Henrik Leopold and Eid-Sabbagh Rami-Habib and Jan Mendling and Leonardo Guerreiro Azevdo and Fernanda Araujo Baião

Detection of Naming Convention Violations in Process Models for Different Languages

Article (Submitted)
(Refereed)

Detection of Naming Convention Violations in Process Models for Different Languages

Henrik Leopold[a,*], Rami-Habib Eid-Sabbagh[b], Jan Mendling[c,e], Leonardo Guerreiro Azevedo[d], Fernanda Araujo Baião[d]

[a]*Humboldt-Universität zu Berlin, Spandauer Straße 1, 10178 Berlin, Germany*
[b]*Hasso Plattner Institute, Potsdam, Germany*
[c]*Wirtschaftsuniversität Wien, Austria*
[d]*Federal University of the State of Rio de Janeiro (UNIRIO), Rio de Janeiro, Brazil*
[e]*University of Ljubljana, Slovenia*

Abstract

Companies increasingly use business process modeling for documenting and redesigning their operations. However, due to the size of such modeling initiatives, they often struggle with the quality assurance of their model collections. While many model properties can already be checked automatically, there is a notable gap of techniques for checking linguistic aspects such as naming conventions of process model elements. In this paper, we address this problem by introducing an automatic technique for detecting violations of naming conventions. This technique is based on text corpora and independent of linguistic resources such as WordNet. Therefore, it can be easily adapted to the broad set of languages for which corpora exist. We demonstrate the applicability of the technique by analyzing nine process model collections from practice, including over 27,000 labels and covering three different languages. The results of the evaluation show that our technique yields stable results and can reliably deal with ambiguous cases. In this way, this paper provides an important contribution to the field of automated quality assurance of conceptual models.

Keywords: Process Modeling Guidelines, Naming Convention Checking, Natural Language Processing, Process Model Quality

1. Introduction

Nowadays business process modeling plays an important role in many organizations [17]. Business process models are for instance used for documenting business operations or supporting the analysis and design of information systems [43]. The strong uptake of process modeling in industry has lead to huge modeling initiatives which often result in thousands of process models [73]. Such an amount of models raises the

*Corresponding author. Tel.: +49 30 2093 5805
Email addresses: henrik.leopold@wiwi.hu-berlin.de (Henrik Leopold), rami.eidsabbagh@hpi.uni-potsdam.de (Rami-Habib Eid-Sabbagh), jan.mendling@wu.ac.at (Jan Mendling), azevedo@uniriotec.br (Leonardo Guerreiro Azevedo), fernanda.baiao@uniriotec.br (Fernanda Araujo Baião)

question of how to assure the quality of these models. A specific problem here is the fact that many modelers in practice are not sufficiently trained and that a significant share of process model collections from practice contains errors [57].

In order to cope with these challenges organizations introduce modeling guidelines capturing different details on how processes should be modeled. The complexity of such guidelines is often a factor that makes manual enforcement and compliance checking difficult. Several quality aspects can be easily checked in an automated fashion. For instance, formal process properties such as soundness or the absence of deadlocks can be reliably and effectively checked [84, 90]. Such automatic analyses are precise and can also be applied on large process model collections in a short amount of time. However, while many formal and control-flow based aspects are well understood, there is a notable gap for checking linguistic aspects such as naming conventions. This is a serious problem since the importance of text labels for the understanding of a process model has been clearly demonstrated in prior research [26, 59]. Hence, an automated solution for checking naming conventions would significantly help to improve the overall quality of process models in practice.

While refactoring of activity labels has been discussed in prior research [53], there are several challenges that hinder the application of automatic label analysis on a broader scale. Label analysis is typically conducted for English process models. Accordingly, any approach for English models can benefit from the rich set of freely available natural language processing tools for the English language, such as the lexical database WordNet for the English language [62]. Although there exist many WordNet solutions for several languages[1], these tools significantly vary with respect to completeness, quality and availability. For instance, the English Wordnet covers 155,287 words, while the German *GermaNet* only covers 93,407 words. In addition, many WordNet solutions do not always cover morphological rules. Hence, a word will be only found in the database if it is given in the base form, e.g. a noun in the singular or a verb in the infinitive. On top of that, most WordNet databases for other languages cannot be freely accessed. As a result, models created with text labels in other important languages such Portuguese, Spanish, or German cannot be checked with prior solutions.

In this paper we address these problems by introducing a flexible technique for automatically checking naming conventions in process models. This technique does not depend on WordNet and can hence be adapted to other languages. In order to demonstrate the applicability of the technique, we conduct an extensive evaluation with in total nine process model collections from practice. The employed collections contain English, German, and Portuguese labels in order to illustrate the language independence of this technique.

Building on the recommendations from [88, 89], the remainder of this paper is structured as follows. Section 2 discusses the background on process modeling guidelines and the potential of natural language

[1] http://www.globalwordnet.org/gwa/wordnet_table.html

for checking linguistic guideline aspects. In addition, we investigate the different label styles of process model elements. Section 3 defines a technique for automatically checking naming conventions in a language independent manner. Section 4 presents the results from the empirical evaluation of the nine process model collections. Finally, section 5 concludes the paper and gives an outlook on future research.

2. Background

In this section, we discuss the research background. First, we summarize the general idea of establishing process modeling guidelines. Then, we discuss the potential of natural language processing for guideline checking. Finally, we identify different styles for labeling process model elements in different languages.

2.1. Process Modeling Guidelines

Process modeling guidelines aim at assuring the quality and consistency of process models which are created by different and heterogeneously skilled users. They capture best practices which have proven to be beneficial to support readers in understanding the models [29, 76, 1]. In their simplest forms, they can be formulated as rules such as "A model must have one start event" or "A model should not make use of complex gateways." The rationale behind such guidelines is the insight that some modeling practices are easier to understand and hence are superior in terms of clarity.

Guidelines refer to different aspects of process models. Four of the major aspects are formal model properties, model layout, the use of modeling elements and the use of natural language. Thereby, *formal model properties* refer to the correctness of the model structure. It is often suggested to keep a model as structured as possible [58] and to avoid deadlocks [83]. The *model layout* discusses the proper arrangement of the elements. Good layout typically minimizes the number of crossing arcs and utilizes a clear direction of flow either from right to left or from top to bottom [74]. The *usage of modeling elements* dimension defines which elements should or should not be used. Languages like BPMN offer different options to express the same behaviour [70], so one option might be preferred. Elements with complicated semantics might be forbidden in order to guarantee a good model understanding also by unexperienced model readers [76]. The *usage of natural language* can be considered from two perspectives. The first refers to the set of terms which can be used in the model. Some guidelines try to ensure the term consistency by introducing glossaries [67] or forbidden unspecific verbs [74]. The second perspective is concerned with the structure in which these terms are presented in the label. Guidelines usually suggest certain naming conventions as for instance the verb-object style for activities [59].

In order to illustrate these dimensions and to discuss in how far they can be automatically checked, we consider the exemplary BPMN process model from Figure 1. The process starts with a start event which is then followed by the activity *Cost Planning* and proceeds with an XOR-split gateway. Such a

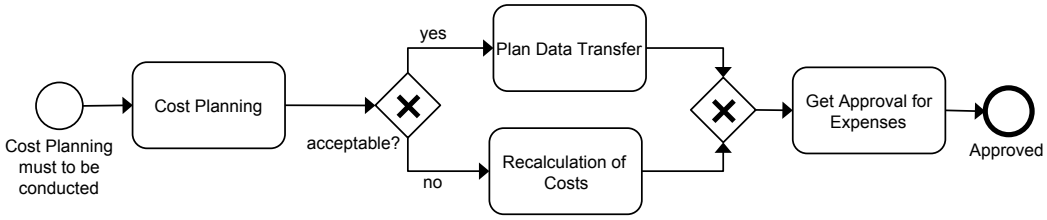

Figure 1: Example Model

diamond-shaped element defines a decision point such that either the upper or the lower branch of the process will be executed. Accordingly, either the activity *Plan Data Transfer* or the activity *Recalculation of Costs* is conducted. Afterwards, the control is passed to the XOR-join gateway. This diamond-shaped element waits for one of the incoming branches to complete. Once one branch has been executed, the process continues with the activity *Get Approval for Expenses* before the process is terminated. It is quite apparent that this exemplary process model contains some considerable weaknesses which may prevent a reader from understanding it correctly.

Starting with the formal properties perspective, we might want to check whether the process model suffers from structural errors like deadlocks. In fact, such formal issues are well-understood and can be efficiently checked in an automatic way using Petri-net concepts [25]. Further techniques area available for checking the correctness of the data flow [79, 87, 75] or the satisfiability of resource constrains [9, 15, 78]. Also automatic techniques for refactoring model structure are available [86, 68]. The layout of the model can be discussed in terms of flow direction and crossing arcs. The depicted model appears to be well organized according to these criteria. Poorly arranged models can be laid out using concepts from graph drawing [4], which have been customized for process models [23]. The usage and exclusion of certain model elements is intensively discussed for BPMN [65, 76]. We observe that the example model from Figure 1 only uses a basic set of elements, which is often recommended. Technically, checking the inclusion and exclusion of elements of a certain type is trivial.

If we consider the usage of the natural language in the model we can observe two main problems. The first is concerned with inconsistent terms. While two activities refer to the term *costs* the last activity mentions the term *expenses*. To avoid such inconsistencies an approach for automatically creating and enforcing the usage of glossary terms has been proposed [67]. The second problem is the inconsistent usage of labeling styles. While a verb-object labeling is typically suggested [76, 58, 1], the example model of Figure 1 shows different deviations. The activities *Cost Planning* and *Recalculation of Costs* capture the actions *plan* and *recalculate* as nouns at different positions in the label. The activity *Get Approval for Costs* is compliant with the verb-object requirement. This style mix actually causes that the activity *Plan Data Transfer* could be misinterpreted. It could either advice to *plan* a *data transfer* or to *transfer* a record of *plan data*. Also the event and gateway labels can be improved. The gateway label *acceptable?* lacks a reference to a business

object, and also the end event label *Approved* does not precisely define the result of the process execution.

The significance of these linguistic problems is emphasized by various process modeling guidelines [74, 58, 29, 76, 1]. Different research has been conducted to at least partially tackle problems with labeling styles. For instance, Delfmann et al. proposed approaches for avoiding naming convention conflicts by enforcing certain phrase structures [20] or by introducing auto-completion [7]. Leopold et al. specify a technique for labeling style recognition and refactoring [53]. A limitation of these approaches is their restriction to the English language and the dependence on natural language processing tools like WordNet, which are not available in other languages, at least in terms of the quality of the content. In order to design a technique which is based on all relevant insights on natural language processing and conceptual models, we use the next section to give a detailed overview of this research field.

2.2. Natural Language Processing in Conceptual Models

This section gives an overview of the intersection of natural language processing and conceptual models. In Section 2.2.1 we present the current state of the art. Afterwards, in Section 2.2.2, we discuss the particular challenges for the automated checking of naming conventions for different languages in process models.

2.2.1. State of the Art

The automated analysis of natural language, which is usually referred to as natural language processing (NLP), has a long tradition and has been applied to many fields. Examples include the processing of natural language texts [40, 30], machine translation [39, 12] and speech recognition [69, 41]. In the past, mature and powerful tools as for instance the Stanford Parser [45] or the lexical database WordNet [62] have been developed. Thanks to their free availability these tools have been utilized in different research domains including the field of conceptual modeling. In conceptual modeling these tools are often used to support and improve the design of the models. Two different directions have been considered in the related work for accomplishing this goal: approaches applying NLP on external text material to infer useful information about the design of the model, and approaches applying NLP on the model itself in order to facilitate different kinds of analyses. The difference between the two directions becomes clear if we consider how the NLP techniques are employed in each scenario.

Many of the approaches applying NLP techniques on *external text material* aim for the automatic inference of conceptual models. Examples include the (semi)-automatic creation of process models [31, 37, 35, 34, 77], conceptual dependency diagrams [33], entity-relationship models [36, 66], and UML diagrams [3, 18, 19, 64]. Some authors also propose inference approaches which are not limited to a single model type, but can be adapted for different kinds of conceptual models [**?** 63]. Works which are not concerned with the extraction of conceptual models typically aim for supporting the model designer in different ways. For instance, Richards et al. propose a technique for visualizing natural language requirements in order to better compare multiple

viewpoints [71]. Bolloju et al. go a step further by introducing a technique which automatically indicates inconsistencies with requirement documents [11]. Lahtinen and Peltonen complement UML tools with a speech interface such that the conceptual models can be easily edited via spoken language [50]. In some cases the extraction of the conceptual models is only an intermediate step: Tseng and Chen propose a methodology for mapping natural language constructs into SQL statements via the inference of UML class diagrams [82]. Similarly, Tseng et al. map natural language constructs into relational algebra via the extraction of entity-relationship models [81]. The important advantage that all these works have in common is that they apply standard NLP tools such as taggers, parsers, or speech recognition to infer the required information from written or verbal sources. As such sources usually contain grammatically rich sentences, the authors obtain satisfying results with this strategy.

Techniques applying NLP tools on *a conceptual model itself* have to meet different requirements. Typically, conceptual models do not contain full and grammatically correct sentences. As a result, authors restrain from employing traditional NLP tools as parsers and taggers. Some authors explicitly recommend to avoid the application of NLP tools because of these issues [5, 47]. However, there are several attempts to make use of the natural language in conceptual models. For instance, Becker et al. propose an approach for enforcing naming conventions in process models [7]. Other authors also look at the terminology used in conceptual models to improve the overall quality [22, 49, 48, 28, 85]. Going beyond the syntactic quality dimension of process models, Gruhn and Laue employ a Prolog based algorithm that is capable of identifying semantic errors in process models labels [38]. For achieving the best possible semantic congruence with the modeled domain some authors use the natural language in models to generate human-readable texts [8, 60, 16]. While many works applying NLP on conceptual models are concerned with model quality, there are also approaches which aim for eliciting knowledge that is implicitly captured by the model. Examples include the identification of activity correspondences between models [24, 6], the discovery of services [46, 52] and the elicitation of process patterns [32].

Table 1 summarizes the discussed approaches from both use case classes. The double line separates the approaches applying NLP on text material (first half) from those applying NLP directly on the models (second half). The table highlights that NLP tools for eliciting the syntactical structure of sentences, such as parser and taggers, are only employed by approaches which work on text material. In fact, none of the previously mentioned approaches applying NLP on model elements, makes use of parsers or taggers. Instead, they either assume a certain format of the language in the model [60, 16, 8] or they ignore the syntax. In the latter case the approaches typically remove stop words such as articles and conjunctions and then use the remaining words for their analyses. While this might be sufficient for some use cases, it yet inhibits the precise usage of natural language in conceptual models. Hence, these approaches cannot be used for an ex-post determination of the label structure. For the checking of naming conventions it is necessary to clearly infer the syntactical structure of a label and properly accomplish the grammatical disambiguation of terms.

Table 1: Overview of Approaches Applying NLP

Approach	Author	NLP Tools
Construction of Models		
BPMN Process Model from Text	Friedrich et al. [31]	Parser, WordNet
BPMN Process Model from Group Stories	Goncalves et al. [37]	Parser
BPMN Process Model from Textual Sources	Ghose et al. [35, 34]	Parser
BPMN Process Model from Use Cases	Sinha et al. [77]	Parser
Dependency Diagram from Text	Gangopadhyay [33]	Parser
ER-Model from Text	Gomez et al. [36]	Parser
ER-Model from Text	Omar et al. [66]	Tagger, Parser
UML Class Model from Text	Bajwa and Choudhary [3]	Tagger, Parser
UML Model from Text	More and Phalnikar [64]	Tagger, Parser
UML Model from Text	Deeptimahanti and Babar [18, 19]	Parser, WordNet
Conceptual Model from Requirements	Fliedl et al. [27]	Tagger, Parser
Conceptual Model from Requirements	Montes et al. [63]	Tagger, Parser
Designer Support		
Visualization of Use Case Descriptions	Richards et al. [71]	Parser
Consistency of Object Models	Bolloju et al. [11]	Parser
Speech Recognition for UML	Lahtinen and Peltonen [50]	Speech Analysis
Construction of Formal Specification		
SQL Statements	Tseng and Chen [82]	Parser
Relational Algebra	Tseng at al. [81]	Parser
Quality Assurance		
Term Inconsistency Detection in Process Models	Koschmider and Blanchard [48]	WordNet
Recommendation-based User Support for Process Models	Koschmider et al. [49]	WordNet
Linguistic Consistency Checking of Conceptual Models	van der Vos et al. [85]	WordNet
Naming Convention Enforcement in Process Models	Becker et al. [7]	Domain Thesauri
Reducing Linguistic Variations in Process Models	Breuker et al. [22]	WordNet
Semantic Annotation of BPMN Process Models	Francescomarino and Tonella [28]	WordNet
Detection of Semantic Errors in EPCs	Gruhn and Laue [38]	Proprietary
Generation of Text		
Generation from UML Class Diagrams	Meziane et al. [60]	WordNet
Generation from Object Models	Lavoie et al. [8]	Proprietary
Generation from Conceptual Models	Dalianis [16]	Proprietary
Information Elicitation		
Service Discovery from Conceptual Model	Knackstedt et al. [46]	WordNet
Service Discovery from Process Models	Leopold and Mendling [52]	WordNet
Detection of Process Patterns	Gacitua-Decar and Pahl [32]	WordNet
Similarity Measurement in Process Models	Ehrig et al. [24]	WordNet
Business Process Activity Mapping Identification	Becker et al. [6]	Proprietary

In the light of this literature review we discuss the resulting challenges for the automated checking of naming conventions in the following subsection.

2.2.2. Challenges for Process Model Guideline Checking

Considering the existing work on NLP and conceptual modeling, it as apparent that the automated analysis of the text structure in conceptual models is associated with considerable challenges. Aiming for the automated checking of naming conventions for different languages we face three main challenges: the non-applicability of standard NLP tools, the lack of availability of linguistic tools for other languages and the linguistic complexity impeding the grammatical disambiguation of terms.

The *non-applicability of standard NLP tools* is mainly caused by the shortness of process model element labels. Typically, linguistic parsers employ statistical methods for computing the most likely part of speech using a manually pre-tagged corpus. Thus, they require a certain degree of context to perform well [45, 80]. Further, they are usually trained with corpora consisting of book texts and newspaper articles. Hence, these corpora hardly contain the specific patterns of word sequences that we find in process models. As a result, the tagging algorithm does not even have the possibility to find an according match in the corpus.

As pointed out in the last subsection, previous works addressing the analysis of linguistic information in conceptual models mainly relied on WordNet. However, the *general availability* of such linguistic tools for less frequently spoken languages is limited. Hence, techniques building on these tools cannot be effectively adapted to other languages. Although English is the predominant business language, we experience that companies typically model their processes in the native language, partially driven by legal requirements. Hence, we consider the possibility of adapting a technique to other languages as important quality criterion.

The *linguistic complexity* of the target language influences how well information can be extracted from short fragments like process model labels. As pointed out by McWhorter, the complexity of a language can be assessed from four different angles: phonology (sounds of the language), syntax (structure of the sentences), semantics (meaning of words) and morphology (structure of word forms) [56]. A frequently discussed problem is for instance the phenomenon of homonymy when words and meanings are not uniquely connected. However, for the identification of part of speeches in process model elements especially the morphological complexity is of major importance: the more complicated the morphological rules of the target language are, the more different word forms are available to differentiate words and their part of speech from each other. If a language is poor in terms of morphological changes, many words will suffer from the so-called zero-derivation ambiguity [21]. This phenomenon refers to the fact that one syntactical word can represent multiple part of speeches without adding any suffixes. Examples are the English words *the order* and *to order* or *the plan* and *to plan*. While this phenomenon can be frequently found among English words, also other languages suffer from this type of ambiguity. For instance the Portuguese word *estado* can represent the noun *state* or the participle form of the verb *to be* (estar). As such cases require an appropriate disambiguation, it

is important to be aware of the morphological complexity of the target language. Different authors tried to adequately assess this characteristic with different metrics [44, 10, 14]. The bottom line is usually that English is one of the languages with the least morphological complexity. German and also Slavic languages can be rather found at the other end of the scale. For the analysis of process model labels this means that English is one of biggest challenges. Especially for English, sophisticated disambiguation algorithms must be introduced.

In order to be aware of the particular linguistic challenges in process model elements, we require a clear definition of common patterns for specific languages. Hence, such label styles are discussed in the following subsection.

2.3. Process Model Element Label Styles

Considering the findings of the previous section, it becomes clear that the automatic checking of naming conventions and the adaptation of such a technique to other languages is still a considerable challenge. The design of effective algorithms for checking naming conventions requires a thorough understanding of the use of natural language in process models not only for English, but also for other languages. To this end, we manually analyzed three English, three German and three Portuguese process model collections to gain insights into differences and commonalities of labeling practices. Focussing on labeled elements with syntactical variations, the detection of naming convention violations relates to three major elements of process models: activities, events, and gateways.

2.3.1. Activity Label Styles

The activity label styles can be categorized in four classes based on how the action is captured in the label. The focus on the action is particularly important for activity labels as they instruct the model reader to actually perform a business related activity. Table 2 shows an overview of the observed activity label styles. It shows the core structure of each style, different example labels and the languages for which we have found the styles. Note that we abstract from additional information fragments such as *for customer* in the label *Create invoice for customer* and focus on the core structure consisting of action and business object. The reason why some of styles from Table 2 cannot be found in every language is given by the different usage of the considered languages. While a certain grammatical structure can be very common in English, it might be totally uncommon in German. Hence, Table 2 also indicates the typical usage of the investigated languages.

In the first of the four classes the action is given as a verb. In *verb-object* labels as *Create invoice* the verb is given as an imperative verb in the beginning of the label. In *infinitive-style* labels the action is also positioned in the beginning of the label but given as an infinitive. Examples are the German label *Erstellen Rechnung* and the Portuguese label *Criar nota fiscal* (both mean *Create invoice*). A typical pattern for

Table 2: Activity labeling styles

Labeling Style	**Core Structure**	**Example**	**Languages**
Verb-Object VO	A(imperative) + O	Create invoice, Erstelle Rechnung, Crie nota fiscal	EN, GER, PT
Infinitive Style IS	A(infinitive) + O	Erstellen Rechnung, Criar nota fiscal	GER, PT
Objective-Infinitive OI	O + A(infinitive)	Rechnung erstellen	GER
Action-Noun AN(np)	O + A(noun)	Invoice creation	EN
Action-Noun AN(of)	A(noun) + 'of' + O	Creation of invoice	EN
Action-Noun AN(gerund)	A(gerund) + [article] + O	Creating invoice, Erstellung der Rechnung,	EN, GER
Action-Noun AN(irregular)	*anomalous*	LIFO: Valuation: Pool level	EN
Descriptive DES	[role] + A(3P) + O	Clerk creates Invoice, Sachbearbeiter erstellt Rechnung	EN, GER
No-Action NA	*anomalous*	Error, Protokoll	EN, GER

German activities is given by the *object-infinitive style* where the action is positioned at the end of the label. An example is given by *Rechnung erstellen* (literally *Invoice create*).

In labels belonging to the second class, the action is captured as a noun. In particular, four different styles can be observed. The first is the *action-noun style (np)*. In such activities the nominalized action is provided at the end of the label as in *Invoice creation*. Another frequent pattern for English activities is the *action-noun style (of)* where the preposition *of* is used to separate the nominalized action from the business object. As examples consider *Creation of invoice* or *Notification of customer*. The *action-noun style (gerund)* contains a gerund in the beginning of the label as in *Creating invoice*. All labels which contain a nominalized action but cannot be assigned to on of the three previously introduced styles are categorized as *action-noun (irregular)*.

The third class contains the *descriptive label style*. In such labels the action is provided is a verb in the third person form. In many cases a role is mentioned in the beginning of the label. Examples are the English label *Clerk creates Invoice* or the according German translation *Sachbearbeiter erstellt Rechnung*.

All labels which do not contain any action are assigned to the *no-action style*. Examples are usually given by single nouns such as *Error* or *Protokoll* (protocol).

2.3.2. Event Label Styles

While activity labels instruct the reader to do something, event labels indicate a state. As this can be expressed in different ways, we derived five different event label styles. Four of them follow a regular structure. Note that some events are also labeled using activity styles. As those have been defined above, we do not revisit them here.

Table 3 shows an overview of the observed event label styles. The first event label style is the *participle style*, which contains a business object followed by a participle verb. In some cases an auxiliary verb is inserted before the participle verb form. Examples are *Invoice created* or *Customer notified*. In general, the *modal style* is quite similar to the participle style. However, labels of the modal style always contain a

Table 3: Event labeling styles

Labeling Style	Core Structure	Example	Languages
Participle PS	O + [aux. verb] + A(participle)	Invoice created, Rechnung erstellt, Nota fiscal criada	EN, GER, PT
Modal MS	O + A(modal verb construction)	Invoice must be created, Rechnung muss erstellt werden	EN, GER
Attribute AS	O + [aux. verb] + adjective	Invoice correct, Rechnung korrekt, Nota fiscal correta	EN, GER, PT
Categorization CS	O + aux. verb + noun	Customer is member, Kunde ist Mitglied	EN, GER
Irregular	anomalous	Inquiry, Qualität, Solicitação	EN, GER, PT

modal verb such as *can*, *must*, or *shall*. In English labels such as *Invoice must be created* this modal verb is then followed by an auxiliary and a participle verb. In the German version of that label, *Rechnung muss erstellt werden*, the order of auxiliary and participle verb is swapped. Labels following the *attribute style* are characterized by an adjective at the end of the label. In some cases an auxiliary verb is used to obtain a correct sentence. Examples are *Invoice correct* or *Status is ok*. The last regular style is the *categorization style* which is characterized by the usage of two nouns which are associated with each other. As an example consider the label *Customer is member*.

Besides these regular event label styles we also observed *irregular* labels. In many cases these labels contain a single noun such as *Inquiry*.

2.3.3. Gateway Label Styles

The role of gateway labels is to properly indicate what kind of decision must be made in order to follow a particular path in a model. Hence, gateway labels usually end with a question mark to explicitly show that a decision must be made. From this analysis we derived four regular gateway label patterns.

Table 4 shows an overview of the observed gateway label styles. Labels following the *participle-question style* can be considered to be participle state style labels with a question mark at the end. Accordingly, they consist of a business object which is followed by an action in the participle verb form. Examples are *Invoice created?* or *Customer notified?* The *infinitive-question style* is the question counterpart of the verb-object and the object-infinitive activity styles. The infinitive verb is either positioned in the first or in the last position of the label. In English labels such as *Approve Contract?* the verb is usually positioned in the beginning. In German labels like *Vertrag akzeptieren?* (literally *Contract approve?*) the verb is mostly given at the end. Gateway labels of the *adjective-question style* are characterized by an adjective at the of the label. Hence, they are the question counterpart of the attribute event style. As examples consider *Parts available?* or *ID valid?* While the previously introduced styles can be associated with event or activity styles, the *equation question style* is gateway specific. Such labels contain a logical evaluable statement, in many cases by using logical operators such as $>$ or $<$. An example is given by the label *Amount* $>$ *€200?* However, in some cases the logical operator is replaced by a verbal construct. Thus, the given example could

Table 4: Gateway labeling styles

Labeling Style	**Core Structure**	**Example**	**Languages**
Participle-Question PQ	O + [aux. verb] + A(participle)	Invoice created?, Rechnung erstellt?, Nota fiscal criada?	EN, GER, PT
Infinitive-Question IFQ	A(infintive) + O / O + A(infinitive)	Approve Contract?, Vertrag akzeptieren?	EN, GER
Adjective-Question AQ	O + [aux. verb] + adjective	Parts available?, Teil verfügbar?, Parte disponível?	EN, GER, PT
Equation-Question EQ	O + logical construction + no.	Amount is greater than €200?, Betrag > €200?, Valor é maior do que R$ 1.000?	EN, GER, PT
Irregular	anomalous	Result?, Neuberechnung?, Crédito?	EN, GER, PT

be also written as *Amount is greater than €200?*

As all investigated model elements, also some gateway labels do not follow a regular structure. Such *irregular* labels usually contain a single word such as *Result?*

3. Checking of Naming Conventions for Different Languages

In this section, we present a technique for automatic and language-independent detection of naming convention violations. In Section 3.1 we describe the manual preparation steps that are required. In Section 3.2 we introduce the technique on a conceptual level and discuss the comprised components in detail.

3.1. Pattern Formalization

In order to automatically detect naming convention violations, we first need to operationalize the verbally described rules. In particular, we have to convert the rules into a pattern which can be checked in an automatic fashion. Hence, the preparation phase includes the transformation of the given rules into so-called linguistic patterns. As linguistic pattern we understand a sequence of part of speeches. In order to illustrate this step consider the following naming conventions which are frequently suggested for English process models[1] [58, 61, 55]:

1. Activities must start with an imperative verb which is followed by a business object, e.g. *Send Documents* or *Inform Customer*.

2. Events must start with a business object which is followed by a passive verb construction or a verb in the past tense, e.g. *Customer is informed* or *Documents were sent*.

3. Gateways must represent a question which is starting with a verb, followed by a business object and an adjective, e.g. *Is customer informed?* or *Were documents sent?*

[1]For a complete discussion of the naming conventions of the investigated languages please refer to Section 4.1.

Having such naming conventions at hand, we can manually transform them to a sequence of required part of speech tags. For the above stated rules, we accordingly derive the following linguistic patterns:

1. Activity Pattern: *Verb (Imperative) + Noun*
2. Event Pattern: *Noun + Auxiliary Verb + Verb (Participle)*
3. Gateway Pattern: *Auxiliary Verb + Noun + Verb (Participle) + '?'*

As these formal representations of linguistic guideline rules can be automatically compared against existing process model elements, they form the basis of the technique. However, before the derived linguistic patterns are defined as a benchmark, we recommend an exploratory analysis of the model collection which needs to be checked. Without knowing about existing patterns, it is hard to decide about minor deviations which would not be considered to violate the above defined rules. As an example consider the event label *Document sent to Customer*. In comparison to the defined event pattern we can identify two essential differences. First, the label contains an additional information fragment which is introduced by the preposition *to*. Second, the label does not contain an auxiliary verb before the participle *sent*. We may encounter a similar situation with activity labels. For instance the activity label *Notify Customer of Rejection* contains an additional information fragment introduced by the preposition *of*. Also gateways might be complemented with an additional fragment. As example, consider the label *Were documents send to customer?* In general, it is essential to be aware of such deviations in order to decide whether they should be considered as violation or not. Hence, by analysing and classifying the labels in the considered process model collection, the linguistic patterns can be extended and adapted to the actual needs. Allowing for the introduced deviations for events, activities and gateways, we introduce optional elements which are denoted by square brackets:

1. Activity Pattern: *Verb (Imperative) + Noun [+ Preposition + Noun]*
2. Event Pattern: *Noun + [Auxiliary Verb] + Verb (Participle) [+ Preposition + Noun]*
3. Gateway: *Auxiliary Verb + Noun + Verb (Participle) [+ Preposition + Noun] + '?'*

Once the valid linguistic patterns have been determined, they serve as input for the automatic violation detection approach.

3.2. A Technique for Checking Naming Conventions

Based on a set of required linguistic patterns, we define a technique for automatically detecting naming convention violations. Figure 2 provides an overview of the architecture. The core idea is to first assign the corresponding part of speech tags to a given process model element. In case the label contains ambiguous words and can hence be represented by multiple linguistic patterns, the subsequent disambiguation component

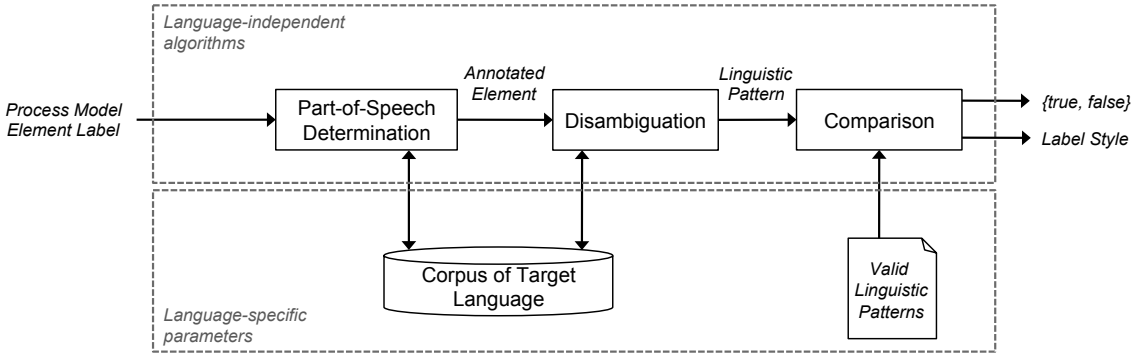

Figure 2: General Architecture

resolves the ambiguity and decides about the correct pattern. By comparing the linguistic pattern from the process model element with the required linguistic patterns from the preparation phase, violations can be automatically detected. As a result, the technique returns whether a naming convention was violated and which style was detected. It is important to note that the comprised components of the technique can be subdivided into language-specific and language-independent elements. While the algorithms of the technique are language-independent, language-specific input is required to enable the algorithms to check the specific rules of the target language. However, since the incorporation of new languages only requires an incremental one-time effort, we are convinced that this does not prevent users from introducing new languages. In the following subsections we will explain each component of the introduced architecture in detail.

3.2.1. Part of Speech Determination

The goal of the part of speech determination component is the automatic annotation of the given process model element with the according part of speech tags. Thereby, words with multiple possible part of speeches are annotated with all potential tags. As already pointed out earlier, the part of speech determination for process model elements is associated with considerable challenges. While databases like WordNet can be used to determine all potential part of speeches of a given word in English, many WordNet solutions for other languages have limitations in terms of quality and availability. In order to provide a language independent and flexible solution, we decided to employ text corpora which are easily obtainable. As examples consider the *OANC* (English), *Floresta* (Portuguese), or the *Tiger Corpus* (German). Such corpora usually contain texts from different sources along with manually created part of speech tags. Due to the availability of text corpora, we use them for developing a part of speech tagging solution which is appropriate for the purpose of detecting naming convention violations.

Figure 3 illustrates how a text corpus can be used for extracting the required data for determining the part of speeches of words. The strategy is to derive two data tables from the corpus: a part of speech table and a verb table. The *part of speech table* contains a list of all verbs from the corpus and how often they

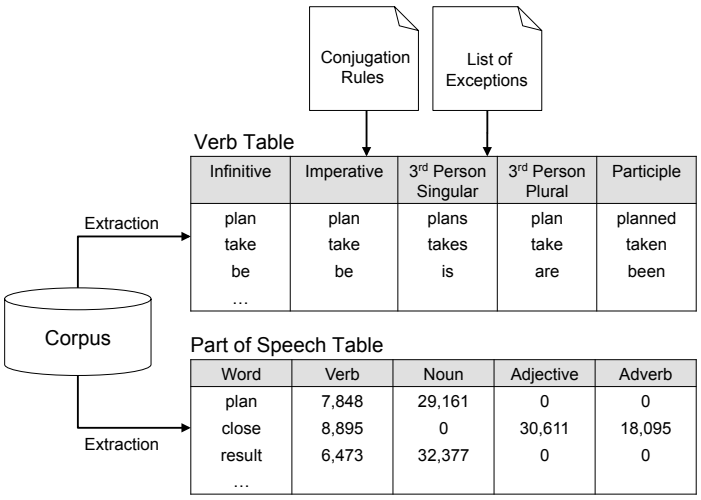

Figure 3: Generation of Part of Speech Tagging Sources

occur as verb, noun, adjective or adverb. As text corpora have been manually tagged by humans, this information can be easily and automatically obtained. The *verb table* includes all verbs from the corpus and the according conjugation forms. In order to completely obtain this information, the verb data in the corpus is complemented with additional information. In particular, we employ conjugation rules of the target language to compute all required verb forms. In addition, we use an exception list for appropriately determining the form of irregular verbs. For many languages such as English, German, and Portuguese according lists are readily available online or in linguistic resources such as language books. Since we build on a standard CSV format, available lists can be easily incorporated. In case no conjugation lists are available, the algorithm for creating theses tables must be manually implemented. However, since this is a one-time effort with reasonable complexity, we do not consider this as a barrier for adapting the presented technique.

Using the data of these tables, two important goals can be achieved. First, all possible part of speeches of a given word can be determined. This is accomplished by checking the different entries in the part of speech tables. As an example, consider the word *plan*. Figure 3 shows the occurrences from the *OANC*. From the numbers in the illustrated table we learn that *plan* has 7,848 occurrences as a verb and 29,161 occurrences as noun. Hence, plan could be a verb as well as a noun. Second, in addition to the determination of the possible part of speeches, the verb table provides the possibility to reliably decide about the word form of a given verb. Since naming conventions typically ask for specific verb forms such as participles or imperatives, this is a key task for detecting violations of a given naming convention.

Algorithm 1 formalizes the usage of this technique. It requires a process model element and the two created tables as input. As a result the algorithm returns the element with the derived part of speech tags as annotation. The algorithm consists of a loop which iteratively goes through the element label and tags each word (lines 2-10). Therefore, we first use the part of speech table to infer all possible part of speech tags (line

Algorithm 1: Part of Speech Determination

```
1: tagProcessModelElement(Label element, Table posTable, Table verbTable)
2: for i=0 to i < element.getLength() do
3:     List posList = posTable.getPartOfSpeeches(element.get(i));
4:     for each tag ∈ posList do
5:         element.get(i).addPOS(tag);
6:         if tag = 'VERB' then
7:             String verbForm = verbTable.getForm(element.get(i));
8:             element.get(i).setVerbForm(verbForm);
9:     if posList.getLength() > 1 then
10:        element.get(i).hasMultipleTags = true;
11: return element;
```

3). Subsequently, we use a loop to annotate the considered word with each tag from the derived list (line 5). In case the considered tag represents a verb, the verb table is further used to derive the according verb form (line 7). The inferred verb form is then added to the annotation of the word (line 8). Finally, if the derived part of speech list contains more than a single entry, the variable *hasMultipleTags* is set to *true* (line 10).

3.2.2. Ambiguity Resolution

The ambiguity resolution component aims for adequately resolving cases where one word of the element label is associated with multiple part of speech tags. This is a crucial step as the compliance of a given label with the required linguistic pattern is based on their part of speeches. As an example consider the English activity pattern. If the first word in the label (e.g. *plan*) can be classified as verb and as noun, a wrong classification leads to a wrong decision with regard to its compliance with the naming convention. A similar situation applies for the event pattern which is used for Portuguese process models. If a Portuguese event ends with the word *estado*, it can represent the noun *state* or the participle of the word *estar* (to be). If it is erroneously classified as a noun, the considered event label will be incorrectly classified as naming convention violation.

For the solution of this problem we adapt a technique from prior research [53]. The core idea is to use different stages of label context for determining the part of speech of a given word (see Figure 4). In total we consider four different stages, organized from most local context to most generic: (I) the label itself, (II) the process model, (III) the process model collection and (IV) knowledge on word frequencies derived from the employed text corpus. In the beginning the algorithm tries to use the information which is included in the label. If we for instance consider the Portuguese event label *Alteração do estado*, the word *estado* can be doubtlessly recognized as noun since it is preceded by the preposition *do*. If the information in the label is not sufficient for disambiguating the considered word, the scope is broadened to the process model. For instance, the English word *plan* could be identified as noun since the process model contains several other activities using the business object *plan*. Similarly, the whole process model collection can be checked for the typical part of speech of the considered word. If the process model collection is still not sufficient to

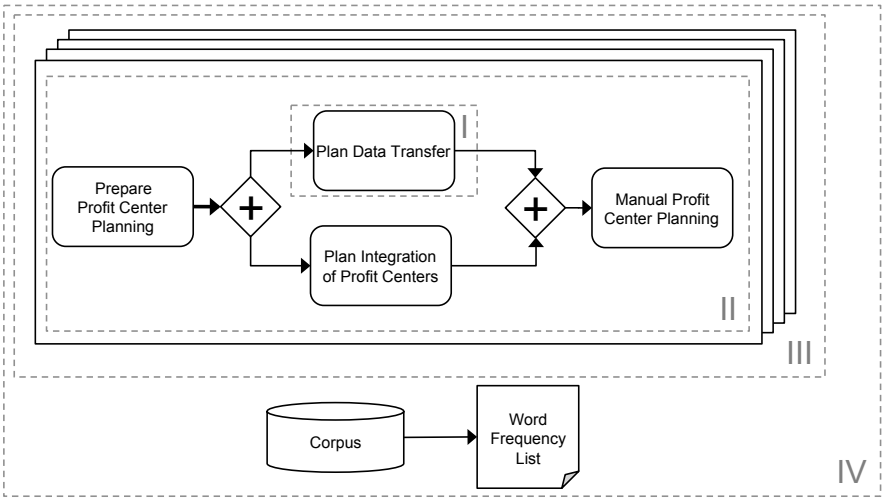

Figure 4: Disambiguation Strategy for Zero-Derivation Cases

Algorithm 2: Derivation of Linguistic Pattern

```
1: deriveLinguisticPattern(Label element)
2: String linguisticPattern;
3: for i=0 to i < element.getLength() do
4:     if element.get(i).hasMultipleTags = true then
5:         String actualPOS = resolveAmbiguity(element.get(i));
6:         linguisticPattern = linguisticPattern + actualPOS;
7: return linguisticPattern;
```

determine the part of speech, we employ knowledge about the frequency of each part speech in the target language. By consulting the part of speech table we derived from text corpus we can easily learn about the frequency of each part of speech. For instance the *OANC* corpus contains 226 occurrences for *credit* as verb and 1,672 as noun. Accordingly, *credit* is in general more likely to be a noun.

Algorithm 2 illustrates the derivation of the linguistic pattern from the element and its annotation. To accomplish this, each word of the input element is investigated in the context of a loop (lines 3-6). If the considered word carries more than a single part of speech tag, the above introduced technique is used to decide on the actual part of speech (line 5). Afterwards, the derived tag is added to the linguistic pattern variable (line 6). Once all words have been investigated and the linguistic pattern was constructed, the pattern is returned (line 7). This pattern then serves as input for the comparison component.

3.2.3. Comparison

In the comparison component we check whether the linguistic pattern from the input element is equivalent to the required linguistic pattern from the naming convention. Therefore, we iteratively go through the label and check the compliance of the current part of speech with the linguistic pattern. Algorithm 3 formalizes this procedure.

Algorithm 3: Checking of Naming Conventions

```
1: fulfillsNamingConvention(String element, String convention)
2: boolean correct = true;
3: int elemIndex = 1;
4: int convIndex = 1;
5: while elemIndex ≤ element.getLength() do
6:     String requiredPOS = convention.get(convIndex);
7:     List currentPOSList = getPartOfSpeech(element.get(elemIndex));
8:     if convention.get(convIndex).isOptional() then
9:         if requiredPOS ≠ currentPOS then
10:            convIndex = convIndex+1;
11:    else
12:        if requiredPOS ≠ currentPOS then
13:            correct = false;
14:            break;
15:        if currentPOS = 'NOUN' ∧ containsCompositeNoun(elemIndex,element.getLength()) then
16:            elemIndex = getEndOfCompositeNoun(elemIndex,element.getLength())+1;
17:            convIndex = convIndex+1;
18:        else
19:            elemIndex = elemIndex+1;
20:            convIndex = convIndex+1;
21: return correct;
```

The algorithm requires two input parameters: the labeled element which needs to be checked and the linguistic pattern derived from the naming convention. In the beginning of the algorithm the boolean variable *correct* is set to *true* and the index variable for the element and the convention are initialized (line 2-4).

If the required part of speech belongs to an optional element in the linguistic pattern and the current and required part of speech do not match, the index of the convention string is increased (lines 8-10). As a result, the next iteration of the loop compares the same word from the element string with the subsequent word in the convention string. In case the required part of speech belongs to a mandatory element, the current and the required part of speech have to match. Otherwise, a violation is detected and the variable *correct* is set to *false* (lines 11-14). If the two part of speeches match, the subsequent lines handle the occurrence of composite nouns (lines 15-17). This is important as in many languages composite nouns are not represented by a single word but by a sequence of words. As examples consider the English word *service order* or the Portuguese noun *regras de negócio* (business rule). Accordingly, the function *containsCompositeNouns* checks whether the current word and the subsequent words from the element string represent a composite noun (line 15). If that is the case, the index of the element string is adjusted to the first word after the composite noun (line 16). Otherwise, if no composite noun is detected, both indexes are increased by one (lines 19-20).

As long as the index of the element string has not reached the last word, the next iteration is triggered. Finally, the value of the variable *correct* is returned (line 21). In case the user wishes to receive feedback on the applied label style, the algorithm is accordingly used to categorize the detected linguistic pattern.

Altogether, the pattern formalization is a preparatory step that has to be conducted once. The technique

with its three steps part of speech determination, disambiguation, and comparison can then be applied in an automatic fashion to analyze large collections of process models.

4. Empirical Evaluation

To demonstrate the capability of the technique for checking naming conventions, we conduct an evaluation experiment. We test the technique on nine process model collections from practice covering three different languages. The goal of the experiment is to learn how well the automatically recognized violations match a manually created benchmark. Section 4.1 describes the general setup of the experiment. Section 4.2 provides an overview of the utilized process model collections. Section 4.3 investigates the technique from a runtime performance perspective before Section 4.4 discusses the overall experiment results.

4.1. Evaluation Setup

The setup of the evaluation consists of a prototypical implementation of the presented algorithms and a manually created benchmark.

In order to test the technique, we implemented the algorithms in the context of a Java prototype. Since the technique is not depending on any linguistic tools, it was not necessary to include further libraries. As the test collection includes German, English and Portuguese process models, we accordingly implemented the part of speech determination solution for each of these languages. Therefore we used the publicly available corpora *OANC* (English), *Tiger* (German) and *Floresta* (Portuguese). We developed a simple parser for automatically extracting the part of speech and the verb table from the corpora. We complemented the verb table by implementing the according conjugation rules of the target language with a complete list of exceptions. The information required for this implementation can be easily obtained online.[2,3,4]

For the proper assessment of the violation detection technique, we created a benchmark for the test collection. We manually inspected each label in the test collection and derived the linguistic pattern. In addition, we categorized each label as *guideline compliant* or as *naming convention violation*. In order to decide on the required naming conventions we employed two sources. First, we used existing guidelines and recommendations from research and practice [58, 76, 1, 59]. Second, we consulted different industry partners who provided us with their process model collections. From their internal process model guidelines we were able to derive specific naming conventions.

Table 5 gives an overview of the allowed naming conventions for the experiment. Note that this configuration can be adapted to the preferences of organizations. Further, it can be also extended with patterns for other elements such as data objects. In general, the table reveals that there is a consensus with

[2]http://www.usingenglish.com/reference/irregular-verbs/
[3]http://www.orbilat.com/Languages/Portuguese/Grammar/ Verbs/index.html
[4]http://www.evertype.com/gram/german-verbs.html

Table 5: Allowed Naming Conventions for the Evaluation Experiment

	English	**German**	**Portuguese**
Activities	VO	OI	IS
Events	PS, MS, AS	PS, MS, AS	PS, AS
Gateways	PQ, IFQ, AQ, EQ	PQ, IFQ, AQ, EQ	PQ, IFQ, AQ, EQ

respect to event and gateway labels while there are different requirement for activity labels. For English activities there are several guidelines and recommendations available suggesting the use of the verb-object style as introduced in Section 2.3 [58, 76, 1, 59]. However, for other languages such as German or Portuguese the verb-object recommendation for activities is not used. A guideline from a large German health insurer suggests to employ the object-infinitive style. As the German language has two different imperative modes, the object-infinitive style conveys the information in a more neutral manner. A similar observation can be made for the Portuguese models. A guideline from a large Brazilian Energy Corporation asks for the application of the infinitive style. Accordingly, we determined the required naming conventions with these styles. For events and gateways the requirements from practice and academia are usually less specific. Silver [76] suggests that event labels should refer to a state and must be clearly separable from activities. The guideline from the Brazilian Energy Corporation explicitly requires the participle or the attribute style. Accordingly, we restrict the set of allowed patterns to these for the Portuguese models. For gateways the consensus is rather that gateway labels should be named with a question that refers to an action and a business object. Consequently, we consider all event question styles to be valid.

4.2. Test Collection Demographics

We designed the test collection with the aim to cover diverse types of naming conventions. Therefore, we included model collections varying in dimensions such as domain, natural language, modeling notation, and the share of labels violating the naming conventions defined in the previous section. Table 6 presents details of the used model collections. In total, we classify the models into three groups based on the natural language used:

1. English Process Model Collections

    - *SAP Reference Model (SAP)*: The SAP Reference Model represents the business processes of the SAP R/3 system in its version from the year 2000 [42, pp. 145-164]. It contains 604 Event-driven Process Chains (EPCs) which are organized in 29 functional branches such as sales and accounting.
    - *Insurance Model Collection (CH)*: The insurance model collection contains 119 EPCs dealing with the claims handling activities of an Australian insurance company. The insurance model set contains rather large processes with a high density of events.

Table 6: Details about used model collections

	English			*German*			*Portuguese*		
	SAP	**CH**	**AC**	**ITSM**	**HI**	**AC**	**EC**	**RF**	**AC**
Number of models	604	119	518	88	46	311	11	19	29
No. of Activities	2433	1655	4109	293	2117	2959	89	230	216
Average no. of activities per model	4.03	13.91	7.93	3.33	44.1	8.09	9.82	12.11	7.45
Average no. of words per label	3.50	6.98	3.66	3.98	4.65	2.92	4.1	4.79	3.6
Minimum no. of words. per label	1	2	1	2	1	1	2	1	1
Maximum no. of words per label	12	22	15	15	13	18	11	11	11
Share of naming convention violations	88.7%	41.0%	25.8%	31.7%	30.3%	33.7%	2.8%	33.0%	34.3%
No. of Events	6933	2325	553	584	1279	544	89	99	95
Average no. of events per model	11.48	19.54	1.07	6.64	26.65	1.75	8.09	5.21	3.28
Average no. of words per label	5.37	4.78	2.78	4.94	4.48	2.47	3.1	2.98	2.78
Minimum no. of words. per label	2	1	1	2	1	1	2	2	1
Maximum no. of words per label	13	21	11	13	12	7	9	6	8
Share of naming convention violations	40.4%	38.7%	67.1%	42.8%	33.2%	59.0%	2.8%	24.2%	71.6%
No. of (labeled) Gateways	-	-	235	-	-	220	-	-	37
Average no. of gateways per model	-	-	0.45	-	-	0.71	-	-	1.28
Average no. of words per label	-	-	2.57	-	-	2.36	-	-	3.03
Minimum no. of words. per label	-	-	1	-	-	1	-	-	1
Maximum no. of words per label	-	-	9	-	-	9	-	-	8
Share of naming convention violations	-	-	46.8%	-	-	35.0%	-	-	43.2%
Notation	EPC	EPC	BPMN	EPC	EPC	BPMN	EPC	EPC	BPMN

- *Academic Collection (AC)*: The Academic Collection includes 518 process models created with the Business Process Model and Notation (BPMN). The models stem from academic training and cover diverse domains.

2. German Process Model Collections

   - *Incident Management Collection (ITSM)*: This collection contains 88 EPCs covering the incident management processes from a large German IT service provider.
   - *Health Insurance Model Collection (HI)*: The Health Insurance Model Collection includes 48 EPCs, covering customer-oriented processes from a large German health insurer.
   - *Academic Collection (AC)*: The Academic Collection contains 311 processes modeled with BPMN. Similarly to the English Academic collection, the models stem from academic training.

3. Portuguese Process Model Collections

   - *Energy Corporation Collection (EC)*: This collection contains 11 EPCs from a large Brazilian Energy Corporation. As opposed to the other collections, it has been manually maintained in order to avoid naming convention violations.
   - *Research Funding Collection (RF)*: This collection contains EPC processes regarding a Brazilian research funding program including, for example, the activities of funding request and grant.

Table 7: Computation Performance

	English			*German*			*Portuguese*		
	SAP	**CH**	**AC**	**ITSM**	**HI**	**AC**	**EC**	**RF**	**AC**
All activities (ms)	134,722	63,370	169,617	2,977	29,323	26,403	290	310	475
Single activity (avg. ms)	55.37	38,29	41,27	10.16	9.91	8.92	3.26	1.35	2.19
All events (ms)	412,445	131,927	35,984	2,351	14,720	961	236	437	234
Single event (avg. ms)	59.49	56.74	153.12	4.03	11.51	1.77	2.65	4.41	2.46
All gateways (ms)	-	-	6,461	-	-	109	-	-	20
Single gateway (avg. ms)	-	-	27.49	-	-	0.5	-	-	0.54

- *Academic Collection (AC)*: The Academic Collection contains 29 processes modeled with BPMN. Similarly to the English and German Academic collection, the models stem from academic training.

4.3. Performance Results

We designed the technique to support modelers in the context of a modeling tool during the modeling process. Hence, in order to immediately indicate naming convention violations, the implementation must be adequately fast. Accordingly, we measure the computation time for each investigated model collection. We tested the computation on a MacBook Pro with a 2.26 GHz Intel Core Duo processor and 4 GB RAM, running on Mac OS X 10.6.8 and Java Virtual Machine 1.5. In order to avoid distortions due to one-off setup times, we ran the algorithms twice and measured the second run only. Table 7 summarizes the total and average execution times of the naming convention violation detection for each element type.

The numbers illustrate two essential points. First, the execution time depends on the ambiguity of the target language. We observe significantly higher execution times for the English models for all element types. This is in line with the parsing complexity of English. As many disambiguation runs are necessary to determine the label style, the overall computation time is longer. As a second point we learn that also the share of violating labels effects the overall execution time. This is especially emphasized by the events of the English Academic Collection. The share of 67.1% of convention violations results in an increase of computation time. The reason for this relates to the algorithm design. In order to classify a given label as *violation*, all other possibilities have to be excluded. However, all in all the algorithm yields execution times which are well suited for supporting modelers during the modeling process. Even the worst average execution time of 153 milliseconds is fast enough for directly indicating a violation. Hence, we consider the implementation to have a sufficient performance in terms of computation time.

4.4. Experiment Results

We furthermore assess the performance of the technique using the information retrieval metrics precision, recall, and f-measure. In the context of naming convention violation detection, the precision value is the number of correctly identified violations divided by the number of retrieved violations. The recall is the number of correctly identified violations divided by the total number of violations in the considered model

Table 8: Evaluation Results

		English			*German*			*Portuguese*		
		SAP	**CH**	**AC**	**ITSM**	**HI**	**AC**	**EC**	**RF**	**AC**
Activities	Recall	98.0%	96.6%	90.0%	96.9%	99.8%	96.7%	100%	100%	100%
	Precision	95.5%	99.7%	93.7%	96.9%	92.4%	95.5%	100%	100%	95.9%
	F-Measure	96.7%	98.1%	91.8%	96.9%	96.0%	96.1%	100%	100%	97.9%
Events	Recall	99.3%	93.0%	99.5%	100%	94.1%	96.6%	100%	100%	100%
	Precision	98.2%	98.7%	99.0%	96.5%	99.8%	99.0%	100%	92.0%	100%
	F-Measure	98.7%	95.8%	99.2%	98.2%	96.8%	97.8%	100%	96.0%	100%
Gateways	Recall	-	-	97.3%	-	-	97.4%	-	-	100%
	Precision	-	-	95.5%	-	-	94.9%	-	-	94.1%
	F-Measure	-	-	96.4%	-	-	96.2%	-	-	97.0%

collection. As it is important for our scenario that both metrics yield sufficiently high values, we also calculate the f-measure, which is the harmonic mean of precision and recall [2].

Table 8 summarizes the results of the automatic violation detection for all considered process model collections. In general, the numbers show that the technique for detecting naming convention violations works satisfactory. Even the lowest obtained f-measure is above 91%. However, we still observe notable differences among the different model collections and the considered languages.

For a considerable difference among *model collections* consider the activity related f-measures for the English collections. While for instance the f-measure of activities in the CH collection yields a f-measure of 98.1%, the f-measure for the AC collection is only 91.8%. Such differences can be explained with the different modeling style and emphasizes the importance of including different collections in the evaluation experiment. Depending on the modeling style, ambiguous cases can be more or less frequent. As the AC collection was mainly created by students, these models contain more ambiguous cases which in turn results in a slightly poorer performance. Nevertheless, as indicated by the recall value, still 90% of all naming convention violations could be successfully detected.

The *effect of the target language* is especially highlighted by the difference between the English and Portuguese model collections. Although the English language has a huge vocabulary, its grammar is rather simple and there are many words suffering from the zero-derivation ambiguity. In other languages, such as Portuguese or German, we observe a lot more inflectional changes. For instance, an infinitive verb in Portuguese changes if it is used as an imperative. In English this is not the case. As example consider the Portuguese verb *auditar* (to audit). While the English imperative of *to audit* is given by *audit* the Portuguese verb *auditar* turns into the imperative form *audite*. Thus, labels violating the naming convention for activities (to start with an infinitive verb) could be reliably detected. However, also Portuguese suffers from the zero-derivation ambiguity. As an example consider the Portuguese Label *Alteração do estado* (Change of Status) where the word *estado* can be a noun and a participle verb. Although this example represents a zero-derivation case, it can be quite easily resolved. The preposition *do* before estado reveals

that we face a composite noun and that hence *estado* cannot be a verb. As a result of this, we partially achieved recall and precision values of 100%.

Surprisingly, there is no notable effect of the process model element type. In some cases the f-measure for the activities is higher than the corresponding event value as for instance in the CH and RF collection. Nevertheless, we also observe the opposite. In the HI collection and the Portuguese AC collection the f-measure of the events is higher than the corresponding activity value.

Based on the results of this analysis, we investigated the reasons for misclassification. In total, we identified four main reasons why a violation was not detected or why a correct label was classified as violation:

- *Erroneously resolved ambiguity:* As extensively discussed in this paper, the proper resolution of ambiguous words is one of the major challenges. Although the technique works reliably, ambiguity still represents a major error source. The importance of this problem is heavily depending on the language. While we have not found any erroneously resolved ambiguities for Portuguese models, ambiguity is in many cases the error source among English models. For German models the problem is less drastic but still present. As an example for a wrongly resolved German zero-derivation case consider the label *Rate überfällig* (Instalment overdue). The problem here is that *Rate* can represent a German noun with the meaning *instalment* or a German verb with the meaning *to guess*. In order to overcome these problems, text corpora could be used to identify common combinations of words. In the previously introduced case of the German label, a corpus could for instance reveal that the word *Rate* in combination with *überfällig* indicates that *Rate* is more likely to represent a noun.

- *Typographic errors:* If words in the labels contain typographic errors, it is not possible to find them in the employed corpora. Hence, their part of speech cannot be determined. As an example consider the label *Parts avalable*. Here the missing *i* could cause a misclassification of the label. In order to overcome this problem, we employed the Levenshtein string edit distance [54] and allow words to slightly deviate from the terms in the corpora. For instance with a Levenshtein distance of 1, the word *avalable* with a missing *i* is still correctly associated with the word *available* in the corpus. However, for bigger mistakes that strategy cannot be employed. If we for instance allow for a Levenshtein distance of 2 the word *test* could also be associated with its participle *test* or even completely different words as for instance *tear*. As a solution for such cases, we propose to refine the Levenshtein metric. By for instance including the closeness of keys on the keyboard the metric could be improved to better reflect mistakes which really stem from typing errors.

- *Abbreviations:* Similar to typographic errors, most abbreviations will not be found in corpora. Thus, a considered verb cannot be reliably assigned to a part of speech class. Standard abbreviations are usually not a problem as they are in many cases also used in the underlying corpora. However, non-standard

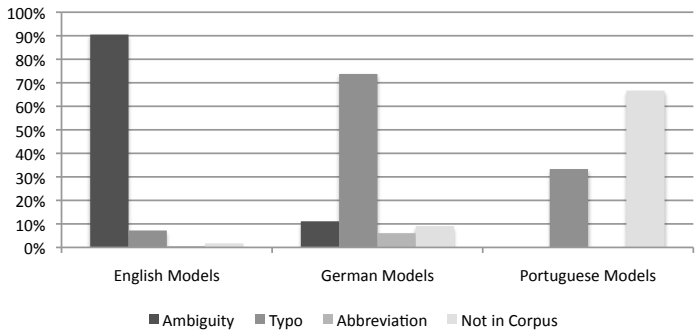

Figure 5: Distribution of Error Sources

abbreviations will always result in a misclassification. An example for an arbitrarily abbreviated label is given by the German label *Prüfungsverfahren abgeschl.* (Audit completed). Here the German participle verb *abgeschlossen* (completed) is abbreviated in the middle of the word. Although such cases are rare, they can lead to a decrease of the overall performance. As a solution for this problem, we propose to identify the abbreviated word in the corpus using a substring match. For the introduced example, the search for *abgeschl* would accordingly result in a match. In case of multiple matches, the context of the label could be used to select the most likely option.

- *Word not in corpus:* Apart from typos and abbreviations, we also found completely correct words which were not correctly assigned to the according part of speech. We encountered that problem in two cases. The first case refers to domain-specific or rather modern words. Especially older corpora do not contain words such as *digitalize*. For domain specific words, as for instance system terms (e.g SAP or ECPA), the problem is similar. To resolve this problem, the corpora could be complemented with a list of domain-specific terms. As many companies maintain such lists, we consider this to be a reasonable solution. The second case is language specific and concerned with German prefixes. The German language allows to complement verbs with a variety of prefixes (similarly to e.g. *grade* and *upgrade*). Thus, for instance *machen* (to make) can turn into *weitermachen* (to continue). Unfortunately, not all of these cases are covered by corpora. As a solution, we could employ a list of prefixes to identify composed verbs. After removing the prefix, the word can be accordingly identified in the corpus.

Figure 5 illustrates the distribution of the error sources grouped by the language of the model collections. In general, it highlights that the language has an important influence on the distribution of the error sources. In English models, particularly ambiguity remains the major problem. This can be explained by the general degree of ambiguity of the English language. In comparison to German and Portuguese, English suffers much more from the zero-derivation ambiguity. As a result, English models contain more ambiguous cases that are subject to misinterpretation. Although the German performance suffers from ambiguity as well, typos

play a bigger role here. Especially the German Academic collection contains a lot of typographical mistakes. Due to the partially complex spelling rules of German, this can also be considered as a language-specific problem. In Portuguese, the majority of errors corresponds to words not being present in the corpus. Here, we observed many domain-specific words in the models that are not covered by the corpus. In contrast to English and German models, no ambiguity and abbreviation errors were found. Altogether, it can be stated that English models are particularly subject to errors caused by ambiguity and German models suffer from the more complex spelling. The performance of the Portuguese models cannot be explained by a particular characteristic of the Portuguese language. It is rather the result of the specificity of the vocabulary in the investigated models.

To demonstrate the applicability of the proposed mechanisms for dealing with typographical errors, abbreviations, and missing words, we conducted an additional test. Since missing words and abbreviations can be easily added to the corpus, it was not necessary to change the implementation. Hence, we added the observed cases of missing words and abbreviations to the corpus. In order to more sophistically deal with typographical errors we added the possibility of making use of an advanced Levenshtein metric. We implemented a flag for accordingly turning it off and on. As a result of the change, the 27 errors caused by missing words and abbreviations could be reduced to zero. Since missing words or abbreviations only need to be added once, such a tuning represents a reasonable one-time effort. The effect of the advanced Levenshtein metric is little less significant. However, out of 104 typos 69 could be successfully identified. The reason for non-identified typos lies in the complexity of the typographical errors. Some can not be traced back to wrong key strokes on the neighbour key. Nevertheless, as previously discussed, a further relaxation of the Levenshtein metric would result in a decrease of the quality. The runtime of the technique was only slightly affected by these changes. It increased by less than one percent. This result can be explained based on two observations. First, adding a small set of new words to a corpus consisting of several thousand words does not significantly increase the search space. Second, the Levenshtein metric is only computed if a word is not found in the corpus. Hence, the additional mechanisms do not significantly increase the runtime. Altogether, these extensions further increased the accuracy of our technique.

4.5. Implications

The development of an automatic technique for checking naming conventions has implications for research and practice. In particular, we discuss implications for the automated quality assurance of process models and the general applicability of natural language techniques for languages other than English. Furthermore, implications for practice are highlighted.

The work presented in this paper complements prior research on *automatic quality assurance* and corresponding techniques for checking formal model properties, model layout and use of model elements. The importance of naming conventions is highlighted by guidelines such as 7PMG [58] and supported by

experimental research on process model understanding [59]. Once techniques for the automatic checking of formal model properties became available, this sparked an important stream of research on the quality of process models in practice [57]. In a similar way, we expect that automatic techniques for checking naming conventions, as the one presented in this paper, will inform model quality assurance and reveal novel insights into when and why designers create good or bad models. Such insights are likely to be beneficial for modeling education and for refining existing guidelines.

The design principles introduced in this work do also have implications for the *applicability* of natural language techniques to languages other than English. The results of the evaluation demonstrate that language processing in the context of conceptual models is not necessarily bound to linguistic tools such as WordNet. The corpus-based technique presented in this paper provides the foundation for adaptations to other languages for guideline checking as it enables the tool independent part of speech tagging of small language fragments. Hence, the presented technique could also help to enforce and check language specific aspects of use case diagrams, feature models, or goal models. For use case diagrams, it is recommended to formulate use cases as verb phrases similar to the verb-object style [13]. In feature diagrams it is typically required to name features as noun phrases or single nouns [51]. For the definition of goals in the context of goal models, it is also suggested to use verb phrases with in imperative verb in the beginning [72]. Altogether, all these modeling techniques face challenges comparable to checking naming conventions in process models, and can therefore directly benefit from the results presented in this paper. Nevertheless, where applicable, also approaches beyond the area of conceptual modeling could benefit from the corpus-based part of speech determination and hence become more flexible in terms of language adaptability.

From a *practical perspective*, the presented technique helps companies to effectively and efficiently assure the language quality of their models even if they cannot make use of the English WordNet. As we experienced that companies typically model their processes using their local language, this is an important feature. Considering concrete implementation scenarios, we can differentiate between two main cases. First, the technique could be implemented in a modeling tool and indicate convention violations to the modeler already during the modeling process. As a result, modelers are directly provided with feedback and can accordingly adapt their style of modeling. This is likely to avoid modeling errors and increase the quality of models right from the start. This can help to address pitfalls of process modeling which relate to the often low expertise of modelers in practice [73]. Second, the technique could be employed for the analysis of existing modeling collections. A resulting report can be used by process modeling experts to fix quality issues which are spotted as convention violations. However, independently of the actual implementation scenario, our approach helps to check naming conventions automatically, which used to be a quality aspect that had to be checked manually. This in turn assures that the process models can actually meet their goals of effectively communicating the company's operations.

As a concrete scenario from practice we can report on our experience of introducing the tool in a a global

medical engineering company with 3,000 employees. The company has 25 different modelers working at different sites. In order to harmonize the processes, the company further employs a team of three experts leading the overall modeling initiative. Recently, this company decided to integrate the here presented automatic naming convention violation detection along with other automatic checks into their modeling tool. Due to the introduction of the automatic checks, the time for assuring the quality of the models could be significantly decreased. As a result, the whole modeling initiative gained in acceptance since the modelers could work more independently. This example vividly illustrates the huge effect of the automated support technique presented in this paper.

4.6. Threats to Validity

In this section, we provided an extensive evaluation of the introduced technique for detecting naming convention violations. Nevertheless, threats to validity are an important perspective for discussing empirical research [88, 89]. Hence, we use this subsection to reflect on the limitations of the presented work. In particular, there are two important points that need to be highlighted in this context: the representativeness of the investigated process model collections and the language independence of the technique.

Concerning the *representativeness*, it must be noted that the analyzed process model collections are not representative in a statistical sense. This means that we may encounter additional labeling styles in the collections of other organizations. However, for two reasons we are confident that the presented classification extensively covers the labeling styles from practice. First, in the context of further research projects, we also work with collections of other companies and never encountered styles that deviate from those we presented in the classification. Second, considering the possible options of combining verb and subject, there are no grammatically meaningful combinations left. Hence, we have strong reasons to believe that the classification is complete although we can not prove this in a formal or statistical manner.

With regard to the *language independence* we have to highlight that we only considered three languages, namely English, German, and Portuguese. Hence, the findings cannot be necessarily adapted to other languages. Nevertheless, our experience with Dutch and Slovene models shows that we observe the same styles in these languages. For languages that are not part of the Indo-European language family, it is harder to assess whether the concepts can be directly transferred as such languages may have other grammatical features. Still, even if such languages require an adaptation, for instance in the ambiguity resolution component, the general strategy of building on a labeling style classification can be transferred and applied. Thus, we are convinced that the presented concepts represent a valuable starting point for any given language.

5. Conclusion

In this paper we addressed the problem of automatically checking naming conventions of process model elements. We designed a technique which is independent of WordNet and can hence be easily transferred

to other languages. In particular, we employed text corpora and linguistic knowledge to create a part of speech determination component. Using the computed part of speech information, we then decided about the according linguistic patterns and respective violations. We evaluated the technique using nine business process model collections from practice, covering three different languages. The results demonstrate the applicability of our technique for reliably detecting naming convention violations. Although the target languages differed in the morphological complexity, we consistently obtained f-measure values between 91.8% and 100%.

Despite the promising results, our work has to be reflected from the perspective of some limitations. In this work we tested the approach with languages belonging to the Romanian and Germanic sub-branches of the Indo-European language family. Hence, our evaluation results cannot be automatically transferred to other language families as for instance Asian languages. However, our approach is designed as a language independent solution, which can be theoretically applied to any language. The adaptation to different languages in this paper showed that the main challenge is given by the proper resolution of ambiguous cases. As even the results for English, a highly ambiguous language, were very reliable, we believe that our technique is also valuable for other language families.

The results of our evaluation are also bound to the employed process model collections as our collections are not representative in a statistical sense. However, we tried to avoid biased results by carefully selecting collections with diverse characteristics in various dimensions. Altogether, we investigated more than 27,000 labels in the context of our evaluation. Hence, we are confident that our results properly reflect the potential of the violation detection technique in practice.

In future work we plan to adapt the technique to additional languages including Slavic and Asian languages. Moreover, we aim to further examine the usability perspective and learn how well companies can benefit from such automated quality assurance techniques.

References

[1] T. Allweyer, BPMN 2.0 - Business Process Model and Notation, 2nd ed., Books on De-mand GMBH, Norderstedt, 2009.

[2] R.A. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, ACM Press / Addison-Wesley, 1999.

[3] I.S. Bajwa, M.A. Choudhary, From natural language software specifications to uml class models, LNBIP, Springer, Heidelberg, 2012, pp. 224–237.

[4] G.D. Battista, P. Eades, R. Tamassia, I.G. Tollis, Graph Drawing: Algorithms for the Visualization of Graphs, 1st ed., Prentice Hall PTR, 1998.

[5] J. Becker, P. Bergener, D. Breuker, M. Räckers, An Empirical Assessment of the Usefulness of Weakness Patterns in Business Process Redesign, in: Proceedings of the 20th European Conference on Information Systems (ECIS 2012), AIS, 2012. Barcelona, Spain.

[6] J. Becker, D. Breuker, P. Delfmann, H.A. Dietrich, M. Steinhorst, Identifying Business Process Activity Mappings by Optimizing Behavioral Similarity, in: Proceedings of the 18th Americas Conference on Information Systems (AMCIS 2012), AIS, 2012. Seattle, USA.

[7] J. Becker, P. Delfmann, S. Herwig, L. Lis, A. Stein, Towards Increased Comparability of Conceptual Models - Enforcing Naming Conventions through Domain Thesauri and Linguistic Grammars, in: Proceedings of the 17th European Conference on Information Systems (ECIS 2009), AIS, 2009.

[8] E.R. Benoit Lavoie, Owen Rambow, The ModelExplainer, in: Proceedings of the 8th international Workshop on Natural Language Generation, pp. 9–12.

[9] E. Bertino, E. Ferrari, V. Atluri, The Specification and Enforcement of Authorization Constraints in Workflow Management Systems, ACM Transactions on Information and System Security 2 (1999) 65–104.

[10] B. Bickel, J. Nichols, Inflectional Synthesis of the Verb, in: The World Atlas of Language Structures, Oxford University Press, 2005, pp. 94– 97.

[11] N. Bolloju, C. Schneider, V. Sugumaran, A knowledge-based System for Improving the Consistency between Object Models and Use Case Narratives, Expert Systems with Applications 39 (2012) 9398 – 9410.

[12] P.F. Brown, J. Cocke, S.A. Della Pietra, V.J. Della Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, P.S. Roossin, A Statistical Approach to Machine Translation, Computational Linguistics 16 (1990) 79–85.

[13] A. Cockburn, Writing Effective Use Cases, Addison-Wesley Boston, 2001.

[14] B. Comrie, Language Universals and Linguistic Typology, 2nd edition ed., University of Chicago, 1989.

[15] J. Crampton, H. Khambhammettu, Delegation and Satisfiability in Workflow Systems, in: Proceedings of 13th ACM Symposium on Access Control Models and Technologies, ACM, New York, NY, USA, 2008, pp. 31–40.

[16] H. Dalianis, A Method for Validating a Conceptual Model by Natural Language Discourse Generation, in: Proceedings of the Conference on Advanced Information Systems Engineering, LNCS, Springer, 1992, pp. 425–444.

[17] I. Davies, P. Green, M. Rosemann, M. Indulska, S. Gallo, How do Practitioners use Conceptual Modeling in Practice?, Data and Knowledge Engineering 58 (2006) 358–380.

[18] D.K. Deeptimahanti, M.A. Babar, An Automated Tool for Generating UML Models from Natural Language Requirements, in: Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering, ASE '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 680–682.

[19] D.K. Deeptimahanti, R. Sanyal, Semi-automatic Generation of UML Models from Natural Language Requirements, in: Proceedings of the 4th India Software Engineering Conference, ISEC '11, ACM, New York, NY, USA, 2011, pp. 165–174.

[20] P. Delfmann, S. Herwig, L. Lis, A. Stein, Supporting Distributed Conceptual Modelling through Naming Conventions - A Tool-based Linguistic Approach, Enterprise Modelling and Information Systems Architectures 4 (2009) 3–19.

[21] R. Dixon, Deriving Verbs in English, Language Sciences 30 (2008) 31–52.

[22] J.B. Dominic Breuker, Daniel Pfeiffer, Reducing the Variations in Intra- and Interorganizational Business Process Modeling - An Empirical Evaluation, in: Proceedings of the Internationale Tagung Wirtschaftsinformatik.

[23] P. Effinger, N. Jogsch, S. Seiz, On a Study of Layout Aesthetics for Business Process Models Using BPMN, in: Proceedings of the Second International Workshop on BPMN, volume 67 of *Lecture Notes in Business Information Processing*, Springer, Potsdam, Germany, 2010, pp. 31–45.

[24] M. Ehrig, A. Koschmider, A. Oberweis, Measuring Similarity between Semantic Business Process Models, in: APCCM 2007, Australian Computer Science Communications, 2007, pp. 71–80.

[25] D. Fahland, C. Favre, J. Koehler, N. Lohmann, H. Völzer, K. Wolf, Analysis on Demand: Instantaneous Soundness Checking of Industrial Business Process Models, Data & Knowledge Engineering 70 (2011) 448–466.

[26] C. Fillies, G. Wood-Albrecht, F. Weichhardt, Pragmatic Applications of the Semantic Web using SemTalk, Computer Networks 42 (2003) 599–615.

[27] G. Fliedl, C. Kop, H.C. Mayr, A. Salbrechter, J. Vöhringer, G. Weber, C. Winkler, Deriving Static and Dynamic Concepts from Software Requirementsusing Sophisticated Tagging, Data & Knowledge Engineering 61 (2007) 433–448.

[28] C. Francescomarino, P. Tonella, Supporting Ontology-Based Semantic Annotation of Business Processes with Automated

Suggestions, in: Enterprise, Business-Process and Information Systems Modeling, volume 29 of *Lecture Notes in Business Information Processing*, Springer Berlin Heidelberg, 2009, pp. 211–223.

[29] J. Freund, B. Rücker, Praxishandbuch BPMN 2.0, 3rd ed., Carl Hanser Verlag GmbH & CO. KG, 2012.

[30] C. Friedman, P. Alderson, J. Austin, J. Cimino, S. Johnson, A General Natural-language Text Processor for Clinical Radiology, Journal of the American Medical Informatics Association 1 (1994) 161–174.

[31] F. Friedrich, J. Mendling, F. Puhlmann, Process Model Generation from Natural Language Text, in: CAiSE 2011, volume 6741 of *LNCS*, Springer, 2011, pp. 482–496.

[32] V. Gacitua-Decar, C. Pahl, Automatic Business Process Pattern Matching for Enterprise Services Design, in: IEEE Congress on Services Part II, IEEE Computer Society, 2009, pp. 111–118.

[33] A. Gangopadhyay, Conceptual Modeling from Natural Language Functional Specifications, Artificial Intelligence in Engineering 15 (2001) 207 – 218.

[34] A. Ghose, G. Koliadis, A. Chueng, Rapid Business Process Discovery (R-BPD), in: ER 2007, volume 4801 of *LNCS*, Springer, Berlin, Heidelberg, 2007, pp. 391–406.

[35] A.K. Ghose, G. Koliadis, A. Chueng, Process Discovery from Model and Text Artefacts, in: 2007 IEEE Congress on Services, IEEE Computer Society, 2007, pp. 167–174.

[36] F. Gomez, C. Segami, C. Delaune, A System for the Semiautomatic Generation of E-R Models from Natural Language Specifications, Data and Knowledge Engineering 29 (1999) 57 – 81.

[37] J.C. Goncalves, F.M. Santoro, F.A. Baiao, Business Process Mining from Group Stories, International Conference on Computer Supported Cooperative Work in Design (2009) 161–166.

[38] V. Gruhn, R. Laue, Detecting Common Errors in Event-Driven Process Chains by Label Analysis, Enterprise Modelling and Information Systems Architectures 6 (2011) 3–15.

[39] W.J. Hutchins, Machine Translation: Past, Present, Future, John Wiley & Sons, Inc., New York, NY, USA, 1986.

[40] P. Jackson, I. Moulinier, Natural Language Processing for Online Applications. Text Retrieval, Extraction and Categorization, volume 5 of *Natural Language Processing*, John Benjamins, 2002.

[41] F. Jelinek, Statistical Methods for Speech Recognition, The MIT Press, 1998.

[42] G. Keller, T. Teufel, SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping, Addison-Wesley, 1998.

[43] W. Kettinger, J. Teng, S. Guha, Business Process Change: a Study of Methodologies, Techniques, and Tools, MIS quarterly (1997) 55–80.

[44] K. Kettunen, M. Sadeniemi, T. Lindh-Knuutila, T. Honkela, Analysis of EU Languages Through Text Compression, in: FinTAL, volume 4139 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 99–109.

[45] D. Klein, C.D. Manning, Accurate Unlexicalized Parsing, 41st Meeting of the Association for Computational Linguistics (2003) 423–430.

[46] R. Knackstedt, D. Kuropka, O. Müller, An Ontology-based Service Discovery Approach for the Provisioning of Product-Service Bundles, in: Proceedings of the European Conference on Information Systems.

[47] E. Knauss, D. L., Using the Friction between Business Processes and Use Cases in SOA Requirements, Computer Software and Applications Conference, Annual International 0 (2008) 601–606.

[48] A. Koschmider, E. Blanchard, User Assistance for Business Process Model Decomposition, in: In First IEEE International Conference on Research Challenges in Information Science, pp. 445–454.

[49] A. Koschmider, T. Hornung, A. Oberweis, Recommendation-based Editor for Business Process Modeling, Data and Knowledge Engineering 70 (2011) 483 – 503.

[50] S. Lahtinen, J. Peltonen, Adding Speech Recognition Support to UML Tools, J. Vis. Lang. Comput. 16 (2005) 85–118.

[51] K. Lee, K.C. Kang, J. Lee, Concepts and Guidelines of Feature Modeling for Product Line Software Engineering, in: Proceedings of the Seventh Conference on Software Reuse: Methods, Techniques, and Tools, Springer-Verlag, 2002, pp.

62–77.

[52] H. Leopold, J. Mendling, Automatic Derivation of Service Candidates from Business Process Model Repositories, in: Business Information Systems, pp. 84–95.

[53] H. Leopold, S. Smirnov, J. Mendling, On the Refactoring of Activity Labels in Business Process Models, Information Systems 37 (2012) 443 – 459.

[54] V. Levenshtein, Binary Codes Capable of Correcting Deletions, Insertions, and Reversals, Cybernetics and Control Theory 10 (1966) 707–710.

[55] T. Malone, K. Crowston, G. Herman (Eds.), Organizing Business Knowledge: The MIT Process Handbook, The MIT Press, 2003.

[56] J. McWhorter, The World's Simplest Grammars are Creole Grammars, Linguistic Typology 5 (2001) 125–166.

[57] J. Mendling, Empirical Studies in Process Model Verification, LNCS Transactions on Petri Nets and Other Models of Concurrency II, Special Issue on Concurrency in Process-Aware Information Systems 2 (2009) 208–224.

[58] J. Mendling, H.A. Reijers, W.M.P. van der Aalst, Seven Process Modeling Guidelines (7PMG), Information and Software Technology 52 (2010) 127–136.

[59] J. Mendling, H.A. Reijers, J. Recker, Activity Labeling in Process Modeling: Empirical Insights and Recommendations, Information Systems 35 (2010) 467–482.

[60] F. Meziane, N. Athanasakis, S. Ananiadou, Generating Natural Language Specifications from UML Class Diagrams, Requirements Engineering 13 (2008) 1–18.

[61] L. Miles, Techniques of Value Analysis and Engineering, McGraw-Hill, 1961.

[62] G.A. Miller, WordNet: a Lexical Database for English, Communications of the ACM 38 (1995) 39–41.

[63] A. Montes, H. Pacheco, H. Estrada, O. Pastor, Conceptual model generation from requirements model: A natural language processing approach, in: Natural Language and Information Systems, volume 5039 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2008, pp. 325–326.

[64] P. More, R. Phalnikar, Article: Generating uml diagrams from natural language specifications, International Journal of Applied Information Systems 1 (2012) 19–23.

[65] M. zur Muehlen, J. Recker, How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation, in: Z. Bellahsene, M. Léonard (Eds.), 20th International Conference on Advanced Information Systems Engineering, volume 5074 of *Lecture Notes in Computer Science*, Springer, Montpellier, France, 2008, pp. 465–479.

[66] N. Omar, R. Hassan, H. Arshad, S. Sahran, Automation of Database Design through Semantic Analysis, in: Proceedings of the 7th WSEAS international conference on Computational intelligence, man-machine systems and cybernetics, CIMMACS'08, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 2008, pp. 71–76.

[67] N. Peters, M. Weidlich, Automatic Generation of Glossaries for Process Modelling Support, Enterprise Modelling and Information Systems Architectures 6 (2011) 30–46.

[68] A. Polyvyanyy, L. García-Bañuelos, M. Dumas, Structuring Acyclic Process Models, in: R. Hull, J. Mendling, S. Tai (Eds.), Business Process Management, volume 6336 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2010, pp. 276–293.

[69] L. Rabiner, B.H. Juang, Fundamentals of Speech Recognition, Prentice Hall, 1993.

[70] J. Recker, J. Mendling, On the Translation between BPMN and BPEL: Conceptual Mismatch between Process Modeling Languages, in: Proceedings of the CAiSE Workshops at the 18th Conference on Advanced Information Systems Engineering (CAiSE 2006), pp. 521–532.

[71] D. Richards, A.B. Fure, O. Aguilera, An Approach to Visualise and Reconcile Use Case Descriptions from Multiple Viewpoints, in: Proceedings of the 11th IEEE International Conference on Requirements Engineering, RE '03, IEEE

Computer Society, 2003, pp. 373–374.

[72] C. Rolland, C. Souveyet, C. Achour, Guiding goal modeling using scenarios, Software Engineering, IEEE Transactions on 24 (1998) 1055–1071.

[73] M. Rosemann, Potential Pitfalls of Process Modeling: Part A, Business Process Management Journal 12 (2006) 249–254.

[74] A. Sharp, P. McDermott, Workflow Modeling: Tools for Process Improvement and Application Development, Artech House Publishers, 2001.

[75] N. Sidorova, C. Stahl, N. Trcka, Soundness Verification for Conceptual Workflow Nets with Data: Early Detection of Errors with the Most Precision Possible, Information Systems 36 (2011) 1026–1043.

[76] B. Silver, BPMN Method and Style, with BPMN Implementer's Guide, 2nd ed., Cody-Cassidy Press, 2011.

[77] A. Sinha, A. Paradkar, Use Cases to Process Specifications in Business Process Modeling Notation, in: 2010 IEEE International Conference on Web Services, IEEE, pp. 473–480.

[78] M. Strembeck, J. Mendling, Modeling Process-related RBAC Models with Extended UML Activity Models, Information & Software Technology 53 (2011) 456–483.

[79] S. Sun, J. Zhao, J. Nunamaker, O. Liu Sheng, Formulating the Data-Flow Perspective for Business Process Management, Information Systems Research 17 (2006) 374–391.

[80] K. Toutanova, C.D. Manning, Enriching the knowledge sources used in a maximum entropy part-of-speech tagger, in: Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, pp. 63–70.

[81] F.S. Tseng, A.L. Chen, W.P. Yang, On mapping natural language constructs into relational algebra through e-r representation, Data and Knowledge Engineering 9 (1992) 97 – 118.

[82] F.S. Tseng, C.L. Chen, Extending the UML Concepts to Transform Natural Language Queries with Fuzzy Semantics into SQL, Information and Software Technology 48 (2006) 901 – 914.

[83] W.M.P. van der Aalst, Business Process Management, Business Process Management, volume 1806 of *LNCS*, Springer, 2000, pp. 161–183.

[84] H. Verbeek, T. Basten, W.M.P. van der Aalst, Diagnosing Workow Processes using Woflan, The Computer Journal 44 (2001) 246–279.

[85] B. van der Vos, J.A. Gulla, R. van de Riet, Verification of Conceptual Models based on Linguistic Knowledge, Data and Knowledge Engineering 21 (1997) 147 – 163.

[86] B. Weber, M. Reichert, J. Mendling, H.A. Reijers, Refactoring Large Process Model Repositories, Computers in Industry 62 (2011) 467–486.

[87] I. Weber, J. Hoffmann, J. Mendling, Beyond Soundness: on the Verification of Semantic Business Process Models, Distributed and Parallel Databases 27 (2010) 271–343.

[88] R. Wieringa, Designing Technical Action Research and Generalizing from Real-world Cases, in: Advanced Information Systems Engineering, volume 7328 of *Lecture Notes in Computer Science*, Springer Verlag, London, UK, 2012, pp. 697–698.

[89] R. Wieringa, A. Morali, Technical Action Research as a Validation Method in Information Systems Design Science, in: Design Science Research in Information Systems. Advances in Theory and Practice, volume 7286 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2012, pp. 220–238.

[90] M. Wynn, H. Verbeek, W.M.P. van der Aalst, A. Hofstede, D. Edmond, Soundness-preserving Reduction Rules for Reset Workflow Nets, Information Science 179(6) (2009) 769–790.