Automatic Detection and Resolution of Lexical Ambiguity in Process Models

Fabian Pittke, Henrik Leopold, and Jan Mendling

Abstract—System-related engineering tasks are often conducted using process models. In this context, it is essential that these models do not contain structural or terminological inconsistencies. To this end, several automatic analysis techniques have been proposed to support quality assurance. While formal properties of control flow can be checked in an automated fashion, there is a lack of techniques addressing textual quality. More specifically, there is currently no technique available for handling the issue of lexical ambiguity caused by homonyms and synonyms. In this paper, we address this research gap and propose a technique that detects and resolves lexical ambiguities in process models. We evaluate the technique using three process model collections from practice varying in size, domain, and degree of standardization. The evaluation demonstrates that the technique significantly reduces the level of lexical ambiguity and that meaningful candidates are proposed for resolving ambiguity.

Index Terms—Identification of lexical ambiguity, resolution of lexical ambiguity, business process models

1 Introduction

Conceptual models play an important role in various system-related management activities including requirements elicitation, domain analysis, software design as well as in documentation of databases, business processes, and software systems [1]. However, it has been found that the correct and meaningful usage of conceptual models appears to be a challenge in practical settings. There are several reasons for this. First, large scale projects often rely on modelers who have limited experience [2] and who often create models with correctness issues [3]. Second, modelers in large projects work in a distributed fashion, which makes it difficult to maintain consistency in terms of terminology and granularity. Third, the complexity of modeling initiatives with thousands of models [4] makes it impossible to perform quality assurance in a manual way. These issues cause rework and clarifying communication at later stages, or even worse: wrong design decisions are taken based on incorrect, incomplete, or inconsistent models [5], [6].

The quality assurance of large and complex model collections requires the usage of automatic analysis techniques. Up until now, such automatic quality assurance is mainly available for checking formal properties of particular classes of conceptual models. For instance, there is a rich set of verification techniques to check control-flow-related properties of process models [7], [8]. There are only a few techniques available for checking guidelines on text labels within models, e.g., for behavioural models [9], [10]. These techniques, however, miss important concerns of quality assurance, most notably terminological ambiguity. Indeed, the severity of the ambiguity problem is emphasized by several authors [11], [12]. Bolloju and Leung [13] find ambiguous descriptions and inconsistent element names in 64 percent of the UML models they evaluated. In the field of electronic assessment of UML activity diagrams, Jayal and Sheppard [14] also face the challenge of ambiguous label names when matching them to semantically similar ones.

A prominent instance of ambiguity is the usage of homonymous or synonymous words. Homonyms are words that have more than a single meaning. The word *ball* is an example as it can refer to a *round object* or to a *dance event*. Synonyms are different words which refer to the same meaning. As an example, consider the words *bill* and *invoice*. Although the problem of synonyms and homonyms is well established in various areas of system analysis and design including requirements engineering [11], [12], use case descriptions [15], [16], model transformations [17], code and design smells [18], matching code and documentation [19], or system components reuse [20], [21], there has been only limited research on identifying such problems in conceptual models automatically.

In this paper, we address the need for automatic techniques to detect and resolve textual ambiguities in large model collections. In contrast to natural language documents, conceptual models often include phrase fragments like “check invoice” or “receipt of message”, such that techniques that require full sentences are not directly applicable. We address this challenge by building on ideas that have initially been described previously [22]. Our focus will be on process models, which typically use labels made up of phrase fragments. The contribution of this paper is an automatic detection and resolution technique that significantly improves terminological quality metrics of a model collection. We demonstrate this capability based on an user evaluation with more than 2,500 process models from practice.

- *F. Pittke and J. Mendling are with the Institute for Information Business, WU Vienna, Austria. E-mail: {fabian.pittke, jan.mendling}@wu.ac.at.*
- *H. Leopold is with the Department of Computer Science, VU University Amsterdam, The Netherlands. E-mail: h.leopold@vu.nl.*

Manuscript received 15 Nov. 2013; revised 18 Dec. 2014; accepted 20 Jan. 2015. Date of publication 28 Jan. 2015; date of current version 17 June 2015. Recommended for acceptance by A. Egyed.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSE.2015.2396895

In the rest of the paper, Section 2 details the ambiguity problem, discusses research on detecting and resolving lexical ambiguities and illustrates the challenges based on a motivating example. Section 3 defines the formal preliminaries before Section 4 introduces the automatic detection and resolution techniques. Section 5 presents the results of applying our technique for the analysis of three process model collections. In Section 6, we discuss implications and limitations of our work. Finally, Section 7 concludes the paper.

2 Background

In this section, we discuss the background of our research. In Section 2.1, we highlight the problem of ambiguity in process models and present existing solutions for detecting and correcting ambiguity. In Section 2.2, we discuss the challenges of detecting and resolving ambiguity in process models.

2.1 Managing Ambiguity in Process Models

It is widely acknowledged that process models are important to understand how business activities are carried out [23], to document relevant parts of the organization [24], or to design process-aware information systems [25]. Moreover, it is also acknowledged that poor process modeling causes wrongly-taken business decisions [23] and misunderstandings among several parts of the organization [26]. This ultimately leads to delayed product and service releases [27] or poor system design [28]. One particular cause is the presence of ambiguity in process models which significantly influences process model understanding [29].

Despite the fact that semi-formal languages, such as process models or UML activity diagrams, are meant to restrict ambiguity [30], these process models still include elements (such as activities or events) with natural language fragments. Similar to natural language text [31], [32], [33], these language fragments are a reason of ambiguity. Moreover, the ambiguity problem is even bigger in process models since they provide only limited context to recognize and resolve ambiguities [9], [29]. Also, models often focus on isolated aspects of the system. Once a system integrates several parts of the organization, a number of models has to be considered before inconsistencies can be detected [34].

In order to improve process models and to ensure the quality of process-aware information systems, several authors propose an integration of process modeling with requirements engineering [28], [35]. This is particularly promising since approaches to ambiguity management have been discussed in the latter discipline. In requirements engineering, two classes of approaches can be distinguished. The first class includes techniques that *detect ambiguities* in requirements documents and explain them to the user. The second class encompasses techniques that attempt to *resolve ambiguities*. Each class can be subdivided into approaches focusing on detecting ambiguity in *requirements documents* and in *models* (see Table 1).

Ambiguity detection approaches aim at the reduction of ambiguity by identifying requirements statements that can be interpreted in multiple ways. For that purpose, various techniques have been proposed to assess requirements documents of different input types. In case *textual requirements* are written in natural language, different reading techniques, such as inspection-based reading [30], [36], scenario-based reading [11], or object-oriented inspections [37], can be employed. Denger et al. [27] as well as Gleich et al. [33] utilize natural language patterns for ambiguity detection. Moreover, several authors have developed metrics that indicate ambiguity in requirements documents. Available approaches can measure the degree of readability, understandability, and ambiguity [39], [40] as well as underspecifications, vagueness, and implicit sentence structures [31]. Metrics can also be employed to identify and to create a ranking of ambiguities according to different criteria [41]. Using a dataset of ambiguous requirements documents and human judgments about their interpretation, Chantree et al. [32] propose heuristics to automatically replicate these judgments. Finally, Yang et al. [42], [43] employ a machine learning approach and build a classifier that automatically determines the ambiguity of requirement statements. For *models*, Anda and Sjøberg [44] apply the concept of reading techniques to use case models and propose a checklist-based approach to detect defects such as inconsistencies and ambiguities. Mens et al. [45], [46] propose an inconsistency detection approach by using graph transformation rules to support the detection task in UML class models and state machine diagrams.

Once ambiguities have been detected, approaches for the *systematic resolution* are required to restore the consistency of requirements documents. The resolution of ambiguous requirements needs to identify the correct interpretation of the requirement statement and to rewrite it according to the stakeholders intention. For *textual requirements*, several authors, such as Chantree et al. [32] or Gleich et al. [33], acknowledge mental capabilities of humans to resolve these ambiguities. To facilitate this task, Gleich et al. [33] propose

TABLE 1
Approaches for Ambiguity Management in Requirements Engineering

Technique	Author
Ambiguity Detection	
Requirements in Text	
Reading Techniques	Kamsties et al. [11], [30], [36] Shull et al. [37]
Natural Language Patterns	Denger et al. [27], Gleich et al. [33] Rolland and Ben Achour [38]
Ambiguity Metrics	Fantechi et al. [39], Ceccato et al. [40], Fabbrini et al. [31], Wang et al. [41]
Ambiguity Heuristics	Chantree et al. [32]
Classifier Construction	Yang et al. [42], [43]
Requirements in Models	
Reading Techniques	Anda and Sjøberg [44]
Rule-based Detection	Mens et al. [45], [46]
Ambiguity Resolution	
Requirements in Text	
Notification	Gleich et al. [33]
Pattern-based rephrasing	Rolland and Ben Achour [38]
Requirements in Models	
Rule-based Resolution	Mens et al. [45], [46]

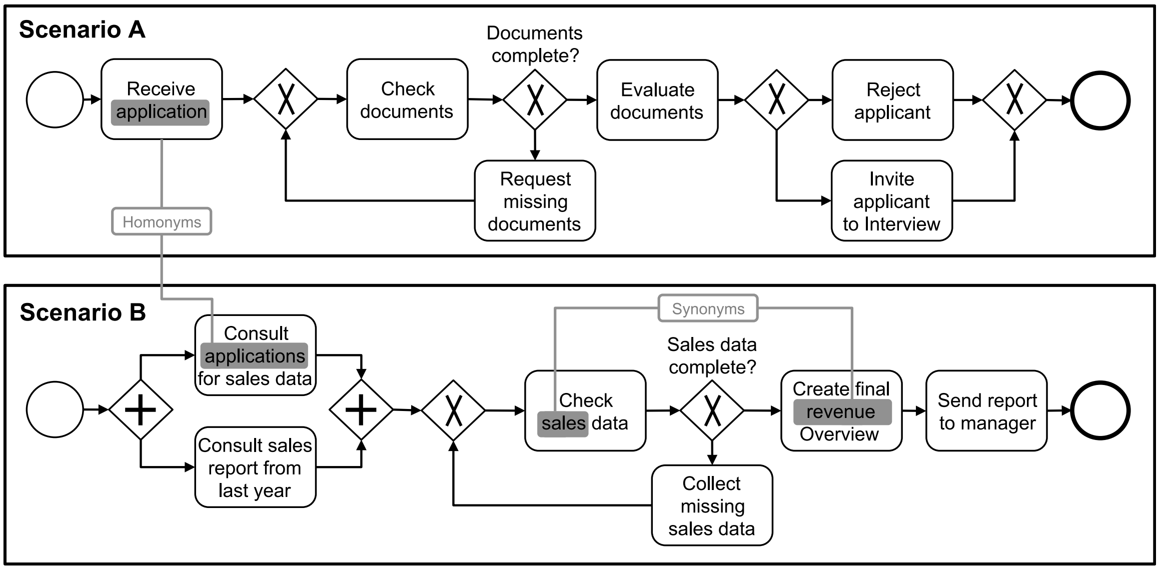


Fig. 1. Example for ambiguous process models.

a notification approach that highlights and explains ambiguous requirements statements and explicitly encourages the requirements engineers to handle the detected ambiguity. Going one step further, Rolland and Ben Achour propose a rephrasing technique that makes use of unambiguous and precise natural language patterns [38] to resolve the ambiguity in the requirements document. In order to resolve such ambiguity in *models*, Mens et al. [45], [46] enrich the previously mentioned detection technique with resolution rules that automatically rework the defects in class and state machine diagrams.

Despite their merits, the presented techniques for the detection and resolution of lexical ambiguity suffer from two main shortcomings that impede their application to process model repositories: the required manual effort and their dependence on a full natural language text.

The first shortcoming, the *required manual effort*, refers to the extensive amount of manual work that is required to detect and resolve ambiguities in process models. Since companies tend to maintain several hundreds or even thousands of process models [2], the human effort can be tremendous. In particular, the previously discussed reading techniques can hardly be applied to such a number of models. Similarly, this also applies for the resolution of the detected ambiguities, where the requirements engineers need to understand the context of the detected ambiguity and come up with a more precise alternative. The large manual effort also impedes the application of heuristic-based or machine learning approaches as each of these approaches requires a manually created data set from which heuristics can be derived or from which machine-learning approaches can be trained.

The second shortcoming, the *dependence on a full natural text*, refers to the fact that many approaches rely on text fragments taken from a grammatically correct natural language text. However, the elements of process models contain only short textual fragments that do not exhibit a complete sentence structure [9]. In addition, they follow different language patterns that may even hide the verb [29]. As a result, the discussed approaches, as the ones from Denger et al. [27], Gleich et al. [33], or Rolland and Ben Achour [38], are not applicable for process models.

Despite the findings and the diversity of approaches, there is no technique available that can address the detection and resolution of lexical ambiguity in process models. Therefore, it is the objective of this paper to close this gap and to present a technique for automatically detecting and resolving ambiguous words in process models. To this end, we discuss specific challenges to meet this objective in the following section.

2.2 Challenges and Applicable Concepts

There are several challenges associated with the automatic detection and resolution of lexical ambiguity in process models. In order to illustrate these challenges, we use BPMN process models as depicted in Fig. 1. Scenario A describes a job application. The process starts with the receipt of an application followed by a check for completeness. In case the documents are not complete, the missing documents are requested and checked again. Otherwise, the documents are evaluated and it is decided whether the application is rejected or the applicant is invited to an interview. Scenario B depicts the generation of an annual sales report. The process starts with two activities that are conducted in parallel: consulting applications for sales data and the last year's sales report. Then, the sales data are checked for completeness. If the data are incomplete, the missing sales data are collected. If everything is complete, the final revenue overview is created and sent to the manager.

These models are subject to lexical ambiguity. First, each process model use the word *application*. Scenario A utilizes it in the sense of a job application, i.e., a written request for employment in an enterprise. We can infer this meaning from the context in which documents are checked and an interview takes place. In scenario B, the word *application* refers to a software program since it is consulted for data extraction and for the creation of a sales report. Apparently, this small collection of two models uses the word *application* with two different meanings, i.e., *application* is a homonym.

TABLE 2
Challenges for Ambiguity Detection and Resolution

Challenge	Applicable Concepts
Sense Disambiguation	
Supervised Methods	Agirre and Martinez [47]
	Quinlan [48]
	Véronis and Ide [49]
	Navigli and Ponzetto [50]
Word and Context Clustering	Schütze [51]
	Lin [52]
	Pantel and Lin [53]
Ambiguity Operationalization	
Vector Space Model	Salton et al. [54]
Word Space Dimensions	Schütze [55]
Ambiguity Definitions	Deissenboeck and Pizka [56]
Ambiguity Resolution	
Additional Information	Frakes and Pole [57]
	Lin and Chan [58]
Interaction	Bouzeghoub et al. [59]
Alternative Selection	Hayne and Ram [60]
Rephrasing	Ben-Ari et al. [61]

Moreover, we observe inconsistencies in scenario B. The word *sales* refers to the turnover of the enterprise. The same semantics can be assigned to the word *revenue*. Thus, the two words are used to express the same concept and are hence considered to be synonyms.

The example also points to specific challenges for detecting homonyms and synonyms. In general, we face three challenges for the identification and resolution of lexical ambiguities: sense disambiguation, sense operationalization, and ambiguity resolution. Table 2 provides an overview of techniques addressing these challenges.

The challenge of *sense disambiguation*, also referred to as word sense disambiguation (WSD), relates to determining the sense of a word in a given context [62]. In general, we distinguish between WSD techniques that employ supervised machine-learning techniques (e.g., [47], [48], [49], [50]) and clustering approaches that identify context similar words (e.g., [51], [52], [53]). However, each existing approach requires an extensive amount of natural language text as input. As the textual information of process model elements is typically very short, it provides limited context for identifying the correct sense of a word. This insight is also confirmed by Zong and Ng [63] who report worse sense disambiguation results for short queries in information retrieval. Thus, the presented approaches need to be adapted or extended for handling short textual information of process models.

The challenge of *ambiguity operationalization* is concerned with representing word senses in such a way that ambiguities can be identified. Such a representation would show various aspects of a particular object. A prominent representation is the vector space model, which was introduced by Salton et al. [54] to describe documents in an information retrieval system. The model expresses indexing terms of a document as dimensions of the vector space. The degree to which a document is associated with a specific index term is expressed with numerical values. A similar approach, the *word space model* by Schütze [55], applies this concept to the linguistic field. The word space model interprets the vector dimensions as words. Following Schütze, a word in a corpus is represented as a vector whose components counts the number of occurrences within a fixed context (for example a sentence or a larger context). As a result, the distributional profile of words implicitly expresses their semantics. Deissenboeck and Pizka [56] introduce a formalization of ambiguities to deal with synonyms and homonyms in identifier naming. Although their formalization captures essential characteristics of ambiguity, they are not sufficiently precise for automatically identifying synonyms and homonyms.The mere existence of a word with multiple senses does not necessarily imply that it also represents an ambiguous case. To this end, we need to redefine the vector space model, to transfer it to word senses and to refine the conditions that allow for confirming or rejecting the hypothesis of ambiguity.

After the identification, the challenge of *resolution* needs to be addressed. Ideally, the detected ambiguities can be removed in order to achieve consistency and correctness among a set of process models. Reconsidering the example from Fig. 1, it is desirable to replace the homonym *application* with *job application* in scenario A and with *software program* in scenario B. In the literature, there are four different strategies for resolving ambiguities. Frakes and Pole [57] as well as Lin and Chan [58] propose to append additional information to the ambiguous word. Bouzeghoub et al. [59] suggest the incorporation of user-interaction in order to manually resolve ambiguity. A third strategy was developed by Hayne and Ram [60]. Their approach provides the user with the possibility to choose from automatically created recommendations. The last strategy suggests the automatic rephrasing or replacement of ambiguous words with more precise alternatives. An exemplary approach has been developed by Ben-Ari et al. [61] who propose an interactive approach to rephrase ambiguities. However, these strategies either require manual efforts ([57], [58]), or have been applied in specific scenarios ([58], [59], [60]), or have been developed for grammatically correct sentences ([61]). Hence, we require strategies that can deal with short text fragments that are common in process models.

Building on these challenges, we develop techniques for automatically detecting and resolving lexical ambiguities in process models in the following sections.

3 Preliminaries

This section introduces the formal preliminaries of our approach. In Section 3.1, we discuss formal preliminaries of process models. In Section 3.2, we formalize lexical ambiguity before formulating explicit conditions for ambiguity detection in Section 3.3.

3.1 Formalization of Process Models

In this paper, we adapt the canonical process model format from Leopold [64], and solely focus on process model elements containing natural language text. Further, we use the label definition as proposed by Mendling et al. [29]. According to this definition, every label of a process model contains two components: an action and a business object on which the action is applied. As an example, consider the activity label *Receive Application* from Fig. 1. It contains the action *to receive* and the business object *application*. It is important to note that these components can be communicated in different grammatical variations. For instance, the label *Application Receipt* contains the same action and business object as *Receive Application*, but uses a different grammatical structure. In order to be independent of grammatical structures, we use a technique which automatically extracts action and business object of the respective label to a decent degree of accuracy (Avg. precision: 91 percent, avg. recall: 90.6 percent) [64]. Furthermore, we assume actions to be captured as verbs and business objects as nouns. Altogether, we formalize a process model as follows.

Definition 3.1 (Process Model). *A process model $P = (A, E, G, F, W_V, W_N, L, \alpha, \beta, \lambda)$ consists of six finite sets A, E, G, W_V, W_N, L, a binary relation $F \subseteq (A \cup E \cup G) \times (A \cup E \cup G)$, a partial function $\alpha : L \nrightarrow W_V$, a partial function $\beta : L \nrightarrow W_N$, and a partial function $\lambda : (A \cup E \cup G) \nrightarrow L$, such that*

- *A is a finite non-empty set of activities.*
- *E is a finite set of events.*
- *G is a finite set of gateways.*
- *W_V is a finite set of verbs.*
- *W_N is a finite set of nouns.*
- F *is a finite set of sequence flows. Each sequence flow $f \in F$ represents a directed edge between two nodes.*
- *L is a finite set of text labels.*
- *The partial function α assigns a verb $w_V \in W_V$ to a label $l \in L$.*
- *The partial function β assigns a noun $w_N \in W_N$ to a label $l \in L$.*
- *The partial function λ defines the assignment of a label $l \in L$ to an activity $a \in A$, event $e \in E$ or gateway $g \in G$.*

As the techniques presented in this paper focus on sets of process models, we further introduce the notion of a process model collection. We denote a process model collection with C. Moreover, we define the following subsets for a process model collection C:

- The set of all labels: $L^C = \bigcup_{P \in C} L$.
- The set of all actions: $L^C_A = \bigcup_{l \in L^C} \alpha(l)$.
- The set of all business objects: $L^C_{BO} = \bigcup_{l \in L^C} \beta(l)$.
- The set of all actions and business objects $\mathcal{L}^C = L^C_A \cup L^C_{BO}$.

So far, we introduced only process models and their element labels. However, we have observed that the sense of a label may change in a different context. In the motivating example of Fig. 1, the word *application* might either refer to a *job application* or a *software program*. Thus, we require a formal baseline for word senses and for lexical ambiguities which is the main task of the following section.

3.2 Formalization of Lexical Ambiguity

In order to automatically identify ambiguities, namely homonymy and synonymy, it is essential to adequately conceptualize the sense of a word. In this paper, we adopt the *enumerative approach* by Navigli [62]. According to this approach, distinct word senses are identified and listed in a sense inventory. The connection between word senses and words of a dictionary is formalized as follows.

Definition 3.2 (Senses). *Let D be a dictionary of words, the senses of a word are defined by the function $Senses_D : \mathcal{W} \times POS \rightarrow 2^S$, such that*

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

Senses are defined in dictionaries such as the *Oxford Advanced Learner's Dictionary of Current English* [65], the *Oxford Dictionary of English* [66], and the *Longman Dictionary of Contemporary English* [67]. For automatic analysis, lexical databases such as *WordNet* [68], [69] are available, which capture various semantic relationships in a structured way. A similar resource is the *BabelNet database* [70], which combines WordNet senses with the web encyclopedia Wikipedia and allows also multilingual word sense retrieval and disambiguation. To illustrate the enumerative approach, we consider the word *application* from Fig. 1. WordNet enumerates the following seven senses:

- s_1: the act of bringing something to bear
- s_2: a verbal or written request for assistance or employment or admission to a school
- s_3: the work of applying something
- s_4: a program that gives a computer instructions that provide the user with tools to accomplish a task
- s_5: liquid preparation having a soothing or antiseptic or medicinal action when applied to the skin
- s_6: a diligent effort
- s_7: the action of putting something into operation

Using the introduced sense conceptualization, we further formalize homonymy and synonymy. As already mentioned, a homonym is a word with multiple senses and synonyms are words sharing a common sense. We regard the word *application* as a homonym as it may refer to multiple senses such as a *job application*, a *software program*, an *application in the medical sense*, or *diligence and effort*. The words *bill*, *invoice*, and *account* are synonyms since they share a common meaning. In order to formalize these properties, we reformulate the ambiguity formalization of homonyms and synonyms by Deissenboeck and Pizka [56].

Definition 3.3 (Homonyms). *Given a word $w \in \mathcal{L}^C$, the word w is a homonym iff $|Sense_D(w, *)| > 1$ where the symbol * denotes a wildcard for a part of speech $p \in POS = \{n, v, a, r\}$. The set of all homonyms Hom^C from a process model collection C is accordingly given by $Hom^C = \{w \,||\, (Sense_D(w, *)| > 1\}$.*

Definition 3.4 (Synonyms). *Given two words $w_1, w_2 \in \mathcal{L}^C$, the words w_1 and w_2 are synonyms iff $(Sense_D(w_1, *) \cap Sense_D(w_2, *)) \neq \emptyset$. Accordingly, the set of all synonym pairs Syn^C from a process model collection C is given by $Syn^C = \{(w_1, w_2) \,|\, Sense_D(w_1) \cap Sense_D(w_2) \neq \emptyset\}$.*

The previous definitions emphasize that a decision on homonymy and synonymy requires the consideration of all senses of a given word. Hence, a word is best represented by a vector of its senses, which is inline with the concepts of the vector space model by Salton et al. [54] and the word space model by Schütze [55]. The vector space model describes a document with the help of indexing terms and treats them as vector dimensions. The degree to which a document is associated with a specific index term is expressed with numerical values in the vector. The word space model interprets the vector dimensions as words and counts the occurrences of a word within a fixed context. Building on these concepts, we introduce the notion of a semantic vector SV^w of a word w. A semantic vector, represents word senses as vector dimensions which corresponds to a distinct word sense of a dictionary. If the word is used with a specific meaning, its value in the semantic vector is non-zero.

Definition 3.5 (Semantic Vector). *Let w be a word and s_i^w the ith sense of the word w retrieved from the dictionary D, the semantic vector is given by $SV_D^w = <sv_1^w, sv_2^w, \ldots, sv_n^w>$, such that*

- *The ith dimension of SV_D^w corresponds to the ith sense $s_i^w \in Sense_D(w,*)$.*
- *sv_i^w is a supporting score that indicates to which degree the word is used with the ith sense in a given context.*

To illustrate these concepts, we refer to the word *application* of Fig. 1. As shown, *application* has seven distinct word senses, such as job application (s_2) and software program (s_4). Taking the process model of scenario A as context, we assume that the context of the process model leads to the following vector $SV_{WordNet}^{application} = <0, 10, 0, 9, 2, 1, 0>$.

3.3 Ambiguity Detection Conditions

The homonym and synonym definitions from Section 3.2 capture only the basic characteristics of lexical ambiguity. In particular, they do not consider the context of a given word. The implications of the missing context is explained best by reconsidering the previously discussed word *application*. According to WordNet, this word has seven different senses. Hence, following Definition 3.3, *application* is a homonym. However, only because a word *may* refer to multiple senses, this does not necessarily mean that it actually does. An ambiguous case exists only, if the context of the word does not allow the reader to infer the correct sense. Therefore, Definitions 3.3 and 3.4 constitute only necessary conditions.

In order to finally decide about the homonymous or synonymous character of a word, we refine the notion of ambiguity with two consecutive steps. The first step involves the determination of the *dominant word sense* which is inline with standard WSD approaches, which typically assign a supporting score to an input target word given a set of words as a context [71], [72]. The word sense with the highest supporting score is typically assumed to be the most appropriate one and thus dominating all other word senses. Therefore, we formally state this condition as follows.

Definition 3.6 (Dominant Word Sense). *Given a word $w \in \mathcal{L}^C$, its semantic vector SV_D^w along with the supporting scores sv^w for a given context, a sense $s_i^w \in Sense_D(w,*)$ is* dominant *iff there is no other sense $s_i^w \in Sense_D(w,*)$ with a higher supporting score: $\forall s_i^w, s_j^w \in Sense_D(w,*), s_j^w \neq s_i^w : sv_i^w > sv_j^w$. We denote the dominant sense with s_{max}^w.*

Let us again consider the word *application* in Fig. 1 and its semantic vector with regard to scenario A, i.e., $SV_{WordNet}^{application} = <0, 10, 0, 9, 2, 1, 0>$. According to the definition of the dominant word sense, the 2nd sense is considered to be dominant since its supporting score $sv_2^{application} = 10$ is the highest among all supporting scores.

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

Definition 3.7 (Quasi-Dominant Word Senses). *Given a word $w \in \mathcal{L}^C$, its semantic vector SV_D^w along with the supporting scores sv^w for a given context and its dominating sense s_{max}^w, a sense $s_i^w \in Sense_D(w,*)$ is* quasi-dominant *iff $sv_i^w \geq \epsilon \cdot sv_{max}^w$. Accordingly the set of all quasi-dominant word senses is given by $sense_{QD}^w = \{s_i^w \in Sense_D(w,*) | sv_i^w \geq \epsilon \cdot sv_{max}^w\}$. Further, we define $domSense^w$ as being the set that contains all dominant and quasi-dominant word senses: $domSense^w = \{s_{max}^w\} \cup sense_{QD}^w$.*

Reconsidering the example of the word *application*, we already identified the 2nd sense to be dominant. If we set the confidence score ϵ to 0.9 and apply the previous definition to the remaining supporting scores $s_i^{application}$, we also find the 4th sense as quasi-dominant with regard to the other vector values. Obviously, since more word senses appear to be appropriate, the word *application* actually represents a homonym. Thus, we need to extend the necessary conditions of homonyms and synonyms with these concepts. The extended definitions formulate a sufficient condition for synonyms and homonyms that build upon the necessary conditions in the Definitions 3.3 and 3.4. The extended definitions consider the dominant senses of the word in the respective context and thus decide if the respective synonym or homonym candidate is confirmed or rejected.

Definition 3.8 (Sufficient Homonym Condition). *Given a word $w \in \mathcal{L}^C$, the semantic vector space for that word w, its vector scores sv^w and its set of dominant and quasi-dominant word senses $domSense^w$, the word w is a* confirmed homonym *iff Definition 3.3 holds and $|domSense^w| > 1$. Otherwise, it is a* rejected homonym.

Definition 3.9 (Sufficient Synonym Condition). *Given two words $w_1, w_2 \in \mathcal{L}^C$ that fulfill Definition 3.4. Let $domSense^{w_1}$ and $domSense^{w_2}$ be the set of the dominant and quasi-dominant word senses of w_1 or w_2, respectively. The words w_1 and w_2 are* confirmed synonyms *iff Definition 3.4 holds and iff $domSense^{w_1} \cap domSense^{w_2} \neq \emptyset$. Otherwise, the words are* rejected synonyms.

Building on these definitions, we introduce a technique that identifies and resolves lexical ambiguities in process models in the following section.

4 Conceptual Approach

In this section, we introduce the approaches for identifying and resolving homonymy and synonymy in process models. We start with presenting the techniques for detecting homonyms and synonyms. Afterwards, we present the respective resolution techniques.

4.1 Detection of Lexical Ambiguity

In order to properly identify lexical ambiguity, we employ the concepts and definitions we introduced in the previous section. We first discuss the detection of homonyms. Then, we present the approach for detecting synonyms.

4.1.1 Homonym Detection

Building on our arguments from the previous sections, we formulate three essential requirements a homonym detection technique has to fulfill. First, it has to identify words with several meanings. Second, it has to select those words that are actually used with different meanings in similar contexts. Third, it needs to preserve the quality and consistency of the considered model collection, i.e., process models with a similar contexts need to be treated equally.

To address *the first requirement*, we employ lexical databases that store several distinct word senses according to the enumerative approach. We already mentioned WordNet and BabelNet as examples of such lexical databases. In contrast to WordNet, BabelNet also includes encyclopedic knowledge of Wikipedia and further enriches WordNet with contemporary senses. Moreover, it has been shown that the combination of these two knowledge bases handles specific terms and vocabulary better [73]. For these reasons, we use the BabelNet to retrieve the word senses and check Definition 3.3 for any term.

In order to fulfill *the second requirement*, we need to operationalize the process model context and to derive the supporting scores for each dimension of the semantic vector. For this purpose, we employ an advanced multilingual WSD approach by Navigli and Ponzetto [50]. This approach exploits the BabelNet database to perform a graph-based disambiguation across different languages bringing together empirical evidence from other languages by ensemble methods. The evaluation of this approach demonstrates that the combination of a lexical knowledge base with graph algorithms provides partially better results than classical WSD methods, also including specific words. Equipped with the supporting scores we check Definition 3.8 and finally confirm or reject a word as homonym.

To appropriately deal with the *third requirement*, we need to consider the fact that a word occurs in several process models and that the word can have a different sense in another model. Thus, we need to find those models in which the target word is used with similar senses, i.e., in which the semantic vectors are close to each other. Afterwards, we decide for a group of several process models if a target term is used ambiguously or not. For the resolution, we ensure that the ambiguity is resolved consistently through all process models within a group.For this purpose, we employ the XMeans clustering approach by Pelleg and Moore [74] as it efficiently estimates the number of clusters automatically and is capable of finding smaller groups of vectors with similar characteristics. Finally, we check the necessary and the sufficient condition for each cluster center and decide on the homonymy of a word.

These steps are formalized in Algorithm 1. The algorithm requires the process model collection C and a target word w as input and starts with basic initializations (Steps 2-4). The set $SemanticVectors$ stores the semantic vectors that are created from disambiguating the word w with respect to the process model p which serves as the context. The set $homonymCandidates$ stores the final results of the algorithm and contains the detected homonyms. After the initializations, the necessary condition for homonyms is checked. For this purpose, we employ the BabelNet database and retrieve the synsets of the word w (Step 5). If this check does not evaluate to false, the semantic vectors for each process model p and word w are calculated using the graph-based WSD approach with BabelNet. We denote this with the function $babel(w,p)$ that takes the word w and a process model p as input and returns the semantic vector according to Definition 3.5. Each semantic vector is then added to the set $SemanticVectors$ (Steps 5-7). In Step 9, the semantic vectors are clustered to identify groups of context similar vectors. We denote this with the function $xmeans\ (semanticVectors)$ that takes the set of all semantic vectors of word w as input and returns a set of identified clusters. Afterwards, the sufficient homonym condition is checked for each cluster (Steps 10-11). The sufficient homonym evaluation makes use of the function $DominantSense(sc)$ that returns all dominant senses. If it evaluates to true, the word and the respective cluster are considered to be a potential homonym candidate and added to the result set (Step 12). At the end, the algorithm returns the final result set and terminates (Step 16).

Algorithm 1. Identification of Homonyms for a Set of Process Models

```
1:  identifyHomonyms(Word $w$, ProcessModels $C$)
2:  $semanticVectors \leftarrow \emptyset$
3:  $semanticClusters \leftarrow \emptyset$
4:  $homonymCandidates \leftarrow \emptyset$
5:  if BabelNet.RetrieveSynsets($w$) $\geq 2$ then
6:      for all $p \in C$ do
7:          $semanticVectors \leftarrow semanticVectors \cup$ babel ($w$,$p$)
8:      end for
9:      $semanticClusters \leftarrow$ xmeans($semanticVectors$)
10:     for all $sc \in semanticClusters$ do
11:         if |DominantSenses($sc$)| $\geq 2$ then
12:             $homonymCandidates \leftarrow homonymCandidates \cup$ ($w$,
                $sc$)
13:         end if
14:     end for
15: end if
16: return $homonymCandidates$
```

4.1.2 Synonym Detection

Regarding the identification of synonyms, the technique has to consider pairs of words that have at least one meaning in common. Analogously to the homonymy problem, these two words can be synonymous only if their context is approximately identical.

For this purpose, we again employ the BabelNet database to learn about the word senses of two words and to decide if they fulfill Definition 3.4. If so, we use the multilingual WSD approach and create the semantic vectors and the supporting scores. Afterwards, we employ the XMeans clustering approach for each semantic word space and identify the dominant and quasi-dominant meanings. In contrast to the homonym approach, we have to compare the resulting clusters of each candidate word with each other to check the sufficient synonym condition. If two clusters share at least one meaning, i.e., the intersection of dominant and quasi dominant sense is not empty, we confirm the two words in the respective clusters as synonym candidates.

The synonym detection approach is formalized in Algorithm 2. It takes two words w_1 and w_2 as well as the set of process models C as input. Afterwards, it starts with the basic initializations, i.e., the initialization of the semantic vector and cluster set for w_1 and w_2 and the result set (Steps 2-5). In Step 6, the necessary synonym condition is checked according to Definition 3.4. If true, the semantic vectors of each word are calculated using the BabelNet WSD approach which we denote again with the function $babel(w, p)$ (Steps 8-11). The algorithm continues with clustering the vectors of each semantic vector space denoted by the function $xmeans(semanticVectors)$ (Steps 12-13). Finally, the algorithm checks the sufficient synonym condition for each cluster (Step 15). If the dominant senses of one cluster of word w_1 intersect with the dominant senses of one cluster of word w_2, the sufficient condition evaluates to true and the two words as well as the respective clusters are stored in the result map (Steps 15-16). The algorithm terminates by returning the result set $synonymClusters$ (Step 20).

Algorithm 2. Detection of Synonyms in a Set of Process Models

```
1: identifySynonyms(Word $w_1$, $w_2$, ProcessModels $C$)
2: $semanticVectors_1 \leftarrow \emptyset$
3: $semanticVectors_2 \leftarrow \emptyset$
4: $semanticClusters_1 \leftarrow \emptyset$
5: $semanticClusters_2 \leftarrow \emptyset$
6: $synonymCandidates \leftarrow \emptyset$
7: if $(Sense_D(w1,*) \cap Sense_D(w2,*)) \neq \emptyset$ then
8:     for all $p_1, p_2 \in \mathcal{C}$ do
9:         $semanticVectors_1 \leftarrow semanticVectors_1 \cup$ babel $(w_1, p_1)$
10:        $semanticVectors_2 \leftarrow semanticVectors_2 \cup$ babel $(w_2, p_2)$
11:    end for
12:    $semanticClusters_1 \leftarrow$ xmeans($semanticVectors_1$)
13:    $semanticClusters_2 \leftarrow$ xmeans($semanticVectors_2$)
14:    for all $sc_1 \in semanticClusters_1$,
       $sc_2 \in semanticClusters_2$ do
15:        if (DominantSenses($sc_1$)
           $\cap$ DominantSenses($sc_2$)) $\neq \emptyset$ then
16:            $synonymCandidates \leftarrow synonymCandidates \cup$ ($w_1$,
               $sc_1$, $w_2$, $sc_2$)
17:        end if
18:    end for
19: end if
20: return $synonymCandidates$
```

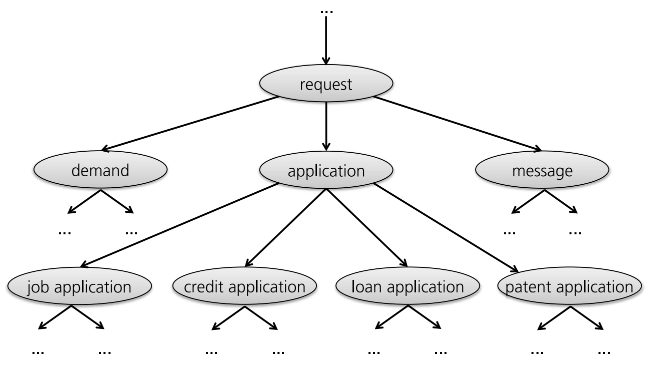


Fig. 2. Example fragment of the hyponymy tree.

4.2 Resolution of Lexical Ambiguity

A fully automatic resolution of lexical ambiguity is associated with considerable challenges and thus not free of errors. Therefore, it is often more appropriate to rely on human resolution instead and to notify them about identified cases of ambiguity [32], [75]. In this line, we propose a resolution technique that aims at providing sensible refactoring suggestions for the identified synonyms and homonyms. In the following, we first discuss strategies for generating resolution suggestions for homonyms. Then, we present a strategy for synonyms.

4.2.1 Homonym Resolution

The resolution of homonyms is addressed using different perspectives. We propose three resolution strategies.

The first strategy exploits the semantic relationship of *hyponomy*. The hyponymy relation describes a transitive relation between word senses that organizes them into a sub-word hierarchy. It starts with a general word and becomes more and more specific. As an example, consider the fragment of BabelNet's hyponymy tree for the word *application* in Fig. 2. It illustrates that BabelNet provides four different hyponyms that can be used to resolve the ambiguity. Accordingly, this resolution strategy would suggest the depicted hyponyms, i.e., *job application*, *credit application*, *loan application*, and *patent application*. Building on this concept, we formalize the resolution strategy as follows:

$$Suggestion_{Hypo} = hypo_D(w_0, s_0),$$

where $hypo_D(w_0, s_0)$ denotes a function that returns the set of all hyponyms of the original word w_0 with the word sense s_0 in a dictionary D.

In the second homonym resolution strategy we employ again the hyponym relation. However, instead of replacing the ambiguous term with a hyponym (a more specific word), we add a more general word from the hyponym tree as a qualifier. In linguistics, such a word is referred to as hypernym. Reconsidering the hyponym tree in Fig. 2, we pick the hypernym *request* and create *application [request]* as a suggestion. Formally, the hypernym strategy is described as follows:

$$Suggestion_{Hyper} = hyper_D(w_0, s_0),$$

where $hyper_D(w_0, s_0)$ denotes a function that returns hypernym of the original word w_0 with the word sense s_0 in a dictionary D.

The third strategy also implements the qualifier concept. In particular, we aim at identifying qualifiers in dictionaries and glosses of the respective word. In order to automatically select an appropriate candidate, the approach picks the word that is most similar to the original word. To measure the closeness between the words, we employ the Lin metric [76] as is it has been shown to best correlate with the human judgements. For the word *application*, we can retrieve words such as *employment*, *patent*, *job*, or *admission*. According to the Lin measure, the word *patent* has the highest similarity score which subsequently leads to the suggestion *application (patent)*. Formally, this strategy can be described as follows:

$$Suggestion_{Lin} = \{w| \max_{w \in gloss_D(w_0, s_0)} \{sim_{Lin}(w_0, w)\}\},$$

where $gloss_D(w_0, s_0)$ denotes a function that retrieves all gloss words of the original word w_0 with the word sense s_0 in a dictionary D and $sim_{Lin}(w_1, w_2)$ a function that calculates the similarity score.

4.2.2 Synonym Resolution

The synonym resolution strategy is motivated by the research of Plachouras et al. [77] and Rijsbergen [78] who measured the generality of a search query in an information retrieval system. The authors argue that the generality of a search query is determined by the number of documents that are retrieved by this query. We adapt this concept in the following way. The search query is interpreted as a word that is looked up in the BabelNet database, while the query results represent the word senses. Accordingly, the more synsets the query word retrieves, the more general it is. The rationale is that a general word can be used in several contexts with a broader set of meanings. In order to finally suggest an alternative word for replacement, the approach determines the word with the minimal number of word senses. For example, consider the synonyms *sales* and *revenue* from Fig. 1. The approach retrieves four senses for sales and six senses for revenue. Accordingly, the word *sales* is chosen as a suggestion to resolve the synonym conflict. This strategy is formalized as follows:

$$Suggestion_S = \{w| \min_{w \in syn(w_0)} |\{Sense_D(w, *)|\},$$

where $syn(w_0)$ denotes the set of all words that have the same meaning as the original word w_0.

5 Evaluation

To demonstrate the capability of the presented techniques, we challenge them against real-world data. More specifically, we test them on three process model collections from practice with a varying degree of standardization. The overall goal of the evaluation is to learn whether the techniques can effectively identify homonyms and synonyms and successfully resolve them by suggesting more specific words. To this end, we study different evaluation dimensions. Section 5.1 provides detailed information about the employed process model collections. Section 5.2 investigates the techniques from a run time perspective. Section 5.3 discusses the evaluation of the identification algorithms. Finally, Section 5.4 elaborates on the evaluation of the resolution algorithms.

TABLE 3
Details of the Test Collections

Characteristic	SAP	TelCo	AI
No. of Models	604	803	1,091
No. of Labels	2,432	12,088	8,339
No. of. Unique Actions	321	728	1,567
No. of. Unique Business Objects	891	3,955	3,268

5.1 Model Repository Demographics

In order to achieve a high external validity, we employ three different model collections from practice. We selected collections differing with regard to standardization, the expected degree of terminological quality, and the domain. Table 3 summarizes their main characteristics. These collections include:

- *SAP reference model*. The *SAP Reference Model* contains 604 Event-Driven Process Chains organized in 29 different functional branches [79]. Examples are procurement, sales, and financial accounting. The model collection includes 2,432 activity labels with 321 unique actions and 891 unique business objects. Since the SAP Reference Model was designed as an industry recommendation with a standardized terminology, we expect a small number of homonyms and synonyms.
- *TelCo collection*. The *TelCo* collection contains the processes from an international telecommunication company. It comprises 803 process models with in total 12,088 activities. We identified 728 unique actions and 3,955 unique business objects. We assume the TelCo collection to contain more heterogeneous terminology as it is not based on a standardized glossary.
- *AI collection*. The models from the AI collection cover diverse domains and stem from academic training (see http://bpmai.org). From the available models, we selected those with proper English labels. The resulting subset includes 1,091 process models with in total 8,339 activity labels. The activities contain 1,567 unique actions and unique 3,268 business objects. Since the collection targets no specific industry and has been mainly created by students, the number of synonyms and homonyms is expected to be the highest among all considered collections.

5.2 Performance Results

The main application scenario of the presented techniques is the automatic assurance of an unambiguous terminology in process model repositories. Thus, the detection and resolution of ambiguous terms is not necessarily time critical. A user may start the algorithms, and assess the terminology conflicts at a later point. However, if the detection and resolution approaches are used in an interactive setting, the computation must be adequately fast. Hence, we tested the

TABLE 4
Average Processing Time for Identification and Resolution

		Characteristic	Avg (s)	Min (s)	Max (s)	Total (s)
Homonyms	SAP	Detection + Resolution	0.03	<0.01	2.12	40.46
		WSD	0.88	<0.01	48.26	1,055.35
	TelCo	Detection + Resolution	0.04	0.01	4.39	206.96
		WSD	1.88	<0.01	253.9	8,750.98
	AI	Detection + Resolution	0.05	0.01	3.79	216.46
		WSD	1.51	<0.01	113.94	7,107.16
Synonyms	SAP	Detection + Resolution	0.81	<0.01	6.01	967.06
		WSD	0.93	<0.01	291.19	1,109.10
	TelCo	Detection + Resolution	4.22	<0.01	54.64	19,634.08
		WSD	1.95	<0.01	480.94	9,055.74
	AI	Detection + Resolution	2.60	<0.01	40.45	12,233.39
		WSD	0.78	<0.01	271.29	3,664.51

resolution technique on a MacBook Pro with a 2.4 GHz Intel Core Duo processor and 4 GB RAM, running on Mac OS X 10.7 and Java Virtual Machine 1.7.

Table 4 summarizes the average, minimum, and maximum execution times of the text generation technique for a single word of a specific collection. The last column indicates the required time to perform a complete run on the collection. We split the processing time into two components for transparency. The results show that the synonym technique consumes significantly more time than the homonym technique, which is due to pairwise comparison. An average detection and resolution run consumes about 0.04 seconds for homonyms and 2.54 seconds for the synonyms. Large deviations from these values are observed only for words that are used in several process models containing a higher number of elements. In general, we learn that the disambiguation of word senses is the most time consuming step. Although the average disambiguation consumes only 1.54 seconds, there are deviations that would be tedious when used in an interactive mode. However, if required, these results can be improved by storing and reusing the calculated supporting scores. Since these scores have to be calculated only if a new process model is created or an existing process model was changed, standard indexing techniques could be used to effectively improve the overall run time.

5.3 Evaluation of Identification

In this section, we discuss the evaluation of the proposed identification techniques. First, Section 5.3.1 elaborates on our evaluation setup. Then, Section 5.3.2 presents the results for the homonym identification. Finally, Section 5.3.3 discusses the results of the synonym identification.

5.3.1 Evaluation Setup

We assess the performance of the identification algorithms by comparing the performance of the basic approach and the advanced approach. As a baseline, the basic approach uses a lexical database such as WordNet to learn whether or not the respective word is ambiguous. The advanced approach has already been described in the previous section, whereby the advanced approach explicitly uses WSD capabilities to decide upon the ambiguity of a word. In order to assess the performance of these approaches, we use precision, recall and the f-measure as metrics [80]. In our context, the precision value is the number of correctly recognized ambiguity cases divided by the total number of ambiguity cases retrieved by the algorithms. The recall is the number of correctly recognized ambiguity cases divided by the total number of ambiguity cases. The f-measure is the harmonic mean between precision and recall.

In order to be able to compute precision and recall for the collections introduced in Section 5.1, we require a benchmark with the human interpretations of the comprised terms. However, due to the high number of terms and process models in the test collections, a complete benchmark would require the manual judgment and annotation of 29,438 potential homonym term-model pairs and 80,609,259 potential synonym term-model pairs. It would be extremely difficult and time consuming to annotate all the homonyms and synonyms in the test collection.

To solve this problem, we draw a random sample from the test collections and apply statistical methods in order to make valid propositions for precision and recall. In particular, we apply the following procedure. We notice that each combination of a term and a process model is either ambiguous or not. Hence, repeatedly drawing instances from our test collections and assessing the ambiguity represents a binomial experiment $X \sim B(n, p)$ with a population size of n and the probability of finding an ambiguity p. Assuming a homonym probability of 0.01 and a synonym probability of 0.0001 for our overall test sample, we follow the recommendations by Berger [81] and Brown et al. [82] and apply the Jeffrey interval, which is most suitable for binomial references from large populations with a small probability p. Aiming for a significance level $\alpha = 0.05$ and a margin of error $\epsilon \leq 0.02$, we draw a random sample of 120 term-model pairs for homonyms and 125 term-model pairs for synonyms based on the sample size calculation formula provided by Piegorsch [83]. Additionally, we created two types of samples. The first sample includes term-model pairs from the algorithmic results, while the second one includes term-model pairs from the overall test set. With this strategy, we assess the performance of the algorithmic output and the overall test set for the synonym as well as the homonym detection technique.

TABLE 5
Identification Results for Homonyms and Synonyms

	Sample	User Assessment		Basic Approach		Adv. Approach		Basic Approach			Adv. Approach		
		AW	NAW	AW	NAW	AW	NAW	P	R	F	P	R	F
Homonyms	Precision	58	62	115	5	104	16	0.5	0.98	0.66	0.55	0.98	0.7
	Recall	26	94	104	16	17	103	0.2	0.84	0.32	0.53	0.35	0.42
Synonyms	Precision	33	95	100	28	36	92	0.29	0.88	0.44	0.56	0.58	0.57
	Recall	2	126	0	128	0	128	-	0	-	-	0	-

AW: Number of Ambiguous Words, NAW: Number of Non-Ambiguous Words P: Precision, R: Recall, F: F-Measure

Taking into account the literature of WSD evaluation (see for example [84], [85]), we asked a set of six native English speakers to provide us with their interpretation of the term-model pairs. We randomly assigned them to the test samples in such a way that each native speaker had to assess one homonym and one synonym sample. Thus, we got three judgments for each sample of term-model pairs. Each participant was provided with a target word and a process models in which this word was highlighted. For homonyms, we randomly selected four word senses and asked participants whether the target word could be used with the respective sense a) in general and b) in the context of the particular process model. For synonyms, we analogously selected alternative terms and asked users whether a replacement of the target term by an alternative term is meaningful a) in general and b) in the context of the particular process model. The participants were asked to provide feedback for each question on a 4-point-Likert-scale from *Strongly agree* to *Strongly disagree*. By using a 4-point-scale, we intentionally forced participants to make a final decision, which is necessary for the calculation of precision and recall. For the evaluation, we considered only the answers that relate to the particular process model and calculated the average of these which marked the term as ambiguous or not.

5.3.2 Homonym Identification Results

The results of the homonym identification approach are summarized in Table 5. The results show that the advanced approach works satisfactorily in comparison to the basic approach. The advanced approach achieves a precision of 53 to 55 percent, while the basic approach is much more unstable and retrieves a considerable high number of false positives (precision: 20 to 50 percent). In contrast however, the recall scores of the basic approach ranges between 84 and 98 percent in the samples. This observation relates directly to the high number of ambiguous words detected by the basic approach (115 in the precision sample and 104 in the recall sample) which apparently include most of the ambiguous words. If we take the f-measure into account, we observe again a higher performance of the advanced approach. The f-measure amounts to 42 percent in the recall sample and 70 percent in the precision sample in contrast to 32 and 66 percent.

We further reflect upon the results by including the nature of the term-model samples into our consideration. In the precision sample, we chose from those term-model pairs that have been retrieved by the algorithm. Thus, this sample included pairs with a fairly good chance of being ambiguous. In such a setting, each approach is equally capable of identifying a considerable number of ambiguities that are also relevant for the user. When looking at the details, we observe that the advanced approach is capable of reducing the number of false positives and keeping the relevant terms at the same time. In the recall sample, we selected term-model pairs from all three process model collections and also included terms that may remain unrecognized by the algorithm. In this case, we observed a low precision and a high recall for the basic approach. In contrast, the results of the advanced approach are more balanced with a moderate precision. However, we observe a surprisingly low recall at the same time. The details revealed that our participants spotted specific word senses to be similar to each other and thus identified an ambiguous term. The algorithm drew a clear distinction between specific word senses and found only one to be appropriate. For example, the participants agreed that the term *link* in the activity *link pairs order to case* complies to the word senses *to be or to become joined or united* as well as *to make a logical or causal connection*, while the algorithm agrees only on the first word sense. Despite this, we still consider the algorithm to be superior to the basic approach and to retrieve more meaningful candidates to the users.

In addition to the quantitative analysis, we discuss qualitative results of the identification approach. Table 6 presents the Top five homonymous actions and business objects among all test collections. The most frequent homonym action is given by *to process*. For this word, our approach identified two dominant word senses. First, the word is used as a general expression to perform a set of operations to create a required output or information. Second, it also identifies the activity of officially delivering a legal notice or summons. For the business objects, the technique identifies the noun *notice* as the most frequent homonym. In this case, it is unclear if the incident refers to a single event or a disturbance, for example in IT applications. These words are good examples of homonyms that are frequently used and should rather be avoided.

5.3.3 Synonym Identification Results

The results of the synonym detection technique are summarized in Table 5. Similarly to the homonym detection approach, the results show that the advanced approach of this paper exceeds the capabilities of the basic approach. In terms of precision, the basic approach achieves 29 percent while the advanced approach reaches at least 56 percent. In terms of recall, the basic approach appears to outperform the advanced approach (88 percent compared to 58

TABLE 6
Top Five of Homonymous Actions and Business Objects

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

percent). As discussed earlier, the high recall value is the result of the extensive number of retrieved instances that are produced by the basic approach. Taking the f-measure into account, we observe that the advanced approach is more balanced (57 percent in contrast to 44 percent).

We also discuss the results with regard to the samples of the user evaluation. In the precision sample, the advanced approach dominates the capabilities of the basic approach. It does not only reduce the number of false positives by a significant amount (36 instances compared to 100 instances), but it also keeps a large share of those synonym pairs that are relevant for the user. In the recall sample, the participants have detected two synonyms which have not been found by the basic or the advanced approach. Accordingly, it is not possible to calculate any values for precision or the f-measure. Interestingly, the users found the term pairs *(fill out, complete and submit)* as well *(document, register and communicate)*. In these cases, the participants interpreted the combination of two distinct verbs as a synonym to the single term. Since neither of these two approaches cannot recognize the combination of verbs correctly, they do not detect a synonym in these cases. Overall, we consider the advanced approach to be superior to the basic one and to retrieve the relevant cases for the user.

In addition to the quantitative perspective, it is again interesting to investigate qualitative examples. Table 7 gives an overview of the Top five synonymous actions and business objects. It is interesting to note that the approach identified a group of three pairwise synonym words. The words *make*, *create*, and *produce* all refer to the activity of manufacturing products. For the business objects, the technique identifies *customer*, *market* and *client* as synonyms for persons that pay for goods and services. Another example are the synonyms *purchase order* and *order* which refer to a commercial delivery document. Overall, these examples also illustrate the capability of our approach to identify synonyms that should be avoided.

5.4 Evaluation of Resolution

In this section, we discuss the evaluation of the resolution strategies. First, Section 5.4.1 elaborates on our evaluation

TABLE 7
Top Five of Synonymous Actions and Business Objects

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

setup. Then, Section 5.4.2 presents the results for the homonym resolution. Finally, Section 5.4.3 discusses the results of the synonym resolution.

5.4.1 Evaluation Setup

We assess the performance of the resolution strategies by comparing the degree of ambiguity before and after applying it to the test collections. Since we do not aim at a fully automatic resolution of ambiguity and still require humans to perform this task, we can abstract from the suggestions themselves and turn to the effects of ambiguity resolution, i.e., a reduction of ambiguity in the test collections. Therefore, we define metrics that measure the basic characteristics of homonyms or synonyms respectively in order to operationalize the effects of ambiguity.

Since a homonym represents a word referring to multiple senses, we measure the degree of homonymy by using the number of senses per word. Given a set of process models P, we capture the degree of homonymy as follows:

$$SpW^P = \frac{|\bigcup_{w \in \mathcal{L}^P} Sense_D(w,*)|}{|\mathcal{L}^P|}. \tag{1}$$

Differentiating between action and business object, we further calculate the number of senses per word for actions (SpW^P_A) and for business objects (SpW^P_{BO}):

$$SpW^P_A = \frac{|\bigcup_{w \in \mathcal{L}_A} Sense_D(w,v)|}{|\mathcal{L}_A|} \tag{2}$$

$$SpW^P_{BO} = \frac{|\bigcup_{w \in \mathcal{L}_{BO}} Sense_D(w,n)|}{|\mathcal{L}_{BO}|}. \tag{3}$$

For the degree of synonymy, we calculate the number of words for each word sense. Given a process model set P, the degree of synonymy is defined as follows:

$$WpS^P = \frac{|\mathcal{L}^P|}{|\bigcup_{w \in \mathcal{L}^P} Sense_D(w,*)|}. \tag{4}$$

Differentiating between action and business object, we calculate the number of words per sense for actions (WpS^P_A) and for business objects (WpS^P_{BO}):

$$WpS^P_A = \frac{|\mathcal{L}_A|}{|\bigcup_{w \in \mathcal{L}_A} Sense_D(w,v)|} \tag{5}$$

$$WpS^P_{BO} = \frac{|\mathcal{L}_{BO}|}{|\bigcup_{w \in \mathcal{L}_{BO}} Sense_D(w,n)|}. \tag{6}$$

5.4.2 Homonym Resolution Results

Table 8 summarizes the results for the introduced metrics. It illustrates the significant effect of the homonym resolution technique for both, actions and business objects. The biggest effect can be seen for the TelCo collection. In the TelCo collection, the number of senses per action is reduced from 6.27 to 1.47 and the number of senses per business object is reduced from 8.96 to 1.76. However, similar results can be observed for the SAP and AI collection. Fig. 3 illustrates this effect by showing the senses per word distribution for all collections. The distribution figures demonstrate that the technique reduces the number of words with multiple sense. As a result of its application, almost all words with more than five senses are replaced. Consequently, the technique also significantly increases the number of words with a single sense. Thus, the degree of homonymy is considerably reduced. Note that not every word having multiple senses is necessarily ambiguous. Many words are associated with senses that a closely related. Hence, the technique will never result in a sense distribution in which every word points to a single sense.

TABLE 8
Results of Homonym Resolution

		SAP	**TelCo**	**AI**
SpW^P	Before	6.79	8.67	7.29
	After	1.71	1.47	1.86
SpW^P_A	Before	5.37	6.27	5.70
	After	1.83	1.47	1.54
SpW^P_{BO}	Before	7.31	8.96	7.60
	After	1.66	1.76	1.92

In addition to the quantitative results, we provide resolution examples for each of the Top five homonyms (see Table 9). For the sake of readability, we list only the first hyponym that is suggested. In general, the resolution strategies are capable of creating a decent number of suggestions to resolve the ambiguous words. However, we also note differences in the performance of these strategies. For the hyponym strategy, we note that it fails to retrieve hyponyms in eight cases due to the fact that there are simply no hyponyms existing. Looking at the suggestions, we observe that the suggestions are quite specific for most of the cases and might miss the originally intended word senses. The hypernym strategy fails in only two cases. Moreover, if we compare the word sense and the suggestion, we more likely tend to accept the suggestion as it best describes the intended word sense. The Lin strategy is always capable to create a suggestion. However, we also notice that it fails in nine cases. This typically occurs when each gloss word has a similarity score of 0. Nevertheless, we can consider the remaining Lin suggestions as meaningful suggestions when combining it with the other strategies.

5.4.3 Synonym Resolution Results

Table 10 summarizes the results for the introduced metrics. From the data we learn that the technique decreases the degree of synonymy for all three collections. The biggest effect can be observed for the AI collection. In this collection, the WpS^P_A is reduced by 0.036 and the WpS^P_{BO} is reduced by 0.08. While we yield comparable results for the TelCo collection, the effect for the SAP collection is significantly smaller. In the SAP collection, the WpS^P_A metric deceases by 0.04 and the WpS^P_{BO} decreases by 0.05. These differences are a result of the lower ex ante degree of synonymy of the SAP collection. The, in general, small absolute differences can be explained by the fact that many words do

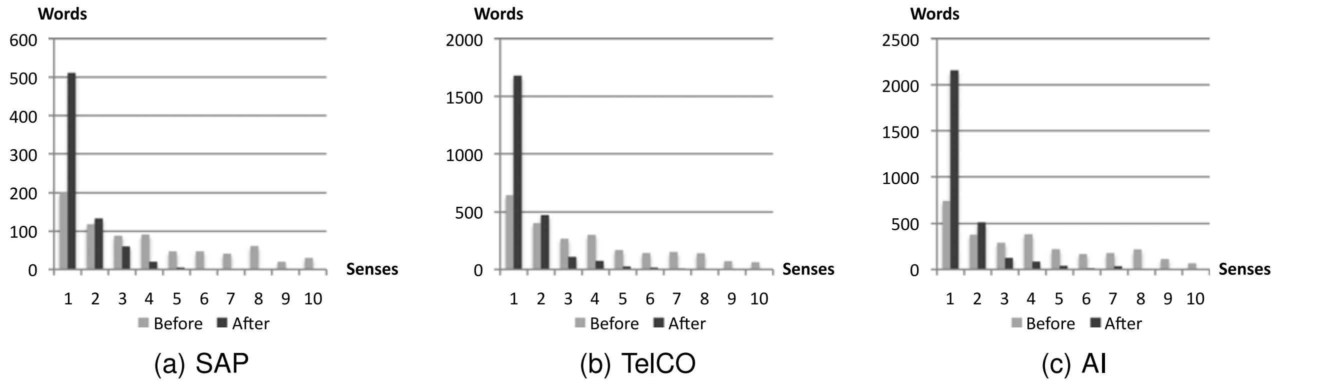


Fig. 3. Senses per word distribution before and after homonym resolution.

not have synonyms, i.e., their words per sense value is one. Those words having synonyms typically have one or two. Hence, the effect of resolving a synonym results in a rather small difference of the metric. However, the numbers still show the positive effect of the synonymy resolution technique. Fig. 4 visualizes the results by showing the words per sense distribution. It confirms the explanation for the small absolute values. Only a few senses are represented by more than one word. Again note that not every case with $WpS > 1$ is necessarily ambiguous. Hence, not all of them are replaced.

We also discuss the qualitative results of the synonym resolution strategy. Similar to the homonym case, we create the resolution suggestions for the Top five synonymous actions and business objects as depicted in Table 11. The table shows on the one hand that the strategy is always capable of creating a suggestion based on the specificity of a word. On the other hand, we can distinguish two cases. First, the strategy can suggest a new alternative that comprises two words, such as the verb *go over* for the synonym pair survey and review. Second, it can select the most unambiguous alternative among two words, as for example for the words invoice and account. In general, we could not find counter-intuitive suggestions which lets us conclude that the specificity strategy fulfills its purpose.

5.4.4 Joint Application Results

The homonym as well as the synonym resolution technique modify the content of the investigated collections. In this section, we investigate whether the results are negatively affected by applying the proposed techniques in sequence. Table 12 summarizes the results.

In general, it must be stated that the techniques appear to be complementary. The results show that the joint application of the resolution techniques reduces the degree of synonymy and homonymy and hence improves the result. As there are overlaps between cases of homonymy and synonymy, i.e., there are words that are synonyms and homonyms at the same time, this is a desirable result. In addition, the data illustrates the importance of application order. Considering the metrics SpW_A^P and WpS_A^P, it can be seen that resolving synonymy first yields better results. However, the values of the metric SpW_{BO}^P suggest the opposite.

TABLE 9
Resolution Suggestions for the Top Five Homonymous Actions and Business Objects

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

TABLE 10
Results of Synonym Resolution

		SAP	TelCo	AI
WpS^P	Before	1.087	1.083	1.089
	After	1.075	1.072	1.076
WpS^P_A	Before	1.100	1.187	1.191
	After	1.096	1.155	1.155
WpS^P_{BO}	Before	1.082	1.062	1.067
	After	1.077	1.055	1.059

In general, the synonymy resolution technique may create new homonyms, but it is very unlikely that the homonymy resolution creates new synonyms since it typically introduces compounds. Hence, the synonymy resolution should be executed first. This is also emphasized by the negligibly small differences of the SpW metrics.

6 Discussion

In this section, we discuss the implications of our research for research and practice. Finally, we reflect upon the limitations of our work.

6.1 Implications for Research

The results of this research have implications for software and requirements engineering, the quality assurance of conceptual models, and for conceptual model analysis techniques.

In the domain of software engineering, conceptual models are frequently used for formally specifying system requirements [86]. These models have to be consistent and correct in order to avoid risks for the software development project [6]. The proposed techniques provide means for supporting the software and requirements engineering process by automatically detecting ambiguous terminology in formal requirements documents and by automatically proposing more accurate terms. As a result, misconceptions of requirements are avoided and the likelihood of a successful completion of the software engineering initiative is increased. It should be noted that the application of our technique is not limited to process models. Since also other conceptual models use natural language fragments in the same fashion as process models [87], [88], [89], our technique can be also applied to goal models, feature diagrams, or use case diagrams.

For assuring the quality of conceptual models, several guidelines and frameworks have been proposed [90], [91], [92], [93]. Building on the requirements of such guidelines, many techniques have been introduced that check and correcting these guidelines in an automatic fashion. Among others, this includes techniques for checking and correcting syntactical aspects of natural language in conceptual models [10], [94], [95]. The research in this paper complements these techniques by addressing the semantic dimension of natural language. Thus, it represents an important step towards fully automated quality assurance of conceptual models.

There are several techniques that automatically analyze conceptual models for different purposes. Examples include techniques for computing the overall similarity between two models [96], [97], for the automatic identification of activity correspondences between two process models [98], [99], [100], for the identification of service candidates from process models [101], [102], [103] and for the automatic assessment of diagrams in an educational scenario [104]. What all these techniques have in common is that they need to syntactically compare the natural language content of the analyzed models. Our technique can improve the performance of these techniques by first assuring a consistent usage of language. As a result, model analysis techniques can avoid an erroneous association of semantically unrelated words.

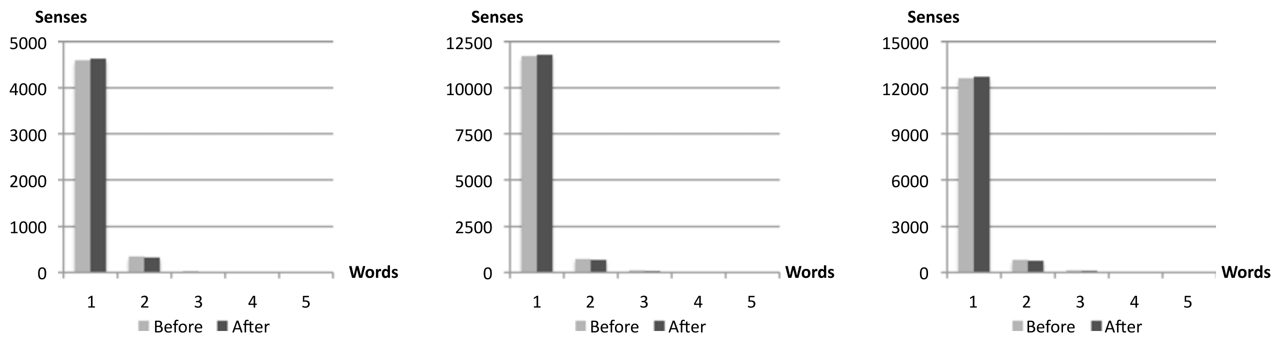
6.2 Implications for Practice

The results of this paper have considerable implications for practice. Most importantly, the proposed techniques can be integrated into commercial modeling tools. In such a context, it can point modelers to ambiguous words and automatically generate more accurate alternatives among which the modeler may choose. As a result, linguistic quality issues can be avoided right from the start.

For already existing model repositories, our technique can help modelers in cleaning up terminology. Given the size of model repositories in practice with thousands of models [2], our techniques make an important contribution to the effective and efficient model management. This is particularly useful when multiple modelers create models concurrently.

6.3 Limitations

The findings from this paper are subject to some limitations. In particular, we discuss the representativeness of the collections, the transfer to other conceptual models, and user evaluation.



(a) SAP (b) TelCO (c) AI

Fig. 4. Word per sense distribution before and after synonym resolution.

TABLE 11
Top Five of Synonymous Actions and Business Objects

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

The three process model collections can hardly be seen as *representative* in a statistical sense. Therefore, we cannot completely rule out that other model collections would yield different results. We tried to minimize this risk by selecting process model collections that vary along different dimensions such as degree of standardization, domain, and size. Hence, we are confident that the successful application of our techniques is not limited to a particular set of models.

We evaluated the techniques using only process models. Consequently, the applicability to other conceptual models might still require further demonstration. Nevertheless, related work reveals that the naming conventions for other conceptual models are identical to those of process models. Examples include naming conventions of goal models [87], feature diagrams [88], and use case diagrams [89]. Since our technique uses only a set of labels as input, this provides us with confidence that the presented techniques are applicable also to other conceptual models.

Although the findings suggest the usefulness of the technique, there is still a need to conduct an evaluation in an end user scenario from practice. Nevertheless, we know that the detection and resolution tasks are typically conducted by human reviewers who perceive it as time-consuming, ineffective and boring [105], [106]. Since our techniques can reduce the detection time significantly and increase the effectiveness of resolving ambiguities, we are confident that human reviewers will find these techniques useful to fulfill this task.

TABLE 12
Results of Joint Application

		SAP	TelCo	AI
SpW^P	Homonym only	1.71	**1.47**	1.86
	Synonym-Homonym	1.53	1.49	1,64
	Homonym-Synonym	**1.53**	1.52	**1,61**
SpW^P_A	Homonym only	1.83	1.47	1.54
	Synonym-Homonym	**1.54**	**1.24**	1.35
	Homonym-Synonym	1.64	1.30	**1.34**
SpW^P_{BO}	Homonym only	1.66	1.76	1.92
	Synonym-Homonym	1.53	**1.52**	1.68
	Homonym-Synonym	**1.49**	1.54	**1.67**
WpS^P	Synonym only	1.076	1.072	1.077
	Synonym-Homonym	**1.072**	**1.068**	**1.070**
	Homonym-Synonym	1.077	1.073	1.077
WpS^P_A	Synonym only	1.096	1.155	1.155
	Synonym-Homonym	**1.056**	**1.102**	**1.094**
	Homonym-Synonym	1.071	1.158	1.155
WpS^P_{BO}	Synonym only	1.077	**1.055**	1.059
	Synonym-Homonym	**1.077**	1.062	1.067
	Homonym-Synonym	1.079	1.056	**1.061**

7 Conclusion

In this paper, we addressed the problem of automatically identifying and resolving homonyms and synonyms in conceptual models. Using a literature review, we identified that current techniques cannot be easily transferred to models as they cannot deal with their specific characteristics. In particular, the shortness of the natural language text labels represents a major challenge. In order to adequately address this issue, we introduced a technique that operationalizes and exploits the model context. Moreover, we proposed necessary and sufficient conditions that allow us to automatically identify truly ambiguous homonyms and synonyms. Using the lexical database BabelNet, we further implemented different resolution strategies, which automatically suggest replacement terms. The evaluation with English native speakers illustrated that the technique identifies a significant number of homonyms and synonyms within the test collections. The introduced metrics *Sense per Word* and *Word per Sense* further highlighted the positive effect of the resolution approach. As a result, the overall ambiguity could be significantly reduced.

In future research, we first aim to transfer our technique to a professional environment. In such an environment, process modelers are capable to use our technique complement to their work with modeling tools in order to refine the terminology within processes. We are currently discussing different opportunities with industry partners that already started with this task. Second, we aim to study the performance of the techniques when applied for other types of models. Since, for instance, goal models have similar characteristics with regard to natural language, the techniques should be readily applicable. Therefore, the research presented in this paper can be regarded as an important step towards the automatic quality assurance of conceptual models altogether.

Acknowledgments

F. Pittke is the corresponding author.

REFERENCES

[1] I. Davies, P. Green, M. Rosemann, M. Indulska, and S. Gallo, "How do practitioners use conceptual modeling in practice?" *Data Knowl. Eng.*, vol. 58, no. 3, pp. 358–380, 2006.

[2] M. Rosemann, "Potential pitfalls of process modeling: Part A," *Bus. Process Manage. J.*, vol. 12, no. 2, pp. 249–254, 2006.

[3] J. Mendling, "Empirical studies in process model verification," in *Transaction on Petri Nets and Other Models of Concurrency II*, vol. 2. New York, NY, USA: Springer, 2009, pp. 208–224.

[4] R. M. Dijkman, M. L. Rosa, and H. A. Reijers, "Managing large collections of business process models—Current techniques and challenges," *Comput. Ind.*, vol. 63, no. 2, pp. 91–97, 2012.

[5] B. W. Boehm, "Understanding and controlling software costs," *J. Parametrics*, vol. 8, no. 1, pp. 32–68, 1988.

[6] A. Abran, P. Bourque, R. Dupuis, J. W. Moore, and L. L. Tripp, *Guide to the Software Engineering Body of Knowledge-SWEBOK*, 2004th ed. Piscataway, NJ, USA: IEEE Press, 2004.

[7] W. M. P. van der Aalst, "Workflow verification: Finding control-flow errors using petri-net-based techniques," in *Business Process Management*, vol. 1806. New York, NY, USA: Springer, 2000, pp. 161–183.

[8] I. Weber, J. Hoffmann, and J. Mendling, "Beyond soundness: On the verification of semantic business process models," *Distrib. Parallel Databases*, vol. 27, no. 3, pp. 271–343, 2010.

[9] H. Leopold, S. Smirnov, and J. Mendling, "On the refactoring of activity labels in business process models," *Inf. Syst.*, vol. 37, no. 5, pp. 443–459, 2012.

[10] H. Leopold, R.-H. Eid-Sabbagh, J. Mendling, L. G. Azevedo, and F. A. Baião, "Detection of naming convention violations in process models for different languages," *Decision Support Syst.*, vol. 56, pp. 310–325, 2014.

[11] E. Kamsties, "Understanding ambiguity in requirements engineering," in *Engineering and Managing Software Requirements*, A. Aurum and C. Wohlin, Eds. New York, NY, USA: Springer, 2005, pp. 245–266.

[12] D. M. Berry, E. Kamsties, and M. M. Krieger. (2003). From contract drafting to software specification: Linguistic sources of ambiguity—A handbook [Online]. Available: http://se.uwaterloo.ca/ dberry/handbook/ambiguityHandbook.pdf

[13] N. Bolloju and F. S. K. Leung, "Assisting novice analysts in developing quality conceptual models with UML," *Commun. ACM*, vol. 49, no. 7, pp. 108–112, 2006.

[14] A. Jayal and M. Shepperd, "The problem of labels in e-assessment of diagrams," *J. Edu. Resources Comput.*, vol. 8, no. 4, p. 12, 2009.

[15] F. Törner, M. Ivarsson, F. Pettersson, and P. Öhman, "Defects in automotive use cases," in *Proc. IEEE Int. Symp. Emperical Softw. Eng.*, 2006, pp. 115–123.

[16] K. Cox and K. Phalp, "Replicating the CREWS use case authoring guidelines experiment," *Empirical Softw. Eng.*, vol. 5, no. 3, pp. 245–267, 2000.

[17] M. Wimmer, G. Kappel, A. Kusel, W. Retschitzegger, J. Schönböck, and W. Schwinger, "From the heterogeneity jungle to systematic benchmarking," in *Proc. Int. Conf. Models Softw. Eng.*, 2011, pp. 150–164.

[18] N. Moha, Y.-G. Guéhéneuc, L. Duchien, and A.-F. Le Meur, "DECOR: A method for the specification and detection of code and design smells," *IEEE Trans. Softw. Eng.*, vol. 36, no. 1, pp. 20–36, Jan./Feb. 2010.

[19] A. Marcus, J. I. Maletic, and A. Sergeyev, "Recovery of traceability links between software documentation and source code," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 15, no. 5, pp. 811–836, 2005.

[20] T. Lehmann, "A framework for ontology based integration of structured IT-systems," in *Proc. Workshop Semantic Web Enabled Softw. Eng.*, 2007, pp. 1–15.

[21] A. Mili, R. Mili, and R. T. Mittermeir, "A survey of software reuse libraries," *Ann. Softw. Eng.*, vol. 5, pp. 349–414, 1998.

[22] F. Pittke, H. Leopold, and J. Mendling, "Spotting terminology deficiencies in process model repositories," in *Enterprise, Business-Process and Information Systems Modeling*. New York, NY, USA: Springer, 2013, pp. 292–307.

[23] J. Gordijn, H. Akkermans, and H. Van Vliet, "Business modelling is not process modelling," in *Proc. Int. Workshops Conceptual Model. e-Business Web*, 2000, pp. 40–51.

[24] T. H. Davenport and J. E. Short, "The new industrial engineering: Information technology and business process redesign," *Sloan Manage. Rev.*, vol. 31, no. 4, pp. 11–27, 1990.

[25] M. Dumas, W. M. Van der Aalst, and A. H. Ter Hofstede, *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Hoboken, NJ, USA: Wiley, 2005.

[26] H. Mili, G. Tremblay, G. B. Jaoude, É. Lefebvre, L. Elabed, and G. E. Boussaidi, "Business process modeling languages: Sorting through the alphabet soup," *ACM Comput. Surveys*, vol. 43, no. 1, p. 4, 2010.

[27] C. Denger, D. M. Berry, and E. Kamsties, "Higher quality requirements specifications through natural language patterns," in *Proc. IEEE Int. Softw.: Sci., Technol. Eng.*, 2003, pp. 80–90.

[28] J. Barjis, "The importance of business process modeling in software systems design," *Sci. Comput. Program.*, vol. 71, no. 1, pp. 73–87, 2008.

[29] J. Mendling, H. A. Reijers, and J. Recker, "Activity labeling in process modeling: Empirical insights and recommendations," *Inf. Syst.*, vol. 35, no. 4, pp. 467–482, 2010.

[30] E. Kamsties and B. Peach, "Taming ambiguity in natural language requirements," in *Proc. 13th Int. Conf. Softw. Syst. Eng. Appl.*, 2000, pp. 89–101.

[31] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami, "Quality evaluation of software requirement specifications," in *Proc. Softw. Internet Quality Week*, 2000, pp. 1–18.

[32] F. Chantree, B. Nuseibeh, A. De Roeck, and A. Willis, "Identifying nocuous ambiguities in natural language requirements," in *Proc. IEEE 14th Int. Conf. Requirements Eng.*, 2006, pp. 59–68.

[33] B. Gleich, O. Creighton, and L. Kof, "Ambiguity detection: Towards a tool explaining ambiguity sources," in *Requirements Engineering: Foundation for Software Quality*. New York, NY, USA: Springer, 2010, pp. 218–232.

[34] J. C. Sampaio do Prado Leite, and P. A. Freeman, "Requirements validation through viewpoint resolution," *IEEE Trans. Softw. Eng.*, vol. 17, no. 12, pp. 1253–1269, Dec. 1991.

[35] M. Attaran, "Exploring the relationship between information technology and business process reengineering," *Inf. Manage.*, vol. 41, no. 5, pp. 585–596, 2004.

[36] E. Kamsties, D. M. Berry, and B. Paech, "Detecting ambiguities in requirements documents using inspections," in *Proc. 1st Workshop Inspections Softw. Eng.*, 2001, pp. 68–80.

[37] F. Shull, G. H. Travassos, J. Carver, and V. R. Basili, "Evolving a set of techniques for OO inspections," University of Maryland Tech. Rep. CS-TR- 4070, pp. 1–36, 1999.

[38] C. Rolland and C. Ben Achour, "Guiding the construction of textual use case specifications," *Data Knowl. Eng.*, vol. 25, no. 1, pp. 125–160, 1998.

[39] A. Fantechi, S. Gnesi, G. Lami, and A. Maccari, "Applications of linguistic techniques for use case analysis," *Requirements Eng.*, vol. 8, no. 3, pp. 161–170, 2003.

[40] M. Ceccato, N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry, "Ambiguity identification and measurement in natural language texts," Dept. Inf. Eng. Comput. Sci., Univ. Trento, Trento, Italy, Tech. Rep. DIT-04-111, 2004.

[41] Y. Wang, I. L. M. Gutièrrez, K. Winbladh, and H. Fang, "Automatic detection of ambiguous terminology for software requirements," in *Natural Language Processing Information Systems*. New York, NY, USA: Springer, 2013, pp. 25–37.

[42] H. Yang, A. Willis, A. De Roeck, and B. Nuseibeh, "Automatic detection of nocuous coordination ambiguities in natural language requirements," in *Proc. IEEE/ACM Int. Conf. Autom. Softw. Eng.*, 2010, pp. 53–62.

[43] H. Yang, A. De Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, "Analysing anaphoric ambiguity in natural language requirements," *Requirements Eng.*, vol. 16, no. 3, pp. 163–189, 2011.

[44] B. Anda and D. I. Sjøberg, "Towards an inspection technique for use case models," in *Proc. 14th Int. Conf. Softw. Eng. Knowl. Eng.*, 2002, pp. 127–134.

[45] T. Mens, R. Van Der Straeten, and M. D'Hondt, "Detecting and resolving model inconsistencies using transformation dependency analysis," in *Proc. 9th Int. Conf. Model Driven Eng. Lang. Syst.*, 2006, pp. 200–214.

[46] T. Mens, G. Taentzer, and O. Runge, "Analysing refactoring dependencies using graph transformation," *Softw. Syst. Model.*, vol. 6, no. 3, pp. 269–285, 2007.

[47] E. Agirre and D. Martinez, "Exploring automatic word sense disambiguation with decision lists and the web," in *Proc. COLING Workshop Semantic Annotation Intell. Content*, 2000, pp. 11–19.

[48] J. R. Quinlan, *C4. 5: Programs for Machine Learning*, vol. 1. San Mateo, CA, USA: Morgan Kaufmann, 1993.
[49] J. Véronis and N. Ide, "Large neural networks for the resolution of lexical ambiguity," in *Computer Lexical Semantics Natural Language Processing Series*. Cambridge, U.K.: Cambridge Univ. Press, 1995, pp. 251–269.
[50] R. Navigli and S. P. Ponzetto, "Joining forces pays off: Multilingual joint word sense disambiguation," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn.*, 2012, pp. 1399–1410.
[51] H. Schütze, "Automatic word sense discrimination," *Comput. Linguistics*, vol. 24, no. 1, pp. 97–123, 1998.
[52] D. Lin, "Automatic retrieval and clustering of similar words," in *Proc. 17th Int. Conf. Comput. Linguistics*, 1998, pp. 768–774.
[53] P. Pantel and D. Lin, "Discovering word senses from text," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2002, pp. 613–619.
[54] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, 1975.
[55] H. Schütze, "Dimensions of meaning," in *Proc. ACM/IEEE Conf. Supercomput.*, 1992, pp. 787–796.
[56] F. Deissenboeck and M. Pizka, "Concise and consistent naming," *Softw. Quality Control*, vol. 14, no. 3, pp. 261–282, 2006.
[57] W. B. Frakes and T. P. Pole, "An empirical study of representation methods for reusable software components," *IEEE Trans. Softw. Eng.*, vol. 20, no. 8, pp. 617–630, Aug. 1994.
[58] X. Lin and L. M. Chan, "Personalized knowledge organization and access for the web," *Library Inf. Sci. Res.*, vol. 21, no. 2, pp. 153–172, 1999.
[59] M. Bouzeghoub, G. Gardarin, and E. Métais, "Database design tools: An expert system approach," in *Proc. 11th Int. Conf. Very Large Data Bases*, 1985, pp. 82–95.
[60] S. Hayne and S. Ram, "Multi-user view integration system (MUVIS): An expert system for view integration," in *Proc. 6th Int. Conf. Data Eng.*, 1990, pp. 402–409.
[61] D. Ben-Ari, D. M. Berry, and M. Rimon, "Translational ambiguity rephrased," in *Proc. 2nd Int. Conf. Theoretical Methodol. Issues Mach. Translation Natural Lang.*, 1988, pp. 175–183.
[62] R. Navigli, "Word sense disambiguation: A survey," *ACM Comput. Surveys*, vol. 41, no. 2, pp. 1–69, 2009.
[63] Z. Zhong and H. T. Ng, "Word sense disambiguation improves information retrieval," in *Proc. 50th Annu. Meeting Assoc. Comput. Linguistics*, 2012, pp. 273–282.
[64] H. Leopold, "Natural language in business process models," Ph.D. dissertation, Dept. Inf. Syst. Oper., Humboldt Universität zu Berlin, Berlin, Germany, 2013.
[65] A. S. Hornby, A. P. Cowie, A. C. Gimson, and J. W. Lewis, *Oxford Advanced Learner's Dictionary of Current English*, vol. 1428, Cambridge, U.K.: Cambridge Univ. Press, 1974.
[66] A. Stevenson, *Oxford Dictionary of English*. London, U.K.: Oxford Univ. Press, 2010.
[67] P. Procter, "Longman dictionary of contemporary English," London, Longman Group Limited, 1981.
[68] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[69] G. Miller and C. Fellbaum, *WordNet: An Electronic Lexical Database*. Cambridge, MA, USA: MIT Press, 1998.
[70] R. Navigli and S. P. Ponzetto, "Multilingual WSD with just a few lines of code: The BabelNet API," in *Proc. ACL Syst. Demonstrations*, 2012, pp. 67–72.
[71] D. McCarthy, R. Koeling, J. Weeds, and J. Carroll, "Finding predominant word senses in untagged text," in *Proc. 42nd Annu. Meeting Assoc. Comput. Linguistics*, 2004, p. 279.
[72] D. McCarthy, R. Koeling, J. Weeds, and J. Carroll, "Unsupervised acquisition of predominant word senses," *Comput. Linguistics*, vol. 33, no. 4, pp. 553–590, 2007.
[73] S. P. Ponzetto and R. Navigli, "Knowledge-rich word sense disambiguation rivaling supervised systems," in *Proc. 48th Annu. Meeting Assoc. Comput. Linguistics*, 2010, pp. 1522–1531.
[74] D. Pelleg and A. W. Moore, "X-means: Extending K-means with efficient estimation of the number of clusters," in *Proc. 17th Int. Conf. Mach. Learn.*, 2000, pp. 727–734.
[75] C. L. Heitmeyer, R. D. Jeffords, and B. G. Labaw, "Automated consistency checking of requirements specifications," *ACM Trans. Softw. Eng. Methodol.*, vol. 5, no. 3, pp. 231–261, 1996.
[76] D. Lin, "An information-theoretic definition of similarity," in *Proc. 15th Int. Conf. Mach. Learn.*, 1998, pp. 296–304.
[77] V. Plachouras, F. Cacheda, I. Ounis, and C. J. V. Rijsbergen, "University of Glasgow at the web track: Dynamic application of hyperlink analysis using the query scope," in *Proc. Text Retrieval Conf.*, 2003, pp. 636–642.
[78] C. J. van Rijsbergen, *Information Retrieval*, 2nd ed. London, U.K.: Butterworths, 1979.
[79] G. Keller and T. Teufel, *SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping*. Reading, MA, USA: Addison-Wesley, 1998.
[80] R. A. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Reading, MA, USA: Addison-Wesley, 1999.
[81] J. O. Berger, *Statistical Decision Theory and Bayesian Analysis*. New York, NY, USA: Springer, 1985.
[82] L. D. Brown, T. T. Cai, and A. DasGupta, "Interval estimation for a binomial proportion," *Statist. Sci.*, vol. 16, pp. 101–117, 2001.
[83] W. W. Piegorsch, "Sample sizes for improved binomial confidence intervals," *Comput. Statist. Data Anal.*, vol. 46, no. 2, pp. 309–316, 2004.
[84] A. Kilgarriff, "Gold standard datasets for evaluating word sense disambiguation programs," *Comput. Speech Lang.*, vol. 12, no. 4, pp. 453–472, 1998.
[85] P. Resnik and D. Yarowsky, "Distinguishing systems and distinguishing senses: New evaluation methods for word sense disambiguation," *Natural Lang. Eng.*, vol. 5, no. 2, pp. 113–133, 1999.
[86] E. C. Cardoso, J. P. A. Almeida, and G. Guizzardi, "Requirements engineering based on business process models: A case study," in *Proc. 13th Enterprise Distrib. Object Comput. Conf.*, 2009, pp. 320–327.
[87] C. Rolland, C. Souveyet, and C. Achour, "Guiding goal modeling using scenarios," *IEEE Trans. Softw. Eng.*, vol. 24, no. 12, pp. 1055–1071, Dec. 1998.
[88] K. Lee, K. C. Kang, and J. Lee, "Concepts and guidelines of feature modeling for product line software engineering," in *Proc. 7th Int. Conf. Softw. Reuse: Methods, Tech., Tools*, 2002, pp. 62–77.
[89] A. Cockburn, *Writing Effective Use Cases*, vol. 1. Reading, MA, USA: Addison-Wesley, 2001.
[90] J. Mendling, H. A. Reijers, and W. M. P. van der Aalst, "Seven process modeling guidelines (7PMG)," *Inf. Softw. Technol.*, vol. 52, no. 2, pp. 127–136, 2010.
[91] R. Schuette and T. Rotthowe, "The guidelines of modeling—An approach to enhance the quality in information models," in *Proc. 17th Int. Conf. Conceptual Model.*, 1998, pp. 240–254.
[92] O. I. Lindland, G. Sindre, and A. Solvberg, "Understanding quality in conceptual modeling," *IEEE Softw.*, vol. 11, no. 2, pp. 42–49, Mar. 1994.
[93] J. Becker, M. Rosemann, and C. Uthmann, "Guidelines of business process modeling," in *Business Process Management, Models, Techniques, and Empirical Studies*. New York, NY, USA: Springer, 2000, pp. 30–49.
[94] J. Becker, P. Delfmann, S. Herwig, L. Lis, and A. Stein, "Towards increased comparability of conceptual models—Enforcing naming conventions through domain thesauri and linguistic grammars," in *Proc. Eur. Conf. Inf. Syst.*, Jun. 2009, pp. 2636–2647.
[95] P. Delfmann, S. Herwig, L. Lis, and A. Stein, "Supporting distributed conceptual modelling through naming conventions—A tool-based linguistic approach," *Enterprise Model. Inf. Syst. Archit.*, vol. 4, no. 2, pp. 3–19, 2009.
[96] M. Ehrig, A. Koschmider, and A. Oberweis, "Measuring similarity between semantic business process models," in *Proc. 4th Asia-Pacific Conf. Comceptual Model.*, 2007, pp. 71–80.
[97] R. M. Dijkman, M. Dumas, B. F. van Dongen, R. Käärik, and J. Mendling, "Similarity of business process models: Metrics and evaluation," *Inf. Syst.*, vol. 36, no. 2, pp. 498–516, 2011.
[98] J. Becker, D. Breuker, P. Delfmann, H.-A. Dietrich, and M. Steinhorst, "Identifying business process activity mappings by optimizing behavioral similarity," in *Proc. Amer. Conf. Inf. Syst.*, 2012, pp. 1–10.
[99] R. M. Dijkman, M. Dumas, and L. García-Bañuelos, "Graph matching algorithms for business process model similarity search," in *Proc. 7th Int. Conf. Bus. Process Manage.*, 2009, pp. 48–63.
[100] H. Leopold, M. Niepert, M. Weidlich, J. Mendling, R. M. Dijkman, and H. Stuckenschmidt, "Probabilistic optimization of semantic process model matching," in *Proc. 10th Int. Conf. Bus. Process Manage.*, 2012, pp. 319–334.

[101] L. G. Azevedo, F. Santoro, F. Baião, J. Souza, K. Revoredo, V. Pereira, and I. Herlain, "A method for service identification from business process models in a SOA approach," in *Enterprise, Business-Process and Information Systems Modeling*, vol. 29. New York, NY, USA: Springer, 2009, pp. 99–112.

[102] R. Knackstedt, D. Kuropka, and O. Müller, "An ontology-based service discovery approach for the provisioning of product-service bundles," in *Proc. Eur. Conf. Inf. Syst.*, 2008, pp. 1965–1977.

[103] H. Leopold and J. Mendling, "Automatic derivation of service candidates from business process model repositories," in *Business Information Systems*. New York, NY, USA: Springer, 2012, pp. 84–95.

[104] A. Jayal and M. Shepperd, "An improved method for label matching in e-assessment of diagrams," *Innovation Teaching Learning Inf. Comput. Sci.*, vol. 8, no. 1, pp. 3–16, 2009.

[105] G. Lami, "QuARS: A tool for analyzing requirements," Softw. Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-2005-TR-014, 2005.

[106] V. Ambriola and V. Gervasi, "On the systematic analysis of natural language requirements with CIRCE," *Autom. Softw. Eng.*, vol. 13, no. 1, pp. 107–167, 2006.

Fabian Pittke received the master's degree in business informatics from the Institute of Information Systems (IWI), Universitat des Saarlandes, Germany, in 2010. He is currently working toward the PhD degree at WU, Vienna. He has been a research fellow with Humboldt-Universitat zu Berlin until 2012 and is, since then, an external research fellow at the Institute for Information Business at Wirtschaftsuniversitat Wien. His research interests include business process models and natural language processing.

Henrik Leopold received the bachelor's degree from the Berlin School of Economics, the master's and the PhD (Dr. rer. pol.) degrees from Humboldt University, Berlin, all in information systems. He is currently an assistant professor at the Department of Computer Science, VU University Amsterdam. In July 2013, he completed his doctoral thesis on Natural Language in Business Process Models. Before his graduation, he was with several departments of the pharmaceutical division of Bayer in Germany and the US. His current research interests include business process management and modelling, natural language processing, and process architectures. The results of his research have been published, among others, in *Decision Support Systems*, *IEEE Transactions on Software Engineering, and Information Systems*.

Jan Mendling studied business computer science from the University of Trier, Germany and UFSIA Antwerpen, Belgium. He received the PhD degree from WU Vienna, Austria. He is currently a full professor and the head of the Institute for Information Business at WU Vienna. After being a postdoc with QUT Brisbane (Australia) and a junior professor at HU Berlin (Germany), he moved back to WU in 2011. His research interests include business process management, conceptual modelling, and enterprise systems. He has published more than 200 research papers and articles, among others in *ACM Transactions on Software Engineering and Methodology*, *IEEE Transactions on Software Engineering, Information Systems, Data & Knowledge Engineering*, and *Decision Support Systems*. He is a member of the editorial board of three international journals, one of the founders of the Berlin BPM Community of Practice (www.bpmb.de), and a board member of the Austrian Gesellschaft fur Prozessmanagement. His PhD thesis has won the Heinz-Zemanek-Award of the Austrian Computer Society and the German Targion Award for dissertations in the area of strategic information management.