Aligning Process Model Terminology with Hypernym Relations

Stefan Bunk, Fabian Pittke, Jan Mendling

Aligning Process Model Terminology with Hypernym Relations

Stefan Bunk, Fabian Pittke, and Jan Mendling

WU Vienna, Welthandelsplatz 1, A-1020 Vienna, Austria
bunk@ai.wu.ac.at,{fabian.pittke,jan.mendling}@wu.ac.at

Abstract. Business process models are intensively used in organizations with various persons being involved in their creation. One of the challenges is the usage of a consistent terminology to label the activities of these process models. To support this task, prior research has proposed quality metrics to support the usage of consistent terms, mainly based on linguistic relations such as synonymy or homonymy. In this paper, we propose a new approach that utilizes hypernym hierarchies. We use these hierarchies to define a measure of abstractness which helps users to align the level of detail within one process model. Moreover, we define two techniques to detect specific terminology defects, namely process hierarchy defects and object hierarchy defects, and give recommendations to align them with hypernym hierarchies. We evaluate our approach on three process model collections from practice.

Keywords: inconsistency detection, model quality, process models

1 Introduction

Documenting business operations with process models has become a common practice of many organizations resulting in process collections of a considerable size. An important aspect of such collections is to keep them understandable for all stakeholders and free of contradictions and inconsistencies. This is difficult due to the size of these collections [27, 15]. Quality management therefore has to rely as much as possible on automatic, computer-assisted analysis.

While the structural information of business process models has been intensively studied [19, 29], recent research has focused on the quality of the natural language in these models [18, 12]. Recent techniques support the analysis of grammatical structures [13, 16] or the detection of semantic ambiguities [6, 26]. While the latter set of techniques use synonymy and homonymy, the potential of using other semantic relations is not well understood [17]. Specifically, hierarchy information, as given by so called hyponyms and hypernyms, are not considered so far and impose a notable gap of linguistic analysis techniques.

In this paper, we propose a novel approach that uses hypernym relations to analyze process models and find inconsistencies. Hypernyms, or the opposite hyponyms, are words that have a broader meaning than other words [25]. An

example of a hypernym-hyponym relationship is the verb pair *create* – *manufacture*. While *create* is an abstract word with many possibilities of making an object, *manufacture* captures the fact that the object is created with the hands. Our proposed technique will make use of word hierarchies to determine their level of abstraction in a given business process. Based on this, we identify two potential defects in a process model: First, we detect process hierarchy defects, which occur when a verb and one of its hypernyms is used in the same process model. Second, we identify object hierarchy defects that affect business objects which occur with verbs from the same hypernym hierarchy. Afterwards, we discuss means to resolve the detected defects in process models. In order to demonstrate the capabilities of our approach, we use three process model collections from practice and evaluate the proposed technique with them.

The rest of the paper is organized as follows. In Section 2, we illustrate the problem at hand and discuss previous approaches to tackle this issue. Then, Section 3 defines the necessary concepts of this work, before Section 4 introduces the details of our approach. In Section 5 we evaluate the metric on three real-world datasets and find defects in the data, showing the applicability of our approach. Finally, Section 6 concludes the paper.

2 Background

In this section, we discuss the background of our research. First, we illustrate the terminology problem based on two process models. Then, we will discuss existing research approaches for refactoring text labels in business process models.

2.1 Problem Statement

The problem of inconsistent terminology is best explained with an example. Figure 1 depicts two simple process models: *Bug Report* (Process A) and *New Feature* (Process B). Process A starts with the receipt of a bug report. Then, an employee of the first-level-support tries to reproduce the reported bug. If successful, the bug is delegated to a developer trying to find its cause. Subsequently, the developer implements a bug fix and changes the documentation which terminates the process. If the bug could not be reproduced, the process also terminates. Process B depicts the necessary steps to include a new feature in a software. After a suitable use case has been identified, the software is changed and its documentation updated which also resembles the end of the process.

The depicted process models have several problems with regard to their terminology. First, we observe that the actions in process A do not seem to be on the same level of detail. While *reproduce*, *delegate*, and *implement* are rather specific, the actions *change* and *find* refer to tasks on a higher level of abstraction. Thus, these actions might point to a wider range of possible tasks and leave the modeler with much space of interpretation. Depending on the required abstraction level, the modeler may wish to adjust the wording (e.g. *identify* instead of *find*), to group some activities together, or to split them into more activities.

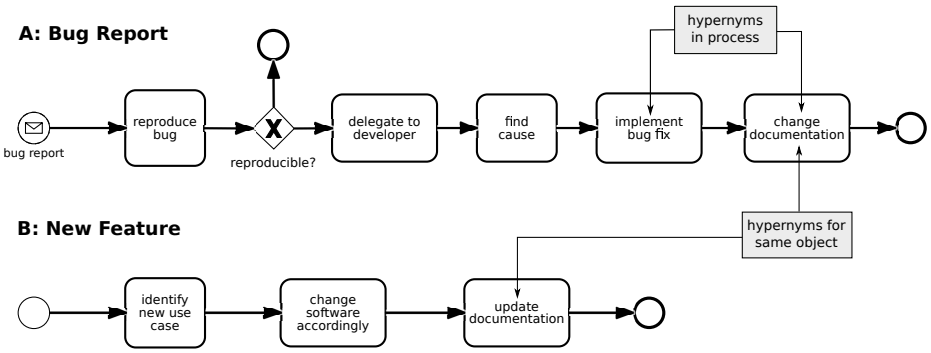


Fig. 1. Example for business processes with terminological conflicts

Second, we observe a mismatch in the abstraction level that relates to the usage of verbs. In Process A, the verb *implement* is a hyponym of the verb *change* because the implementation of something always involves that a process or a product is changed. We also observe this problem between several process models of a process model collection. In the example, Process A and B both use the business object *documentation* with, however, two different actions, i.e. *change* and *update*. Similar to the previous case, *update* is a more specific expression of *change*. For the reader, it might be unclear, whether these two actions refer to the same task or not.

The problem of inconsistent terminology has been highlighted a.o. in the research of Kamsties et al. [10] or Berry et al. [2], who identify syntax, semantic, and pragmatic ambiguities as the major source of unsound software requirements. In consequence, such unsound software requirements may lead to misconceptions within the organization with regard to the wrong selection of systems and components or the wrong implementation of system functionality [20]. Finally, as pointed out by Denger et al. [4], the correction and re-implementation will lead to expensive reworks and software deployments delays.

2.2 Related Work

Research in the field of requirements engineering has brought forth a plethora of approaches to improve the terminology of requirements documents. These approaches particularly focus on the issue of ambiguity detection. One option is to manually check requirements techniques by employing specific reading techniques, such as inspection-based reading [11], scenario-based reading [10], or object-oriented reading [28]. Besides the reading techniques, there are also automatic approaches. These make use of metrics to evaluate the requirements document based on its understandablility, consistency, and readability [5] or on its ambiguity [3]. Another class of approaches uses natural language patterns to manage ambiguity in requirements documents. Among them, Denger et al. [4] propose generic sentence patterns to describe events or reactions of a software.

Table 1. Overview of Ambiguity Management Approaches

Category	Approach	Author
Ambiguity Detection in Documents	Inspection-based Reading	Kamsties et al. [11]
	Scenario-based Reading	Kamsties et al. [10]
	Object-oriented Reading	Shull et al. [28]
	Metrics on Understandablility, Consistency, and Readability	Fantechi et al. [5]
	Ambiguity Metric	Ceccato et al. [3]
	Sentence Patterns for Ambiguity Prevention	Denger et al. [4]
	Ambiguity Detection with Regular Expressions and Keywords	Gleich et al. [8]
Ambiguity Detection in Process Models	Ambiguity and Specificity Measurement Metrics	Friedrich [6]
	Verifying Concept Relations with a Lexicon	Van der Vos [30]
	Consistency Verification with Semantic Annotations	Weber et al. [31]
	Preventing Ambiguity with a Domain Thesaurus	Becker et al. [1]
	Synonym Correction	Havel et al. [9]
	Detection and Correction of Semantic Ambiguity	Pittke et al. [26]

These patterns provide a template for a specific situation which are instantiated with the necessary concepts of the software to be developed. Gleich et al. [8] use regular expressions and keywords to detect ambiguities. For example, the expressions *many* or *few* might point to vaguely formulated requirements.

There are also techniques that specifically focus on natural language labels in process models. Friedrich [6] defines a semantic quality metric to identify activity labels that suffer from ambiguity. For that purpose, the author employs the hypernym relation of WordNet and the depth of a term in the hypernym hierarchy. The metric punishes terms if their depth is too low or too high. Van der Vos [30] uses a semantic lexicon to check the quality of model elements. It ensures that words of element labels are used in a linguistically meaningful way. The technique checks if the label correctly refers to a specific object and if the combination between several objects makes sense in the context of a model. Weber et al. [31] propose a semantic annotation approach that enriches activities with preconditions and effects and propagate those for semantic consistency verification. For example, it is only possible to send a cancellation of a purchase order if it has not been confirmed yet. The approach of Becker et al. [1] enforces naming conventions in process models. Based on a domain thesaurus, the tool is capable of proposing alternative terms and preventing naming conflicts. The research prototype of Havel et al. [9] also corrects synonym terminology by selecting the dominant synonym among a set of ex-ante defined synonyms.

Pittke et al. [26] automatically find synonyms and homonyms in a collection and propose a resolution with the help of word sense disambiguation and BabelNet.

The aforementioned approaches have mainly two shortcomings. First, many approaches rely on text fragments taken from a grammatically correct natural language text. However, the activity labels of process models contain only short text fragment that do not even resemble a grammatically correct sentence [16]. In addition, they follow different language patterns that can hide the action of a specific activity [18]. Therefore, the approaches for requirements documents are not directly applicable for process models. Second, most of the approaches address a particular type of ambiguity to ensure the terminological consistency, i.e. ambiguity caused by synonym and homonym usage of terms. However, the inconsistencies depicted in Figure 1 will not be detected with them. The reason is that two words suffering from a hypernym conflict are not necessarily synonyms or homonyms such that the inconsistency is not detected at all.

Against this background, we propose an approach that explores the hypernym hierarchies between terms and uses these relations to detect inconsistencies related to the level of abstraction and to word usage.

3 Preliminaries

The detection of hypernym inconsistencies requires a basic assumption about the syntactic structure of process model activity labels. Although there is no restriction on the syntactic structure of activity labels, research has identified a set of reoccurring activity labeling styles [18, 12]. Based on the analysis of more than 1400 models of 6 collections from different industries [14], activities may be described in the verb-object, the action-noun style, the descriptive style, or an arbitrary style. Despite this variety, there are two essential components that form the nucleus of each activity label, i.e. an action and a business object. The action specifies a task that needs to be conducted in the process models, while the business object names the target of the task. Based on these insights, the our approach also assumes that each activity uses these two components.

Given a specific process model p of a process model collection P, we denote that a process model comprises a set of specific activities A_p. While activities $a_p \in A_p$ can be formulated in different labeling styles, they contain two essential components [18], i.e. an action (a_p^A) and a business object (a_p^{BO}) on which the action is applied. As an example, consider the activity label *Reproduce bug* from Figure 1. It contains the action *to reproduce* and the business object *bug*. It is important to note that these components can be communicated in different grammatical variations. For instance, the label *Bug reproduction* contains the same components as *Reproduce bug*, but uses a different grammatical structure. In order to be independent of grammatical structures, we use the technique of Leopold [12] to automatically extract these components. Furthermore, we assume actions to be captured as verbs and business objects as nouns.

In order to determine the hypernym/hyponym relation between words, we use BabelNet. BabelNet [23] is a large multi-lingual database of words and their

senses. It combines data from WordNet [21], Wikipedia, and OmegaWiki, along with multilingual features. BabelNet organizes words in so called synsets, i.e. sets of synonymous words. Each synset describes one particular meaning of a word, which we refer to as *word sense*. These word senses are organized in an enumerative way, which means that they have been defined disjoint to each other [22]. As an example, the word senses of the verb *to reproduce* are given as follows:

- s_1: Make a copy or equivalent of
- s_2: Have offspring or produce more individuals of a given animal or plant
- s_3: Recreate a sound, image, idea, mood, atmosphere, etc.
- s_4: Repeat after memorization

Formally, we refer to these word senses as given by the following definition:

Definition 1 (Word Senses). *Let S denote the set of word senses and let W be the set of all words. Furthermore, let $POS = \{verb, noun\}$ be the set of part of speech tags. Then, the possible senses of a given word and a given part of speech tag are given by the function $Senses_{ENUM} : W \times POS \rightarrow 2^S$.*

A first important implication from this sense-based view is that hierarchies do not work on words themselves, but rather on word senses. For example, depending on the context, the verb *to destroy* may have the hypernym *to undo* or *to defeat*. Therefore, before retrieving the hypernyms from the BabelNet database, the respective word sense of the word has to be determined in the current context. This problem is known as *word sense disambiguation* (WSD). We employ the multi-lingual word sense disambiguation method by Navigli and Ponzetto [24]. WSD approaches typically require a target word, a POS tag and a context of a word for the disambiguation [22]. In our setting, we might use all the components of activity labels of a process model to perform the disambiguation task. The POS tag is determined as mentioned before: actions are verbs and business objects are nouns. The output of the sense disambiguation is a set of most likely word senses for the respective target word. WSD algorithms return a set of multiple senses if two or more senses fit the current context. We formalize the word sense disambiguation process as follows:

Definition 2 (Word Sense Disambiguation). *Let A_p be the activities of a process model and $a_p \in A_p$ a specific activity of p. Further, let $w \in \{a_p^{BO}, a_p^A\}$ be a word of the label components along with its part of speech tag $pos \in POS$, such that $S_w = Senses_{ENUM}(w, pos)$ describes the available word senses of w. Then, let $C = \{a_p^{BO},\ a_p^A \mid a_p \in A_p\}$ denote the set of all words given by the label components of the process model's activities A_p. The word sense disambiguation function $WSD : W \times C \rightarrow 2^{S_w}$ selects a subset of the available senses, such that each sense describes the meaning of the word w in its context.*

We explain the definition by referring to the verb *to reproduce* from Process A in the motivating example. The senses of the verb have been listed above and together form the set $Senses_{ENUM}(reproduce, verb) = \{s_1, s_2, s_3, s_4\}$. By leveraging the context, which contains the words *bug*, *developer*, *implement* and

fix, we are able to correctly disambiguate the third sense as the only correct one, i.e. $WSD(reproduce, C) = \{s_3\}$.

WSD enables us to explore the hypernym relation between two words, or more specifically between two word senses. In lexical databases, a hypernym relation is typically indicated if two senses are directly connected with each other. Thus, the hypernym relation can be described as follows:

Definition 3 (Hypernyms). *Given a set S of all senses, the BabelNet hypernym relationship $Hyper_{BN} \subset S \times S$ is a relation, such that $(s_1, s_2) \in Hyper_{BN}$ iff. s_1 is an immediate hypernym of s_2.*

We recursively define the function $Hypernyms : S \rightarrow 2^S$, which contains the set of all hypernyms of a sense, such that a sense $s_h \in Hypernyms(s)$ iff. $(s_h, s) \in Hyper_{BN}$ or $\exists s_m \in S : s_h \in Hypernyms(s_m) \wedge s_m \in Hypernyms(s)$.

Let us again consider the word *reproduce* with the disambiguated sense s_3 from Process A. Following the hypernym hierarchy step by step we find $Hypernyms(s_3) = \{s_{3,h1}, s_{3,h2}, s_{3,h3}\}$:

- $s_{3,h1}$: Re-create, form anew in the imagination, recollect and re-form in the mind
- $s_{3,h2}$: Create by mental act, create mentally and abstractly rather than with one's hands
- $s_{3,h3}$: Make or cause to be or to become

As another example from Process A, the $Hypernyms(s)$ set of the verb *change*, disambiguated as *cause to change, make different, cause a transformation*, is the empty set, because no hypernyms of that verb sense exist. In such a case, we have encountered a word that describes a general concept covering all the other concepts. Opposite to that, we might find words which describe very specific concepts that cannot be refined further. Examples of such words would be the words *to implement* or *to subscribe*.

Following this argument, we define root hypernyms and leaf hyponyms.

Definition 4 (Root and Leaf Hypernyms). *Given a sense $s \in S$ we define:*

$$RootHypernyms(s) = \{s_r \in Hypernyms(s) | \nexists s' : (s', s_r) \in Hyper_{BN}\}$$
$$LeafHypernyms(s) = \{s_l \in Hyponyms(s) | \nexists s' : (s_l, s') \in Hyper_{BN}\}$$

It is important to note that a hypernym hierarchy does not necessarily consist of a direct line to a root hypernym. A sense may have two hypernyms, whose hypernym hierarchies join again later, or it may have two different root hypernyms. Thus, there might be several senses resulting in a set of hypernym or hyponym senses. For example the verb *to execute* in the sense of *to put sth. in effect* has two root hypernyms in the BabelNet hierarchy, i.e. the senses *cause a transformation* and *make or cause to be or to become*. Leaf hypernyms of *to execute* are for example *applying oneself diligently*, *bringing something to perfection*, or *achieving a greater degree of success than expected*.

Finally, we need to retrieve the word corresponding to the identified hypernym senses from the previous definition. Taking the definition of word senses

(Definition 3.2) into account, we observe that each word is linked to a set of senses. In other words, each sense might be expressed with a specific set of words. This relation is defined as follows:

Definition 5 (Words of a Given Word Sense). *Let $s \in S$ be a specific word sense from the set of all senses and $POS = \{verb, noun\}$ be the set of part of speech tags. Then, the possible words W of a given word sense and a given part of speech tag is given by the function words : $S \times POS \rightarrow 2^W$.*

Applying this definition to the previous examples of root hypernnyms, we retrieve the word *to change* from the sense *cause a transformation* and *to create* from the sense *make or cause to be or to become*. We receive the words *to ply*, *to consummate*, or *to overachieve* for the exemplary senses of leaf hypernyms.

4 Conceptual Approach

In this section, we use the hypernym relationship to measure the abstractness of a process model and to identify two possible defects in business processes, i.e. *process hierarchy defects* and *object hierarchy defects*. Moreover, we explain how the detected deficiencies may be resolved in a subsequent step.

4.1 Measuring Verb Abstractness

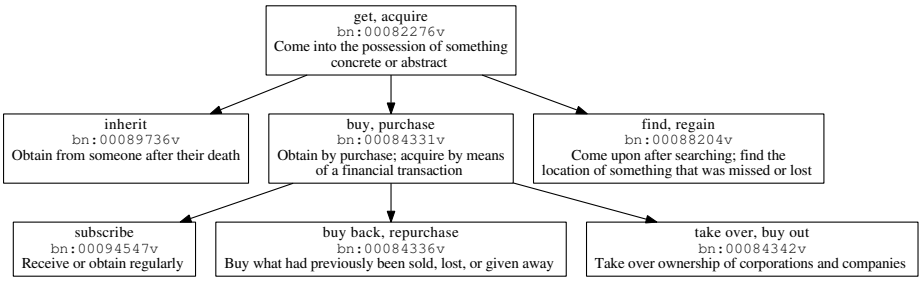


Fig. 2. Example extract from the BabelNet hypernym hierarchy. The first row shows the verbs in the synset, the second line the synset id, and the last line shows the gloss of that synset.

In order to explain our measure for verb abstractness, we use the example hypernym hierarchy in Figure 2. Intuitively, the verb *to get* is more abstract than the verb *to repurchase*, while both can describe the act of getting into possession of something. This is also represented in the hypernym hierarchy of the verb *to get* which is at a higher hierarchy level than the verb *to repurchase*. Thus, our definition of verb abstractness has to explicitly consider the position of the verb in the hierarchy, which is given by its depth and height. In our operationalization, we require both depth and height, because one concept alone does not suffice

to assess the position of the node in the hierarchy. This follows the intuition that a leaf hyponym at depth 5 should be considered as more concrete than a hyponym, which is also at depth 5 but has 5 sub-levels of hyponyms below.

Definition 6 (Sense Abstractness). *For a sense $s \in S$, we measure the abstractness.*

$$abstractness : S \to [0, 1]$$
$$abstractness : s \mapsto \frac{height(s)}{depth(s) + height(s)},$$

such that

- *$depth(v)$ is the length of the longest path to a root hypernym starting at s and only following hypernym relations.*
- *$height(s)$ is the length of the longest path to a leaf hypernym starting at s and only following hyponym relations.*
- *The abstractness value is 1 for root hypernyms and 0 for leaf hypernyms.*

If we again look at the verbs from above, we find $abstractness(s_{get}) = 1.0$ and $abstractness(s_{repurchase}) = 0.0$. The word *to buy*, which lies between *get* and *repurchase* in this hypernym hierarchy, is assigned $abstractness(s_{buy}) = 0.5$.

The abstractness value of a single sense is then extended to an abstractness measure for processes by first averaging the sense abstractness over all senses of the word resulting in a measure for word abstractness. Then, we average again over all words in a process model, which is shown in the following definition:

Definition 7 (Word and Process Abstractness).

$$word\text{-}abstractness : w \mapsto \frac{1}{|WSD(w)|} * \sum_{s \in WSD(w)} abstractness(w)$$
$$process\text{-}abstractness : P \mapsto \frac{1}{|P|} * \sum_{w \in P} word\text{-}abstractness(w)$$

In the processes of Figure 1, the *process-abstractness* values are quite similar: $process\text{-}abstractness(P_A) = 0.63$ and $process\text{-}abstractness(P_B) = 0.60$.

4.2 Process Hierarchy and Object Hierarchy Defects

We have already seen in Figure 1 that process hierarchy defects occur, when a verb and its hypernym is used in the same process. We can automatically determine this type of defect with the following definition:

Definition 8 (Process Hierarchy Defect). *Let v_1 and v_2 be two verbs in the activity labels of one process model. Further, let the sets S_1 and S_2 be the senses of these verbs determined after WSD. We say there exists a process hierarchy defect between the verbs v_1 and v_2 iff.:*

$$\exists s_1 \in S_1, s_2 \in S_2 : s_1 \in Hypernyms(s_2) \vee s_2 \in Hypernyms(s_1)$$

We call the tuple (v_1, s_1, v_2, s_2) *a process hierarchy defect tuple, where* s_1 *is the hypernym and* s_2 *is the hyponym.*

Looking at Process A, we find a process hierarchy defect for $v_1 = change$ and $v_2 = implement$. WSD yields $S_1 = \{s_{1,1} = (\text{cause to change; make different})\}$ and $S_2 = \{s_{2,1} = (\text{apply in a manner consistent with its purpose}), s_{2,2} = (\text{pursue to a conclusion})\}$. Now, the process hierarchy defect is the fact that the second sense of *implement* (pursue to a conclusion) is a hyponym of the first sense of *to change* (cause to change), i.e. $s_{1,1} \in Hypernyms(s_{2,2})$.

Figure 1 also depicts an object hierarchy defect. This type of defect occurs, when a verb and its hypernym is used in conjunction with the same business object. Please note that this problem only occurs for verbs that occur together with business objects from different process models. We formalize this defect as follows:

Definition 9 (Object Hierarchy Defect). *Let* w_{BO} *be a business object, which is used with the sense* s_{BO} *in a business process model collection. Then, let* $S_{w_{BO}}$ *be the set of all verb senses, which are used in activity labels with* (w_{BO}, s_{BO}) *as the business object. We say there exists an object hierarchy defect for the object/sense tuple* (w_{BO}, s_{BO}) *iff.:*

$$\exists s_1, s_2 \in S_{w_{BO}} : s_1 \in Hypernyms(s_2) \vee s_2 \in Hypernyms(s_1)$$

The example processes in Figure 1 contain an object hierarchy defect for the object *documentation*, which is used in the same sense in both of its occurrences. The verb senses used with *documentation* are $S_{documentation} = \{(\text{cause to change; make different}), (\text{bring up to date})\}$. Here, the first sense (cause to change) is a hypernym of the second (bring up to date), i.e. $s_1 \in Hypernyms(s_2)$.

4.3 Resolving Hierarchy Defects in Process Models

The presented detection techniques identify inconsistent terminology instances in process models. These deficiencies hinder the proper understanding and sense-making of process models since the specification of verbs on different hierarchy levels point to a varying range of possible tasks and are thus open to several interpretations. These deficiencies have to be aligned in the next step by repository managers. Accordingly, repository managers can use the BabelNet system to align deficient verbs, process hierarchy defects, and object hierarchy defects.

Regarding the alignment of *deficient verbs*, the defined abstractness metrics point to particular verbs that violate the average degree of abstractness in the respective process model. This violation might either involve verbs that are too general or too specific compared to the average *process-abstractness* score. In case of too general verbs, the repository manager can employ the BabelNet system and look for hyponym alternatives. Too specific verbs may be resolved with hyponym alternatives. For example, consider the activity *Change documentation* in Process A of our motivating example. The word-abstractness score amounts

to almost 1 indicating a very general verb. In order to meet the average *process-abstractness* of 0.63, the repository manager may choose the verbs *to adapt* or *to edit* as suitable alternatives from the BabelNet hypernym tree.

Regarding the alignment of *process hierarchy defects*, the detection technique identifies verb pairs within a process model that are in a hypernym relationship with each other. Typically, this defect implies that a given task already involves the doing of another. We already mentioned the verbs *to change* and *to implement* suffering from this type of defect, because the process of changing something always involves an implementation. Accordingly, the repository manager has two alternatives. On the one hand, he could merge the two activities into one since the implementation of the bug fix also involves the change of the software documentation. On the other hand, he might keep both activities and align the action of the general activity *change documentation* by replacing it with more specific alternatives, such as *to update* or *to edit*.

Regarding the alignment of *object hierarchy defects*, the detection technique retrieves pairs of process models, in which one business objects is used with different verbs being in a hypernym relation with each other. Typically, this defect causes confusion whether or not the same task has to be applied to the respective business object. Looking at our example processes, our technique identifies the activities *update documentation* from process A and *change documentation* from process B, which leave us unclear whether changing the documentation also involves the same task as updating it. Accordingly, the repository manager might either align the two actions by choosing only one action, e.g. *to update*, or by replacing the general action *to change* with one of the aforementioned alternatives.

By applying these alignment techniques, the specificity and the terminological quality of the process model and the repository will be increased and the understandability can be improved.

5 Evaluation

This section presents the results of our evaluation. We first describe the test data and then the results of measuring abstractness and detecting hierarchy defects.

5.1 Evaluation Setup

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 2 summarizes their main characteristics. These collections include:

- **AI collection:** The models of this collection originate from *BPM Academic Initiative*[1]. The collection was built by students and lecturers in a series of projects and lectures. From the available models, we selected those with

[1] See `http://bpmai.org/BPMAcademicInitiative/`

Table 2. Characteristics of the test collections

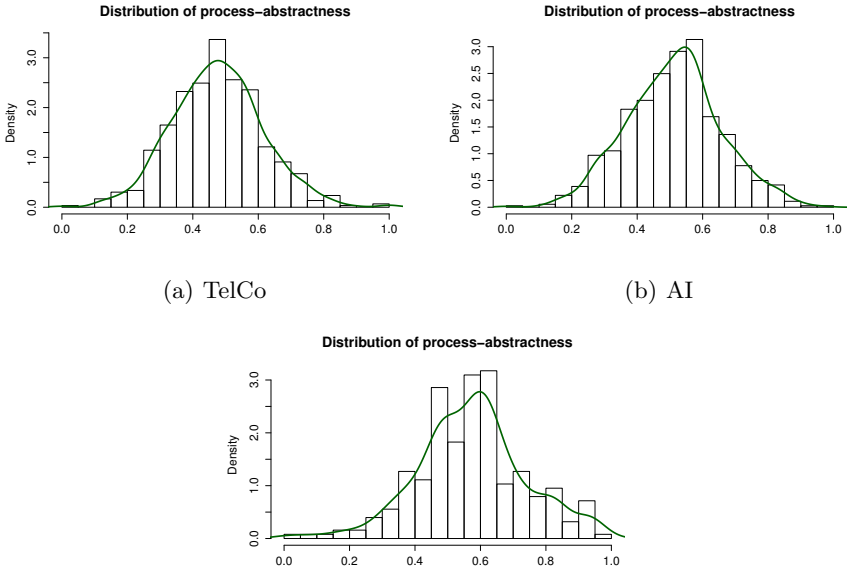
	AI	**SAP**	**TelCo**
No. of Processes	949	575	774
No. of Activities	4,193	1,545	4,972
Avg. No. of Verbs per Process	4.42	2.69	6.42
No. of Unique Verb Senses	513	354	413
No. of Unique Object Senses	1,979	556	1,641
Modeling Language	BPMN	EPC	EPC
Domain	Training	Independent	Telecommunication
Terminology Quality	Low	High	Medium

proper English labels. The resulting subset includes 949 process models with in total 4,193 activity labels. We expect this collection to have the lowest terminology quality among our three datasets, because no common guidelines or standards exists.

- **SAP:** The *SAP Reference Model* contains 575 Event-Driven process chains in 29 different functional branches [7]. Examples are procurement, sales, and financial accounting. The model collection includes 1,545 activity labels. Since the SAP Reference Model was designed as an industry recommendation with a standardized terminology, we expect a small number of terminological defects.
- **TelCo:** The *TelCo* collection contains the processes of an international telecommunication company. It comprises 774 process models with in total 4,972 activities. We assume the TelCo collection to contain more heterogeneous terminology as it is not based on a standardized glossary. In terms of number of defects, we would expect this to be between the previous two model collections.

5.2 Evaluation of Abstractness

In the first step of the evaluation, we focus on the distribution of the *abstractness* value. In Figure 3 we show its distribution for all collections. On average, the distribution of all our test collections shows a mean of 0.52 and standard deviation of 0.31. These values indicate that most of the word senses of all verbs in a single process models are in the middle of the hypernym hierarchy making use of narrow or broad verbs in a balanced way. However, we also observe process models making extensive use of verbs with either a very narrow or a very broad sense It is also interesting to note that each collection appears to be equally affected by process models with either very narrow or very broad verbs. Since, these cases more unlikely and hard to spot, the affected models require further investigation and alignment based on the overall degree of abstraction within a process model. Interestingly, this phenomenon appears in all of the three collections alike and thus being independent on the assumed labeling quality of the collections. We therefore conclude that verb abstractness has not been considered in industry process models yet and emphasizes the necessity of further investigation.



(a) TelCo

(b) AI

(c) SAP

Fig. 3. Distribution of *process-abstractness* value for the test collections.

Additionally, we now look at the processes with boundary values of *process-abstractness*. Table 3 shows the verbs used in these processes along with their respective abstractness value. Following intuition, the abstract processes are dominated by words such as *to create*, *to determine*, and *to analyze*, while *to capture* and *to negotiate* appear in the concrete processes. Just by looking at the abstractness values and the verbs, it is possible to determine the level of the process in a process hierarchy.

5.3 Evaluation of Defects

Regarding the evaluation of hierarchy defects, we first have a look at the quantitative extent within our test collections. Table 4 lists the number of process and object hierarchy defects for each collection. Surprisingly, the TelCo collection has the highest number of process hierarchy defects per model (226 affected models with 0.85 defects on average). This might be explained by the high number of verbs per process (see Table 2): the probability of process hierarchy defects rises with more verbs in a process. For the object hierarchy defects, the initially assumed terminological quality matches with the results. The SAP collection clearly has the least number of object defects per business process, followed by TelCo and AI both lying closely together.

After presenting the quantitative extent of defects in the test collections, we provide qualitative examples of word pairs that frequently cause process

Table 3. Abstract and concrete processes in the test collections

	process-abstractness	Verbs in the process
TelCo	0.00	answer, launch, test
	0.00	capture, shop, test
	0.11	capture, check, negotiate
	0.15	appoint, report, review, update, validate
	0.17	capture, check, send, store
	0.87	change, close, create, handle, register, take
	0.87	analyze, change, create, inform, request
	0.88	analyze, create, initiate
	0.89	change, determine, provide
	1.00	create, determine, perform
SAP	0.02	close, index, revalue
	0.06	assign, process, requisition
	0.06	assign, process, requisition
	0.06	assign, process, requisition
	0.06	post, park, receipt
	0.82	create, determine, plan
	0.83	analyze, direct, transfer
	0.89	change, determine, value
	0.94	adjust, create, determine
	1.00	change, create, maintain
AI	0.04	cheese, fruit, milk
	0.09	check, return, tick
	0.11	blaze, check, prepare
	0.11	design, sap, test
	0.11	click, flight, open, validate
	0.84	access, change, choose, create, remove
	0.84	complete, confirm, create, evaluate
	0.86	determine, enter, search
	0.89	close, make, solve
	1.00	apply, close, start

Table 4. Defects per process collection, normalized by number of processes, and number of processes with defects.

	SAP	**AI**	**TelCo**
Avg. No. of Process Hierarchy Defects	0.397	0.356	0.848
No. of Affected Process Models	72 (12.52%)	185 (19.49%)	226 (29.2%)
Avg. No. of Object Hierarchy Defects	0.171	1.620	1.303
No. of Affected Process Models	31(5.39%)	218(22.97%)	211 (27.26%)

hierarchy and object hierarchy defects. For that purpose, we provide tabular overviews of the five most frequent defects of each type. Table 5 gives an overview of the most frequent *process hierarchy defects* in our test collections. We, for example, identified bad combinations of the verbs *to handle* and *to manage*, *to produce* and *to generate*, or *to check* and *to confirm*. In these cases, the hypernym is conflicting with one hyponym, which causes the hierarchy defect. However, we also find hierarchy defects that involve several conflicting hyponyms. When looking at all repositories, the verb *to create* is conflicting with four different hyponyms, i.e. *to initiate, to plan, to generate, and to decide*. Even worse, the defects caused by the verb *to create* involve completely different word senses. Consider for example the TelCo collection. The verb *to create* is conflicting with the hyponyms *to initiate* and *to plan*, which emphasizes the necessity to carefully use the terminology of the model collection.

Table 5. Top five detected process hierarchy defect tuples in the test collections

	Count	Hypernym word and sense	Hyponym word and sense
TelCo	28	handle: Be in charge of, act on	manage: Watch and direct
	20	create: Make or cause to be or to become	initiate: Bring into being
	17	confirm: Establish or strengthen as with new evidence or facts	check: Make certain of something
	16	create: Make or cause to be or to become	plan: Make a design of; plan out in systematic form
	15	change: Cause to change; make different	implement: Pursue to a conclusion or bring to a successful issue
SAP	9	create: Make or cause to be or to become	plan: Make or work out a plan
	8	transmit: Send from one person or place to another	process: Deliver a warrant or summons to someone
	7	transfer: Send from one person or place to another	process: Deliver a warrant or summons to someone
	6	permit: Consent to, give permission	confirm: Support a person for a position
	5	create: Make or cause to be or to become	decide: Influence or determine
AI	26	produce: Create or manufacture a man-made product	generate: Give or supply
	21	confirm: Establish or strengthen as with new evidence or facts	check: Make certain of something
	19	create: Make or cause to be or to become	generate: Give or supply
	11	inform: Impart knowledge of some fact, state or affairs	prepare: Create by training and teaching
	9	get: Come into the possession of something concrete or abstract	receive: Come into possession of something

We continue with a deeper discussion of *object hierarchy defects* as depicted in Table 6. The results show that our technique is capable to uncover hypernym conflicts for business objects that are frequently used in the respective process model collection. In most of the cases, the technique sees hypernym conflicts for the processing or completion of orders, the identification and targeting of relevant customers, and the creation of pricing options. These examples illustrate that a clear distinction between the tasks applied to one object is necessary since, for example, the creation of pricing options may be done in an automatic way (*to generate*) or by a human clerk (*to create*).

Moreover, it is interesting to note that the business object *order* is involved in a notable number of object hierarchy defects throughout all test collections. For example, it conflicts with the verbs *to execute* and *to complete* in SAP or with the verbs *to check* and *to confirm* in TelCo. Both cases highlight the necessity for carefully choosing actions since the object *order* appears to play a central role in several processes. However, it is worth pointing out that the defects of the business object order in the SAP collection are far less prominent. This resembles the purpose of the SAP collection to be a an industry recommendation with a standardized terminology.

Table 6. Top five detected object hierarchy defect tuples for each collection

	Rank	Word	Count	Examples
TelCo	1	order	339	complete – change, check – confirm
	2	customer	100	identify – refer
	3	appointment	43	plan – make
	4	management	32	contract – change
	5	case	24	generate – create
SAP	1	order	8	execute – complete
	2	invoice	6	receipt – verify
	3	notification	5	print – create
	4	budget	2	release – transfer
	5	project	2	schedule – plan
AI	1	pricing options	298	generate – create
	2	order	138	ship – place, process – change
	3	claim	110	review – evaluate
	4	goods	100	release – move, ship – move
	5	application	66	review – evaluate

6 Conclusion

In this paper, we addressed the problem of inconsistent terminology in activity labels of business process models. We approached the problem from the perspective of hypernym/hyponym relations between verb senses. We argued that there is more freedom in choosing the verbs of a business process model, which

motivates our focus on verb senses. By using the lexical database BabelNet, we operationalized the abstractness of a verb sense and used the height and the depth of the node in the hypernym hierarchy to measure the sense abstractness. Generalizing this abstractness to words and processes, we can detect abstract and concrete processes, and compare them with their level of detail in a process hierarchy if available. We use the abstractness measure to uncover two types of defects in a model collection, dealing with the usage of hypernyms. Process hierarchy defects occur if one business process contains a verb sense and one of its hypernyms, indicating different abstraction levels. Object hierarchy defects occur if one business object is used with words from the same hypernym hierarchy. The evaluation showed that these defects exist and deserve the attention of modelers.

Our technique can be used throughout the entire life cycle of a model collection. Using it during development already prevents the defects we address here. It may also increase the awareness for word selection in general and thereby increase the quality not only for verb hypernym defects. As we presented in the evaluation section, our technique can also be applied to complete collections to find ambiguities after process model creation.

In future work, we first aim to examine the effects of our method in an end user study. In such a study, we want to focus on the impact of our techniques on the understandability and maintainability of process models. Moreover, we want to gather insights and requirements from a realistic scenario that helps us to transfer our technique into existing modeling tools. Second, we also want to address the resolution of process and object hierarchy defects. The idea is to develop a recommendation-based approach that helps the modeler to correct the spotted defects in an efficient way. For that purposes, this work provides a basis for further investigation in this important research area.

References

1. Becker, J., Delfmann, P., Herwig, S., Lis, L., Stein, A.: Towards increased comparability of conceptual models - enforcing naming conventions through domain thesauri and linguistic grammars. In: ECIS 2009. pp. 2231–2242 (2009)
2. Berry, D.M., Kamsties, E., Krieger, M.M.: From contract drafting to software specification: Linguistic sources of ambiguity. Tech. rep., Technical Report, School of Computer Science, University of Waterloo, Waterloo, ON, Canada (2003)
3. Ceccato, M., Kiyavitskaya, N., Zeni, N., Mich, L., Berry, D.M.: Ambiguity identification and measurement in natural language texts (2004)
4. Denger, C., Berry, D.M., Kamsties, E.: Higher quality requirements specifications through natural language patterns. In: Conference on Software: Science, Technology and Engineering, 2003. pp. 80–90. IEEE (2003)
5. Fantechi, A., Gnesi, S., Lami, G., Maccari, A.: Applications of linguistic techniques for use case analysis. Requirements Engineering 8(3), 161–170 (2003)
6. Friedrich, F.: Measuring semantic label quality using wordnet. In: 8. Workshop der Gesellschaft für Informatik eV (GI) und Treffen ihres Arbeitskreises Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (WI-EPK)“, Nüttgens, M. et al.(eds.), Berlin. pp. 7–21. Citeseer (2009)

7. Gerhard, K., Teufel, T.: SAP R/3 Process Oriented Implementation: Iterative Process Prototyping (1998)
8. Gleich, B., Creighton, O., Kof, L.: Ambiguity detection: Towards a tool explaining ambiguity sources. In: Requirements Engineering: Foundation for Software Quality, 16th International Working Conference, REFSQ 2010. pp. 218–232 (2010)
9. Havel, J., Steinhorst, M., Dietrich, H., Delfmann, P.: Supporting terminological standardization in conceptual models - a plugin for a meta-modelling tool. In: 22nd European Conference on Information Systems, ECIS 2014 (2014)
10. Kamsties, E.: Understanding ambiguity in requirements engineering. In: Engineering and Managing Software Requirements, pp. 245–266. Springer (2005)
11. Kamsties, E., Berry, D.M., Paech, B., Kamsties, E., Berry, D., Paech, B.: Detecting ambiguities in requirements documents using inspections. In: Proceedings of the first workshop on inspection in software engineering (WISE'01). pp. 68–80 (2001)
12. Leopold, H.: Natural language in business process models. Springer (2013)
13. Leopold, H., Eid-Sabbagh, R., Mendling, J., Azevedo, L.G., Baião, F.A.: Detection of naming convention violations in process models for different languages. Decision Support Systems 56, 310–325 (2013)
14. Leopold, H., Eid-Sabbagh, R., Mendling, J., Azevedo, L.G., Baião, F.A.: Detection of naming convention violations in process models for different languages. Decision Support Systems 56, 310–325 (2013), http://dx.doi.org/10.1016/j.dss.2013.06.014
15. Leopold, H., Mendling, J., Günther, O.: Learning from quality issues of BPMN models from industry. IEEE Software 33(4), 26–33 (2016), http://dx.doi.org/10.1109/MS.2015.81
16. Leopold, H., Smirnov, S., Mendling, J.: On the refactoring of activity labels in business process models. Inf. Syst. 37(5), 443–459 (2012)
17. Mendling, J., Leopold, H., Pittke, F.: 25 challenges of semantic process modeling. International Journal of Information Systems and Software Engineering for Big Companies (IJISEBC) 1(1), 78–94 (2015)
18. Mendling, J., Reijers, H.A., Recker, J.: Activity labeling in process modeling: Empirical insights and recommendations. Information Systems 35(4), 467–482 (2010)
19. Mendling, J., Verbeek, H., van Dongen, B.F., van der Aalst, W.M., Neumann, G.: Detection and prediction of errors in EPCs of the SAP reference model. Data & Knowledge Engineering 64(1), 312–329 (2008)
20. Mili, H., Tremblay, G., Jaoude, G.B., Lefebvre, É., Elabed, L., Boussaidi, G.E.: Business process modeling languages: Sorting through the alphabet soup. ACM Computing Surveys (CSUR) 43(1), 4 (2010)
21. Miller, G.A.: WordNet: a lexical database for english. Communications of the ACM 38(11), 39–41 (1995)
22. Navigli, R.: Word sense disambiguation: A survey. ACM Computing Surveys (CSUR) 41(2), 10 (2009)
23. Navigli, R., Ponzetto, S.P.: BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artificial Intelligence 193, 217–250 (2012)
24. Navigli, R., Ponzetto, S.P.: Joining forces pays off: Multilingual joint word sense disambiguation. In: EMNLP-CoNLL 2012. pp. 1399–1410. Association for Computational Linguistics (2012)
25. Oxford Dictionaries: "hypernym". Oxford Dictionaries., http://www.oxforddictionaries.com/de/definition/englisch_usa/hypernym
26. Pittke, F., Leopold, H., Mendling, J.: Automatic detection and resolution of lexical ambiguity in process models. Transactions on Softw. Eng. 41(6), 526–544 (2015)

27. Rosemann, M.: Potential Pitfalls of Process Modeling: Part A. Business Process Management Journal 12(2), 249–254 (2006)
28. Shull, F., Travassos, G.H., Carver, J., Basili, V.R.: Evolving a set of techniques for oo inspections (1999)
29. Van Der Aalst, W.M.: Workflow verification: Finding control-flow errors using petri-net-based techniques. In: Business Process Management, pp. 161–183. Springer (2000)
30. van der Vos, B., Gulla, J.A., van de Riet, R.: Verification of conceptual models based on linguistic knowledge. Data & Knowledge Engineering 21(2), 147 – 163 (1997)
31. Weber, I., Hoffmann, J., Mendling, J.: Beyond soundness: on the verification of semantic business process models. Distributed and Parallel Databases 27(3), 271–343 (2010)