Information Systems 37 (2012) 443–459

Contents lists available at SciVerse ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

On the refactoring of activity labels in business process models

Henrik Leopold [a,*], Sergey Smirnov [b], Jan Mendling [c]

[a] *Humboldt-Universität zu Berlin, Germany*
[b] *Hasso Plattner Institute, Potsdam, Germany*
[c] *Wirtschaftsuniversität Wien, Vienna, Austria*

A R T I C L E I N F O

Article history:
Received 11 October 2011
Received in revised form
12 December 2011
Accepted 5 January 2012
Recommended by: M. Weske
Available online 14 January 2012

Keywords:
Activity label refactoring
Business process modeling
Model quality

A B S T R A C T

Large corporations increasingly utilize business process models for documenting and redesigning their operations. The extent of such modeling initiatives with several hundred models and dozens of often hardly trained modelers calls for automated quality assurance. While formal properties of control flow can easily be checked by existing tools, there is a notable gap for checking the quality of the textual content of models, in particular, its activity labels. In this paper, we address the problem of activity label quality in business process models. We designed a technique for the recognition of labeling styles, and the automatic refactoring of labels with quality issues. More specifically, we developed a parsing algorithm that is able to deal with the shortness of activity labels, which integrates natural language tools like WordNet and the Stanford Parser. Using three business process model collections from practice with differing labeling style distributions, we demonstrate the applicability of our technique. In comparison to a straightforward application of standard natural language tools, our technique provides much more stable results. As an outcome, the technique shifts the boundary of process model quality issues that can be checked automatically from syntactic to semantic aspects.

1. Introduction

Business process modeling is increasingly used in organizations for supporting documentation, organizational redesign, and information system development [1]. Typically, several expert modelers and a considerable number of casual modelers from diverse lines of business are involved in the creation of process models. Often, the outcome of such an initiative are several hundred process models. Maintenance and quality assurance of these process models is a critical challenge given this big number of models and modelers. Indeed, different quality issues have been observed: up to 20% of all models in collections from practice contain errors [2], and many casual modelers are not sufficiently trained [3].

There are several measures that organizations can take for addressing these problems including staff training, enforcement of modeling guidelines, and usage of formal analysis tools. Modeling expertise is an important factor for the success of process modeling projects [4]. It has also been shown that experienced modelers perform significantly better in process model comprehension experiments [5]. Training of modelers is therefore a suitable measure to assure model quality. Modeling guidelines are also helpful for assuring quality. The Guidelines of Process Modeling [6] and the Seven Process Modeling Guidelines [7] summarize essential modeling rules. They are typically extended and refined by organizations for their modeling projects. Finally, there are various formal analysis techniques that help to detect modeling errors. Properties like soundness can be used to check whether

* Corresponding author. Tel.: +49 30 2093 5805.
E-mail addresses: henrik.leopold@wiwi.hu-berlin.de (H. Leopold), sergey.smirnov@hpi.uni-potsdam.de (S. Smirnov), jan.mendling@wu.ac.at (J. Mendling).

0306-4379/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.is.2012.01.004

a process model contains deadlocks [8]. Several modeling tools provide soundness analysis by the click of a button.

Automatic analysis techniques such as checking soundness are very attractive for quality assurance. The results are precise and unambiguous, and automatic analysis tools can easily work on a large set of models in a short period of time. It is a problem that formal techniques so far only cover a small share of the properties that need to be checked. From the perspective of the SEQUAL quality framework [9], soundness relates to the syntactic layer of a model since it refers to its formal structure. Many guidelines though point to the textual content, e.g., in terms of a consistent design [6] or a verb–object style for constructing labels [7], which also covers semantical aspects. The importance of naming conventions [10,11] and the potential of using natural language processing techniques for checking them has been recently recognized [12,13]. The specific structure of activity labels in process models remains a challenge though. Due to the shortness of labels, natural language tools like the Stanford Parser [14] cannot be directly applied. As the labels are not full sentences, such parsers lack context to perform well.

This paper addresses the problem of analyzing textual activity labels in process models. Our contribution is an activity label refactoring technique that utilizes different aspects of context of a label for an accurate classification. We then utilize the parsing results for an automatic rework of the labels. This paper generalizes and extends the prior work on EPC-specific label parsing [12,15]. We provide a thorough evaluation of our technique based on a prototypical implementation and three process model collections from practice containing more than 10,000 labels. The results show that our technique achieves high precision and recall in parsing labels, and performs substantially better than an unspecific application of the Stanford Parser.

The remainder of this paper is structured as follows. Section 2 discusses labeling styles of process models and parsing challenges. Section 3 defines our three-step approach to label refactoring based on labeling style recognition, action and business object derivation, and label composition. Section 4 presents the results of an empirical evaluation studying three process model collections from practice. Section 5 discusses our contribution in the light of related work. Section 6 summarizes the paper and concludes with an outlook on the future research.

2. Background

In this section we discuss the background of our research. Section 2.1 introduces different perspectives on business process model quality, including the quality of text labels. Section 2.2 defines different styles of activity labeling in process models. Finally, Section 2.3 discusses challenges for an automatic refactoring of activity labels.

2.1. Perspectives on process model quality

Business process modeling is often conducted in companies that have reached a certain level of size and complexity. Business process models help documenting and analyzing the division of labour as well as the interactions and handovers between actors working in a process. Against this background, business process modeling is challenged by several factors. First, the companies have to maintain collections of several hundred process models. Second, the modeling initiative in such companies typically involve dozens of modelers with many of them not being modeling experts. The complexity of this setting demands dedicated measures to assure that process models of high quality are created. The high number of models motivates an extensive automation of the quality assurance.

Automatic analysis of process models typically focuses on the process behavior. Fig. 1 shows the example of a simple business process modeled in BPMN. The process starts with the activity *Make decision* and proceeds with the XOR-split gateway. This diamond-shaped element defines a decision point such that control is either forwarded to the upper or the lower branch based on a condition resolved at runtime. Subsequently, either *Alternative 1 Execution* or *Executing alternative 2* are conducted, but never both. Control is then passed to the AND-join gateway. This diamond-shaped element with the plus sign waits for both branches to complete. Only afterwards *Synchronization of both completed branches* can be executed. If a formal analysis of this model is conducted, for instance a soundness check [8], it becomes clear that the process can never be executed from the start to the end: the AND-join requires both previous branches to terminate while the XOR-split takes care that only one of them becomes activated. This means that the model contains a deadlock. The modeler should be informed about this problem, and would best replace the AND-join with an XOR-join gateway.

Correctness is critical for the specification of a process [16]. Quality issues like the deadlock in Fig. 1 can be efficiently found for different classes of process models. Languages, like BPMN, EPCs, or UML Activity Diagrams, that can be mapped to free-choice Petri nets can be checked for soundness in quadratic time [17], structured models in linear time [18]. There are also techniques available for checking the correctness of data flow [19–21], satisfiability of constraints on the resource perspective [22–24], or the interoperability of cross-organizational workflows [25]. These correctness aspects are well understood and efficiently supported by tools for a wide range of process model classes.

A problem is that verification tools do not address the full spectrum of process model quality aspects. This holds

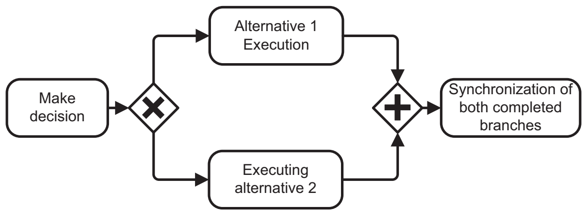


Fig. 1. Example of a business process model in BPMN.

most notably for the textual content of process models. The modeling guidelines recommend activity labels of a verb–object style [26,7]. Labels following this style start with the action referenced by a verb in imperative form followed by the corresponding business object which is involved in the activity. Empirical research witnessed that verb–object labels are superior for comprehension than labels of other styles [11]. If we consider Fig. 1 again, we see that only the first activity *Make decision* is compliant with the verb–object style, while other labels define the action using a noun. Labels like *Alternative 1 Execution* can be reformulated as verb–object labels with similar meaning (*Execute alternative* 1 in this case). A problem is though that process models in practice include on average 20 activities [27]. For a process model collection with several hundred models, this means that thousands of activity labels have to be inspected and potentially corrected to achieve verb–object style compliance. An automatic technique for refactoring of labels would help to conduct quality assurance much more efficiently.

2.2. Labeling styles

The design of effective and precise algorithms for labeling style refactoring requires a thorough understanding of the current labeling practices. In particular, it is essential to understand what label structures we may encounter and how these structures can be reliably recognized. Prior works list labeling styles we consider in our approach [11–13]. Altogether, there are seven labeling styles summarized in Table 1: verb–object, three types of action-noun, a descriptive style, an irregular category, and a no-action class.

The labels of *verb–object style* contain an action that is followed by a business object. Examples of these labels are *verb phrases* like *Create invoice* and *Validate order*. In the first case the action is *create* and the business object is *invoice*, while in the second example the action is *validate* and the business object is *order*. Notice that a business object may be absent such as in the labels *Analyze* or *Notify*. As these labels are also verb phrases, we relate them to the same style. A special case is given by verb phrases which contain a prepositional phrase, e.g., *Create order for received request*. A prepositional phrase in such labels brings additional information to the reader; it is optional.

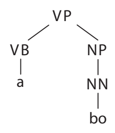
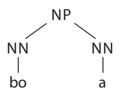
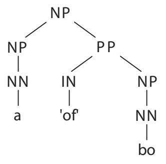
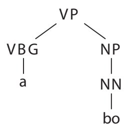
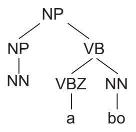
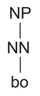
The labels of *action-noun style* (*np*) start with a business object followed by an action. Examples of labels adhering to this style are noun phrases like *Vendor evaluation* and *Schedule approval*. In the first case the action is *evaluate* and the business object is *vendor*, while in the second example the action is *approve* and the business object is *schedule*. Notice that a business object may be absent as in the labels *Analysis* or *Notification*. As these labels are also noun phrases, we relate them to the same style. Another special case is given by noun phrases which contain a prepositional phrase, e.g., *Revenue planning in work breakdown structure*. A prepositional phrase in such labels brings additional information to the reader; it is optional.

The labels of *action-noun style* (*of*) are a specific kind of a noun phrase with an *of* prepositional phrase. However, the action is represented by a noun which comes first and is succeeded by a prepositional phrase. The prepositional phrase starts with a preposition *of* and refers to a business object. Examples are *Creation of specification* and *Settlement of order*. For the two given examples the actions are *create* and *settle*, respectively, and business objects are *specification* and *order*. Similar to the labels of the previous labeling style, the labels of noun phrase with *of* can have an optional prepositional phrase, e.g., *Creation of specification for budget planning*.

The labels of *action-noun style* (*ing*) are indicated by a verb in *-ing* form. This gerund is succeeded by the business object captured as a noun. The following labels are representatives of this class: *Creating version* and *Processing requisition for projects*. For the first label the action is *create* and the business object is *version* while in the second example the action is *process* and the business object is *requisition*. Notice that the label of this style may have an optional prepositional phrase (e.g., as *for projects* in *Processing requisition for projects*).

The majority of labels adheres to the action-noun styles described above [13]. However, a small share of the labels that are noun phrases belongs to none of them. We assign such labels to *action-noun style (irregular)*. These labels have an anomalous structure. For instance,

Table 1
Activity labeling styles.

Labeling style	**Structure**	**Example**
Verb-object VO		Create invoice
Action-noun AN (np)		Invoice creation
Action-noun AN (of)		Creation of invoice
Action-noun AN (ing)		Creating invoice
Action-noun AN (irregular)	⟨*anomalous*⟩	LIFO: valuation: pool level
Descriptive DES		Clerk answers call
No-action NA		Error

the majority of these labels contains punctuation symbols such as ':' or ';' connecting the parts of the label in a special manner. As examples consider *Profit Center Assessment: Plan* or *LIFO: Valuation: Pool Level*.

The labels of *descriptive style* describe activities from a third person perspective. Most of these labels start with the name of the resource executing the activity. The resource is followed by the action in the third person form and the business object. Examples of descriptive labels are *Accounting creates invoice* and *Applies for registration*. While the first label example contains the actor *Accounting*, the second label directly starts with the action succeeded by the business object. Notice that a prepositional phrase may follow after the business object.

Although activity labels should point to an action to provide transparent instructions to the model reader, there are activity labels which do not refer to any action. We assign these labels to the *no-action style*. Examples of no-action labels are nouns without any reference to an action such as *Error*. Further examples of no-action labels from an industrial model collection analyzed by Mendling et al. [11] are *Information System* and *Data Basis*, both obviously including no reference to a particular activity.

2.3. Limitations of natural language processing methods for activity label analysis

Natural language text fragments in activity labels demonstrate two properties significant in the context of label analysis. First, activity labels are succinct and typically do not exhibit a complete sentence structure. In this way, they expose a limited amount of information about their grammatical structure. Second, activity labels tend to suffer from the zero derivation phenomenon of verbs [28]. While some verbs in the English language are formed from nouns by adding suffixes, e.g., *-ize* or *(i)fy*, verbs of zero derivation belong to a special kind of homonyms. As such, zero derivation verbs are syntactically identical to the corresponding noun. Zero derivation is a particular challenge in the context of business process models, since a lot of nouns referencing business objects are homonyms of zero-derivation verbs. Examples are *an order* (noun) and *to order* (verb) or *a process* (noun) and *to process* (verb).

The two aforementioned properties impede activity label analysis. In particular, they limit the application of ideas developed for analysis of natural language text. For instance, Leopold et al. [29] argue that the Stanford Parser has to be adapted for activity label analysis in order to be able to deal with short labels and zero derivation. Therefore, the activity label refactoring proposed in this paper builds on a specific approach to label parsing which makes use of concepts from natural language processing (NLP). NLP is a research field that studies how software systems can analyze, understand, and produce the human language [30]. NLP is a highly interdisciplinary research area as its foundations lie in disciplines such as computer science, linguistics, logic, and psychology [31]. The application fields of NLP are diverse and can be found in natural language text processing [32,33], machine translation [34,35], and speech recognition [36,37]. Based on NLP concepts, we design a technique for working with the specifics of short activity labels. We also integrate two existing NLP tools: The Stanford Parser and WordNet.

The Stanford Parser is a probabilistic natural language parser recognizing the grammatical structure of sentences. To analyze texts in a natural language, the probabilistic parser requires a prior initialization with a statistical model of this natural language. The Stanford Parser works with a pre-tagged version of the Penn Treebank corpus [38]. This text collection includes a set of Wall Street Journal articles tagged with grammatical information by humans. Once initialized, the Parser is capable of discovering groups of related words within a sentence or identification of a grammatical role a word plays in a sentence [14,39]. The straightforward application of the Stanford Parser for discovery of activity labels is interfered by the shortness of activity labels. However, we use the Stanford Parser for analysis of descriptive style labels, which are often full sentences.

The second tool we incorporate into our approach is WordNet. WordNet is a lexical database for the English language developed at the Princeton University [40]. WordNet organizes nouns, verbs, adjectives, and adverbs into logical groups called *synsets*. Each synset is a set of synonymous words or collocations (a combination of words with a specific meaning such as *fast food* or *think tank*). As one word may have various meanings, each synset comprises only words or collocations having the same meaning in a particular context, such that they are interchangeable. Furthermore, WordNet specifies several lexical and semantical relations for words, e.g., hyponymy, meronymy, and holonymy. Using these relations, it is for instance possible to discover a verb for its corresponding noun. Our technique for activity label refactoring uses WordNet in several ways. First, we use it as a dictionary to check if a word can exist in a particular part of speech. Second, WordNet facilitates stemming. Finally, we use the relations between verbs and nouns to express an action captured with a noun by means of a corresponding verb.

3. Activity label refactoring

This section discusses our automatic approach to activity label refactoring. The different algorithms of this approach are designed to work accurately in terms of precision and recall. This means that the percentage of misclassified labels needs to be small, since otherwise the refactoring operation produces damage. Our approach works in four general phases:

1. Recognition of activity labeling style.
2. Analysis of action-noun labels.
3. Derivation of an action and a business object from activity label.
4. Composition of a verb–object activity label.

In the remainder of this section, we elaborate on these phases step by step. Among the four phases, the initial activity labeling style recognition is the most critical, as a

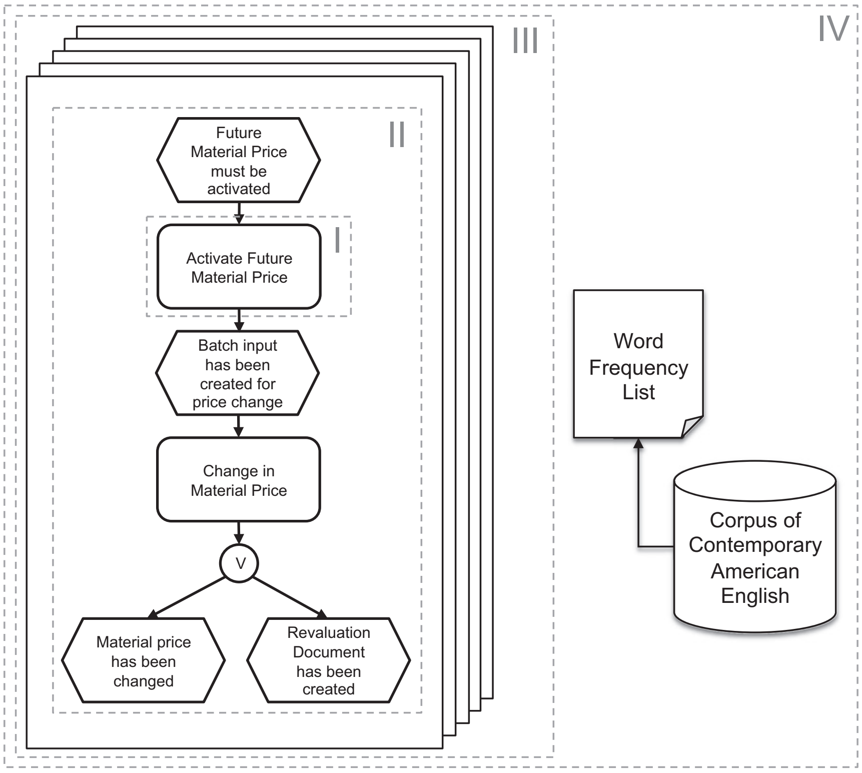

IV
III
II
I
Future Material Price must be activated
Activate Future Material Price
Batch input has been created for price change
Change in Material Price
V
Material price has been changed
Revaluation Document has been created
Word Frequency List
Corpus of Contemporary American English

Fig. 2. Four levels of context used within activity labeling style recognition.

wrong classification of a label leads to incorrect action and object derivation and a wrong verb–object label. Accordingly, style recognition is discussed in detail. It is subdivided into four subordinate stages.

3.1. Recognition of activity labeling style

The initial phase of activity labeling style recognition aims to assign a label to its correct labeling style. Because of the challenges of short labels and zero-derivation ambiguities, we design a specific approach that is tailored to the requirements of process models.

The key feature of the labeling style recognition algorithm is the analysis of label context information. We structure the context into four levels: (1) activity label itself, (2) process model containing the activity, (3) process model collection, and (4) knowledge on word frequencies (see Fig. 2). These levels are organized in a sequence from the most local context towards most generic. The algorithm tries to classify an activity label starting from the most local context, i.e., the label, and broadening the context, once the previous level is insufficient. If required it takes more generic, yet increasingly uncertain information into account. Against this background, the labeling style recognition is organized into four subordinate stages:

- *Stage*1: Label analysis.
- *Stage*2: Model analysis.
- *Stage*3: Model collection analysis.
- *Stage*4: Natural language analysis.

In the remainder of this section we elaborate on each stage.

3.1.1. Stage 1: label analysis

Algorithm 1. Activity labeling style recognition, label analysis.

```
1:  recognizeUsingLabel(Set modelActivityLabels)
2:  Set unrecognizedLabels = ∅;
3:  global Set verbs = ∅;
4:  global Set nouns = ∅;
5:  global Set $labels_{vo}$ = ∅;
6:  global Set $labels_{an}$ = ∅;
7:  global Set $labels_{des}$ = ∅;
8:  for all label ∈ modelActivityLabels do
9:    if getStyleByRule(label)! =UNCLASSIFIED then
10:     if getStyleByRule(label)= =VERB-OBJECT then
11:       $labels_{vo}$=$labels_{vo}$ ∪ {label};
12:     if getStyleByRule(label)= =ACTION-NOUN then
13:       $labels_{an}$ = $labels_{an}$ ∪ {label};
14:     if getStyleByRule(label)= =DESCRIPTIVE then
15:       $labels_{des}$ = $labels_{des}$ ∪ {label};
16:   else if isVerb(label.words[1]) then
17:     if isInfinitive(label.words[1]) then
18:       Set potentialVerbs=getOtherPotentialVerbs(label);
19:       if potentialVerbs= = ∅ then
20:         verbs = verbs ∪ {label.words[1]};
21:         $labels_{vo}$=$labels_{vo}$ ∪ {label};
22:       else
23:         unrecognizedLabels = unrecognizedLabels ∪ {label};
24:       else
```

```
25:         $nouns = nouns \cup \{label.words[1]\}$;
26:         $labels_{an} = labels_{an} \cup \{label\}$;
27:     else
28:       $nouns = nouns \cup \{label.words[1]\}$;
29:       $labels_{an} = labels_{an} \cup \{label\}$;
30: recognizeUsingModel(unrecognizedLabels)
```

This step tries to assign each activity label either to the verb–object style, any of the action-noun styles, or the descriptive style. The algorithm's input is a set of activity labels of one process model *modelActivityLabels*. The core idea of the algorithm is to reject a hypothesis that a label is of verb–object style. Furthermore, the global set of verbs and nouns is populated for later usage. Algorithm 1 illustrates the steps of the first stage.

Algorithm 1 starts by applying a set of structural rules against the considered label which doubtlessly indicates the used labeling style (line 9). We included the following rules as a part of the *getStyleByRule* function to predetermine the labeling style:

- *Of-rule*: The usage of the preposition *of* in the second position of the label requires the first word to be a noun representing the action. In case a phrasal verb is used, the *of* can consequently be found in the third position of the label. As examples consider the labels *Approval of concept* or *Notification of Customer* which can be, based on this rule, directly assigned to the action-noun style.
- *If-rule*: The usage of *if* at the second or, in case of a phrasal verb, at the third position of the label implies an imperative action as the first word. The algorithm classifies such labels, as, for instance, *Check if customer is satisfied*, as verb–object labels.
- *Condition rule*: Some labels provide extensive information by using a combination of a main and a subordinate clause. This is, for instance, the case if a condition for the instructed action is stated. Such conditions are frequently introduced by fragments as *in case of*, *in the event of* or *if*. These subordinate clauses are then succeeded by a main clause starting with an imperative verb. Hence, labels as *In case of continuous delay escalate* are directly assigned to the verb–object style.
- *Descriptive rule*: Stage 1 also uses rules to detect descriptive labels. As descriptive labels are fragments of natural language text, we analyze them by means of the Stanford Parser. In this way, labels without an actor, as *writes down annual fee*, and also labels with actor, as *Seller processes order*, can be allocated to the descriptive style. Once the Parser recognizes the first possible verb in the label as a third person form of a verb, the label is categorized accordingly.

If no structural rule applies to the label, Algorithm 1 proceeds by considering potential verbs at different positions. First, it checks whether the first word of the label is a potential verb (line 16). If this is the case, we investigate whether the first word also equals the respective imperative of the verb it may represent. As the imperative in English always equals the infinitive, we implement this check using WordNet. If this assumption is confirmed, we consider a case of zero-derivation ambiguity and proceed with further analysis steps. To reliably decide about the zero-derivation labels, the algorithm detects other potential verbs in the label (line 18). As an example consider the label *Order System*. The word *order* suffers from zero-derivation ambiguity and without examining the rest of the label, an automatic decision is impossible. However, if we assume facing an action-noun label, we are able to figure out that *system* cannot be an action. Thereafter, the only potential verb in this label is given by *order*. In this case the label is assigned to the verb–object style (lines 20–21). Nevertheless, if multiple potential verbs are identified, the classification decision is handed over to the second stage of labeling style analysis: we add the label to the set of unrecognized labels (line 23).

If the first word of a label is not an infinitive, the label is assigned to the action-noun style and the precise style is determined later on (line 26). Similarly, the label is classified as action-noun once its first word is not a verb (line 29). After every label in the set was processed, unrecognized labels are passed as the input to the Algorithm 2 (line 30).

3.1.2. *Stage 2: model analysis*

Algorithm 2. Activity labeling style recognition, model analysis.

```
1:  recognizeUsingModel(Set labels)
2:  Set $unrecognizedLabels = \emptyset$;
3:  for all $label \in labels$ do
4:    $assigned =$ false;
5:    Set $processModelLabels =$
      $label.getProcessModel().getActivityLabels()$;
6:    for all $pmLabel \in processModelLabels$ do
7:      if $pmLabel.words[1] == label.words[1]$ and $pmLabel \in$
        $VOLabels$ then
8:        $verbs = verbs \cup \{label.words[1]\}$;
9:        $labels_{vo} = labels_{vo} \cup \{label\}$;
10:       $assigned =$ true;
11:     if $pmLabel.words[1] == label.words[1]$ and $pmLabel \in$
        $ANLabels$ then
12:       $nouns = nouns \cup \{label.words[1]\}$;
13:       $labels_{an} = labels_{an} \cup \{label\}$;
14:       $assigned =$ true;
15:   if !assigned then
16:     $unrecognizedLabels = unrecognizedLabels \cup \{label\}$;
17: recognizeUsingModelCollection(unrecognizedLabels)
```

Algorithm 2 implements the second stage of activity labeling style recognition. It takes information on label styles in the whole model into account. Its input is the set of labels that have not been recognized in Stage 1. To classify a label, the algorithm inspects the process model in which the activity label is observed. In particular, for each label to be classified the algorithm checks its first word. All the labels of the activities in this model that start with the same word are investigated. If the process model includes such an activity which was already assigned to a style, the current label is allocated accordingly (lines 8–10, 12–14). If the label was not assigned using this strategy it is added to the newly created set of unrecognized labels. After every label in the set was processed, the Stage 3 of recognition is triggered (line 17).

3.1.3. Stage 3: model collection analysis

Algorithm 3. Activity labeling style recognition, model collection analysis.

```
1:  **recognizeUsingModelCollection(Set** $labels$**)**
2:  **Set** $unrecognizedLabels = \emptyset$;
3:  **for all** $label \in labels$ **do**
4:     **if** $label.words[1] \in verbs$ **and** $label.words[1] \notin nouns$ **then**
5:        $labels_{vo} = labels_{vo} \cup \{label\}$;
6:     **else if** $label.words[1] \in nouns$ **and** $label.words[1] \notin verbs$ **then**
7:        $labels_{an} = labels_{an} \cup \{label\}$;
8:     **else**
9:        $unrecognizedLabels = unrecognizedLabels \cup \{label\}$;
10: **recognizeUsingLanguage**($unrecognizedLabels$)
```

The Stage 3 is described by Algorithm 3. The input of the algorithm is the set of labels unrecognized in Stage 2. Algorithm 3 broadens the context from the model to the process model collection level. Technically, this context is captured by sets of *verbs* and *nouns*, created in the preceding stages. These two sets contain the first words of verb–object and action-noun labels of the process model collection. Inspecting the sets of *verbs* and *nouns* the algorithm checks if the first word of an unrecognized label has been considered a verb or a noun earlier. If the word is exclusively contained in one of these sets, the label is assigned to the respective labeling style (lines 5 and 7). Labels that are not classified within the first three algorithm stages are handled by Stage 4 (line 10).

3.1.4. Stage 4: natural language analysis

Algorithm 4. Activity labeling style recognition, natural language analysis.

```
1:  **recognizeUsingLanguage**(**Set** $labels$)
2:  **for all** $label \in labels$ **do**
3:     **int** $verbFrequency$ = **getVerbTagsInCorpus**($label.words[1]$)
4:     **int** $nounFrequency$ = **getNounTagsInCorpus**($label.words[1]$)
5:     **if** $verbFrequency \geq nounFrequency$ **then**
6:        $labels_{vo} = labels_{vo} \cup \{label\}$;
7:     **else**
8:        $labels_{an} = labels_{an} \cup \{label\}$;
```

Algorithm 4 finalizes labeling style recognition. In order to decide upon the labeling style, it is necessary to disambiguate the first word and decide whether it represents an action. In Stage 4, we exploit properties of the English language. In this language a word may belong to several parts of speech. However, analysis of large natural language text collections allows us to learn frequencies for each word. Therefore, we have considered a frequency list derived from the Corpus of Contemporary American English [41]. This list contains 500,000 words with their parts of speech and the respective frequencies in the corpus. Having this information at hand we learn for each word its part of speech and its frequency. If the verb frequency is higher or equal to the noun usage count, the label is allocated to the verb–object style. If it is lower, the label is consequently assigned to the action-noun style. In this way every remaining label is assigned to a labeling style. As an example consider the label *Credit Check* in which both words can be used as verbs as well as nouns. In order to decide about the labeling style we access the frequency list and consider the part of speech related occurrences for the word *credit*. As a result, we receive 7169 occurrences for *credit* as a verb and 36,784 for *credit* as a noun. Consequently, we assume *credit* to be a noun and assign the label *Credit Check* to the action-noun style.

3.2. Analysis of action-noun labels

Once the activity labels are separated into verb–object and action-noun labels, it is necessary to determine the exact style of the latter group. Based on this classification, we directly infer the position of verbs and business objects in the label. For determining the labeling style, there is a set of rules to be considered:

- If the label contains irregular characters, the style is set accordingly.
- If conjunctions or prepositions are found, we store the position of their first occurrence.
- If the label starts with a gerund, we check whether it really represents an action. Consider the label *Planning scenario processing* as discussed in Leopold et al. [13]. *Planning* is a gerund, but it can also be a part of a business object. For resolving this ambiguity, we analyze surrounding events or activities preceding and succeeding the considered activity with the inspected label. We might find that the activity is connected to an event labeled with *Planning scenario is processed*. A part of speech analysis identifies *planning* and *scenario* as nouns and *process* as a verb. Therefore, we assume that *processing* defines the action. The label is classified as *action-noun* $AN(ing)$.
- If the label contains prepositions and the first one is an *of*, the label is qualified as *action-noun* $AN(of)$.
- Otherwise, the label is assumed to be *action-noun* $AN(np)$.

3.3. Derivation of action and object

At this stage, all the information is available for deriving the action and the business object from the label. Algorithm 5 defines a derivation method for *action-noun* (np), *action-noun* (of), and *action-noun* (ing) styles and descriptive labels. These are the labels that can be systematically refactored. Labels of irregular style are not addressed. Algorithm 5 also does not address labels that contain coordinating conjunctions. The input of the algorithm is an action-noun label *label* and a corresponding *LabelProperties* object *prop* storing the classification properties of the prior steps. The output of the algorithm is the set *prop* with *action* and *bObject* properties set.

Algorithm 5. Derivation of an action and a business object from a label.

```
1:  deriveActionAndBusinessObject(Label label, LabelProperties prop)
2:  if !prop.hasConjunctions then
3:    $size = label.words.size$;
4:    if $prop.style ==$ NOUN then
5:      if prop.hasPrepositions then
6:        $label = label.words[1] + \cdots + label.words[prop.pIndex-1]$;
7:      if $size == 1$ then
8:        $action = label.words[1]$;
9:      else
10:       if $label.words[size-1] + label.words[size]$ is a phrasal verb then
11:         $prop.action = label.words[size-1] + label.words[size]$;
12:         if $size > 2$ then
13:           $prop.bObject = label.words[1] + \cdots + label.words[size-2]$;
14:       else
15:         $prop.action = label.words[size]$;
16:         $prop.bObject = label.words[1] + \cdots + label.words[size-1]$;
17:   else if $prop.style ==$ PREPOSITION_OF then
18:     $prop.action = label.words[1] + \cdots + label.words[pIndex-1]$;
19:     $pPhrase = label.words[pIndex+1] + \cdots + label.words[size]$;
20:     if pPhrase contains prepositions then
21:       $nIndex$ = index of the next preposition after $pIndex$;
22:       $prop.bObject = label.words[pIndex+1] + \cdots + label.words[nIndex-1]$;
23:     else
24:       $prop.bObject = label.words[pIndex+1] + \cdots + label.words[size]$;
25:   else if $prop.style ==$ GERUND then
26:     if prop.hasPrepositions then
27:       $label = label.words[1] + \cdots + label.words[prop.pIndex-1]$;
28:     if $size == 1$ then
29:       $action = label.words[1]$;
30:     else
31:       if $label.words[1] + label.words[2]$ is a phrasal verb then
32:         $prop.action = label.words[1] + label.words[2]$;
33:         if $size > 2$ then
34:           $prop.bObject = label.words[3] + \cdots + label.words[size]$;
35:       else
36:         $prop.action = label.words[1]$;
37:         $prop.bObject = label.words[2] + \cdots + label.words[size]$;
38:   else if $prop.style ==$ DESCRIPTIVE then
39:     for $i = 1 \rightarrow size$ do
40:       if $label.words[i].getTag() ==$ VERB then
41:         $prop.action = label.words[i]$;
42:         break;
43:     $prop.bObject = label.words[i+1] + \cdots + label.words[size]$;
44: return prop;
```

The algorithm starts with an analysis of labels following noun phrase style (lines 4–16). It checks for an optional prepositional phrase. If the label has a prepositional phrase, the phrase is omitted and not studied any more. If the label has only one word, e.g., *Deployment* or *Classification*, this word is recognized as an action. Otherwise, the algorithm checks if the last two words of the label constitute a phrasal verb, for instance, *set up* and *carry forward*. If the first two words are recognized as a phrasal verb, this verb is perceived as an action. The rest of the label, if it exists, is recognized as a business object (lines 11–13). If the phrasal verb is not revealed, the last word is recognized as an action while the rest as a business object (lines 15–16).

Algorithm 5 proceeds with the analysis of activity labels of *action-noun* (*of*) *style* (lines 17–24). The label part preceding preposition *of* is recognized as an action. The label part between preposition *of* and the next preposition is treated as a business object. Then, the analysis of *action-noun* (*ing*) labels follows (lines 25–37). Analysis of these labels resembles the analysis of labels of noun phrase style. The key difference is that the action is expected to appear in the beginning of the label while the business object in the end.

Finally, Algorithm 5 concludes with the analysis of descriptive labels (lines 38–43). In order to illustrate the subsequent steps, we consider the two descriptive labels *Buyer informs seller to cancel* and *Checks process model.* While the first label specifies the role *buyer* for performing the given task, the second label only states the task from a third person perspective. As already indicated by the two examples, the verb position in descriptive labels is not predefined. However, as descriptive labels were assigned to its style using the Stanford Tagger, the assigned tags can be used to identify the verb. Accordingly, the first step of the derivation is to identify the occurrence of the first verb tag in the label. Beginning with the first word, each tag of the label is considered (lines 39–42). If the current word carries a verb tag, it is saved as action to the *prop* record and the loop is terminated (lines 40–42). As the business object directly follows the action, the remaining words are saved as business object (line 43).

Algorithm 5 does not explicitly deal with irregular action-noun labels and labels containing conjunctions. Therefore, we provide an outlook on how such labels are analyzed. For irregular labels we first identify all potential actions in the label. Then we select the most likely action by investigating surrounding events or activities. For conjunction labels the first step is to identify, if the conjunction coordinates actions or business objects. This can be achieved using information about the labeling style and the position of the conjunction in the label. Afterwards, an algorithm similar to Algorithm 5 can be used for deriving actions and business objects from coordinated components of the label. Notice that a conjunction may appear in the optional prepositional phrase, as in *Creation of proposal for sales and profit planning*. In this case the conjunction is ignored, as it does neither coordinate actions nor business objects.

3.4. Verb–object label construction

Refactoring aims to transform an action-noun or a descriptive label into a verb–object label, which signifies the same action performed on the same business object. The derivation of actions and business objects from activity labels enables construction of labels in verb–object style. In fact, after the analysis of the previous steps the task becomes a straightforward concatenation of a verb representing an action and a noun phrase representing a business object. Notice that the optional prepositional phrase derived from the label at the derivation stage can be preserved in the verb–object label.

To achieve this, the prepositional phrase is concatenated to the label after the business object.

As it follows from the title, action-noun labels capture actions with nouns. At the same time, a verb–object label expects a verb to represent the action. To learn which verb corresponds to the noun capturing the action, we make use of *nominalization*—a linguistic operation of producing a noun from another part of speech via the addition of derivational affixes. In the context of this task we are interested in nominalization relations between nouns and verbs. Technically, nominalization relations can be obtained from WordNet [42]. As an example, consider action-noun label *Invoice verification*. The action is given by *verification* and the business object by *invoice*. Using nominalization we can learn that the action signified by the label is *verify*. Concatenation of the verb *verify* and the business object *invoice* results in verb–object label *Verify invoice*.

Once this phase is completed, we have refactored labels in action-noun and descriptive style to verb–object labels.

4. Empirical evaluation

To demonstrate the capability of our approach for refactoring activity labels, we conduct an evaluation with real-world data. Section 4.1 gives an overview of the process model collections we utilize. The goal of the evaluation is to learn how well the proposed algorithms approximate a human interpretation and a manual refactoring of activity labels. Accordingly, we study different dimensions of the evaluation. Section 4.2 investigates our approach from a runtime performance perspective. Section 4.3 shows the results of the recognition phase. Section 4.4 compares these results to a simple adoption of the Stanford Parser. Section 4.5 assesses the overall improvement of the automatic refactoring. We also discuss cases of misclassification in this context.

4.1. Test collection demographics

In order to achieve a high external validity, we include process model collections which vary in multiple dimensions as, for instance, the domain, modeling language, and the distribution of labeling styles. We designed a test sample that includes three different real-world process model collections and the human interpretation for each activity label in it. Table 2 summarizes the main features of the considered process model collections. They include:

- *SAP reference model*: The SAP Reference Model is a model collection capturing the business processes supported by the SAP R/3 system in its version from the year 2000 [43, pp. 145–164]. The collection contains in total 604 Event-driven Process Chains (EPCs) organized in 29 functional branches of an enterprise such as sales and accounting. This collection contains mainly action-noun labels (81%).
- *TelCo collection*: The TelCo Collection contains a set of 388 ADONIS models from a large telecommunication service provider. With regard to contents, the models capture various aspects from the domain of customer service management. The major share of labels in this collection are following the verb–object style (81%).
- *Signavio collection*: The Signavio Collection consists of 518 process models created with the Business Process Model and Notation (BPMN). The models cover diverse domains and mainly stem from academic training. Most of the labels are in verb–object style (74%).

Table 2
Details about used model collections.

Property	SAP	TelCo	Signavio
Process models	604	388	518
AN labels	81%	8%	9%
VOS labels	11%	81%	74%
DES labels	0%	1%	6%
NA labels	8%	10%	11%
Activity labels	2433	4254	4097
Average no. of activities per model	4.03	10.96	7.91
Average no. words per label	3.50	3.83	3.66
Minimum no. of words per label	1	1	1
Maximum no. of words per label	12	16	15
Modeling language	EPC	ADONIS	BPMN

With respect to the differences between the model collections, we emphasize three dimensions of diversity: labeling style distribution, modeling language, and modeling experience.

The most significant feature of the considered collections is the opposed distribution of *labeling styles*. While the majority of the activity labels in the SAP Reference Model follow the action-noun style and only a small share belongs to the verb–object style, the labeling style distribution in the TelCo and Signavio Collection is the inverted. This fact is important, as a small share of action-noun labels requires the algorithm to work with a high precision in terms of style recognition. Otherwise, verb–object labels are mistakenly classified as action-noun and consequently flawed in the refactoring phase. To demonstrate the capability of our algorithm to cover both extremes in terms of style distribution, the selected process model collections can be considered to be well suited.

As the extensiveness of the information content varies among different *modeling languages*, we also chose process models which differ in this regard. EPCs contain, by definition, numerous events which can be used for inferring information or to validate assumptions about words and their parts of speech in activity labels. By contrast, the event information in BPMN and ADONIS models can be very sparse. To show that our approach does not rely on modeling language specific information, we cover three different modeling languages.

When process models are created in practice, it cannot be assumed that the involved modelers have extensive *modeling experience* or can be considered modeling experts. While the SAP Reference Model and the TelCo collection were created in a professional environment, the Signavio process model collection was mainly created by students as course assignments. In order to cover the aforementioned real-world scenario of heterogeneous modeling quality, we studied both professional and non-professional process model collections.

By mixing the characteristics of the included process model collections along these dimensions, we aim to increase the external validity of the conclusions drawn from our evaluation.

4.2. Performance results

Our refactoring technique is aimed to be part of a modeling tool either as a batch service or an online help indicating problems during the modeling process. Therefore, the computation time should be adequately fast. For this reason we investigate the response time for each model collection in total, and also for one single activity. We tested the execution on a MacBook Pro with a 2.26 GHz Intel Core Duo processor and 4 GB RAM, running on Mac OS X 10.6.7 and Java Virtual Machine 1.5. In order to exclude one-off setup times, we executed the algorithm twice and measured the second run only.

Table 3 summarizes the results for the three process model collections by showing the execution times for recognition, refactoring and total processing for each of the process model collections and for one single label on average. From the numbers we can learn that execution times for both recognition and refactoring depend upon the labeling style distribution of the analyzed process model collections. In particular, the results for the SAP Reference Model indicate that a great share of action-noun labels entails longer execution times. Considering our recognition algorithm, this is not surprising as the detection of action-noun styles and the refactoring of action-noun labels require additional computation steps. Nevertheless, we observe that the algorithm computes refactored labels rapidly. Even for the SAP Reference Model, which requires the most rework because of a great share of action-noun labels, the average execution time for one label is approximately 14 ms. Consequently, within 1 s a process model with about 71 activity labels can be refactored completely. Hence, we consider our algorithm to be well suited in terms of execution time for an online help during modeling.

4.3. Recognition results

We assess the recognition performance of our algorithm using precision, recall, and f-measure as metrics. In our context the precision value is the number of correctly recognized labels of a given style divided by the total number of labels retrieved by the algorithm. The recall is the number of correctly recognized labels of a given style divided by the total number of labels belonging to this style. As our goal is to obtain considerable recall and precision values at the same time, we also compute the f-measure, the harmonic mean of precision and recall [44].

In order to be able to assess the computed results, we created a benchmark using the human interpretations of the comprised activity labels. The human interpretation is captured by two mappings: one mapping from an activity label to a set of corresponding actions and another mapping from an activity label to a set of business objects. This information is stored in a spreadsheet, which can then be read by an application in the evaluation phase.

Within the evaluation we compared:

1. recognition of labeling styles by the algorithm and by humans;
2. derivation of actions and business objects by the algorithm and by humans.

Fig. 3 summarizes the results of all four phases for each collection. The results show that the recognition algorithm works satisfactory. After the last phase of the algorithm each f-measure is higher than 70%, while the average f-measure is above 80%. However, we observe considerable differences among the f-measures within a single and also among the different collections. For instance, for the SAP Reference Model, we observe a comparatively moderate verb–object recognition (70%) but a significantly high action-noun recognition (94%). By contrast, for the TelCo and the Signavio Collection the verb–object recognition shows the higher value.

If we include the style distribution of the collections into our consideration, the details of the results become more transparent. For instance, the f-measure for the verb–object recognition in the SAP Reference Model results from the small amount of verb–object labels in the collection. As only 220 labels belong to this style, an action-noun label misclassified to a verb–object label significantly affects the recognition precision of the verb–object style. For the TelCo and the Signavio collections it is the other way around, as a relatively small share of action-noun labels has to be recognized precisely.

These distribution imbalances asymmetrically impact the eventual refactoring. In case of a verb–object dominant style distribution as in the TelCo Collection, some of the action-noun labels are misclassified as verb–object

Table 3
Performance results.

	Period	**SAP**	**TelCo**	**Signavio**
Whole collection	Recognition (ms)	26,209	15,237	26,214
	Refactoring (ms)	7989	4596	6205
	Total (ms)	34,198	19,833	65,477
Label (**average**)	Recognition (ms)	10.77	3.58	6.40
	Refactoring (ms)	3.28	1.08	1.51
	Total (ms)	14.06	4.66	6.73

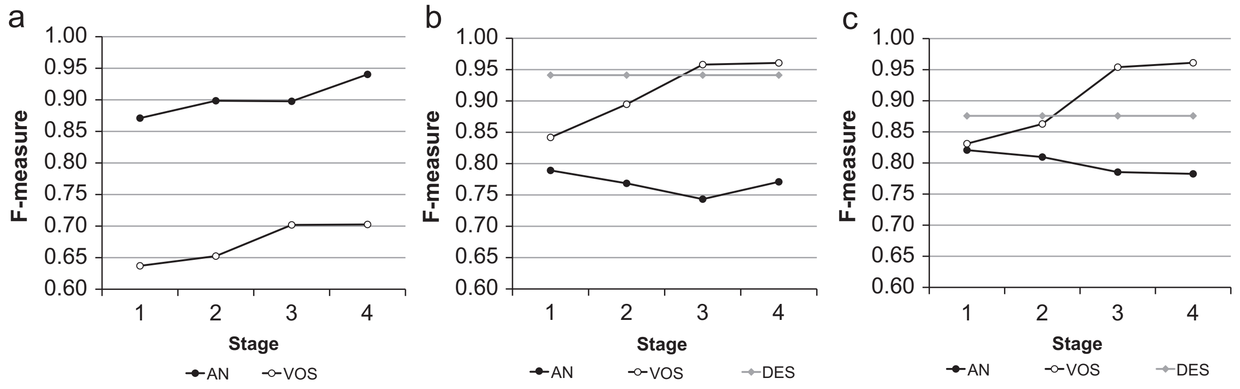


Fig. 3. Results of recognition phase. (a) SAP reference model. (b) TelCo collection. (c) Signavio collection.

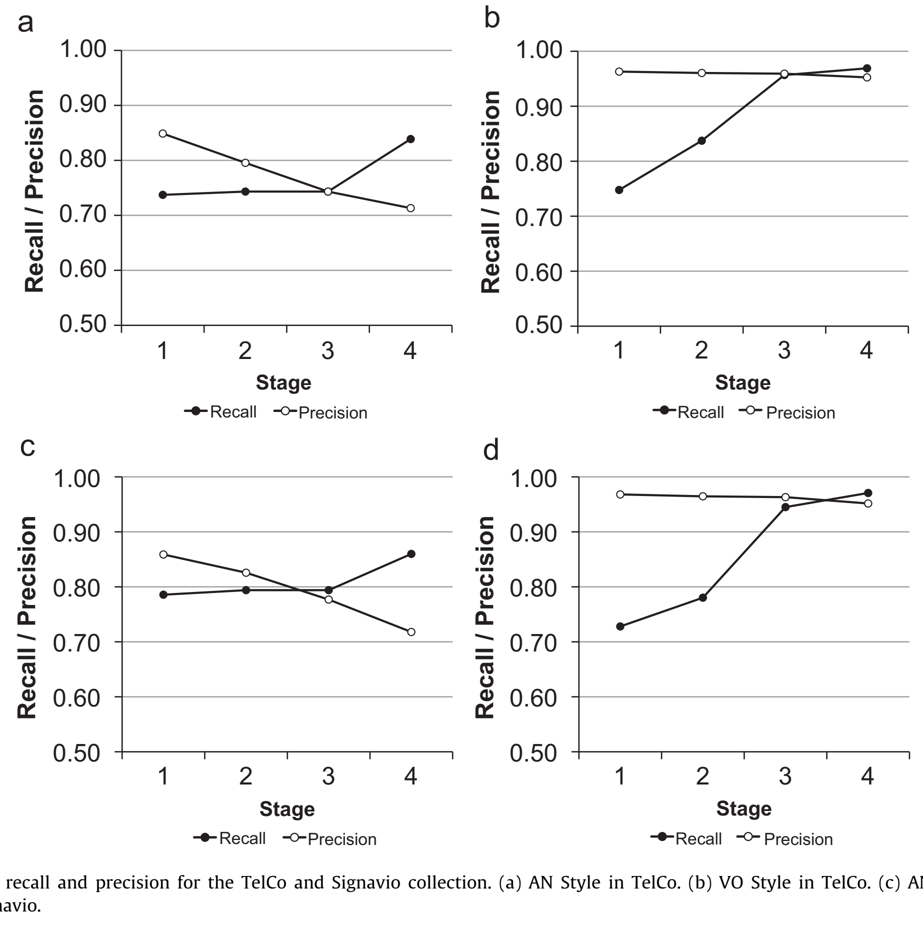


Fig. 4. Recognition recall and precision for the TelCo and Signavio collection. (a) AN Style in TelCo. (b) VO Style in TelCo. (c) AN Style in Signavio. (d) VO Style in Signavio.

labels because of the lower verb–object precision. As a consequence, they are not considered for refactoring. By contrast, if a verb–object label is misclassified, it becomes erroneously subject to refactoring. Hence, the precision for the action-noun recognition has a greater impact on the refactoring outcome. These precision values range from 85% to 95% in phase 1 and from 71% to 93% in phase 4 for the different collections in the different phases (not visualized separately).

Besides the final results, the diagrams also highlight the effect of the four phases. While the recognition of the descriptive style is purely based on the label structure and thus unchanged throughout the phases, the verb–object and action recognition depends upon the phase. However, the curve also indicates differences in this dependency. Whereas the f-measure for the verb–object recognition is strictly increasing in all the three model collections, the action-noun f-measure partially decreases, as for instance in phase 3 in the TelCo and Signavio Collection. This is mainly caused by the inverse development of recall and precision. To assign ambiguous labels to a certain style, increasingly insecure decisions are made. In this way, the recall can be increased while precision decreases. Fig. 4 illustrates this effect by presenting the disaggregated f-measures for the TelCo and Signavio collections. It reveals that the action-noun recall for the TelCo collection already started with a quite high value of 74% and a considerable precision of 85%. Due to the comparable labeling style distribution similar results can be observed for the Signavio collection which starts with an action-noun recall of 79% and a precision of 86%. Nevertheless, not all ambiguous labels are assigned to a style in this phase. For both collections the subsequent phases two and three show an increase of the verb–object recall as well as the significant increase in the action-noun recall in phase four, which coincide with lower precision values. Due to the small size of the action-noun label share, the change of the action-noun precision is higher than the verb–object precision. These results illustrate the trade-off between precision and recall. Nevertheless, the phase structure of our approach offers a flexible way to define preferences for precision or recall depending on the considered application. If a decision for all labels is required, all four phases have to be executed. If the usage scenario demands high precision, the algorithm may be limited to one of the first three phases.

4.4. Stanford Parser recognition results

The results presented so far demonstrate a good performance of the proposed technique. However, for assessing its merit in comparison to a naive application of natural language parsing techniques, we conducted the labeling style recognition experiment using the Stanford Parser. For this experiment we provided the Stanford Parser with all activity labels comprised in the process model collections. Based on its parsing result, we decided upon the style allocation of the label. If the parser recognized a third person verb, the label was allocated to the descriptive style. In any other case we only considered the tag of the first word. If the parser assigned a verb tag, we allocated the label to the verb–object style. If the parser assigned a noun tag, we consequently allocated the label to the action-noun style. Fig. 5 shows the results for the simple application of the Stanford Parser.

If the parser was able to correctly recognize the underlying label structures, we could expect a precise classification. However, the results reveal the opposite. This fact can be explained by the combination of the algorithmic approach of the Stanford Parser and the improper input of activity labels. The Stanford Parser determines the part of speech of a certain word based on the surrounding words, and computes the most likely part of speech using a pre-tagged set of natural language texts, for instance, the Penn Treebank—a tagged collection of the Wall Street Journal articles [38]. As many activity labels are rather short and action-noun labels do not represent proper sentences, we cannot expect that such text fragments can be disambiguated using the Penn Treebank.

Looking into the details, we observe high recall values for the descriptive and action-noun style recognition. Taking the approach of the Stanford Parser into consideration, this is a logical outcome. As descriptive style labels represent proper sentences, they can be adequately analyzed by the parser. The high recall for action-noun labels results from the tendency of the parser to assign noun tags even if the a verb tag would have been theoretically possible. Hence, the majority of the labels are assigned to the action-noun style and the recall value is, therefore, high. As most labels in the SAP Reference Model comply with the action-noun style, we observe a high corresponding precision. Nevertheless, the verb–object recall and each of the precision values among the other collections reveal that the sole application of the Stanford Parser would lead to inaccurate categorization results, with a significant number of labels being corrupted in the refactoring phase. We conclude that our labeling style recognition performs significantly better than the naive adoption of standard NLP tools, essentially because our approach is tailored to the specifics of short activity labels.

4.5. Refactoring results

To evaluate the quality of the final refactoring, we consider two aspects: first, that labels are correctly refactored, and second, that labels can also be erroneously refactored. As a baseline for this discussion, we consider the set of action-noun labels ($labels_{AN}$) and the set of descriptive

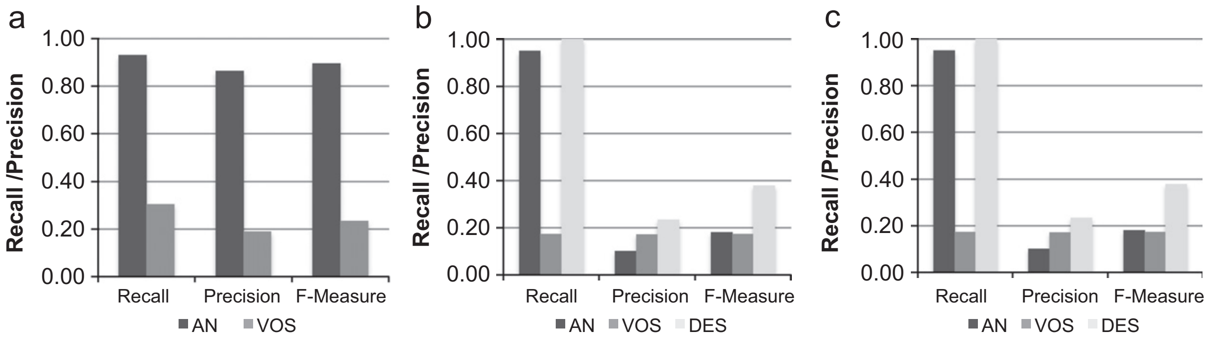


Fig. 5. Stanford Parser results for recognition phase. (a) SAP reference model. (b) TelCo collection. (c) Signavio collection.

labels ($labels_{DES}$). Together they define the set of all labels that *should be* refactored, i.e., $labels_{AN+DES} = labels_{AN} \cup labels_{DES}$. Furthermore, we consider the set of labels that *are actually* refactored. These are either correctly or erroneously refactored. We write $labels_r = labels_{r-correct} \cup labels_{r-erroneous}$. It follows that $labels_{r-correct} = labels_r \cap labels_{AN+DES}$ and $labels_{r-erroneous} = labels_r \backslash labels_{AN+DES}$. Based on these sets we introduce two metrics: refactoring gain and refactoring effect. While the refactoring gain captures only the relative share of correctly refactored labels, the refactoring effect takes into account that labels can be erroneously refactored. We consider it to be important that both metrics yield satisfying results. On the one hand, we aim to maximize the share of mediocre labels that are refactored (refactoring gain). On the other hand, we also want to minimize the damage caused by the erroneous refactoring of good labels (refactoring effect)

$$\text{Refactoring gain } RG = \frac{|labels_{r-correct}|}{|labels_{AN} \cup labels_{DES}|} \tag{1}$$

$$\text{Refactoring effect } RE = \frac{|labels_{r-correct}| - |labels_{r-erroneous}|}{|labels_r|} \tag{2}$$

The refactoring gain RG is the share of labels which has been correctly refactored in relation to the labels to be refactored, i.e., action-noun and descriptive labels. This metric ranges between 0 (no label is correctly refactored) and 1 (all required labels are correctly refactored). The refactoring effect RE takes into account the damage of misclassification. It ranges from -1 (all refactored labels are erroneous) and 1 (all labels have been correctly refactored).

Fig. 6 shows the results for both metrics. In Fig. 6(a) we can see that the *refactoring gain* ranges between 64% and 77% in the different phases for the three collections. The growth of the curves reflects a corresponding increase in the recall of action-noun labels. For all the three collections about 75% of the action-noun and descriptive labels have been properly refactored into verb–object labels. In order to assess the overall *refactoring effect*, we balance correct and erroneous refactorings.

Following on this analysis, we investigated the cases of misclassification. Altogether, we observe 14% erroneously refactored action-noun labels for the SAP Reference Model, 11% for the TelCo Collection and 15% for the Signavio Collection. The sources for this incorrect refactoring can be divided into three classes:

- *Compound words*: In some cases the algorithm does not recognize compound words and treats them as isolated words. For instance, consider the label *New User Registration*. In this label we find the compound noun *new user* and the action *register*. However, the algorithm has only limited contextual information available to determine whether *new* is an adjective of the business object *user* or an adjective characterizing the nominalized action *registration*, which should be transformed to an adverbial conjunct in the refactored label. Although this case is quite clear to humans, the algorithm determines a refactored label as *Register user newly* and not *Register new user*. By contrast, if we examine the label *Informal Sales Conversation* it becomes clear why this decision is non-trivial. Both labels have the same structure. Although the adjective relates to the business object in the first case, it is used to qualify the action in the latter case. Hence, the label does not require the process participant to *converse* about *informal sales*, but to *informally converse* about *sales*.
- *Adjective-noun ambiguity*: We have already discussed the problem of zero-derivation ambiguity between nouns and verbs. However, the ambiguity problem also arises between nouns and adjectives. This becomes critical if we try to identify adjectives which are specifying verbs, since the former must be transformed into adverbial conjuncts in the refactored label. For instance, in the label *Good Receipt* we apparently face the receipt of goods. However, for the algorithm it is not possible to disambiguate *good*. Hence, the refactoring of the label is not determined with *Receive goods* but with *Receive well* due to misinterpretation.
- *Irregular labels*: Although this problem only applies to a minor share, the investigated collections also contain labels following an arbitrary labeling style. For these labels such as *LIFO: Group Formation: Change* it is challenging to infer the action correctly and also to agree on how an adequate refactoring of these labels

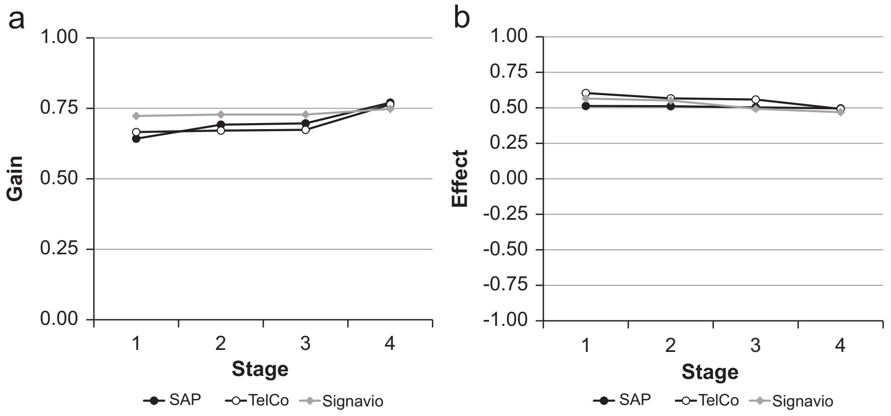


Fig. 6. Results of refactoring. (a) Refactoring gain. (b) Refactoring effect.

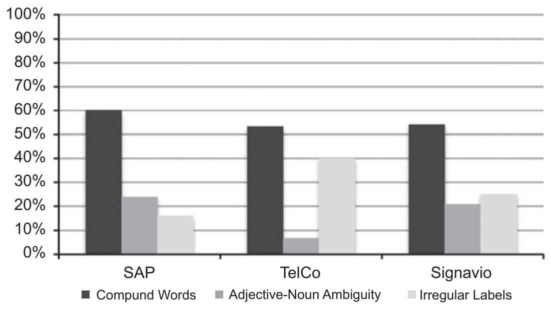


Fig. 7. Distribution of refactoring error sources.

should be defined. Therefore, many of the refactoring results of these irregular labels have been assessed to be improper, either because the action was not correctly inferred or because the structure of the refactored label was not satisfying.

Fig. 7 illustrates the distribution of the introduced error classes among the three considered model collections. It shows that compound words are the most frequent error source. The share of labels being erroneously refactored due to compound words ranges from 53% in the TelCo collection to 60% in the SAP Reference Model. However, in the three model collections also the adjective-noun-ambiguity and the irregular labels affect the overall refactoring result. Hence, in order to overcome these problems in the future, we propose the following strategy.

All the three error classes are based on the problem of ambiguity: for one label, theoretically two or even more interpretations are possible. In order to resolve these ambiguities we suggest the incorporation of a text corpus. Such a corpus can be consulted to check the frequency of the different refactoring outcomes. As a result, the most likely solution can be identified. As an example, consider the above introduced label *New User Registration*. If we look up the words *new user*, we will most likely get more matches than if we search for the word combination *user newly*. Hence, based on the frequency of the word combination in the corpus, we can identify action and business object and hence obtain a proper refactoring. The same strategy can be applied for irregular labels. As the lack of structure permits the reliable determination of action and business object, all possible action—business object combinations must be identified. Then, the corpus can be consulted to determine the most likely combination. An additional solution is the involvement of the user. Being aware of the fact that a very insecure decision will be made, the feedback of the user can incorporated to further improve the quality of the refactoring.

Although the discussed error sources negatively affected the overall refactoring effect, the share of noun labels which suffered from these problems was quite small. If the algorithm is used to point to potential problems, these shortcomings will not affect the modeling results as the user can simply correct the interpretation of the algorithm.

Turning to the refactoring effect, we observe the positive effect the algorithm creates in the process model collection. Consequently, this metric also includes the drawback that some verb–object labels might be flawed because of a wrong classification. Thus, for instance the verb–object label *Plan Reconciliation* is erroneously refactored into *Reconcile Plan*. Although these cases reduce the positive effect of our refactoring, only a small share of the labels is affected. This is also reflected by the overall refactoring gain. The final numbers after phase 4 amount to 50% for the SAP Reference Model, 49% for the TelCo models and 47% for the Signavio collection. Thereby, the development of the metric among the different collections is slightly different. While the refactoring effect for the SAP collections starts with 51% and, hence, remains constant, it decreases by 11 and 10 percentage points in the Telco and the Signavio collections. This development can be explained by the combination of the phase concept of our approach and the different labeling style distribution in the affected model collections. Concerning the phases, we can generally state that the certainty of the classification declines with each phase. Thus, if the algorithm is applied on a model collection with only a few action-noun labels, the number of erroneously refactored labels may exceed the number of correctly refactored labels and lead to an overall decrease of the refactoring effect. However, due to the introduced phase concept the trade off between a high refactoring gain and a suitable refactoring effect can be explicitly reached by reasonably defining the number of phases. Altogether, we can state that the refactoring gain and effect show that our approach significantly improves the labeling quality in the considered process models collections. Although some labels may be flawed, these can be easily and quickly corrected by the user as the amount is relatively small.

In summary, it can be stated that the refactoring approach performs satisfactory on the three model collections. A great share of labels can be corrected by the help of the algorithm and only a manageable amount of labels requires further inspection.

5. Related work

The work presented in this paper relates to research on applying natural language processing techniques to process modeling. We focus on three major aspects of this area, namely activity label quality, process model matching, and process model generation.

The benefits of a particular style of labeling have been discussed in practice. Labels composed of verbs followed by objects are promoted in [45,26,46]. Comparable guidelines have been defined for use cases in requirements engineering [47]. The actual advantage in terms of clarity is demonstrated in an experiment, which contrasts verb–object labels with action-noun labels in process models from practice [11]. Moreover, shorter labels have been shown to yield better process model comprehension in a correlational study [48], which is line with recommendations on sentence length [49,50]. These findings provide the foundation for design-oriented contributions on utilizing natural language techniques for process models. Delfmann et al. employed standard grammar parsers for checking naming conventions [51] and for autocompletion [12]. Gruhn and Laue [52] used linguistic analysis of activity labels for finding semantic errors, while Peters and Weidlich [53] examine the potential to extract glossaries from process model collections. Also the recognition of labeling styles has been inspected in prior publications [29,15,13]. This paper informs these works with a parsing technique that is tailored to the specifics of activity labels. Our evaluation demonstrates the limitations of using off-the-shelf parsers on labels that are not proper sentences. Our extensive evaluation with real-world process model collections provides detailed insight into the parameters that influence the refactoring result. In this way, our contribution is of fundamental importance to the area of activity label analysis.

Linguistic analysis of activity labels is an import step in matching business process models, an area dedicated to finding automatically a set of correspondences between semantically equivalent activities in two different process models. The identification of such correspondences is a prerequisite for merging and integrating process models [54–56]. Inspired by techniques on schema and ontology matching [57], there has been a series of works discussing the specifics of aligning and comparing process models. See Dijkman et al. [58] for an overview. Some of these works assume that an ontology exists for matching the labels [59–61]. Also techniques for semantic annotation have been proposed [62–65]. An intrinsic problem of matching business process models is the complexity of the search space once 1:*n* matches are investigated [66–68]. Our work complements this stream of research. Our recognition technique has the potential to improve matchers significantly in terms of precision, e.g., when only matches between actions and between business objects are considered.

Natural language processing techniques are not only important for analyzing process models, but also for constructing them. There have been various approaches by different groups. A group from Klagenfurt has been working on parsing German text into a generic meta-model, and semi-automatically generating UML activity diagrams from its content [69,70]. A fully automated approach is presented in Yue et al. [71], but it assumes the text to follow the conventions of use-case descriptions. A group from Rio de Janeiro works with group stories provided in Portuguese for generating BPMN models [72]. Syntax parsing and textual patterns are used to create BPMN models in the approach of the *R-BPD* toolkit [73,74]. Further approaches on generating BPMN from text have been presented [75–78]. Our approach is informative to these works, as it provides the basis for a text-to-model model-to-text round-tripping. The challenges of identifying the components of an activity label are essential for building natural language text that appears to be natural. While model generation faces challenges such as anaphora resolution [78], the problem for model-to-text generation will be the other way around to introduce anaphora in an appropriate way.

6. Conclusion

In this paper we addressed the problem of activity label quality in business process models. We designed a technique for the recognition of labeling styles, and the automatic refactoring of labels of undesirable style. More specifically, we developed a parsing algorithm fit to the particularities of activity labels, and utilized standard natural language tools like WordNet and the Stanford Parser where appropriate. Using three samples of business process model collections from practice with differing style distributions, we were able to demonstrate the applicability of our technique. In comparison to a straight-forward application of standard tools (f-measure below 0.20 for certain collections), our technique provided much more stable results (between 0.65 and 0.95 for the different styles in different collections). As a result, the refactoring gain for the collections of our sample ranges from 64% to 77%. The overall refactoring effect is positive with roughly 0.5 on a scale from -1 to $+1$.

The results we achieved demonstrate the potential of utilizing natural language processing techniques for quality assurance of process models. Our work has to be reflected from the perspective of some limitations. Up until now, we focused on activity labels in English language. Therefore, the results are bound to the specifics of this language. We are currently experimenting with other languages, namely German and Dutch. A preliminary observation from these tests is that label parsing can be done more reliably for the latter languages due to a richer morphology of verb forms. Part of speech ambiguities like zero derivation in English are in those languages much less frequent, such that the parsing can identify actions and business objects more precisely. Languages like German are also easier to parse due to the fact that compounds are concatenated into a single word. In the future work, we aim to quantitatively measure the effect of these language differences using German and Dutch process model collections.

The results of our analysis are also bound to the specifics of the three process model collections that we used. They are not representative in a statistical sense. We tried to avoid a bias in whatever aspect by selecting collections with diverse characteristics in various dimensions. We have included collections created by modelers of different degrees of expertise, using different graph-based process modeling languages and different modeling tools, having different skewness in their distribution of labeling styles. Altogether, more than 10,000 activity labels were used in our evaluation. This provides us with confidence that the results reflect the potential of our technique to be applied in practice. We are currently discussing with a vendor who aims to design a labeling convention checking in their process modeling tool. This will help us to investigate our technique from a usability perspective in the future research.

References

[1] W. Kettinger, J. Teng, S. Guha, Business process change: a study of methodologies, techniques, and tools, MIS Quarterly 11 (1997) 55–80.

[2] J. Mendling, Empirical studies in process model verification, Lecture Notes in Computer Science, Transactions on Petri Nets and Other Models of Concurrency II, Special Issue on Concurrency in Process-Aware Information Systems 2 (2009) 208–224.

[3] M. Rosemann, Potential pitfalls of process modeling: part A, Business Process Management Journal 12 (2) (2006) 249–254.
[4] W. Bandara, G.G. Gable, M. Rosemann, Factors and measures of business process modelling: model building through a multiple case study, European Journal of Information Systems 14 (2005) 347–360.
[5] H.A. Reijers, J. Mendling, A study into the factors that influence the understandability of business process models, IEEE Transactions on Systems, Man, and Cybernetics, Part A 41 (3) (2011) 449–462.
[6] J. Becker, M. Rosemann, C. Uthmann, Guidelines of business process modeling, in: W. van der Aalst, J. Desel, A. Oberweis (Eds.), Business Process Management. Models, Techniques, and Empirical Studies, Springer, Berlin, 2000, pp. 30–49.
[7] J. Mendling, H.A. Reijers, W.M.P. van der Aalst, Seven process modeling guidelines (7PMG), Information and Software Technology 52 (2) (2010) 127–136.
[8] W. van der Aalst, Workflow verification: finding control-flow errors using petri-net-based techniques, Business Process Management. Lecture Notes in Computer Science, vol. 1806, Springer, 2000, pp. 161–183.
[9] J. Krogstie, G. Sindre, H. Jørgensen, Process models representing knowledge for action: a revised quality framework, European Journal of Information Systems 15 (1) (2006) 91–102.
[10] C. Fillies, G. Wood-Albrecht, F. Weichhardt, Pragmatic applications of the semantic web using SemTalk, Computer Networks 42 (5) (2003) 599–615.
[11] J. Mendling, H.A. Reijers, J. Recker, Activity labeling in process modeling: empirical insights and recommendations, Information Systems 35 (4) (2010) 467–482.
[12] 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: ECIS 2009, 2009.
[13] H. Leopold, S. Smirnov, J. Mendling, Recognising activity labeling styles in business process models, Enterprise Modelling and Information Systems Architectures 6 (1) (2011) 16–29.
[14] D. Klein, C.D. Manning, Accurate unlexicalized parsing, 41st Meeting of the Association for Computational Linguistics (2003) 423–430.
[15] H. Leopold, S. Smirnov, J. Mendling, Refactoring of process model activity labels, NLDB 2010, Lecture Notes in Computer Science, vol. 6177, Springer, 2010, pp. 268–276.
[16] A. Basu, A. Kumar, Research commentary: workflow management issues in e-business, Information Systems Research 13 (1) (2002) 1–14.
[17] W. van der Aalst, A. Hirnschall, H. Verbeek, An alternative way to analyze workflow graphs, CAiSE 2002, Lecture Notes in Computer Science, vol. 2348, Springer, 2002, pp. 535–552.
[18] 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 (5) (2011) 448–466.
[19] S. Sun, J. Zhao, J. Nunamaker, O. Liu Sheng, Formulating the dataflow perspective for business process management, Information Systems Research 17 (4) (2006) 374–391.
[20] I. Weber, J. Hoffmann, J. Mendling, Beyond soundness: on the verification of semantic business process models, Distributed and Parallel Databases 27 (3) (2010) 271–343.
[21] 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 (7) (2011) 1026–1043.
[22] 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 (1) (1999) 65–104.
[23] J. Crampton, H. Khambhammettu, Delegation and satisfiability in workflow systems, in: SACMAT 2008, ACM, New York, NY, USA, 2008, pp. 31–40.
[24] M. Strembeck, J. Mendling, Modeling process-related RBAC models with extended UML activity models, Information & Software Technology 53 (5) (2011) 456–483.
[25] W. van der Aalst, A. Kumar, XML-based schema definition for support of interorganizational workflow, Information Systems Research 14 (1) (2003) 23–46.
[26] A. Sharp, P. McDermott, Workflow Modeling: Tools for Process Improvement and Application Development, Artech House Publishers, 2001.
[27] J. Mendling, Metrics for process models: empirical foundations of verification, error prediction, and guidelines for correctness, Lecture Notes in Business Information Processing, vol. 6, Springer, 2008.
[28] R. Dixon, Deriving verbs in english, Language Sciences 30 (1) (2008) 31–52.
[29] H. Leopold, S. Smirnov, J. Mendling, On labeling quality in business process models, in: GI-Workshop EPK 2009, 2009.
[30] J.F. Allen, Natural language processing, in: Encyclopedia of Computer Science, John Wiley and Sons Ltd., Chichester, UK, 1993, pp. 1218–1222.
[31] A.K. Joshi, Natural language processing, Science 253 (5025) (1991) 1242–1249.
[32] P. Jackson, I. Moulinier, Natural Language Processing for Online Applications. Text Retrieval, Extraction and Categorization, Natural Language Processing, vol. 5, John Benjamins, 2002.
[33] 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 (2) (1994) 161–174.
[34] W.J. Hutchins, Machine Translation: Past, Present, Future, John Wiley & Sons Inc., New York, NY, USA, 1986.
[35] 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 (2) (1990) 79–85.
[36] L. Rabiner, B.H. Juang, Fundamentals of Speech Recognition, Prentice-Hall, 1993.
[37] F. Jelinek, Statistical Methods for Speech Recognition, The MIT Press, 1998.
[38] M.P. Marcus, M. Marcinkiewicz, B. Santorini, Building a large annotated corpus of english: the Penn Treebank, Computational Linguistics 19 (1993) 313–330.
[39] D. Klein, C.D. Manning, Fast exact inference with a factored model for natural language parsing, NIPS 2003, vol. 15, MIT Press, 2003.
[40] G. Miller, C. Fellbaum, WordNet: An Electronic Lexical Database, MIT Press, Cambridge, MA, 1998.
[41] M. Davies, 〈http://www.wordfrequency.info〉 Word Frequency Data from the Corpus of Contemporary American English (COCA) [online] (2011) [cited 3 May 2011].
[42] G.A. Miller, WordNet: a lexical database for english, Communications of the ACM 38 (11) (1995) 39–41.
[43] G. Keller, T. Teufel, SAP(R) R/3 Process Oriented Implementation: Iterative Process Prototyping, Addison-Wesley, 1998.
[44] R.A. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, ACM Press, Addison-Wesley, 1999.
[45] L. Miles, Techniques of Value Analysis and Engineering, McGraw-Hill, 1961.
[46] T. Malone, K. Crowston, G. Herman (Eds.), Organizing Business Knowledge: The MIT Process Handbook, The MIT Press, 2003.
[47] K.T. Phalp, J. Vincent, K. Cox, Improving the quality of use case descriptions: empirical assessment of writing guidelines, Software Quality Journal 15 (4) (2007) 383–399.
[48] J. Mendling, M. Strembeck, Influence factors of understanding business process models, BIS 2008, Lecture Notes in Business Information Processing, vol. 7, Springer, 2008, pp. 142–153.
[49] H. Gretchen, Readability and computer documentation, ACM Journal of Computer Documentation 24 (3) (2000) 122–131.
[50] R. Flesch, How to Test Readability, Harper and Brothers, New York, NY, USA, 1951.
[51] A. 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 (2) (2009) 3–19.
[52] V. Gruhn, R. Laue, Detecting common errors in event-driven process chains by label analysis, Enterprise Modelling and Information Systems Architectures 6 (1) (2011) 3–15.
[53] N. Peters, M. Weidlich, Automatic generation of glossaries for process modelling support, Enterprise Modelling and Information Systems Architectures 6 (1) (2011) 30–46.
[54] G. Preuner, S. Conrad, M. Schrefl, View integration of behavior in object-oriented databases, Data & Knowledge Engineering 36 (2) (2001) 153–183.
[55] A. Basu, R.W. Blanning, Synthesis and decomposition of processes in organizations, Information Systems Research 14 (4) (2003) 337–355.
[56] M. La Rosa, M. Dumas, R. Uba, R.M. Dijkman, Merging business process models, OTM 2010, Lecture Notes in Computer Science, vol. 6426, Springer, 2010, pp. 96–113.
[57] J. Euzenat, P. Shvaiko, Ontology Matching, Springer, 2007.

[58] R.M. Dijkman, M. Dumas, B.F. van Dongen, R. Käärik, J. Mendling, Similarity of business process models: metrics and evaluation, Information Systems 36 (2) (2011) 498–516.
[59] M. Ehrig, A. Koschmider, A. Oberweis, Measuring similarity between semantic business process models, in: APCCM 2007, Australian Computer Science Communications, Ballarat, Victoria, Australia, vol. 67, 2007, pp. 71–80.
[60] D. Grigori, J. Corrales, M. Bouzeghoub, A. Gater, Ranking BPEL processes for service discovery, IEEE Transactions on Services Computing 3 (3) (2010) 178–192.
[61] M. Lincoln, M. Golani, A. Gal, Machine-assisted design of business process models using descriptor space analysis, in: R. Hull, J. Mendling, S. Tai (Eds.), Business Process Management—Eighth International Conference, BPM 2010, Hoboken, NJ, USA, 13–16 September 2010. Proceedings, Lecture Notes in Computer Science, vol. 6336, Springer, 2010, pp. 128–144.
[62] M. Born, F. Dörr, I. Weber, User-friendly semantic annotation in business process modeling, WISE 2007 Workshops, Lecture Notes in Computer Science, vol. 4832, Springer, 2007, pp. 260–271.
[63] C. Francescomarino, M. Rospocher, L. Serafini, P. Tonella, Semantically-aided business process modeling, ISWC 2009, Lecture Notes in Computer Science, vol. 5823, Springer, Berlin, Heidelberg, 2009, pp. 114–129.
[64] Y. Lin, H. Ding, Ontology-based semantic annotation for semantic interoperability of process models, in: CIMCA 2005, IEEE Computer Society, Los Alamitos, CA, USA, 2005, pp. 162–167.
[65] C. Francescomarino, P. Tonella, Supporting ontology-based semantic annotation of business processes with automated suggestions, BMMDS/EMMSAD 2009, Lecture Notes in Business Information Processing, vol. 29, Springer, 2009, pp. 211–223.
[66] M. Weidlich, R. Dijkman, J. Mendling, The ICoP framework: identification of correspondences between process models, CAiSE 2010, Lecture Notes in Computer Science, vol. 6051, Springer, 2010, pp. 483–498.
[67] S. Smirnov, R. Dijkman, J. Mendling, M. Weske, Meronymy-based aggregation of activities in business process models, ER 2010, Lecture Notes in Computer Science, vol. 6412, 2010, pp. 1–14.
[68] H. Leopold, J. Mendling, H. Reijers, On the automatic labeling of process models, CAiSE 2011, Lecture Notes in Computer Science, vol. 6741, Springer, 2011, pp. 512–520.
[69] G. Fliedl, C. Kop, H. Mayr, From textual scenarios to a conceptual schema, Data & Knowledge Engineering 55 (1) (2005) 20–37.
[70] G. Fliedl, C. Kop, H. Mayr, A. Salbrechter, J. Vöhringer, G. Weber, C. Winkler, Deriving static and dynamic concepts from software requirements using sophisticated tagging, Data & Knowledge Engineering 61 (3) (2007) 433–448.
[71] T. Yue, L.C. Briand, Y. Labiche, An automated approach to transform use cases into activity diagrams, ECMFA 2010, Lecture Notes in Computer Science, vol. 6138, Springer, 2010, pp. 337–353.
[72] J.C. de AR Gonçalves, F.M. Santoro, F.A. Baião, Business process mining from group stories, in: CSCWD 2009, IEEE Computer Society, 2009, pp. 161–166.
[73] A. Ghose, G. Koliadis, A. Chueng, Rapid business process discovery R-BPD, ER 2007, Lecture Notes in Computer Science, vol. 4801, Springer, Berlin, Heidelberg, 2007, pp. 391–406.
[74] A. 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.
[75] H.J. Wang, J.L. Zhao, L.-J. Zhang, Policy-driven process mapping (PDPM): discovering process models from business policies, Decision Support Systems 48 (1) (2009) 267–281.
[76] A. Sinha, A. Paradkar, P. Kumanan, B. Boguraev, An Analysis Engine for Dependable Elicitation on Natural Language Use Case Description and Its Application to Industrial Use Cases, Technical Report, IBM, 2008.
[77] A. Sinha, A. Paradkar, Use cases to process specifications in business process modeling notation, in: 2010 IEEE International Conference on Web Services, IEEE, 2010, pp. 473–480.
[78] F. Friedrich, J. Mendling, F. Puhlmann, Process model generation from natural language text, CAiSE 2011, Lecture Notes in Computer Science, vol. 6741, Springer, 2011, pp. 482–496.