing. In: Proceedings of the Thirtieth international conference on Very large data bases-Volume 30. pp. 408–419. VLDB Endowment (2004)
[274] Wang, Y.Y., Acero, A.: Combination of cfg and n-gram modeling in semantic grammar learning. In: INTERSPEECH (2003)
[275] Wang, Y.Y., Deng, L., Acero, A.: Spoken language understanding. Signal Processing Magazine, IEEE 22(5), 16–31 (2005)
[276] Wang, Y.Y., Waibel, A.: Decoding algorithm in statistical machine translation. In: Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics. pp. 366–372 (1997)
[277] Weidlich, M., Dijkman, R.M., Mendling, J.: The ICoP framework: Identification of correspondences between process models. In: International Conference on Advanced Information Systems Engineering. pp. 483–498. Springer (2010)
[278] Weidlich, M., Mendling, J.: Perceived consistency between process models. Information Systems 37(2), 80–98 (2012)
[279] Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. IEEE Transactions on Software Engineering 37(3), 410–429 (2011)
[280] Weidlich, M., Mendling, J., Weske, M.: Propagating changes between aligned process models. Journal of Systems and Software 85(8), 1885–1898 (2012)
[281] Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J.: Process compliance measurement based on behavioural profiles. In: International Conference on Advanced Information Systems Engineering. pp. 499–514. Springer (2010)

[282] Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J., Weske, M.: Process compliance analysis based on behavioural profiles. Information Systems 36(7), 1009–1025 (2011)

[283] Weidlich, M., Sagi, T., Leopold, H., Gal, A., Mendling, J.: Predicting the quality of process model matching. In: International Conference on Business Process Management, pp. 203–210. Springer (2013)

[284] Weidlich, M., Sheetrit, E., Branco, M.C., Gal, A.: Matching business process models using positional passage-based language models. In: International Conference on Conceptual Modeling. pp. 130–137. Springer (2013)

[285] Weidlich, M., Weske, M., Mendling, J.: Change propagation in process models using behavioural profiles. In: Services Computing, 2009. SCC'09. IEEE International Conference on. pp. 33–40. IEEE (2009)

[286] Wetzstein, B., Ma, Z., Leymann, F.: Towards measuring key performance indicators of semantic business processes. In: Business Information Systems. pp. 227–238. Springer (2008)

[287] Wolter, C., Meinel, C.: An approach to capture authorisation requirements in business processes. Requirements engineering 15(4), 359–373 (2010)

[288] Woods, W.A.: Semantics and quantification in natural language question answering. Advances in computers 17, 1–87 (1978)

[289] Xia, F., Palmer, M.: Converting dependency structures to phrase structures. In: Proceedings of the first international conference on Human language technology research. pp. 1–5. Association for Computational Linguistics (2001)

[290] 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)

[291] Yujian, L., Bo, L.: A normalized levenshtein distance metric. IEEE transactions on pattern analysis and machine intelligence 29(6), 1091–1095 (2007)

[292] Zaiß, K., Schlüter, T., Conrad, S.: Instance-based ontology matching using regular expressions. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". pp. 40–41. Springer (2008)

[293] Zhang, W., Sim, Y.C., Su, J., Tan, C.L.: Entity linking with effective acronym expansion, instance selection, and topic modeling. In: IJCAI. vol. 2011, pp. 1909–1914 (2011)

[294] Zmud, R.W.: Remarks from MIS Quarterly editor. MIS Quarterly 21(2), 261–290 (1997)